Codes Correcting Under- and Over-Shift Errors in Racetrack Memories

Yeow Meng Chee[*], **Han Mao Kiah**[*], **Alexander Vardy**[†*], **Van Khu Vu**[*], and **Eitan Yaakobi**[‡]

[*] School of Physical and Mathematical Sciences, Nanyang Technological University, 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, 32000 Israel

Emails:{`ymchee,hmkiah,vankhu001`}`@ntu.edu.sg`, `avardy@ucsd.edu`, `yaakobi@cs.technion.ac.il`

Abstract—**Racetrack memory is a new technology which utilizes magnetic domains along a nanoscopic wire in order to obtain extremely high storage density. In racetrack memory, each magnetic domain can store a single bit of information, which can be sensed by a head. The memory has a tape-like structure which supports a shift operation that moves the domains to be read sequentially by the head. In order to increase the memory's speed, prior work studied how to minimize the latency of the shift operation, while the no less important reliability of this operation has received only a little attention. There are two dominant kinds of errors in a shift operation, namely *under-shift* and *limited-over-shift* errors.**

In this work, we propose a coding scheme to combat these shift-errors. We first show that under-shift, limited-over-shift errors can be modeled as sticky-insertions, deletion-bursts of limited length, respectively. Therefore, in the model of multiple heads, a non-binary code correcting multiple bursts of deletions and any number of sticky-insertions can be used to tackle this problem. The goal of this work is to design such optimal codes.

I. Introduction

Racetrack memory is an emerging non-volatile memory technology which has attracted significant attention in recent years due to its promising ultra-high storage density and low power consumption [2], [3]. The basic information storage element of a racetrack memory is called a *domain*, also known as a *cell*. The magnetization direction of each cell is programmed to store information. The reading mechanism is operated by many *read ports*, called *heads*. In order to read the information, each cell is shifted to its closest head by a *shift operation*. We note that once a cell is shifted, all other cells are also shifted in the same direction and in the same speed. Normally, along the racetrack strip, all heads are fixed and distributed uniformly [4]. Each head thus reads only a block of consecutive cells which is called a *data segment*. A shift operation might not work perfectly. Our goal in this work is to study a coding scheme to combat these potential errors.

A. Preliminaries

Let $\mathbb{F}_q$ denote the q-ary finite field. A q-ary word of length n over the alphabet $\mathbb{F}_q$ is a vector $\boldsymbol{u}$ in $\mathbb{F}_q^n$. For each word $\boldsymbol{u} = (u_1, \ldots, u_n) \in \mathbb{F}_q^n$, a subvector of the word $\boldsymbol{u}$ is a vector $\boldsymbol{u}[i_1, i_2] = (u_{i_1}, u_{i_1+1}, \ldots, u_{i_2}) \in \mathbb{F}_q^{i_2-i_1+1}$, where $1 \leqslant i_1 \leqslant i_2 \leqslant n$. In case $i_1 = i_2 = i$, we denote the subvector $\boldsymbol{u}[i,i]$ by $\boldsymbol{u}[i]$ to specify the i-th symbol u_i of the vector $\boldsymbol{u}$. *A q-ary code* of length n is a subset $\mathbb{C} \subseteq \mathbb{F}_q^n$. Each element of $\mathbb{C}$ is called *a codeword*. For each code $\mathbb{C}$ of length n, we define the *rate* of the code $\mathbb{C}$ to be $R(\mathbb{C}) = \log_q(|\mathbb{C}|)/n$, where $|\mathbb{C}|$ is the size of the code $\mathbb{C}$.

This work was presented in part at the IEEE International Symposium on Information Theory 2018 [1].

Let ℓ, n, m be three positive integers such that $n = \ell \cdot m$. In this work, we consider a racetrack memory which comprises of n cells, each of which can store a single bit, and m heads which are distributed uniformly. We assume that the information stored in a racetrack memory is represented by a length-n word $\boldsymbol{c} = (c_1, c_2, \ldots, c_n)$ where the i-th cell stores the bit c_i. Initially, the i-th head is placed at location $(i-1)\cdot\ell + 1$ and reads a data segment of ℓ bits $\boldsymbol{c}^i = (c_{(i-1)\cdot\ell+1}, \ldots, c_{i\cdot\ell})$. For example, in Fig. 1, a racetrack memory contains twelve data cells and three heads are placed initially at locations 1, 5, and 9. Let $c_{i,j} = c_{(i-1)\cdot\ell+j}$ for $1 \leqslant i \leqslant m$ and $1 \leqslant j \leqslant \ell$. The output matrix from all m heads (without error) is:

$$\begin{pmatrix} \boldsymbol{c}^1 \\ \boldsymbol{c}^2 \\ \vdots \\ \boldsymbol{c}^m \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,\ell} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,\ell} \\ \vdots & \vdots & \ddots & \vdots \\ c_{m,1} & c_{m,2} & \cdots & c_{m,\ell} \end{pmatrix}$$

