Transverse-Read-Codes for Domain Wall Memories

Yeow Meng Chee, *Senior Member, IEEE*, Alexander Vardy, *Fellow, IEEE*, Van Khu Vu, and Eitan Yaakobi, *Senior Member, IEEE*

***Abstract*—Transverse-read is a novel technique to detect the number of '1's stored in a *domain wall memory*, also known as *racetrack memory*, without shifting any domains. Motivated by the technique, we propose a novel scheme to combine transverse-read and shift-operation such that we can reduce the number of shift-operations while still achieving high capacity. We also show that this scheme is helpful to correct errors in domain wall memory. A set of valid-words in this transverse-read channel is called a *transverse-read code*. Our goal in this work is to study transverse-read codes with respect to their properties, capacity, and applications. We first present several properties of transverse-read codes and show that they are equivalent to a family of constrained codes. Then, we compute the maximal asymptotic rate of transverse-read codes for certain parameters. Furthermore, we also present several constructions of transverse-read codes with high rate. Finally, we design several transverse-read codes that can correct limited-shift-errors and limited-magnitude errors in domain wall memory.**

***Index Terms*—Racetrack memories, constrained codes, transverse-read, shift-errors.**

I. Introduction

SPINTRONIC domain-wall memory (DWM), also referred as *racetrack memory*, is a promising candidate as a memory solution that can overcome the density limitations of spin-transfer torque magnetic memory (STT-MRAM), while still retaining its static energy benefits [2], [3], [4], [5]. DWM is constructed from ferromagnetic nanowires, referred to as *tapes* or *racetracks*, which are separated into domains and are connected to a single or a few access transistors to create access ports. The state of the magnetic domains is accessed by *shifting* them along the nanowire and aligning the target domain to an access device. Unfortunately, due to process variation of deeply-scaled domain-wall memories [2], slight fluctuations in current combined with imperfections in the nanowires can cause faults in the shift process. These faults include *over*- and *under*-shifting of the tape, and thus for domain-wall memory to become viable, the shifting reliability must be addressed. As a result, several innovative approaches have been developed to detect and correct shift-errors in racetrack memory [6], [7], [8], [9], [10]. Besides that, the access latency and the energy consumption in racetrack memory depend on the average number of shift-operations. Several works studied how to reduce the number of shift operations in racetrack memory [11], [12].

Another approach to overcome the faults in the shifting process of the DWM was proposed recently in [13], [14], [15]. In these works, a novel *transverse-read* (TR) mechanism was developed in order to provide global information about the data stored within a nanowire. In particular, transverse-read can detect the number of ones among the data stored in a DWM without shifting any domains, while still requiring ultra-low power. However, detecting only the number of ones in the DWM significantly reduces the information rate that can be stored within the memory. Hence, the authors of [15] also demonstrated how TR can be applied to partial segments of the nanowire, such as from an end to an access point or between two access points. This enables a segmented TR which allows access to all of the bits of an arbitrarily long nanowire in several steps, while maintaining isolated current paths. While independently sensing several segments can increase the memory's information rate, this increase is still far from reaching its full potential.

In this work, we propose a novel scheme that simultaneously combines the two important features of DWM. On one hand, we use transverse-reads in order to sense the number of ones between two consecutive access points, and on the other hand we still shift all the domains so that we can transverse-read to sense the number of ones in different segments every time. In general, we consider a message $\boldsymbol{x} = (x_1, \ldots, x_n)$ of n information bits stored in n domains and consecutive access points such that each time we can transverse-read a segment of length ℓ. That is, in the first read, the Hamming weight of the first length-ℓ segment $x_1, \ldots, x_\ell$ is sensed. Next, we shift all domains in δ positions and sense the Hamming weight of the length-ℓ segment $(x_{\delta+1}, \ldots, x_{\delta+\ell})$ in the second read. We keep shifting and sensing until the last segment $(x_{k\delta+1}, \ldots, x_{k\delta+\ell})$ (for simplicity, we assume that there is an integer k such that $n = k\delta + \ell$). For example, we consider the case $n = 12, \delta = 2$, and $\ell = 4$. If $\boldsymbol{x} = (0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0)$, the output in our reading scheme is $(1, 2, 3, 2, 0)$. There exist other vectors, for example $\boldsymbol{y} = (0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0) \neq \boldsymbol{x}$, that have the same output $(1, 2, 3, 2, 0)$. Hence, we may not obtain the full capacity using this scheme. First, we observe that the information rate in this scheme depends on δ and ℓ. For example, when $\delta = \ell = 2$, we show in this paper that the information rate is $\log(3)/2 \approx 0.7925$. Then, we observe that this scheme significantly reduces the number of

Manuscript received 30 January 2023; revised 8 October 2023 and 3 November 2023; accepted 11 November 2023. Date of current version 10 January 2024. This paper was presented in part at the 2021 IEEE International Symposium on Information Theory (ISIT) [DOI: 10.1109/ISIT45174.2021.9518271]. *(Corresponding author: Van Khu Vu.)*

Yeow Meng Chee and Van Khu Vu are with the Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore (e-mail: ymchee@nus.edu.sg; isevvk@nus.edu.sg).

Alexander Vardy, deceased, was with the Department of Electrical and Computer Engineering, University of California at San Diego, La Jolla, CA 92093 USA (e-mail: avardy@ucsd.edu).

Eitan Yaakobi is with the Department of Computer Science, Technion—Israel Institute of Technology, Haifa 3200003, Israel (e-mail: yaakobi@cs.technion.ac.il).

Digital Object Identifier 10.1109/JSAIT.2023.3334303

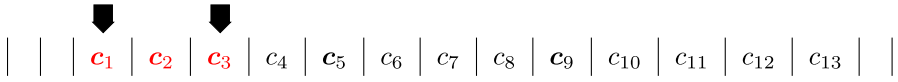

Fig. 1. Racetrack memory with twelve data domains and two heads.

shift-operations by a factor of about δ times. For example, when $\delta = 2$, if we just shift normally about $n/2$ times, we can read only half of the information bits, however using our scheme, we can achieve information rate roughly 0.7925. Our first question of interest is whether we can achieve higher information rates. Hence, we are interested in finding the trade-off between the number of shift-operations and the maximal information rate under this setup. Furthermore, we can show that this scheme is also helpful to correct shift-errors in DWM. From a practical point of view, this scheme captures the two features of DWM in order to significantly reduce the number of shift operations and mitigate the shift errors, while still supporting high information rates. From a theoretical point of view, it poses several interesting challenges in combinatorics and algorithms.

The rest of this paper is organized as follows. In Section II, we present the necessary notations and define the codes formally. In Section III, we study transverse-read codes with respect to their properties, maximal asymptotic rates, and constructions. Then, in Section IV, we show that our scheme of using transverse-read codes is helpful to correct shift-errors in domain wall memories. Finally, in Section V, we summarise our contributions in this work and discuss some future works.

II. Definitions and Preliminaries

Let $\Sigma_q = \{0, 1, \ldots, q-1\}$ denote the alphabet of size q and $[n]$ denote the set $\{1, 2, \ldots, n\}$. For each sequence $\boldsymbol{u} = (u_1, \ldots, u_n) \in \Sigma_q^n$, let $\boldsymbol{u}_{[i;\ell]} = (u_i, u_{i+1}, \ldots, u_{i+\ell-1})$, $1 \leq i \leq n-\ell+1$, denote the length-ℓ substring of $\boldsymbol{u}$, starting from index i. A q-ary code $\mathcal{C}$ of length n is a set of q-ary sequences of length n, that is, $\mathcal{C} \subseteq \Sigma_q^n$. For each code $\mathcal{C}$ of length n, we define its rate to be $R(\mathcal{C}) = \log_q(|\mathcal{C}|)/n$, where $|\mathcal{C}|$ is the size of the code $\mathcal{C}$. Let $\Phi : \Sigma_q^\ell \to \mathbb{N}$ be a mapping from a q-ary codeword of length ℓ to a natural number. Let n, ℓ, δ, k be integers such that $n - \ell = k\delta$. We define the following mapping,

$$\Phi_{\ell,\delta} : \Sigma_q^n \to \mathbb{N}^{k+1},$$

where $\Phi_{\ell,\delta}(\boldsymbol{x}) = (\Phi(\boldsymbol{x}_{[1;\ell]}), \Phi(\boldsymbol{x}_{[\delta+1;\ell]}), \ldots, \Phi(\boldsymbol{x}_{[k\delta+1;\ell]})) \in \mathbb{N}^{k+1}$, given a vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in \Sigma_q^n$. The vector $\Phi_{\ell,\delta}(\boldsymbol{x})$ is called the (Φ, ℓ, δ)-*segment read vector* of $\boldsymbol{x}$. The mapping $\Phi_{\ell,\delta}$ may not be injective, and thus there may be two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $\Phi_{\ell,\delta}(\boldsymbol{x}) = \Phi_{\ell,\delta}(\boldsymbol{y})$. Furthermore, $\Phi_{\ell,\delta}$ is also not surjective, that is, there is a vector $\boldsymbol{v} \in \mathbb{N}^{k+1}$ such that there does not exist any vector $\boldsymbol{x} \in \Sigma_q^n$ such that $\Phi_{\ell,\delta}(\boldsymbol{x}) = \boldsymbol{v}$. A vector $\boldsymbol{u} \in \mathbb{N}^{k+1}$ is called a *valid* (Φ, ℓ, δ)-*segment read vector* if there exists a vector $\boldsymbol{x} \in \Sigma_q^n$ such that $\Phi_{\ell,\delta}(\boldsymbol{x}) = \boldsymbol{u}$. A channel that only accepts the valid (Φ, ℓ, δ)-segment read vectors is called the (Φ, ℓ, δ)-*segment read channel*.

Note that it is also possible to define the cyclic version of these segment-read vectors, however, we prefer the more practical non-cyclic version. In this work, we always assume that $n-\ell = k\delta$, ℓ and δ are fixed while n and k tend to infinity.

