Coding for Transverse-Reads in Domain Wall Memories

Yeow Meng Chee[*], **Alexander Vardy**[†], **Van Khu Vu**[*], and **Eitan Yaakobi**[§]
[*] Department of Industrial Systems Engineering and Management, National University of Singapore, Singapore
[†] Department of Electrical and Computer Engineering, University of California San Diego, La Jolla, CA 92093, USA
[§] Department of Computer Science, Technion — Israel Institute of Technology, Haifa, 3200003 Israel
Emails:`{ymchee,isevvk}@nus.edu.sg, avardy@ucsd.edu, yaakobi@cs.technion.ac.il`

***Abstract*—*Transverse-read* is a novel technique to detect the number of '1's stored in *domain wall memory*, also known as *racetrack memory*, without shifting any domains. Motivated by this technique, we propose a novel scheme to combine transverse-read and *shift-operations* such that the number of shift-operations can be reduced 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*. We first present several properties of transverse-read codes and show that they are equivalent to constrained codes. Then, we compute the maximal asymptotic rate of transverse-read codes for several parameters. Next, we construct achieving capacity codes with efficient encoding/decoding algorithms. Finally, we discuss transverse-read codes which correct shift-errors in domain wall memory.**

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 [1]–[4]. 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 [1], 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 racetracks [5]–[9]. Besides that, the access latency and the energy consumption in racetrack memory depend on the average number of shift-operations. Several works have been done to reduce the number of shift-operations in racetrack memory [10], [11].

Another approach to overcome the faults in the shifting process of DMW was proposed recently in [12]–[14]. 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 DMW without shifting any domains, while still requiring ultra-low power. However, detecting only the number of ones in the DMW significantly reduces the information rate that can be stored within the memory. Hence, the authors of [14] also demonstrated how TR can be applied on 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 can compute that the information rate is about 0.7925. Then, we observe that this scheme significantly reduces the number of shift-operations by about δ times. For example, when $\delta = 2$, if we just shift normally about $n/2$ times, we can only read 50% of information bits but using this scheme, the achievable information rate is at least 0.79.25. 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 in this scheme. Furthermore, we can show that this scheme is also helpful to correct shift-errors in racetrack memories. From a practical point of view, this scheme captures the two features of DMW 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 some interesting challenges in combinatorics and algorithms.

In Section II, we present some necessary notations and define the codes formally. Section III studies properties of transverse-read codes, their maximal asymptotic rates, and propose several 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 future research.

II. Definitions and Preliminaries

Let Σ_q denote the q-ary alphabet $\{0, 1, \ldots, q-1\}$ and $[n]$ denote the set $\{1, 2, \ldots, n\}$. For each sequence $\boldsymbol{u} = (u_1, \ldots, u_n) \in \Sigma_q^n$, let $\boldsymbol{u}_{[i;k]} = (u_i, u_{i+1}, \ldots, u_{i+k-1})$, $1 \leqslant i \leqslant n-k+1$, denote a length-$k$ substring of $\boldsymbol{u}$. The weight of the vector $\boldsymbol{u}$ is $w(\boldsymbol{u}) = \sum_{i=1}^{n} u_i$ and when $q = 2$, the weight of a binary vector is the number of 1's in the vector. 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 the rate of the code $\mathcal{C}$ to be $R(\mathcal{C}) = \log_q(|\mathcal{C}|)/n$.

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

- *The (ℓ, δ)-**transverse-read vector** of a length-n word $\boldsymbol{x} = (x_1, \ldots, x_n) \in \Sigma_2^n$ is the vector $TR_{\ell,\delta}(\boldsymbol{x}) = \big(w(\boldsymbol{x}_{[1;\ell]}), w(\boldsymbol{x}_{[\delta+1;\ell]}), \ldots, w(\boldsymbol{x}_{[k\delta+1;\ell]})\big) \in \Sigma_{\ell+1}^{k+1}$, where $w(\boldsymbol{x}_{[i\delta+1;\ell]})$ is the weight of the length-ℓ substring $\boldsymbol{x}_{[i\delta+1;\ell]}$.*
- *A code $\mathcal{C}(n, \ell, \delta) \subseteq \Sigma_2^n$ is called a **binary** (ℓ, δ)-**transverse-read code** if for all distinct $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{C}(n, \ell, \delta)$, it holds that $TR_{\ell,\delta}(\boldsymbol{x}) \neq TR_{\ell,\delta}(\boldsymbol{y})$.*
- *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*