where $\boldsymbol{c}^i = (c_{i,1}, \ldots, c_{i,\ell})$ is the output of the i-th head for all $1 \leqslant i \leqslant m$. Let the following bijection

$$\Phi_m : \{0,1\}^m \mapsto \mathbb{F}_q,$$

where $q = 2^m$, be an order of all 2^m binary words of length m. For $1 \leqslant i \leqslant \ell$ and $\widehat{\boldsymbol{c}}^i = (c_{1,i}, c_{2,i}, \ldots, c_{m,i}) \in \mathbb{F}_2^m$, let $\Phi_m(\widehat{\boldsymbol{c}}^i) = u_i \in \mathbb{F}_q$. That is, we view each column in the output matrix as an element in the finite field $\mathbb{F}_q$. Thus, we can also view the output matrix as a q-ary word of length ℓ, $\boldsymbol{u} = (u_1, u_2, \ldots, u_\ell) \in \mathbb{F}_q^\ell$.

B. Under-Shift and Limited-Over-Shift

Under the above set up, we consider an event where an over-shift occurs and every head does not read one of the bits. Since there are m heads, m of the n bits are not read. However the positions of these bits are correlated since all heads are fixed and all cells are shifted in the same direction. For example, when an over-shift occurs at the ith position and the bit $c_{1,i}$ is not read in the first head, then the bit $c_{j,i}$ is not read in the j-th head for all $1 \leqslant j \leqslant m$. Thus, the column $\widehat{\boldsymbol{c}}^i$ is deleted in the output matrix, that is, the symbol u_i is deleted in the received q-ary word. Moreover, when an over-shift occurs, it may happen that not only a single bit but a few consecutive bits are deleted. However, the maximum number of consecutive deletions is limited by some small number b. Zhang et al. [4] recently provided an experimental result to show that $b \leqslant 2$ with extremely high probability. We therefore study a model where each over-shift causes a burst of consecutive deletions whose length is at most b and refer to this error as *b-limited-over-shift error* or *b-limited-burst-deletion*. In this model, any two bursts of consecutive deletions are not adjacent since each head always reads at least one bit between two over-shifts. To combat these errors, we use a q-ary deletion-correcting code. In

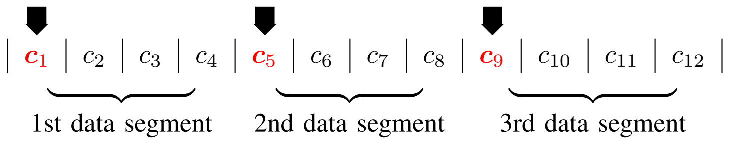

Fig. 1: Racetrack memory with twelve data cells and three heads

particular, we design a q-ary code correcting t bursts of deletions where the length of each burst is at most b and refer to the code as a *q-ary b-limited t-burst-deletion-correcting code.*

On the other hand, when an under-shift occurs, each head reads the same bit again. Hence, in the received q-ary word, a bit is repeated and there is a sticky-insertion. If only under-shift errors happen, we can use q-ary sticky-insertion-correcting codes to correct these errors. Recall that binary sticky-insertion-correcting codes have been well studied [5], [6] and could be easily generalized to q-ary codes. An optimal q-ary code correcting any number of sticky-insertions with high-rate was recently provided [7].

Furthermore, both under-shift and limited-over-shift errors could occur in racetrack memories. Hence, a code correcting a combination of sticky-insertions and multiple bursts of deletions of limited length is required. We also present a q-ary code correcting t_1 bursts of deletions and t_2 sticky-insertions whose length is at most b and refer to the code as a *q-ary b-limited t_1-burst-deletion t_2-sticky-insertion-correcting code*.

C. Related Work

On the theoretical side, deletion/insertion-correcting-codes have been studied for a long time [8]. Although there are several good results on sticky-insertion-correcting-codes [5], [6], deletion-correcting codes [8]–[10], and *single-burst-deletion-correcting codes* [11]–[13], there still is a lack of knowledge on codes correcting a combination of multiple bursts of deletions and sticky insertions.

On the practical side, there are a few work combating shift-errors in racetrack memories [4], [14]. While, Zhang et al. [4] proposed a mechanism scheme of using extra heads to deal with these errors, Vahid et al. [14] proposed to use deletion-correcting-codes to correct shift-errors without using extra heads. In this work, we use coding techniques to combat shift-errors in the model of having multiple heads in racetrack memories.

II. Our Contributions

We first present the following definition.

Definition 1. *A q-ary sequence $\pi = (\pi_1, \dots, \pi_\ell)$ is called a b-bounded de Bruijn sequence of strength h if all length-h subvectors $\pi[i, i+h-1]$ in b consecutive positions are distinct. That is, we can always determine the position i of sub-vector $\pi[i, i+h-1]$ provided the estimation of that position in a segment of length b.*

We may consider b-bounded de Bruijn sequence as a generalization of the well known de Bruijn sequence [15]. To the best of our knowledge, our work is the first to present the notion of b-bounded de Bruijn sequences and the latter forms an important component of the following code construction.