If $\delta = 1$ and $\Phi_{\ell,1}$ is injective, that is $\Phi_{\ell,1}(\boldsymbol{x}) \neq \Phi_{\ell,1}(\boldsymbol{y})$ for all $\boldsymbol{x} \neq \boldsymbol{y}$, then the $(\Phi, \ell, 1)$-segment read vector of $\boldsymbol{x}$ is equivalent to an ℓ-symbol read vector of $\boldsymbol{x}$, defined and studied in [20], [21]. In this case, any vector in Σ_m^{k+1} is a valid $(\Phi, \ell, 1)$-segment read vector, where $m = q^\ell$. We are interested in a code with the ability to correct errors which have been well studied in the context of ℓ-symbol read channel [20], [21], [22], [23], [24]. There are many constructions of codes correcting substitution errors [20], [21], [22], [23], [24], [28] and some other codes correct synchronization errors, including deletions and sticky insertions [18].

In the general case, when Φ can be any mapping (may not injective), finding the maximal number of the (Φ, ℓ, δ)-segment read vectors and the capacity of the (Φ, ℓ, δ)-segment read channel is an interesting challenge. Owing to their application in nanopore sequencing of DNA [25], the (Φ, ℓ, δ)-segment read channel has been studied independently recently. However, only the case $\delta = 1$ was investigated and the codes do not have the ability to correct errors. In this work, we focus on the case where Φ is the weight function owing to the application in racetrack memory and consider various cases of δ.

We now examine a model of domain wall memory of n domains and two access points that are ℓ positions far apart. A message, which is a binary vector of length n, will be stored in these n domains. Two read-ports can transverse-read to sense the weight of a segment of length ℓ. For example, Figure 1 illustrates a domain wall memory with twelve domains and two access ports. To read the information in the domain wall memory, besides the transverse-read technique, we also need the shift operation. In each shift operation in the domain wall memory, all domains together move δ positions. Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \Sigma_q^n$ and let the weight function $\boldsymbol{w} : \Sigma_q^n \to \Sigma_{(q-1)n+1}$ be such that for any $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \Sigma_q^n$ then $\boldsymbol{w}(\boldsymbol{x}) = \sum_{i=1}^n x_i$. So, $\boldsymbol{w}(\boldsymbol{x})$ is the weight of the vector $\boldsymbol{x}$. We define

$$TR_{\ell,\delta} : \Sigma_q^n \to \Sigma_{(q-1)\ell+1}^{k+1}$$

such that $TR_{\ell,\delta}(\boldsymbol{x}) = (\boldsymbol{w}(\boldsymbol{x}_{[1,\ell]}), \boldsymbol{w}(\boldsymbol{x}_{[\delta+1,\ell]}), \ldots, \boldsymbol{w}(\boldsymbol{x}_{[k\delta+1,\ell]}))$. We note that, owing to the application in domain-wall memory, in this work, we only focus on the case $q = 2$ and the stored message $\boldsymbol{x} \in \Sigma_2^n$. The output in the transverse-reading scheme is $TR_{\ell,\delta}(\boldsymbol{x})$. We observe that $TR_{\ell,\delta}(\boldsymbol{x})$ is actually the (Φ, ℓ, δ)-segment read vector of $\boldsymbol{x}$. So, $TR_{\ell,\delta}(\boldsymbol{x})$ is called the (ℓ, δ)-*transverse-read vector* of $\boldsymbol{x}$. The mapping $TR_{\ell,\delta}$ is not injective, that is, there are two vectors $\boldsymbol{x}$ and $\boldsymbol{y}$ such that $TR_{\ell,\delta}(\boldsymbol{x}) = TR_{\ell,\delta}(\boldsymbol{y})$. We are interested in a set of vectors $\boldsymbol{x}$ such that the mapping $TR_{\ell,\delta}$ is injective for this set.

Definition 1: Let n, ℓ, δ, k be integers such that $n-\ell = k\delta$.

1) A binary (ℓ, δ)-*transverse-read code* of length n, denoted by $\mathcal{C}_{TR}(n; \ell, \delta)$, is defined as a set of vectors such that for any two vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}_{TR}(n; \ell, \delta)$,

$TR_{\ell,\delta}(\boldsymbol{x}) \neq TR_{\ell,\delta}(\boldsymbol{y})$. That is,

$$\mathcal{C}_{TR}(n;\ell,\delta) = \left\{\boldsymbol{x} \in \Sigma_2^n : \forall \boldsymbol{x}_i \neq \boldsymbol{x}_j, TR_{\ell,\delta}(\boldsymbol{x}_i) \neq TR_{\ell,\delta}(\boldsymbol{x}_j)\right\}.$$

2) The largest size of a length-n binary (ℓ,δ)-transverse-read code will be denoted by $A(n;\ell,\delta)$ and the maximal asymptotic rate for fixed ℓ and δ is given by

$$\mathcal{R}(\ell,\delta) = \limsup_{k\to\infty} \frac{\log_2(A(n;\ell,\delta))}{k\delta+\ell},$$

where $n = \ell + k\delta$.

Furthermore, the mapping $TR_{\ell,\delta}$ is not always surjective, that is, there exist δ, ℓ and a vector $\boldsymbol{u} \in \Sigma_{\ell+1}^{k+1}$ such that for all $\boldsymbol{x} \in \Sigma_2^n$, it holds that $TR_{\ell,\delta}(\boldsymbol{x}) \neq \boldsymbol{u}$. So, we now define a new class of vectors in $\Sigma_{\ell+1}^{k+1}$.

Definition 2: Let n, ℓ, δ, k be integers such that $n-\ell = k\delta$.

- A vector $\boldsymbol{u} \in \Sigma_{\ell+1}^{k+1}$ is called a *valid* (ℓ,δ)*-transverse-read vector* if there exists a vector $\boldsymbol{x} \in \Sigma_2^n$ such that $TR_{\ell,\delta}(\boldsymbol{x}) = \boldsymbol{u}$.
- The set of all such vectors $\boldsymbol{u}$ of length $k+1$ is called the *valid* (ℓ,δ)*-transverse-read code* of length n and is denoted by $TR(n;\ell,\delta) \subseteq \Sigma_{\ell+1}^{k+1}$.
- The maximal asymptotic rate of the valid (ℓ,δ)-transverse-read code, given ℓ, δ, is

$$\mathcal{R}(\ell,\delta) = \limsup_{k\to\infty} \frac{\log_2(|TR(n;\ell,\delta)|)}{k\delta+\ell},$$

where $n = k\delta + \ell$.

We note that in this work, we only consider the case where $(n-\ell)/\delta = k$ is an integer.

Example 1: Let $n = 13, \ell = 3, \delta = 2, k = 5$. Let $\boldsymbol{x} = (0,0,1,1,1,1,0,1,0,0,1,1,0) \in \Sigma_2^{13}$. Then, $TR_{3,2}(\boldsymbol{x}) = (1,3,2,1,1,2)$ is the $(3,2)$-transverse-read vector of $\boldsymbol{x}$. Hence, $(1,3,2,1,1,2)$ is a valid $(3,2)$-transverse-read vector. Let $\boldsymbol{v} = (0,3,3,0,0,3)$. There does not exist any vector $\boldsymbol{y} \in \Sigma_2^{13}$ such that $TR_{3,2}(\boldsymbol{y}) = \boldsymbol{v}$. Hence, $\boldsymbol{v} = (0,3,3,0,0,3)$ is not a valid $(3,2)$-transverse-read vector.

Each codeword in the (ℓ,δ)-transverse-read code is equivalent to a valid (ℓ,δ)-transverse-read vector. Hence, $|TR(n;\ell,\delta)| = A(n;\ell,\delta)$. The channel that only accepts valid (ℓ,δ)-transverse-read vectors is called *the* (ℓ,δ)*-transverse-read channel*. The capacity of the channel is the maximal asymptotic rate of the (ℓ,δ)-transverse-read code.

We note that to read data in a racetrack memory, we normally read bit by bit and need a shift operation each time we read a bit. Hence, to read a message of length n, we need to shift n times. Using the transverse-read, in each time, we can scan a segment and will shift all domains by δ positions to read the next segment. Thus, to read a message of length n, we only need to shift $k = \lceil n/\delta \rceil$ times. For example, in Figure 1, two access ports can transverse-read a segment of length three and each time, we shift all domains two positions. If the stored message is a binary vector of length 13, $\boldsymbol{c} = (c_1, c_2, \ldots, c_{13})$, the output in our reading scheme is $TR_{3,2}(\boldsymbol{c}) = (\boldsymbol{w}(c_1,c_2,c_3), \boldsymbol{w}(c_3,c_4,c_5), \ldots, \boldsymbol{w}(c_{11},c_{12},c_{13}))$. In this case, we only need to shift all domains six times. However, given δ and ℓ, the maximal information rate in racetrack memories is $\mathcal{R}(\ell,\delta)$, which may not achieve the full capacity. Hence, in this work, we are interested in finding the maximal size $A(n,\ell,\delta)$ and the maximal asymptotic rate $\mathcal{R}(\ell,\delta)$. Given δ, we are also interested in finding the optimal ℓ such that the asymptotic rate $\mathcal{R}(\ell,\delta)$ is maximal. Furthermore, we also seek for some constructions of (ℓ,δ)-transverse-read codes with efficient encoding/decoding algorithms.

Besides that, both shift-operation and transverse-read may not work perfectly and errors may occur. It is known that the shift-errors can be modelled as synchronizations, including sticky-insertions and deletions [7], [8], [9]. We also see that errors in transverse-read vector may cause some substitution errors. Hence, in this work, we also study some transverse-read codes which can correct shift-errors and substitutions errors.

From Definitions 1 and 2, we can see that an (ℓ,δ)-transverse-read code is equivalent to a valid (ℓ,δ)-transverse-read code through the mapping $TR_{\ell,\delta}$. Hence, in this work, we use the term (ℓ,δ)-transverse-read code for both.

III. Transverse-Read Codes

In this section, given ℓ, δ, we study (ℓ,δ)-transverse-read codes, their properties, and aim to find the maximal asymptotic rate of these codes. We are also interested in constructing these codes with efficient encoding and decoding algorithms.

We first present several basic results on $A(n;\ell,\delta)$ and $\mathcal{R}(\ell,\delta)$ in the following theorem.

Theorem 1: Let n, ℓ, δ, and $k = (n-\ell)/\delta$ be all positive integers.