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

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

We now observe that for each word $\boldsymbol{x} \in \Sigma_2^n$, we always find its transverse read vector $TR_{\ell,\delta}(\boldsymbol{x}) \in \Sigma_{\ell+1}^{k+1}$. However, given a vector $\boldsymbol{u} \in \Sigma_{\ell+1}^{k+1}$, there may not exist a binary word $\boldsymbol{x} \in \Sigma_2^n$ that $\boldsymbol{u} = TR_{\ell,\delta}(\boldsymbol{x})$.

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

- *A vector $\boldsymbol{u} \in \Sigma_{\ell+1}^{k}$ is called a **valid** (ℓ, δ)-**transverse read vector** if there exists a binary word $\boldsymbol{x} \in \Sigma_2^n$ such that $\boldsymbol{u} = TR_{\ell,\delta}(\boldsymbol{x})$.*
- *A set of such vectors $\boldsymbol{u}$ is called a **valid** (ℓ, δ)-**transverse-read code**.*

From Definitions 1 and 2, we can easily obtain the following result.

Proposition 3.

- *Let $\mathcal{C}$ be a binary (ℓ, δ)-transverse-read code and $TR_{\ell,\delta}(\mathcal{C}) = \{TR_{\ell,\delta}(\boldsymbol{c}) : \boldsymbol{c} \in \mathcal{C}\} \subseteq \Sigma_{\ell+1}^{k+1}$. Then, $TR_{\ell,\delta}(\mathcal{C})$ is a valid (ℓ, δ)-transverse-read code and $|TR_{\ell,\delta}(\mathcal{C})| = |\mathcal{C}|$.*
- *Let $TR(k, \ell, \delta)$ denote the set of all valid (ℓ, δ)-transverse read vectors of length k. Then, $|TR(k, \ell, \delta)| = A(n, \ell, \delta)$.*

We now examine a model of domain wall memory of n domains which stores a binary word of length n, and two consecutive access points that can transverse read. i.e. can sense the weight of a segment of length ℓ. A shift-operation in racetrack memory shifts all domains together δ positions. So, if $\boldsymbol{x} = (x_1, \ldots, x_n) \in \Sigma_2^n$ is a stored word, the output in our reading scheme is $TR_{\ell,\delta}(\boldsymbol{x})$. Using the new scheme, we can reduce the number of shift-operations by about δ 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-operations and transverse-reads may not work perfectly and errors may occur. It is known that shift-errors, to be defined in Section IV, can be modeled as synchronizations, including sticky-insertions and deletions. We also see that errors in transverse-read may cause some substitution errors. Hence, in this work, we also study some transverse-read codes which can correct shift-errors and substitutions errors.

III. Transverse-Read Codes

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

To study the values of $A(n; \ell, \delta)$ and $\mathcal{R}(\ell, \delta)$, we may consider the maximal valid (ℓ, δ)-transverse-read code $TR(k, \ell, \delta)$ since $|TR(k, \ell, \delta)| = A(n; \ell, \delta)$. We first present several basic results on $A(n; \ell, \delta)$ and $\mathcal{R}(\ell, \delta)$ in the following theorem.

Theorem 4.