Construction 2. *Let $\pi = (\pi_1, \pi_2, \dots, \pi_\ell)$ be a b-bounded de Bruijn sequence of strength one over an alphabet of size q_1. Let $\mathbb{C}_{q_2}(\ell, t)$ be a q_2-ary t-erasure-correcting code of length ℓ. Let $q = q_1 \cdot q_2$. For each word $\boldsymbol{c} = (c_1, c_2, \dots, c_\ell) \in \mathbb{C}_{q_2}(\ell, t)$, we define $\boldsymbol{f}(\boldsymbol{c}, \pi) = (f_1, f_2, \dots, f_\ell)$ such that $f_i = (\pi_i, c_i)$ for all $1 \leqslant i \leqslant \ell$. We construct the following q-ary code of length ℓ, $\mathbb{C}_q(b, \ell, t) = \{\boldsymbol{f}(\boldsymbol{c}, \pi) : \boldsymbol{c} \in \mathbb{C}_{q_2}(\ell, t)\}$.*

We can show that Construction 2 provides asymptotically optimal q-ary b-limited t_1-burst-deletion-correcting codes.

Theorem 3. *Given $0 < \delta, \epsilon < 1$, there exists a q-ary b-limited t_1-burst-deletion-correcting code of length ℓ such that its rate satisfies*

$$R_1 \geqslant (1 - \log_q(b+1)) \cdot (1 - \delta - \epsilon)$$

where $t_1 \cdot b = \delta \cdot \ell$.

Furthermore, we show that the above code also can correct a combination of sticky-insertions and bursts of deletions. We obtain the following result.

Theorem 4. *Given $0 < \delta, \epsilon < 1$, there exists a q-ary b-limited t_1-burst-deletion t_2-sticky-insertion-correcting code of length ℓ for any arbitrarily large t_2 and $t_1 \cdot b = \delta \cdot \ell$ such that its rate*

$$R_2 \geqslant (1 - \log_q(b+2)) \cdot (1 - \delta - \epsilon).$$

These codes are asymptotically optimal.

References

[1] Y. M. Chee, H. M. Kiah, A. Vardy, V. K. Vu, and E. Yaakobi, "Codes correcting limited-shift errors in racetrack memories", presented in *IEEE Int. Symp. Inform. Theory 2018.*

[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," *Design Automation Conference (DAC)*, 2013, pp.1–6.

[4] 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," in *Proc. ACM/IEEE 42nd Annual Int. Symp. on Computer Architecture (ISCA)*, 2015, pp. 694–706.

[5] L. Dolecek and V. Anantharam, "Repetition error correcting sets: explicit constructions and prefixing methods", *SIAM J. Discrete Math.*, vol. 23, no. 4, pp. 2120-2146, 2010.

[6] H. Mahdavifar and A. Vardy, "Asymptotically optimal sticky-snsertion-correcting Codes with Efficient Encoding and Decoding", in *Proc. IEEE Int. Symp. on Inform. Theory,* 2017, pp. 2683–2687.

[7] S. Jain, F. Farnoud, M. Schwartz and J. Bruck, "Duplication-correcting codes for data storage in the DNA of living organisms", in *Proc. IEEE Int. Symp. on Inform. Theory,* 2017, pp. 1028–1032.

[8] V. I. Levenshtein, "Binary codes capable of correcting insertions, deletions and reversals", *Dokl. Akad. Nauk SSSR*, vol. 163, no. 4, pp. 845–848, 1965. Translation: *Sov. Phys. Dokl.*, vol. 10, no. 8, pp. 707–710, 1966.

[9] A. S. J. Helberg and H. C. Ferreira, "On multiple insertion/deletion correcting codes," *IEEE Trans. Inform. Theory*, vol. 48, pp. 305–308, 2002.

[10] J. Brakensiek, V. Guruswami and S. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions", *IEEE Trans. Inf. Theory*, 2017.

[11] C. Schoeny, A. Wachter-Zeh, R. Gabrys, and E. Yaakobi, "Codes for correcting a burst of deletions or insertions," *IEEE Trans. Inf. Theory*, vol. 63, no. 4, pp. 1971–1985, 2016.

[12] L. Cheng, T. G. Swart, H. C. Ferreira, and K. A. S. Abdel-Ghaffar, "Codes for correcting three or more adjacent deletions or insertions, in *Proc. IEEE Int. Symp. on Inform. Theory,* 2014, pp. 1246–1250.

[13] S. K. Hanna and S. E. Rouayheb, "Correcting bursty and localized deletions using guess & check codes", in *Proc. Fifty-Fifth Annual Allerton Conference*, 2017, pp. 9–16.

[14] A. Vahid, G. Mappouras, D. J. Sorin, R. Calderbank, "Correcting two deletions and insertions in racetrack memory", 2017, arXiv:1701.06478.

[15] N. G. De Bruijn, "A combinatorial problem", *Koninklijke Nederlandse Akademie v. Wetenschappen*, vol. 49, pp. 758–764, 1946.