1) For $\ell = 1$, it holds that $A(n;\ell=1,\delta) = 2^{\frac{n-1}{\delta}+1}$ and $\mathcal{R}(\ell=1,\delta) = 1/\delta$.
2) For $\ell \leq \delta$, it holds that $A(n;\ell,\delta) = (\ell+1)^{\frac{n-\ell}{\delta}+1}$ and $\mathcal{R}(\ell,\delta) = \frac{\log_2(\ell+1)}{\delta}$.
3) For $\delta = 1$ and some constant ℓ, it holds that $A(n;\ell,\delta=1) \geq 2^{n-\ell}$ and $\mathcal{R}(\ell,\delta=1) = 1$.

Proof:

1) To prove this claim for $\ell = 1$ and $k = (n-1)/\delta$, we consider a vector $\boldsymbol{x} = (x_1,\ldots,x_n) \in \Sigma_2^n$ and its transverse-read vector $TR_{\ell,\delta}(\boldsymbol{x}) = (x_1, x_{\delta+1}, \ldots, x_{k\delta+1}) \in \Sigma_{\ell+1}^{k+1}$. We observe that for any vector $\boldsymbol{u} \in \Sigma_2^{k+1}$, $\boldsymbol{u}$ is a valid (ℓ,δ)-transverse-read vector. Hence, $A(n,\ell=1,\delta) = |TR(k,\ell=1,\delta)| = 2^{k+1}$ and thus $\mathcal{R}(\ell=1,\delta) = \limsup_{k\to\infty} \frac{k+1}{k\delta+\ell} = \frac{1}{\delta}$.
2) We now consider the case $\ell \leq \delta$. Let $\boldsymbol{x} = (x_1,\ldots,x_n) \in \Sigma_2^n$ and $TR_{\ell,\delta}(\boldsymbol{x}) = (w(\boldsymbol{x}_{[1;\ell]}), w(\boldsymbol{x}_{[\delta+1;\ell]}), \ldots, w(\boldsymbol{x}_{[k\delta+1;\ell]})) \in \Sigma_{\ell+1}^{k+1}$. Since all segments $\boldsymbol{x}_{[i\delta+1;\ell]}$, for $0 \leq i \leq k$, are non-overlapping, any vector $\boldsymbol{u} \in \Sigma_{\ell+1}^{k+1}$ is a valid (ℓ,δ)-transverse-read vector. Hence, $A(n,\ell,\delta) = |TR(k,\ell,\ell)| = (\ell+1)^{k+1} = (\ell+1)^{\frac{(n-\ell)}{\delta}+1}$ and thus $\mathcal{R}(\ell,\delta) = \limsup_{k\to\infty} \frac{(k+1)(\log_2(\ell+1))}{k\delta+\ell} = \frac{\log_2(\ell+1)}{\delta}$.
3) To prove this claim, we consider two length-n vectors $\boldsymbol{u} = (0,\ldots,0,u_1,\ldots,u_{n-\ell}) \in \Sigma_2^n$ and $\boldsymbol{v} = (0,\ldots,0,v_1,\ldots,v_{n-\ell}) \in \Sigma_2^n$ such that $\boldsymbol{u} \neq \boldsymbol{v}$. We observe that $TR_{\ell,\delta=1}(\boldsymbol{u}) \neq TR_{\ell,\delta=1}(\boldsymbol{v})$. Let $\mathcal{C}_{TR}(n,\ell,\delta)$ be a set of all vectors of length n that the first ℓ entries are zeros. So, $\mathcal{C}_{TR}(n,\ell,\delta)$ is a binary $(\ell,\delta=1)$-transverse-read code and $|\mathcal{C}_{TR}(n,\ell,\delta=1)| = 2^{n-\ell}$. Therefore, $A(n,\ell,\delta=1) \geq 2^{n-\ell}$ and $\mathcal{R}(\ell,\delta=1) = \limsup_{k\to\infty} \frac{n-\ell}{k\delta+\ell} = 1$. ∎

For all cases in Theorem 1, we can find the maximal asymptotic rate of (ℓ,δ)-transverse-read codes. In the rest of

the paper, we focus on the more challenging cases when $1 < \delta < \ell$. First, we establish the case where ℓ is a multiple of δ.

Theorem 2: Given two positive integers δ and ℓ such that ℓ is a multiple of δ, it holds that

$$\mathcal{R}(\ell, \delta) = \frac{\log_2(\delta+1)}{\delta}.$$

Before we prove Theorem 2, we show the following result.

Lemma 1: Given two positive integers δ and ℓ such that ℓ is a multiple of δ, it holds that

$$\mathcal{R}(\ell, \delta) \leq \frac{\log_2(\delta+1)}{\delta}. \tag{1}$$

Proof: Let $n_1 = n/\delta$ and $\ell_1 = \ell/\delta$ be two positive integers. Given a vector $\boldsymbol{x} = (x_1, \ldots, x_n) \in \Sigma_2^n$, let $f(\boldsymbol{x}) = (f_1, \ldots, f_{n_1}) \in \Sigma_{\delta+1}^{n_1}$ where $f_i = \boldsymbol{w}(\boldsymbol{x}_{[(i-1)\delta+1;\delta]}) \in \{0, \ldots, \delta\}$ for $1 \leq i \leq n_1$. We see that $TR_{\ell,\delta=2}(\boldsymbol{x}) = (w(f_{[1;\ell_1]}), w(f_{[2;\ell_1]}), \ldots, w(f_{[n_1-\ell_1+1;\ell_1]}) \in \Sigma_{(\ell+1)(\delta)}^{n_1-\ell_1+1}$. Let $\mathcal{C}_{TR}(n, \ell, \delta)$ be a binary (ℓ, δ)-transverse-read code, that is, for two different vectors $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}_{TR}(n, \ell, \delta)$, it holds that $TR_{\ell,\delta}(\boldsymbol{x}) \neq TR_{\ell,\delta}(\boldsymbol{y})$. Hence, $f(\boldsymbol{x}) \neq f(\boldsymbol{y})$. So, $|\mathcal{C}_{TR}(n, \ell, \delta)| \leq |\Sigma_{\delta+1}^{n_1}| = (\delta+1)^{n_1}$, for any (ℓ, δ)-transverse-read code $\mathcal{C}_{TR}(n, \ell, \delta)$. Therefore, $A(n, \ell, \delta) \leq (\delta+1)^{n/2}$, and thus $\mathcal{R}(\ell, \delta) \leq \frac{\log_2(\delta+1)}{\delta}$. ∎

We now construct a binary (ℓ, δ)-transverse-read code $\mathcal{C}_{TR}(n, \ell, \delta)$ as follows.

Construction 1: Let $\mathcal{F} = \{(f_1, f_2, \ldots, f_{n_1}) : f_i = 0 \text{ for all } i = 1, \ldots, \ell_1\} \subseteq \Sigma_{\delta+1}^{n_1}$ be a set of all $(\delta+1)$-ary vectors of length n_1 such that their first ℓ_1 entries are all zeros. We define the mapping $\phi : \mathcal{F} \to \Sigma_2^n$ such that, for each $\boldsymbol{f} = (f_1, \ldots, f_{n_1}) \in \mathcal{F}$, $\phi(\boldsymbol{f}) = \boldsymbol{x} = (x_1, \ldots, x_n) \in \Sigma_2^n$ such that $\boldsymbol{x}_{[(i-1)\delta+1;\delta]} = (0^{\delta-j}1^j)$ if $f_i = j$. Let $n = \delta n_1$, $\ell = \delta\ell_1$, and let $\mathcal{C}_{TR}(n, \ell, \delta) = \phi(\mathcal{F}) = \{\phi(\boldsymbol{f}) : \boldsymbol{f} \in \mathcal{F}\}$ be a set of all vectors $\boldsymbol{x}$ of length n such that there is $\boldsymbol{f} \in \mathcal{F}$ and $\boldsymbol{x} = \phi(\boldsymbol{f})$.

For each $\boldsymbol{x} \in \mathcal{C}_{TR}(n; \ell, \delta)$, we obtain $TR_{\ell,\delta}(\boldsymbol{x}) = TR_{\ell_1,1}(\boldsymbol{f})$ where $\phi(\boldsymbol{f}) = \boldsymbol{x}$. Hence, given two vectors $\boldsymbol{x} \neq \boldsymbol{y} \in \mathcal{C}_{TR}(n; \ell, \delta)$, $TR_{\ell,\delta}(\boldsymbol{x}) \neq TR_{\ell,\delta}(\boldsymbol{y})$. So, the code $\mathcal{C}_{TR}(n; \ell, \delta)$ from Construction 1 is a (ℓ, δ)-transverse-read code of length n. Moreover, $|\mathcal{F}| = (\delta+1)^{n_1-\ell_1}$ and thus $|\mathcal{C}_{TR}(n, \ell, \delta)| = |\mathcal{F}| = (\delta+1)^{n_1-\ell_1}$. Therefore, $A(n, \ell, \delta) \geq (\delta+1)^{n_1-\ell_1}$ and thus, for any even integer ℓ,

$$\mathcal{R}(\ell, \delta) \geq \frac{\log_2(\delta+1)}{\delta}. \tag{2}$$

From inequalities 1 and 2, we obtain $\mathcal{R}(\ell, \delta) = \frac{\log_2 \delta+1}{\delta}$. Hence, Theorem 2 is proven.

In particular, when $\delta = 2$, we obtain the following corollary.

Corollary 1: Let ℓ be an even number. Then, it holds that

$$\mathcal{R}(\ell, \delta = 2) = \frac{\log_2(3)}{2} \approx 0.7925.$$

Next, we continue to study the case where $\delta = 2$, ℓ is an odd integer and provide a construction of a (ℓ, δ)-transverse-read code as follows.

Construction 2: Given two odd integers n and ℓ, let $k = (n-\ell)/2$. We define the mapping

$$g : \Sigma_3^k \to \Sigma_2^n$$

as follows. For each $\boldsymbol{u} = (u_1, \ldots, u_k) \in \Sigma_3^k$, $g(\boldsymbol{u}) = \boldsymbol{c} = (c_1, \ldots, c_n) \in \Sigma_2^n$ such that $c_i = 0$ for all $i = 1, \ldots, \ell$ and for $1 \leq i \leq k$, $(c_{\ell+2i-1}, c_{\ell+2i}) = (0, 0)$ if $u_i = 0$, $(c_{\ell+2i-1}, c_{\ell+2i}) = (0, 1)$ if $u_i = 1$, and $(c_{\ell+2i-1}, c_{\ell+2i}) = (1, 1)$ if $u_i = 2$. Let $\mathcal{C}_{TR}(n, \ell, 2) = \{g(\boldsymbol{u}) : \boldsymbol{u} \in \Sigma_3^k\}$.