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

Proof:

1) 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 $\mathcal{R}(\ell = 1, \delta) = \lim_{n\to\infty} \frac{k+1}{n} = \frac{1}{\delta}$.
2) For $\ell = \delta$, and $k = (n/\delta) - 1$, given $\boldsymbol{x} = (x_1, \ldots, x_n) \in \Sigma_2^n$, $TR_{\ell,\delta}(\boldsymbol{x}) = \big(w(\boldsymbol{x}_{[1;\ell]}), w(\boldsymbol{x}_{[\ell+1;\ell]}), \ldots, w(\boldsymbol{x}_{[k\ell+1;\ell]})\big) \in \Sigma_{\ell+1}^{k+1}$. Since all segments $\boldsymbol{x}_{[i\ell+1;\ell]}$, for $0 \leqslant i \leqslant k$, do not overlap, any vector $\boldsymbol{u} \in \Sigma_{\ell+1}^{k+1}$ is a valid (ℓ, ℓ)-transverse read vector. Hence $A(n, \ell, \delta = \ell) = |TR(k, \ell, \ell)| = (\ell+1)^{k+1} = (\ell+1)^{n/\ell}$ and thus $\mathcal{R}(\ell, \delta = \ell) = \lim_{n\to\infty} \frac{(k+1)(\log_2(\ell+1))}{n} = \frac{\log_2(\ell+1)}{\delta}$.
3) Using the same argument as in part 2), note that all segments $\boldsymbol{x}_{[i\delta+1;\ell]}$, for $0 \leqslant i \leqslant k$, do not overlap, the claim is proven.
4) Consider two length-n vectors $\boldsymbol{u} = (0, \ldots, 0, u_1, \ldots, u_{n-\ell})$, $\boldsymbol{v} = (0, \ldots, 0, v_1, \ldots, v_{n-\ell}) \in \Sigma_2^n$ such that $\boldsymbol{u} \neq \boldsymbol{v}$. We observe that $TR_{\ell,\delta=1}(\boldsymbol{u}) \neq TR_{\ell,\delta=1}(\boldsymbol{v})$. Let $\mathcal{C}(n, \ell, \delta)$ be a set of all vectors of length n that the first ℓ entries are

zeros. So, $\mathcal{C}(n,\ell,\delta)$ is a binary $(\ell,\delta=1)$-transverse-read code and $|\mathcal{C}(n,\ell,\delta=1)|=2^{n-\ell}$. Therefore, $A(n,\ell,\delta=1)\geqslant 2^{n-\ell}$ and $\mathcal{R}(\ell,\delta=1)=\lim_{n\to\infty}\frac{n-\ell}{n}=1$. ∎

For all cases in Theorem 4, 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 $1<\delta<\ell$. Let us start with the case where $\delta=2$ and even values of ℓ, which will be addressed in the following theorem.

Theorem 5. *For $\delta=2$ and ℓ even, it holds that*

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

Proof: Let $n_1=n/2$ and $\ell_1=\ell/2$ be two positive integers. Given a vector $\boldsymbol{x}=(x_1,\dots,x_n)\in\Sigma_2^n$, let $f(\boldsymbol{x})=(f_1,\dots,f_{n_1})\in\Sigma_3^{n_1}$ where $f_i=x_{2i-1}+x_{2i}\in\{0,1,2\}$ for $1\leqslant i\leqslant n_1$. We see that $TR_{\ell,\delta=2}(\boldsymbol{x})=\big(w(f_{[1;\ell_1]}),w(f_{[\ell_1+1;\ell_1]}),\dots,w(f_{[k\ell_1+1;\ell_1]})\big)\in\Sigma_{\ell+1}^{k+1}$. Let $\mathcal{C}(n,\ell,\delta)$ be a binary (ℓ,δ)-transverse-read code, that is, for two different vectors $\boldsymbol{x},\boldsymbol{y}\in\mathcal{C}(n,\ell,\delta)$, it holds that $TR_{\ell,\delta}(\boldsymbol{x})\neq TR_{\ell,\delta}(\boldsymbol{y})$. Hence, $f(\boldsymbol{x})\neq f(\boldsymbol{y})$. So, $|\mathcal{C}(n,\ell,\delta)|\leqslant|\Sigma_3^{n_1}|=3^{n_1}$, for any (ℓ,δ)-transverse-read code $\mathcal{C}(n,\ell,\delta)$. Therefore, $A(n,\ell,\delta)\leqslant 3^{n/2}$, and thus $\mathcal{R}(\ell,\delta=2)\leqslant\frac{\log_2 3}{2}\approx 0.7925$.

On the other hand, we can construct a binary (ℓ,δ)-transverse-read code $\mathcal{C}(n,\ell,\delta)$ as follows. Let $\mathcal{F}\subseteq\Sigma_3^{n_1}$ be the set of all ternary vectors of length n_1 such that their first ℓ_1 entries are all zeros. So, $|\mathcal{F}|=3^{n_1-\ell_1}$. For each $f=(f_1,\dots,f_{n_1})\in\mathcal{F}$, we define $f^{-1}=\boldsymbol{x}=(x_1,\dots,x_n)\in\Sigma_2^n$ such that $(x_{2i-1},x_{2i})=(0,0)$ if $f_i=0$, $(x_{2i-1},x_{2i})=(0,1)$ if $f_i=1$ and $(x_{2i-1},x_{2i})=(1,1)$ if $f_i=2$. Let $\mathcal{C}(n,\ell,\delta)$ be the set of all vectors $\boldsymbol{x}=f^{-1}$ defined above where $f\in\mathcal{F}$. So, $|\mathcal{C}(n,\ell,\delta)|=|\mathcal{F}|=3^{n_1-\ell_1}$. Moreover, we can see that $\mathcal{C}(n,\ell,\delta)$ is a binary (ℓ,δ)-transverse-read code. Therefore, $A(n,\ell,\delta)\geqslant 3^{n_1-\ell_1}$ and thus $\mathcal{R}(\ell,\delta)\geqslant\frac{\log_2 3}{2}$.

In conclusion, we obtain $\mathcal{R}(\ell,\delta=2)=\frac{\log_2 3}{2}\approx 0.7925$ and the theorem is proven. ∎

The results in Theorem 5 can be extended in the following theorem for arbitrary values of ℓ and δ, where ℓ is a multiple of δ. The proof follows similar arguments as in the proof of Theorem 5 and is thus omitted.

Theorem 6. *If ℓ is a multiple of δ, then*

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

Next, we continue with $\delta=2$ and odd values of ℓ. The result in Theorem 5 gives us a lower bound on the maximal asymptotic rate of (ℓ,δ)-transverse-read codes for $\delta=2$. We state the result formally in the following theorem.

Theorem 7. *For all $\ell>\delta>1$ it holds that*

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

Proof: To prove the theorem, we present a construction of an (ℓ,δ)-transverse-read code. Let $k=(n-\ell)/2$ and $\boldsymbol{u}=(u_1,\dots,u_k)\in\Sigma_3^k$ be a ternary vector of length k. Let $g(\boldsymbol{u})=\boldsymbol{c}=(c_1,\dots,c_n)\in\Sigma_2^n$ be such that $c_1=\cdots=c_\ell=0$ and for $1\leqslant i\leqslant k$, $(c_{\ell+2i-1},c_{\ell+2i})=(0,0)$ 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 $\mathcal{C}(n,\ell,\delta=2)=\{g(\boldsymbol{u}):\boldsymbol{u}\in\Sigma_3^k\}$. It is possible to show that if $\boldsymbol{u}\neq\boldsymbol{v}$ then $g(\boldsymbol{u})\neq g(\boldsymbol{v})$ for any $\boldsymbol{u},\boldsymbol{v}\in\Sigma_3^k$. Hence, $|\mathcal{C}(n,\ell,\delta=2)|=|\Sigma_3^k|=3^k$. Moreover, if $g(\boldsymbol{u})\neq g(\boldsymbol{v})$ then $TR_{\ell,\delta=2}(g(\boldsymbol{u}))\neq TR_{\ell,\delta}(g(\boldsymbol{v}))$. Thus, the code $\mathcal{C}(n,\ell,\delta=2)$ constructed above is a (ℓ,δ)-transverse-read code. Hence, $A(n,\ell,\delta=2)\geqslant 3^k$ and thus $\mathcal{R}(\ell,\delta=2)\leqslant\frac{\log_2 3}{2}\approx 0.7925$ for any $\ell\geqslant\delta=2$. ∎

From the above proof of Theorem 7, we can find a construction of an $(\ell,\delta=2)$-transverse-read code with efficient encoding algorithms. Similarly, we can extend the result in Theorem 7 for arbitrary values of ℓ and δ where $\ell>\delta$. We first present a simple construction of a binary (ℓ,δ)-transverse-read code.