We now consider $\boldsymbol{u} \neq \boldsymbol{v} \in \Sigma_3^k$, then $g(\boldsymbol{u}) \neq g(\boldsymbol{v})$. Hence, $|\mathcal{C}_{TR}(n, \ell, \delta = 2)| = |\Sigma_3^k| = 3^k$. Moreover, if $g(\boldsymbol{u}) \neq g(\boldsymbol{v})$ then $TR_{\ell,2}(g(\boldsymbol{u})) \neq TR_{\ell,2}(g(\boldsymbol{v}))$. Thus, the code $\mathcal{C}_{TR}(n, \ell, 2)$ constructed above is an (ℓ, δ)-transverse-read code. Hence, $A(n, \ell, 2) \geq |\mathcal{C}_{TR}(n, \ell, 2)| = 3^k$ and thus, for any odd integer ℓ,

$$\mathcal{R}(\ell, 2) \geq \frac{\log_2 3}{2} \approx 0.7925. \tag{3}$$

From inequalities (2) and (3), we obtain the following result for any integer ℓ.

Lemma 2: For any integer $\ell \geq 2$, we obtain the following lower bound on the rate of the (ℓ, δ)-transverse-read code when $\delta = 2$,

$$\mathcal{R}(\ell, \delta = 2) \geq \frac{\log_2 3}{2} \approx 0.7925.$$

Now, we extend the result in Lemma 2 for arbitrary values of ℓ and δ such that $\ell > \delta$. Namely, we construct a binary (ℓ, δ)-transverse-read code as follows.

Construction 3: Given four integers k, ℓ, δ, n such that $k\delta = n - \ell$. We define the mapping ψ as follows,

$$\psi : \Sigma_{\delta+1}^k \to \Sigma_2^n,$$

for each $\boldsymbol{u} = (u_1, \ldots, u_k) \in \Sigma_{\delta+1}^k$, then $\psi(\boldsymbol{u}) = \boldsymbol{c} = (c_1, \ldots, c_n) \in \Sigma_2^n$ such that $c_i = 0$ for $1 \leq i \leq \ell$ and for $1 \leq i \leq k$, if $u_i = j$ then $\boldsymbol{c}_{[\ell+\delta(i-1)+1;\delta-j]}$ is all-zero vector of length $\delta - j$ and $\boldsymbol{c}_{[\ell+\delta(i)-j+1;j]}$ is all-one vector of length j. Let $\mathcal{C}_{TR}(n, \ell, \delta) = \{\psi(\boldsymbol{u}) : \boldsymbol{u} \in \Sigma_{\delta+1}^k\}$.

We state the result formally as follows.

Theorem 3: The code $\mathcal{C}_{TR}(n, \ell, \delta)$ constructed in Construction 3 is a binary (ℓ, δ)-transverse-read code of length n and thus $A(n, \ell, \delta) \geq (\delta+1)^k$.

Proof: We consider any two vectors $\boldsymbol{u}, \boldsymbol{v} \in \Sigma_{\delta+1}^k$ such that $\boldsymbol{u} \neq \boldsymbol{v}$. Let i be the smallest index such that $u_i \neq v_i$. Hence, $\psi(\boldsymbol{u})_{[\ell+\delta(i-1)+1;\delta]} \neq \psi(\boldsymbol{v})_{[\ell+\delta(i-1)+1;\delta]}$. Thus $\psi(\boldsymbol{u}) \neq \psi(\boldsymbol{v})$. We now consider two vectors $TR_{\ell,\delta}(\psi(\boldsymbol{u}))$ and $TR_{\ell,\delta}(\psi(\boldsymbol{v}))$ and see that $TR_{\ell,\delta}(\psi(\boldsymbol{u}))_{i+1} = \boldsymbol{w}(\psi(\boldsymbol{u})_{[\delta i;\ell]}) \neq \boldsymbol{w}(\psi(\boldsymbol{v})_{[\delta i;\ell]}) = TR_{\ell,\delta}(\psi(\boldsymbol{v}))_{i+1}$. Hence, $TR_{\ell,\delta}(\psi(\boldsymbol{u})) \neq TR_{\ell,\delta}(\psi(\boldsymbol{v}))$.

Therefore, we conclude that the code $\mathcal{C}_{TR}(n, \ell, \delta)$ constructed in Construction 3 is a binary (ℓ, δ)-transverse-read code since for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}_{TR}(n, \ell, \delta)$, we get $TR_{\ell,\delta}(\boldsymbol{x}) \neq TR_{\ell,\delta}(\boldsymbol{y})$. Moreover, $|\mathcal{C}_{TR}(n, \ell, \delta)| = |\Sigma_{\delta+1}^k| = (\delta+1)^k$ since ψ is an injection. Hence, $A(n, \ell, \delta) \geq |\mathcal{C}_{TR}(n, \ell, \delta)| = (\delta+1)^k$ and the theorem is proved. ∎

From Theorem 3, we obtain the following result on the lower bound on the maximal asymptotic rate of (ℓ, δ)-transverse-read codes.

Corollary 2: If ℓ and δ are two integers such that $\ell > \delta > 1$ then

$$\mathcal{R}(\ell, \delta) \geq \frac{\log_2(\delta+1)}{\delta}.$$

Furthermore, from Construction 3, there is a binary (ℓ, δ)-transverse-read code with an efficient encoding algorithm.

In the rest of this section, we present a technique to find the asymptotic rate of (ℓ, δ)-transverse-read codes exactly,

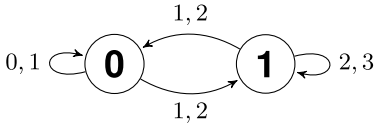

Fig. 2. Non-deterministic finite state transition diagram $\ell = 3, \delta = 2$.

given $\ell > \delta > 1$. To find the asymptotic rate of the above codes, we first prove that these codes are equivalent to a class of constrained codes avoiding some specific patterns and a class of regular languages. Then, we can use some known techniques in constrained codes and regular languages using finite state machines to compute the maximal asymptotic rates.

We first consider the case $\ell = 3$ and $\delta = 2$. We recall that $A(n, \ell, \delta) = |TR(k, \ell, \delta)|$ where $TR(k, \ell, \delta)$ is the set of all valid (ℓ, δ)-transverse-read vectors of length $k + 1$. Let $\boldsymbol{u} = (u_1, \ldots, u_{k+1}) \in TR(k, \ell = 3, \delta = 2) \subseteq \Sigma_4^{k+1}$ be a valid $(\ell = 3, \delta = 2)$-transverse-read vector. So, there exists a vector $\boldsymbol{x} \in \Sigma_2^n$ such that $TR_{\ell,\delta}(\boldsymbol{x}) = \boldsymbol{u}$. Then, for each $1 \leq i \leq k+1$, $u_i = \boldsymbol{w}(x_{2i-1}, x_{2i}, x_{2i+1}) = x_{2i-1} + x_{2i} + x_{2i+1} \in \{0, 1, 2, 3\}$. We observe that $TR(k, \ell = 3, \delta = 2)$ is a regular language. It is recognized by a non-deterministic state machine as in Figure 2. The machine is a graph of two nodes, 0 and 1. For each u_i, there is a corresponding tuple $(x_{2i-1}, x_{2i}, x_{2i+1})$ such that $u_i = \boldsymbol{w}(x_{2i-1}, x_{2i}, x_{2i+1})$. The node j corresponds to the state $x_{2i+1} = j$ for $j = 0, 1$. We start with $u_i = 0$, that is, $x_{2i-1} + x_{2i} + x_{2i+1} = 0$, and thus $x_{2i-1} = x_{2i} = x_{2i+1} = 0$. The machine is at state 0. If $u_{i+1} = 0$ then the machine remains at the same state 0 and there is an edge labelled 0 from node 0 to itself. If $u_{i+1} = 1$, that is, $x_{2i+1} + x_{2i+2} + x_{2i+3} = 1$, then x_{2i+3} can be either 0 or 1. Hence, there is an edge labelled 0 from node 0 to itself and there is an edge labelled 1 from node 0 to node 1. If $u_{i+1} = 2$, that is, $x_{2i+1} + x_{2i+2} + x_{2i+3} = 2$, then since $x_{2i+1} = 0$, it holds that $x_{2i+2} = x_{2i+3} = 1$. Hence, the state of the machine is 1 and there is an edge labelled 2 from node 0 to node 1. Once the machine is at state 1, that is, $x_{2i+3} = 1$, we consider the next symbol $u_{i+2} = x_{2i+3} + x_{2i+4} + x_{2i+5}$. Since $x_{2i+3} = 1$ we have $u_{i+2} \geq 1$. If $u_{i+2} = 1$ then $x_{2i+4} = x_{2i+5} = 0$ and the machine will be at state 0. So, there is an edge labelled 1 from node 1 to node 0. If $u_{i+2} = 2$ then $x_{2i+4} + x_{2i+5} = 1$. Hence, x_{2i+5} can be 0 or 1, that is, the machine will be at state 0 or state 1. So, there is an edge labelled 2 from node 1 to node 0 and a self loop labelled 2 from node 1 to itself. If $u_{i+2} = 3$ then $x_{2i+4} = x_{2i+5} = 1$ and the machine will be at state 1. So, there is a loop labelled 3 from node 1 to itself. Therefore, the state machine in Figure 2 is a non-deterministic finite state machine. It is well known that for any regular language which can be recognized by a non-deterministic finite state machine, it can be expressed by a deterministic state machine. For example, in the case $\ell = 3$ and $\delta = 2$, the regular language $TR(n, \ell, \delta)$ is recognized by a deterministic finite state machine as in Figure 3. In this diagram, we have a new node "*" which is the state that x_{2i-1} can be 0 or 1. The adjacency matrix of this deterministic diagram is:

$$A_G = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

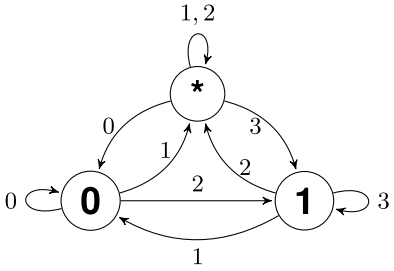

Fig. 3. Deterministic finite state transition diagram $\ell = 3, \delta = 2$.

Next, using the well-known Perron-Frobenius theory [17], we can exactly calculate the maximal asymptotic rate of $(\ell = 3, \delta = 2)$-transverse-read codes to be $(\log_2 \lambda)/2 = 0.8858$ where $\lambda = 3.4142$ is the largest real eigenvalue of A_G.

Additionally, the code $TR(k, \ell, \delta)$, which can be expressed by the state machine in Figure 3, is also a constrained system. We now state the following result.

Theorem 4: We consider the following set

$$\mathcal{F} = \big\{(3, (1, 2)^i, 0), (3, (1, 2)^i, 1, 3), (0, (2, 1)^i, 3), (0, (2, 1)^i, 2, 0)\big\}.$$

A valid $(\ell = 3, \delta = 2)$-transverse-read code is a constrained code avoiding all patterns in $\mathcal{F}$.

Theorem 4 can be proven by showing that both above codes have the same finite state transition diagram as in Figure 3.

Next, we aim to extend the above results for other values of $\ell > \delta > 1$. We now build a non-deterministic finite state machine $G_{\ell,\delta} = (V^{\ell,\delta}, E^{\ell,\delta})$ where $V^{\ell,\delta}$ is the set of all vertices and $E^{\ell,\delta}$ is the set of all edges. The graph $G_{\ell,\delta}$ has $|V^{\ell,\delta}| = 2^{\ell-\delta}$ vertices and each vertex represents a binary word of length $s = \ell - \delta$. If $\ell \geq 2s$, there are directed edges from the vertex $\boldsymbol{x} = (x_1, \ldots, x_s)$ to the vertex $\boldsymbol{y} = (y_1, \ldots, y_s)$ with labels $\{a, a + 1, \ldots, a + \ell - 2s\}$ where $a = \sum_{i=1}^{s}(x_i + y_i)$ for any pair of vertices. If $\ell < 2s$, there is a directed edge from the vertex $\boldsymbol{x} = (x_1, \ldots, x_s)$ to the vertex $\boldsymbol{y} = (y_1, \ldots, y_s)$ if and only if $\boldsymbol{x}_{[\delta+1;s-\delta]} = \boldsymbol{y}_{[1;s-\delta]}$. Such an edge is labelled by b where $b = \sum_{i=1}^{s} x_i + \sum_{j=s-\delta+1}^{s} y_j$. So, in both cases, we can build a non-deterministic finite state machine of the transverse-read channel.

For example, when $\ell = 5$ and $\delta = 2$, we can build a non-deterministic finite state machine of $(\ell = 5, \delta = 2)$ as in Figure 4. In the graph, there are 8 nodes, each of which is a state of the machine. We start with $u_1 = 0$ and the machine is at state $(0, 0, 0)$. If $u_2 = 0$, then the machine stays at state $(0, 0, 0)$ and there is a loop with label 0 from $(0, 0, 0)$ into itself. If $u_2 = 1$, the machine can move to state $(0, 1, 0)$ or $(0, 0, 1)$. There is an edge from state $(0, 0, 0)$ to state $(0, 1, 0)$ with label 1 and an edge from state $(0, 0, 0)$ to state $(0, 0, 1)$ with label 1. If $u_2 = 2$, the machine can move to state $(0, 1, 1)$. There is an edge from state $(0, 0, 0)$ to state $(0, 1, 1)$ with label 2. We can consider other states and build the non-deterministic state machine of 8 nodes and multiple edges as in Figure 4. For simplicity in the illustration, in Figure 4, we only label all edges that go out from nodes (0,0,0) and (1,1,1).

Once we have a non-deterministic finite state machine, it is a folklore that we can convert from a non-deterministic finite

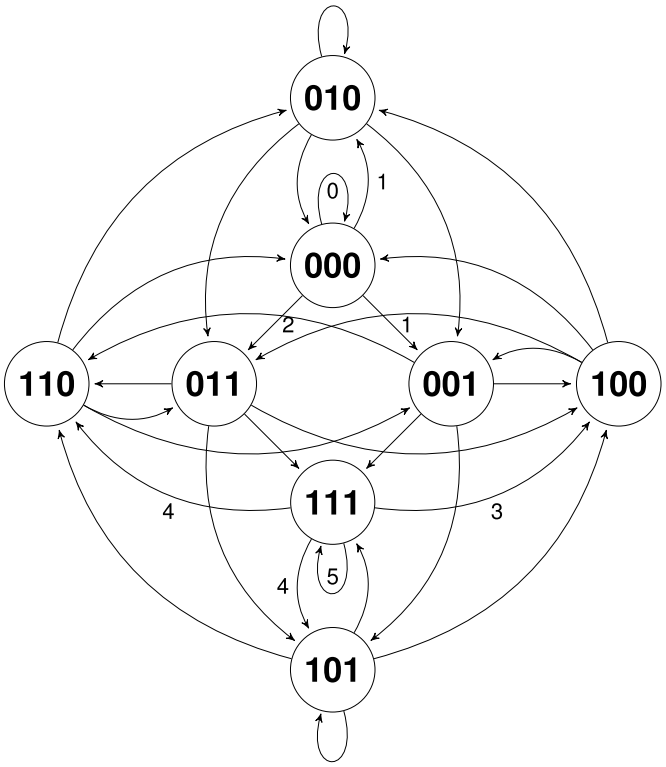

Fig. 4. Non-deterministic finite state transition diagram $\ell = 5, \delta = 2$.

TABLE I
THE MAXIMAL ASYMPTOTIC RATES OF (ℓ, δ)-TRANSVERSE-READ CODES

	$\ell=3$	$\ell=4$	$\ell=5$	$\ell=6$	$\ell=7$	$\ell=8$
$\delta=2$	0.8857	0.7925	0.9258	0.7925	0.9361	0.7925
$\delta=3$	0.6667	0.7298	0.7475	0.6667	0.7702	0.7744
$\delta=4$	0.5	0.5805	0.6243	0.6462	0.6462	0.5805

state machine to a deterministic finite state machine. Hence, we can compute the capacity of the constrained channel represented by this machine, and thus the maximal asymptotic rate of transverse-read codes. Several numerical results were computed and are tabulated in Table I.

From the results in Table I, we see that $0.936 = TR_{\ell=7,\delta=2} > TR_{\ell=5,\delta=2} > TR_{\ell=3,\delta=2} > TR_{\ell=2,\delta=2} = 0.795$. So, using our scheme, even if we reduce the number of shift-operations to 50%, we can still achieve the information rate 93.6% when $\ell = 5$. We observe in Table I that the asymptotic rates of $(\ell, \delta = 2)$ are increasing when ℓ is odd and increasing. For any value of ℓ, the rate satisfies $TR_{\ell,\delta=2} \leq 1$. We are interested in finding the maximum asymptotic rates $TR_{\ell,\delta=2}$ for all odd numbers ℓ. We obtained a trivial lower bound and an upper bound, $0.936 \leq \max_\ell TR_{\ell,\delta=2} \leq 1$. We now state the following conjecture and will study further in future work.

Conjecture 1: Let $\delta = 2$ and ℓ_1, ℓ_2 be two odd number such that $\ell_1 > \ell_2$.

1) The following inequality holds,

$$TR_{\ell_1,\delta=2} > TR_{\ell_2,\delta=2}.$$

2) For any $\epsilon > 0$, there exists an odd number ℓ such that $TR_{\ell,\delta=2} > 1 - \epsilon$.

We remark that the algorithm converts a non-deterministic state machine with v nodes to a deterministic state machine with 2^v nodes. Hence, it is not efficient to compute the capacity of the transverse-read channel when ℓ is large. However, when ℓ is small, it is fast to build a deterministic finite state machine of (ℓ, δ)-transverse-read channel. Using the state machine, it is possible to construct an (ℓ, δ)-transverse-read code achieving the capacity with efficient encoding/decoding algorithms. We may use the well-known finite state splitting algorithms [17] or some rank/unrank algorithms. In the following section, we will study the ability of correcting shift-errors and substitution-errors of these codes.

IV. TRANSVERSE-READ CODES CORRECTING ERRORS

Given ℓ, δ, in this work, we consider the channel that only accepts (ℓ, δ)-transverse-read vectors. In the previous section, we investigated the capacity of the channel which is the maximal asymptotic rate of (ℓ, δ)-transverse-read code and constructed some codes with high rate. In this section, we study and construct some error-correcting codes for the (ℓ, δ)-transverse-read channel. We consider two types of errors in the channel, namely, shift-errors and substitution errors. We construct codes correcting shift-errors in Section IV-A and codes correcting substitution errors in Section IV-B.

A. Limited-Shift-Errors

In this subsection, we start with the (ℓ, δ)-transverse-read channel when $\ell = 2$ and $\delta = 1$. We consider a domain wall memory of n domains and two access ports which are within two locations. These two access ports can transverse-read to detect the number of ones in a segment of length ℓ. Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \Sigma_2^n$ be the stored vector in the domain wall memory. Using the transverse-read technique in each segment of length $\ell = 2$ and shift-operation one position in each step, we obtain the output vector $TR_{2,1}(\boldsymbol{x}) = (x_1 + x_2, x_2 + x_3, \ldots, x_{n-1} + x_n)$ which is the transverse-read vector of $\boldsymbol{x}$. In this scheme, a shift-operation might not work perfectly and errors may occur. For example, an under-shift error occurs at the first position, the first entry in the output is repeated, and thus, we obtain the output $(x_1 + x_2, x_1 + x_2, x_2 + x_3, \ldots, x_{n-1} + x_n)$. This kind of error can be modelled as a sticky-insertion. Besides that, if there is an over-shift error, one entry in the output is deleted. For example, if an over-shift occurs at the second position, we obtain the output $(x_1 + x_2, x_3 + x_4, \ldots, x_{n-1} + x_n)$. So, an over-shift error can be modelled as a deletion in the transverse-read vector. Our goal in this subsection is to correct these errors.