Construction 8. *Let $k=\frac{n-\ell}{\delta}$ and $\boldsymbol{u}=(u_1,\dots,u_k)\in\Sigma_{\delta+1}^k$ be a $(\delta+1)$-ary vector of length k. Let $g(\boldsymbol{u})=\boldsymbol{c}=(c_1,\dots,c_n)\in\Sigma_2^n$ be such that $c_i=0$ for $1\leqslant i\leqslant\ell$ and for $1\leqslant i\leqslant k$, $\boldsymbol{c}_{[\ell+\delta(i-1)+1;\delta]}$ is a subvector of length δ such that its first $\delta-j$ entries are 0 and its last j entries are 1 if $u_i=j$. Let $\mathcal{C}(n,\ell,\delta)=\{g(\boldsymbol{u}):\boldsymbol{u}\in\Sigma_{\delta+1}^k\}$.*

It is possible to show that the code $\mathcal{C}(n,\ell,\delta)$ constructed above is a binary (ℓ,δ)-transverse-read code since for any $\boldsymbol{u},\boldsymbol{v}\in\mathcal{C}(n,\ell,\delta)$, it holds that $g(\boldsymbol{u})\neq g(\boldsymbol{v})$, and thus $TR_{\ell,\delta}(\boldsymbol{u})\neq TR_{\ell,\delta}(\boldsymbol{v})$. Moreover, $|\mathcal{C}(n,\ell,\delta)|=|\Sigma_{\delta+1}^k|=(\delta+1)^k$. So, $A(n,\ell,\delta)\geqslant(\delta+1)^k$. Therefore, we obtain the following result on the lower bound of the maximal asymptotic rate of (ℓ,δ)-transverse-read codes.

Theorem 9. *If ℓ and δ are two integers such that $\ell>\delta>1$ then*

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

Note that Construction 8 also provides binary (ℓ,δ)-transverse-read codes with an efficient encoding algorithm.

In the rest of this section, we present a technique to find the asymptotic rates of (ℓ,δ)-transverse-read codes exactly, 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 $\boldsymbol{u}=(u_1,\dots,u_k)\in TR(k,\ell=3,\delta=2)\subseteq\Sigma_4^{k+1}$ 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 $1\leqslant i\leqslant k$, $u_i=x_{2i-1}+x_{2i}+x_{2i+1}\in\{0,1,2,3\}$. We can view u_i as a path from x_{2i-1} to x_{2i+1}. Here, x_{2i-1} is called a starting point of u_i and x_{2i+1} is called an ending point of u_i. So, a starting point of u_i is also an ending point of u_{i-1}. 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 1, where node j is the state that the ending point of u_{i-1} is $x_{2i-1}=j$ for $j=1,2$. If $x_{2i-1}=0$ and the ending point of u_i is $x_{2i+1}=0$, then $u_i=0$ or $u_i=1$. Hence, from the state 0, if we write $u_i=0$ or $u_i=1$ then we

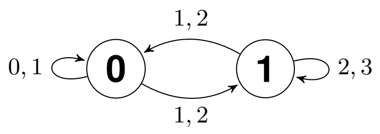

Fig. 1: Non-deterministic finite state transition diagram $\ell = 3, \delta = 2$

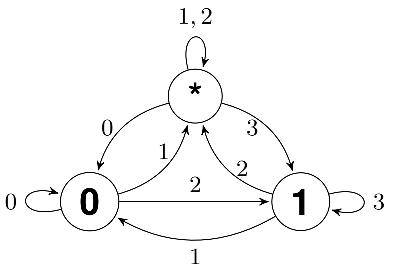

Fig. 2: Deterministic finite state transition diagram $\ell = 3, \delta = 2$

may still stay in the same state 0. If $x_{2i-1} = 0$ and the ending point of u_i is $x_{2i+1} = 1$, then $u_i = 1$ or $u_i = 2$. Hence, from the state 0, if we write $u_i = 1$ or $u_i = 2$ then we may move to the new state 1. We also note that, from state 0, if we write $u_i = 1$, we may stay in the same state or move to the new state. Hence, the state machine in Figure 1 is a non-deterministic finite state machine. We note 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. In this case, the regular language $TR(k, \ell, \delta)$ is recognized by a deterministic finite state machine as in Figure 2. 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:

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

So, using the well-known Perron-Frobenius theory [16], we can find exactly the maximal asymptotic rate of $(\ell = 3, \delta = 2)$-transverse-read codes which is $(\log_2 \lambda)/2 = 0.8858$ where $\lambda = 3.4142$ is the largest real eigenvalue of A_G.

Besides that, $TR(k, \ell, \delta)$, which can be expressed by the state machine in Figure 2, is also a constrained system. We state the following result.

Theorem 10. *We consider the following set*

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

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

Theorem 10 can be proven by showing that both of the above codes have the same finite state transition diagram as in Figure 2.

Furthermore, it is possible to extend the above results for other values of $\ell > \delta > 1$. 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 3. In this diagram, each node is a state that a length-3 substring is started by the corresponding length-3 substring. If a length-5 string starts by (0,0,0) and its weight is 1, it may end by (0,1,0) or (0,0,1). If its weight is 0, it must end by (0,0,0) and if its weight is 2, it must end by (0,1,1). Similar, we can build a non-deterministic finite state machine. For simplicity, in Figure 3, we only label all edges to go out from nodes (0,0,0) and (1,1,1). Once we have a non-deterministic finite state machine, we can build a deterministic finite state machine and compute the maximal asymptotic rate of transverse-read codes. Several numerical results were computed and tabulated in Table I.

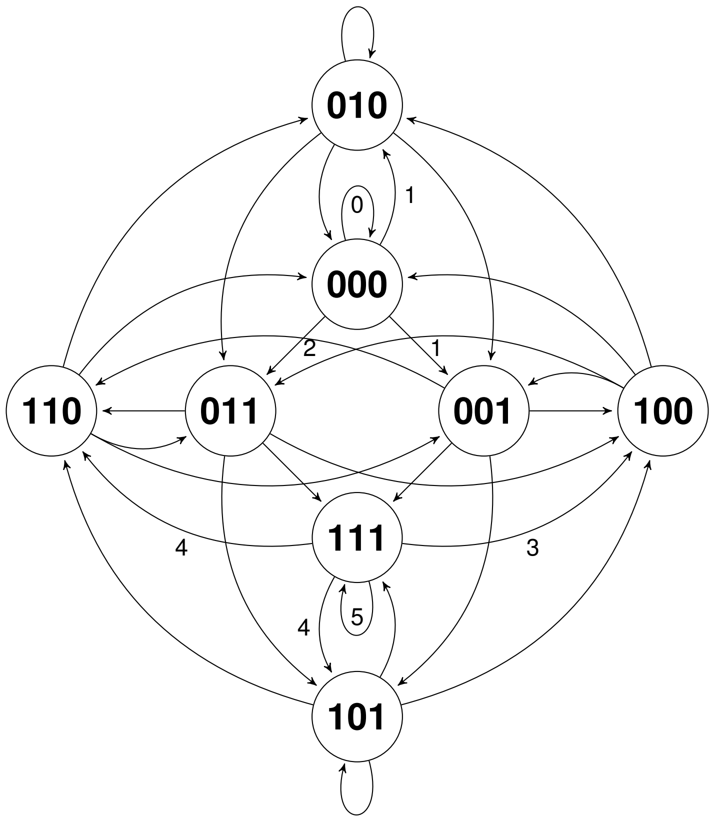

Fig. 3: Non-deterministic finite state transition diagram $\ell = 5, \delta = 2$.