Normally, to correct these errors, one may need to use some classical codes correcting deletions and sticky-insertions. We note that, there are several asymptotically optimal binary codes that correct t sticky-insertion errors with only $t \log n$ redundancy bits. However, it is much more complicated to correct $t > 1$ deletions. In this work, we show that transverse-read codes have some special properties that are useful for correcting these shift-errors. Let us consider a vector $\boldsymbol{x} = (x_1, x_2, x_3, x_4, x_5) = (0, 0, 1, 1, 0)$ and its transverse-read vector $TR_{2,1}(\boldsymbol{x}) = (x_1 + x_2, x_2 + x_3, x_3 + x_4, x_4 + x_5) =$

$(0, 1, 2, 1)$. Once an over-shift occurs, a symbol in $TR_{2,1}(\boldsymbol{x})$ is deleted and we may obtain an invalid word. For example, an over-shift occurs in the second position and the symbol $x_2 + x_3 = 1$ is deleted. Hence, we obtain the vector $(0, 2, 1)$. However, the word $(0, 2, 1)$ is not a valid (2,1)-transverse-read vector since 0 can not be followed by a 2. Hence, we can detect and locate a single deletion in this case. Based on this simple observation, we can show a strong connection between a code correcting sticky-deletions and a code correcting t deletions in our channel where there are no consecutive deletions. We note that, a sticky-deletion is an error that a bit in a run is deleted but the whole run cannot be deleted. Codes correcting sticky-deletions have attracted a lot of attention recently [26], [27] and there are known constructions of codes correcting t sticky-deletions with at most $t\log(n)+o(\log n)$ bits of redundancy. Hence, we are able to design a code correcting t deletions, where there are no consecutive deletions, with at most $t\log(n) + o(\log n)$ bits of redundancy. For simplicity, we first present the result for $\ell = 2, \delta = 1$, and $t = 1$. If there is no error, then we showed in part 4 of Theorem 1 that the maximal size of the (2,1)-transverse-read code is 2^{n-1}. In the following result, we will show that the maximal size of the (2,1)-transverse-read code correcting is $O(2^n/n^t)$.

Theorem 5: Let $\mathcal{C}_1 \subset \Sigma_2^n$ be a binary code correcting a single sticky-deletion. Then, the code $\mathcal{C}_1$ can correct a single deletion in the (2,1)-transverse-read code. That is, if a deletion occurs in a transverse-read vector $TR_{2,1}(\boldsymbol{c})$ where $\boldsymbol{c} \in \mathcal{C}_1$, we can recover the original word $\boldsymbol{c}$.

Proof: Let $\boldsymbol{c} = (c_1, c_2, \ldots, c_n) \in \mathcal{C}_1$ be a stored word. Then, $\boldsymbol{u} = TR_{2,1}(\boldsymbol{c}) = (u_1, u_2, \ldots, u_{n-1})$, where $u_i = c_i+c_{i+1}$, is the (2,1)-transverse-read vector of $\boldsymbol{c}$. We observe that in a valid (2,1)-transverse-read vector, the run of 1's has odd length if it is bounded by two different symbols, that is $(0, 1, \ldots, 1, 2)$ or $(2, 1, \ldots, 1, 0)$, and the run of 1's has even length if it is bounded by the same symbol, that is $(0, 1, \ldots, 1, 0)$ or $(2, 1, \ldots, 1, 2)$. Hence, if the symbol 1 is deleted in the valid (2,1)-transverse-read vector, we can detect and locate the error and thus correct it. We now consider the case where the symbol 0 or 2 was deleted. Note that if $u_i = 0$ then $c_i = c_{i+1} = 0$ and if $u_i = 2$ then $c_i = c_{i+1} = 1$. Hence, if the symbol 0 or 2 is deleted in the transverse-read vector $\boldsymbol{u}$, a sticky-deletion occurs in the stored word $\boldsymbol{c}$. Since $\boldsymbol{c} \in \mathcal{C}_1$ which can correct a single sticky-deletion, we can correct the error and recover the original word $\boldsymbol{c}$. Hence, we can recover the stored word $\boldsymbol{c}$. Therefore, code $\mathcal{C}_1$ can correct a single deletion in the (2,1)-transverse-read code. ∎

It is known that correcting a sticky-deletion is easier than correcting a deletion. Hence, the transverse-read code is helpful in correcting a deletion (over shift error). It is interesting that we can also extend the result for codes correcting multiple deletions. We present the result on codes correcting multiple deletions where there is at most a single deletion in each run as follows.

Theorem 6: Given $t > 1$, let $\mathcal{C}_t$ be a code of length n correcting t sticky-deletions. If there are at most t deletions in a (2, 1)-transverse-read vector $TR_{2,1}(\boldsymbol{c})$ where $\boldsymbol{c} \in \mathcal{C}_t$ such that there is at most a single deletion in each run of same symbols (length of each run can be one), then we can recover the original word $\boldsymbol{c}$.

Proof: To prove the theorem, we just need to follow iteratively the argument in the proof of Theorem 5. Let $\boldsymbol{c} = (c_1, c_2, \ldots, c_n) \in \mathcal{C}_1$ be a stored word. Then, $\boldsymbol{u} = TR_{2,1}(\boldsymbol{c}) = (u_1, u_2, \ldots, u_{n-1})$, where $u_i = c_i+c_{i+1}$, is the (2,1)-transverse-read vector of $\boldsymbol{c}$. Let $\boldsymbol{v} = (v_1, v_2, \ldots, v_{n-t-1})$ be the output that we receive after t deletions occur. Since $\boldsymbol{u}$ is a valid (2, 1)-transverse-read vector, if 01^r0 or 21^r2 is a substring of $\boldsymbol{u}$, then r is an even number. If there is a deletion in a run of symbols 1's in a pattern 01^r0 or 21^r2, we can detect this error by checking the parity of r. To correct this error, we just need to add the symbol 1 in this position of the vector $\boldsymbol{v}$. Furthermore, if 01^s2 or 21^s0 is a substring of $\boldsymbol{u}$ then s is an odd number. Similarly, we can detect and correct a deletion in each run of 1's. After we correct all deletions in each run of 1's in the output, we obtain a valid (2, 1)-transverse-read vector of length m, $\boldsymbol{v}'$. So, there is a vector $\boldsymbol{c}' \in \Sigma_2^m$ such that $TR_{2,1}(\boldsymbol{c}') = \boldsymbol{v}'$. If $m < n$, we can obtain $\boldsymbol{v}'$ from $\boldsymbol{v}$ after deleting $(n-m)$ symbols of 0's or 2's. That is, we can obtain $\boldsymbol{c}'$ from $\boldsymbol{c}$ after $(n-m)$ sticky-deletions. Since the code $\mathcal{C}_t$ can correct at most t deletions and $n - m < t$, we can correct all these errors to recover $\boldsymbol{c}$. This completes the theorem's proof. ∎

From the above proof of Theorem 6, we can obtain a simple decoding algorithm to correct at most t deletions in the transverse-read code. So far, we showed that our scheme of using (ℓ, δ)-transverse-read code is helpful to correct shift-errors for $\ell = 2$ and $\delta = 1$. The main idea is to use codes correcting sticky-deletion to correct deletions, using some special properties of (2,1)-transverse-read codes. This idea is presented in [18] for codes correcting deletions in the symbol-pair read channel. We note that the best known results on codes correcting t deletions require at least $4t\log n+o(t\log n)$ bits of redundancy [29] while it is possible to correct t sticky-deletions using only $t\log n+o(\log n)$ bits of redundancy, given a constant t. Hence, in our scheme for $\ell = 2$ and $\delta = 1$, it is easier to correct shift-errors.

B. Limited Magnitude Errors

In the previous subsection, we studied transverse-read codes correcting shift-errors in domain wall memories. In this section, we focus on substitution errors. A substitution error occurs when there is a misread in the transverse-read and a symbol is read incorrectly. Let $\boldsymbol{x} = (x_1, \ldots, x_n) \in \Sigma_2^n$ and $TR_{3,1}(\boldsymbol{x}) = \boldsymbol{y} = (y_1, \ldots, y_{n-2})$ where $y_i = \boldsymbol{w}(\boldsymbol{x}_{[i;3]})$. If there is an error in the transverse-read vector at the i-th position, then we obtain $y_i' \neq y_i$. In this work, we assume that the magnitude of an error caused by transverse-read is limited by a small number b, that is $|y_i' - y_i| \leq b$. Such an error is called a b-limited magnitude error. Given two vectors $\boldsymbol{u} = (u_1, \ldots, u_n)$ and $\boldsymbol{v} = (v_1, \ldots, v_n)$, we define the *b-limited distance* between $\boldsymbol{u}$ and $\boldsymbol{v}$, denoted $d_b(\boldsymbol{u}, \boldsymbol{v})$, as follows. If there is an index i such that $|u_i - v_i| > b$ then $d_b(\boldsymbol{u}, \boldsymbol{v}) = \infty$. Otherwise, $d_b(\boldsymbol{u}, \boldsymbol{v}) = |\{i : u_i \neq v_i\}|$. That is, the b-limited distance between the two vectors of fixed length is either infinity or the Hamming distance between two vectors. For example,

$\boldsymbol{x} = (0, 0, 1, 1, 1, 0, 1, 1)$ and $TR_{3,1}(\boldsymbol{x}) = \boldsymbol{y} = (1, 2, 3, 2, 2, 2)$. If the output is $\boldsymbol{y}' = (1, 2, 2, 2, 1, 2)$, we observe that the 1-limited distance between $\boldsymbol{y}$ and $\boldsymbol{y}'$ is $d_b(\boldsymbol{y}, \boldsymbol{y}') = 2$. An (ℓ, δ)-transverse-read code $\mathcal{C}_{TR}(n, \ell, \delta)$ is said to be able to correct t b-limited magnitude errors if $\boldsymbol{x} \in \mathcal{C}_{TR}(n, \ell, \delta) \subset \Sigma_2^n$ is a stored message and $\boldsymbol{y}' \in \Sigma_{\ell+1}^{(n-\ell)/\delta+1}$ is the output in the transverse-read channel such that $d_b(\boldsymbol{y}', TR_{\ell,\delta}(\boldsymbol{x})) \leq t$, then it is possible to recover the vector $\boldsymbol{x}$ from the output $\boldsymbol{y}'$. That is, if there are at most t b-limited magnitude errors in the transverse-read channel, we can recover the original vector $\boldsymbol{x}$ in the code $\mathcal{C}_{TR}(n, \ell, \delta)$.

In this subsection, we focus on the case $b = 1$ and design a code to correct 1-limited magnitude errors in this channel. Before we present the construction of our code, we present the following codes.

Construction 4: For $0 < P$, let the code $SVT(n, P)$ be

$$SVT(n, P) = \left\{ \boldsymbol{c} \in \Sigma_q^n : \sum_{i=1}^{n} i c_i \equiv 0 \mod (P+1); \sum_{i=1}^{n} c_i \equiv 0 \mod 3 \right\}.$$

We note that the above code $SVT(n, P)$ is similar to the well-known shifted VT code [19]. The authors in [19] showed that shifted-VT code can correct a deletion, given knowledge of the location of the error within P positions. We now present a similar result for a single 1-limited magnitude error.

Proposition 1: Given a codeword $\boldsymbol{c} \in SVT(n, P)$, and a vector $\boldsymbol{c}'$ such that the 1-limited distance between two vectors is $d_1(\boldsymbol{c}, \boldsymbol{c}') = 1$. If i is an index where $c_i \neq c_i'$ and we know that $i \in \{a+1, \ldots, a+P\}$, then we can recover $\boldsymbol{c}$ from $\boldsymbol{c}'$. Or in other words, the above code $SVT(n, P)$ can correct a single 1-limited-magnitude error given knowledge of the location of the error within P consecutive positions.

Proof: We consider two vectors $\boldsymbol{c}$ and $\boldsymbol{c}'$. Since there is only one index i such that $c_i \neq c_i'$, we obtain $\sum_{j=0}^{n-1}(j+1)c_j - \sum_{j=0}^{n-1}(j+1)c_j' = (i+1)(c_i - c_i')$ and $\sum_{j=1}^{n} c_j - \sum_{j=1}^{n} c_j' = c_i - c_i'$. We note that $|c_i - c_i'| = 1$, that is, $c_i - c_i'$ is either 1 or –1. Furthermore, $\sum_{j=0}^{n-1}(j+1)c_j - \sum_{j=0}^{n-1}(j+1)c_j'$ is either $i+1$ or $-i-1$. Since $\sum_{i=1}^{n} c_i \equiv 0 \mod 3$, we can determine if $c_i - c_i'$ is 1 or –1. And thus, we also can determine if $\sum_{j=0}^{n-1}(j+1)c_j - \sum_{j=0}^{n-1}(j+1)c_j'$ is $i+1$ or $-i-1$. Hence, we can find the exact value of the index i. From the corrupted vector $\boldsymbol{c}'$, we can recover the vector $\boldsymbol{c} \in SVT(n, P)$. ∎

Next, we present the constrained code for limited length of period sub-vector. Let p and m be two positive integers where $p \leq m$. Then, a length-m vector $\boldsymbol{v} \in \Sigma_2^m$ which satisfies $v_i = v_{i+p}$ for all $1 \leq i \leq m-p$ is said to have *period* p. For a vector $\boldsymbol{u} \in \Sigma_2^n$, we denote by $L(\boldsymbol{u}, p)$ the length of its longest subvector which has period p. By definition, $L(\boldsymbol{u}, p) \geq p$ and for $p = 1$, $L(\boldsymbol{u}, 1)$ is the length of the longest run in $\boldsymbol{u}$.

Example 2: Let $\boldsymbol{u} = (u_1, \ldots, u_9) = (0, 0, 1, 1, 0, 1, 0, 1, 1) \in \Sigma_2^9$ be a word of length 9. Since the longest run in $\boldsymbol{u}$ is of length two, we have $L(\boldsymbol{u}, 1) = 2$. The subvector $\boldsymbol{u}[4, 8] = (1, 0, 1, 0, 1)$ of $\boldsymbol{u}$ has period 2 since $u_4 = u_6 = u_8 = 1$ and $u_5 = u_7 = 0$. This is the longest subvector of $\boldsymbol{u}$ of period 2, and hence $L(\boldsymbol{u}, 2) = 5$.

Construction 5 [7]: Let $\mathcal{C}_{pe}(n, p, t)$ be a code of length n such that the length of the longest sub-vector which has period p of every codeword $\boldsymbol{c} \in \mathcal{C}_{pe}(n, p, t)$ is at most t. That is, $\mathcal{C}_{pe}(n, p, t) = \{\boldsymbol{c} \in \Sigma_2^n : L(\boldsymbol{c}, p) \leq t\}$.

The code $\mathcal{C}_{pe}(n, p, t)$ was well-studied in [7]. The authors in [7] showed that it is possible to construct the code $\mathcal{C}_{pe}(n, p, \lceil \log n \rceil)$ with only a single bit of redundancy. Now, we are ready to present a code construction that can correct a single 1-limited magnitude error for the (2,1)-transverse-read channel.

Construction 6: Let the code $\mathcal{C}_{2,1}(n, 1)$ be

$$\mathcal{C}_{2,1}(n, 1) = \left\{ \boldsymbol{c} \in \Sigma_2^n : \boldsymbol{c} \in \mathcal{C}_{pe}(n, 2, \log n) \text{ and } TR_{2,1}(\boldsymbol{c}) \in SVT(n-1, P = 3\log n) \right\}.$$

Theorem 7: The above code $\mathcal{C}_{2,1}(n, 1)$ can correct a single 1-limited magnitude error for the (2, 1)-transverse-read channel.

Proof: Given a vector $\boldsymbol{c} \in \mathcal{C}_{2,1}(n, 1)$, we observe the length of each run of symbol 1's in the vector $TR_{2,1}(\boldsymbol{c})$. A pattern $(0, 1^k, 0)$ and a pattern $(2, 1^k, 2)$ are valid if and only if k is even. A pattern $(0, 1^k, 2)$ and a pattern $(2, 1^k, 0)$ are valid if and only if k is odd. We call this the run-length rule. If a single 1-limited-magnitude error occurs at the index i of $TR_{2,1}(\boldsymbol{c})$, one or two consecutive runs of symbols 1's will violate the run-length rule and thus the output will be an invalid (2,1)-transverse-read vector. The location of the error is within the segment of one or two consecutive runs of symbols 1's. Since $\boldsymbol{c} \in \mathcal{C}_{pe}(n, 2, \log n)$, the longest sub-vector with period two has length at most $\log n$. Thus, the longest run of 1's in $TR_{2,1}(\boldsymbol{c})$ has length at most $\log n - 1$. So, we can locate the error within the segment of length $P = 3\log n$. Then, using the decoder of the code $SVT(n, P)$ in Construction 4, we can correct the 1-limited magnitude error. This concludes the proof. ∎

We note that the above construction of the code $\mathcal{C}_{2,1}(n, 1)$ is a combination of Construction 4 and Construction 5. Since we just need a single bit of redundancy to construct $\mathcal{C}(n, 2, \log n)$ in Construction 5 and at most $\log\log n + c$ bits of redundancy to construct $SVT(n-1, P = 3\log n)$ in Construction 4, we need at most $\log\log n + c + 1$ bits of redundancy to construct $\mathcal{C}_{2,1}(n, 1)$ in Construction 6 for some constant c. The next theorem summarizes this result.

Theorem 8: There is a (2,1)-transverse-read code correcting a single 1-limited magnitude error with at most $\log\log n + c$ bits of redundancy for some constant c.

Lastly, we note that to correct a single substitution of limited magnitude in the classical channel, it is required to use at least $\log n$ bits of redundancy. The remarkable result in Theorem 8 is that we only need $\log\log n + c$ bits of redundancy to correct one error in this channel.

V. Conclusion and Discussion

In this work, we proposed a new scheme of reading information in domain wall memories to reduce the number of shift-operations while still achieving high information rates. We introduce a new family of codes, called (ℓ, δ)-transverse-read codes, and study their properties, maximal asymptotic rates, and constructions. Furthermore, we show that our

scheme of using transverse-read codes is helpful to correct shift-errors and substitution errors in domain wall memories. Lastly, we design several codes which are able to correct multiple over-shift errors and a code correcting a single limited magnitude error.

REFERENCES

[1] Y. M. Chee, A. Vardy, V. K. Vu, and E. Yaakobi, "Coding for transverse-read in domain wall memories," in *Proc. IEEE Int. Symp. Inf. Theory*, 2021, pp. 2924–2929.

[2] S. S. Parkin, M. Hayashi, and L. Thomas, "Magnetic domain-wall racetrack memory," *Science*, vol. 320, no. 5873, pp. 190–194, 2008.

[3] Z. Sun, W. Wu, and H. Li, "Cross-layer racetrack memory design for ultra high density and low power consumption," in *Proc. Des. Autom. Conf. (DAC)*, 2013, pp. 1–6.

[4] S. Parkin and S. H. Yang, "Memory on the racetrack," *Nature Nanotechnol.*, vol. 10, no. 3, pp. 195–198, 2015.

[5] R. Blasing et al., "Magnetic racetrack memory: From physics to the cusp of applications within a decade," *Proc. IEEE*, vol. 18, no. 10, pp. 1303–1321, Aug. 2020.

[6] C. Zhang et al., "Hi-fi playback: Tolerating position errors in shift operations of racetrack memory," in *Proc. ACM/IEEE 42nd Annu. Int. Symp. Comput. Architect. (ISCA)*, 2015, pp. 694–706.

[7] Y. M. Chee, H. M. Kiah, A. Vardy, V. K. Vu, and E. Yaakobi, "Coding for racetrack memories," *IEEE Trans. Inf. Theory*, vol. 64, no. 11, pp. 7094–7112, Nov. 2018.

[8] Y. M. Chee, H. M. Kiah, A. Vardy, K. Van Vu, and E. Yaakobi, "Codes correcting limited-shift errors in racetrack memories," in *Proc. IEEE Int. Symp. Inf. Theory*, 2018, pp. 96–100.

[9] G. Mappouras, A. Vahid, R. Calderbank, and D. J. Sorin, "GreenFlag: Protecting 3D-racetrack memory from shift errors," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, 2019, pp. 1–12.