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

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 the number of shift-operations is reduced by to 50%, we can still achieve information rates of at least 0.936. Since the asymptotic rates of $(\ell, \delta = 2)$-transverse-read codes increase when ℓ is odd and increasing, we are interested in finding the maximal asymptotic rates $\mathcal{R}(\ell, \delta = 2)$ for odd ℓ.

Furthermore, since we can build a deterministic finite state machine of any (ℓ, δ)-transverse-read code, it is possible to construct this code with efficient encoding/decoding algorithms using well-known finite state splitting algorithms [16]. 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

In this section, we discuss the ability of detect and correct errors of transverse-read codes. There are two types of errors we consider under this model: shift-errors and substitution errors. Shift-errors, which may occur when all domains are shifted, can be modeled as sticky-insertions or deletions. Normally, we may need to use some classical deletion correcting codes or use multiple heads to correct these errors. In this work, we show that some transverse-read codes have 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, so we obtain the word $(0, 2, 1)$. However, the word $(0, 2, 1)$ is not a valid (2,1)-transverse-read vector since 0 can not be followed by 2. Hence, we can detect and locate a single deletion in this case. Based on this simple observation, 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 $t = 1$.

Theorem 11. *Let $\mathcal{C}_1 \subseteq \Sigma_2^n$ be a binary code correcting a single sticky-deletion*[1]*. 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 the stored word. Thus $\boldsymbol{u} = TR_{2,1}(\boldsymbol{c}) = ((c_1 + c_2), (c_2 + c_3), \ldots, (c_{n-1} + c_n))$, where $u_i = c_i + c_{i+1}$, is its (2,1)-transverse-read vector. 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 a 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 that the symbol 0 or 2 is 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, in any case, we can recover the stored word $\boldsymbol{c}$. Therefore, the code $\mathcal{C}_1$ can correct a single deletion in a (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 (shift-error). It is interesting that we also can extend the result for transverse-read codes which correct multiple deletions.

Theorem 12. *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 do not exist two-consecutive deletions then we can recover the original word $\boldsymbol{c}$.*

Proof: Let us sketch the main idea of the proof of this theorem. To prove the theorem, we only need to follow the argument in the proof of Theorem 11. If the symbol 1 is deleted in the transverse-read vector $TR_{2,1}(\boldsymbol{c})$, we can detect and correct this error immediately. If the symbol 0 or 2 is deleted in the transverse-read vector, we see that a sticky-deletion occurs in the original word $\boldsymbol{c}$. Then, we use a decoder of the code $\mathcal{C}_t$, which can correct multiple sticky-deletion, to correct these errors. ■

So far, we showed that our scheme of using (ℓ, δ)-transverse-read codes 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 [17] for codes correcting deletions in symbol-pair read channel. We note that the best known results on codes correcting t deletions require at least $8t \log n + o(t \log n)$ bits of redundancy while it is possible to correct t sticky-deletions using only $t \log n + o(\log n)$ bits of redundancy, given some constant t. Hence, in our scheme for $\ell = 2$ and $\delta = 1$, it is easier to correct shift-errors. The results for other values of ℓ and δ are interesting as well and will be studied in the full version of this work.

Besides that, a substitution error occurs when there is a mistake in transverse read and a symbol is read wrongly. For example, $\boldsymbol{x} = (0, 0, 1, 1, 0)$ and $TR_{2,1}(\boldsymbol{x}) = (0, 1, 2, 1)$. If a third symbol in the transverse-read vector $TR_{2,1}(\boldsymbol{x})$ is wrong then we obtain the vector $(0, 1, 0, 1)$ which is invalid. Moreover, we can locate an error in the pattern $(0, 1, 0)$. It is helpful to correct a substitution error with large magnitude. For a substitution error with small magnitude, we propose to study a coding scheme to combine our transverse-read codes with the well-known limited-magnitude error correcting code [15]. We also can show that our scheme for $\ell = 2$ and $\delta = 1$ is helpful to correct a single limited magnitude error. These schemes will be discussed in the full version of our work.