[10] S. Archer, G. Mappouras, R. Calderbank, and D. Sorin, "Foosball coding: Correcting shift errors and bit flip errors in 3D racetrack memory," in *Proc. 50th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, 2020, pp. 331–342.

[11] E. Atoofian and A. Saghir, "Shift-aware racetrack memory," in *Proc. 33rd IEEE Int. Conf. Comput. Design (ICCD)*, 2015, pp. 427–430.

[12] A. Ali Khan, F. Hameed, R. Blasing, S. S. P. Parkin, and J. Castrillon, "Shifts-reduce: Minimizing shifts in racetrack memory 4.0," *ACM Trans. Architect. Code Optim.*, vol. 16, no. 4, pp. 1–23, 2019.

[13] S. Olliver, S. Kline, R. Kawsher, R. Melhem, S. Banja, and A. K. Jones, "Leveraging transverse reads to correct alignment faults in domain wall memories," in *Proc. 49th Annu. IEEE/IFIP Int. Conf. Dependable Syst. Netw. (DSN)*, 2019, pp. 375–387.

[14] S. Ollivier, D. Kline, R. Kawsher, R. Melhem, S. Banja, and A. K. Jones, "The power of orthogonality," in *Proc. IEEE Comput. Soc. Annu. Symp. VLSI*, 2019, pp. 100–102.

[15] K. Roxy, S. Olliver, A. Hoque, S. Longofono, A. K. Jones, and S. Banja, "A novel transverse read technique for domain-wall "race-track" memories," *IEEE Trans. Nanotechnol.*, vol. 19, pp. 648–652, Aug. 2020. [Online]. Available: https://ronny.cswp.cs.technion.ac.il/wp-content/uploads/sites/54/2016/05/chapters1-9.pdf

[16] Y. Cassuto, M. Schwartz, V. Bohossian, and J. Bruck, "Codes for asymmetric limited-magnitude errors with application to multilevel flash memories," *IEEE Trans. Inf. Theory*, vol. 56, no. 4, pp. 1582–1595, Apr. 2010.

[17] B. H. Marcus, R. M. Roth, and P. H. Siegel, *An Introduction to Coding for Constrained System*, 5th ed. 2001. [Online]. Available: https://ronny.cswp.cs.technion.ac.il/wp-content/uploads/sites/54/2016/05/chapters1-9.pdf

[18] Y. M. Chee and V. K. Vu, "Codes correcting synchronization errors for symbol-pair read channels," in *Proc. IEEE Int. Symp. Inf. Theory*, 2020, pp. 746–750.

[19] C. Schoeny, A. W.-Zeh, R. Gabrys, and E. Yaakobi, "Code correcting a burst of deletions or insertions," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 1971–1985, Apr. 2017.

[20] Y. Cassuto and M. Blaum, "Codes for symbol-pair read channels," *IEEE Trans. Inf. Theory*, vol. 57, no. 12, pp. 8011–8020, Dec. 2011.

[21] E. Yaakobi, J. Bruck, and P. H. Siegel, "Constructions and decoding of cyclic codes over b-symbol read channels," *IEEE Trans. Inf. Theory*, vol. 62, no. 4, pp. 1541–1551, Apr. 2016.

[22] Y. M. Chee, L. Ji, H. M. Kiah, C. Wang, and J. Yin, "Maximum distance separable codes for symbol-pair read channels," *IEEE Trans. Inf. Theory*, vol. 59, no. 11, pp. 7259–7267, Nov. 2013.

[23] B. Ding, T. Zhang, and G. Ge, "Maximum distance separable codes for b-symbol read channels," *Finite Fields Appl.*, vol. 49, pp. 180–197, Jan. 2018.

[24] H. Q. Dinh, B. T. Nguyen, A. K. Singh, and S. Sriboonchitta, "On the symbol-pair distance of repeated-root constacyclic codes of prime power lengths," *IEEE Trans. Inf. Theory*, vol. 64, no. 4, pp. 2417–2430, Apr. 2018.

[25] R. Hulett, S. Chandak, and M. Wootters, "On coding for an abstracted nanopore channel for DNA storage," in *Proc. IEEE Int. Symp. Inf. Theory*, 2021, pp. 2465–2470.

[26] S. Wang, V. K. Vu, and V. Y. F. Tan, "Codes for correcting t limited-magnitude sticky deletions," in *Proc. IEEE Int. Symp. Inf. Theory*, 2023, pp. 1–7.

[27] H. Mahdavifar and A. Vardy, "Asymptotically optimal sticky-insertion correcting codes with efficient encoding and decoding," in *Proc. IEEE Int. Symp. Inf. Theory*, 2017, pp. 2683–2687.

[28] O. Elishco, R. Gabrys, and E. Yaakobi, "Bounds and constructions of codes over symbol-pair read channels," in *IEEE Trans. Inf. Theory*, vol. 66, no. 3, pp. 1385–1395, Mar. 2020.

[29] J. Sima, R. Gabrys, and J. Bruck, "Optimal systematic t-deletion correcting codes," in *Proc. IEEE Int. Symp. Inf. Theory*, 2020, pp. 769–774.

Yeow Meng Chee (Senior Member, IEEE) received the B.Math. degree in computer science, and combinatorics and optimization and the M.Math. and Ph.D. degrees in computer science from the University of Waterloo, Waterloo, ON, Canada, in 1988, 1989, and 1996, respectively. He is currently a Professor of Design and Engineering with the National University of Singapore. Prior to this, he was a Professor of Mathematical Sciences with Nanyang Technological University, the Program Director of Interactive Digital Media Research and Development with the Media Development Authority of Singapore, a Postdoctoral Fellow with the University of Waterloo and IBM's Zurich Research Laboratory, the General Manager of the Singapore Computer Emergency Response Team, and the Deputy Director of Strategic Programs with Infocomm Development Authority, Singapore. His research interest lies in the interplay between combinatorics and computer science/engineering, particularly in combinatorial design theory, coding theory, extremal set systems, and their applications. He is an Editor for the *Journal of Combinatorial Theory, Series A*. He is a Fellow of the Institute of Combinatorics and its Applications.

Alexander Vardy (Fellow, IEEE) was born in Moscow, Russia, in 1963. He received the B.Sc. degree (summa cum laude) from the Technion–Israel Institute of Technology, Israel, in 1985, and the Ph.D. degree from Tel-Aviv University, Israel, in 1991. From 1985 to 1990, he was with Israeli Air Force, where he worked on electronic counter measures systems and algorithms. From 1992 to 1993, he was a Visiting Scientist with IBM Almaden Research Center, San Jose, CA, USA. From 1993 to 1998, he was with the University of Illinois at Urbana–Champaign, first as an Assistant Professor and then as an Associate Professor. Since 1998, he has been with the University of California at San Diego (UCSD), where he was the Jack Keil Wolf Chair Professor with the Department of Electrical and Computer Engineering and the Department of Computer Science. While on sabbatical from UCSD, he has held long-term visiting appointments with CNRS, France; EPFL, Switzerland; the Technion–Israel Institute of Technology; and Nanyang Technological University, Singapore. His research interests include error-correcting codes, algebraic and iterative decoding algorithms, lattices and sphere packings, coding for storage systems, cryptography, computational complexity theory, and fun math problems. He received the IBM Invention Achievement Award in 1993 and the NSF Research Initiation and CAREER Awards in 1994 and 1995. In 1996, he was appointed as a Fellow in the Center for Advanced Study, University of Illinois, and received the Xerox Award for Faculty Research. He received the IEEE Information Theory Society Paper Award (jointly with Ralf Koetter) in 2004. In 2005, he received the Fulbright Senior Scholar Fellowship and the Best Paper Award at the IEEE Symposium on Foundations of Computer Science. In 2017, his work on polar codes was recognized by the IEEE Communications and Information Theory Societies Joint Paper Award. From 1995 to 1998, he was an Associate Editor of *Coding Theory* and from 1998 to 2001, he was the Editor-in-Chief of the IEEE TRANSACTIONS ON INFORMATION THEORY. In 1996, he became a Fellow of the David and Lucile Packard Foundation. He was a member of the Board of Governors of the IEEE Information Theory Society from 1998 to 2006 and from 2011 to 2017.

Van Khu Vu received the B.Sc. degree in mathematics from Vietnam National University (VNU), Hanoi, in 2010, and the Ph.D. degree in mathematics from Nanyang Technological University (NTU), Singapore, in 2018. From 2010 to 2012, he was a Lecturer with the VNU University of Sciences, Hanoi. From 2018 to 2019, he was a Research Fellow with the School of Physical and Mathematical Sciences, NTU, where he is currently a Research Fellow with the Department of Industrial Systems Engineering and Management. His primary research interests lie in the areas of algorithms, combinatorics, and coding theory.

Eitan Yaakobi (Senior Member, IEEE) received the B.A. degree in computer science and mathematics and the M.Sc. degree in computer science from the Technion—Israel Institute of Technology, Haifa, Israel, in 2005 and 2007, respectively, and the Ph.D. degree in electrical engineering from the University of California at San Diego, La Jolla, in 2011. He is an Associate Professor with the Computer Science Department at the Technion—Israel Institute of Technology. He also holds a courtesy appointment with the Technion's Electrical and Computer Engineering Department. From 2011 to 2013, he was a Postdoctoral Researcher with the Department of Electrical Engineering, California Institute of Technology, and the Center for Memory and Recording Research, University of California at San Diego. Since 2016, he has been with the Center for Memory and Recording Research, University of California at San Diego, and from 2018 to 2022, he was affiliated with the Institute of Advanced Studies, Technical University of Munich, where he held a four-year Hans Fischer Fellowship, funded by the German Excellence Initiative and the EU 7th Framework Program. His research interests include information and coding theory with applications to non-volatile memories, associative memories, DNA storage, data storage and retrieval, and private information retrieval. He received the Marconi Society Young Scholar in 2009 and the Intel Ph.D. Fellowship in 2010 and 2011. He is a recipient of several grants, including the ERC Consolidator Grant. Since 2020, he has been serving as an Associate Editor for Coding and Decoding for the IEEE TRANSACTIONS ON INFORMATION THEORY.