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 proposed constructions. Furthermore, we also show that our scheme of using these transverse-read codes is helpful to correct shift-errors in domain wall memories. In the full version of our work, we also present some encoding/decoding algorithms in details. The ability of transverse-read codes in correcting substitution-errors will be studied in our future work. Furthermore, we are also interested in finding the maximal asymptotic rates of (ℓ, δ)-transverse-read codes for other values of ℓ and δ.

Acknowledgement

The research of Y. M. Chee is supported by the Singapore Ministry of Education under grant MOE2017-T3-1-007. The research of A. Vardy is supported in part by NSF grant CCF-1764104. The work of E. Yaakobi is supported in part by the Israel Science Foundation (ISF) grant No. 1817/18.

[1]A sticky deletion is a deletion which shortens a run of length at least two by one.

REFERENCES

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

[2] Z. Sun, W. Wu, and H. Li, “Cross-layer racetrack memory design for ultra high density and low power consumption,” in *Design Automation Conference (DAC)*, pp. 1–6, May 2013.

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

[4] R. Blasing, A. Ali Khan, P. Filippou, C. Garg, F. Hameed, J. Castrillon, and S. Parkin, “Magnetic racetrack memory: From physics to the cusp of applications within a decade”, *Proc. of the IEEE*, vol. 18, no. 10, pp. 1303–1321, 2020.

[5] C. Zhang, G. Sun, X. Zhang, W. Zhang, W. Zhao, T. Wang, Y. Liang, Y. Liu, Y. Wang, and J. Shu, “Hi-fi playback: Tolerating position errors in shift operations of racetrack memory,” *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*, pp. 694–706, 2015.

[6] Y.M. Chee, H.M. Kiah, A. Vardy, V.K. Vu, and E. Yaakobi, “Coding for racetrack memories,” *IEEE Trans. on Inform. Theory*, vol. 64, no. 11, pp. 7094–7112, 2018.

[7] Y.M. Chee, H.M. Kiah, A. Vardy, V.K. Vu, and E. Yaakobi, “Codes correcting limited-shift errors in racetrack memories” *Proc. IEEE Int. Symp. on Inform. Theory*, pp. 96–100, 2018.

[8] G. Mappouras, A. Vahid, R. Calderbank, and D. J. Sorin, “GreenFlag: Protecting 3D-Racetrack memory from shift errors”, *2019 49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 1–12, 2019.

[9] S. Archer, G. Mappouras, R. Calderbank, and D. J. Sorin, “Foosball coding: Correcting shift errors and bit flip errors in 3D racetrack memory”, *2020 50th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 331–342, 2020.

[10] E. Atoofian and A. Saghir, “Shift-aware racetrack memory”, *2015 33rd IEEE Int. Conf. on Computer Design (ICCD)*, pp. 427–430, 2015.

[11] A. Ali Khan, F. Hameed, R. Blasing, S. Parkin, and J. Castrillon, “Shifts-Reduce: Minimizing shifts in racetrack memory 4.0”, *ACM Trans. on Architecture and Code Optimization,* vol. 16, no. 4, pp. 56:1–56:23, 2019.

[12] S. Olliver, S. Kline Jr., R. Kawsher, R. Melhem, S. Banja, and A.K. Jones, “Leveraging transverse reads to Correct alignment faults in domain wall memories,” *49th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN)*, pp. 375–387, 2019.

[13] S. Ollivier, D. Jr. Kline, R. Kawsher, R. Melhem, S. Banja, and A. K. Jones, “The power of orthogonality”, *2019 IEEE Annual Symp. on VLSI*, pp. 100–102, 2019.

[14] K. Roxy, S. Olliver, A. Hoque, S. Longofono, A.K. Jones, and S. Banja, “A novel transverse read technique for domain-wall “racetrack” memories,” *IEEE Transactions on Nanotechnology*, vol. 19, pp. 648–652, 2020.

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

[16] B.H. Marcus, R.M. Roth, and P.H. Siegel, *An introduction to coding for constrained system*, 5th edition, Oct. 2001.

[17] Y. M. Chee and V. K. Vu, “Codes correcting synchronization errors for symbol-pair read channels”, *Proc. IEEE Int. Symp. on Inform. Theory,* pp. 746–750, 2020.