Codes Correcting Limited-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*—In this work, we study limited-shift errors in racetrack memories and propose several schemes to combat these errors. There are two kinds of shift errors, namely *under-shift errors*, that can be modeled as sticky-insertions and *limited-over-shift errors*, that can be modeled as bursts of deletions of limited length. One approach to tackle the problem is to use deletion/sticky-insertion-correcting codes. Using this approach, we present a new family of asymptotically optimal codes that correct multiple bursts of deletions of limited length and any number of sticky insertions. We then study another approach that takes advantage of the special features of racetrack memories and the ability to add extra heads for redundancy. Here, we propose how to place the extra heads and construct codes to correct these shift errors.**

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 [1], [2]. 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 [3]. Each head thus reads only a block of consecutive cells which is called a *data segment*.

A shift operation might not work perfectly. When the cells are not shifted (or under-shifted), the same cell is read again in the same head. This event causes a repetition (or sticky-insertion) error. When the cells are shifted by more than a single cell location (or over-shifted), one cell or a block of cells is not read in each head. This event causes a single deletion or a burst of consecutive deletions. We note that the maximal number of consecutive deletions is limited or in other words, the burst of consecutive deletions has limited length. An experimental result shows that the cells are over-shifted by at most two locations with extremely high probability [3]. In this paper, we study both kinds of errors and refer to these errors as *limited-shift errors*.

Since limited-shift errors can be modeled as sticky-insertions and bursts of consecutive deletions with limited length, sticky-insertion/deletion-correcting codes can be applied to combat these limited-shift errors. Although there are several known *sticky-insertion-correcting codes* [4]–[6], *deletion-correcting codes* [7]–[9], *single-burst-deletion-correcting codes* [10], [11], and *multiple-burst-deletion-correcting codes* [12], there is a lack of knowledge on codes correcting a combination of multiple bursts of deletions and sticky insertions. In this paper, motivated by the special structure of having multiple heads in racetrack memories, we study codes correcting multiple bursts of deletions and sticky insertions. To correct shift errors in racetrack memories with only a single head, Vahid et al. [13] recently studied codes correcting two deletions and insertions.

Another approach to combat limited-shift errors is to leverage the special feature of racetrack memories where it is possible to add some extra heads to read cells. If there is no error, the information read in these extra heads is redundant. However, if there are limited-shift errors, this information is useful to correct these errors. Recently, several schemes have been proposed to leverage this feature [3], [14]–[16] in order to tackle this problem. However, in [14]–[16], each head needs to read all the cells while in this model, each head only needs to read a single data segment. The goal of this paper is to present several schemes to add extra heads and construct codes to correct limited-shift errors.

II. 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 $\mathbb{C}$.

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 the 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:

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

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

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

978-1-5386-4780-6/18/$31.00©2018 IEEE

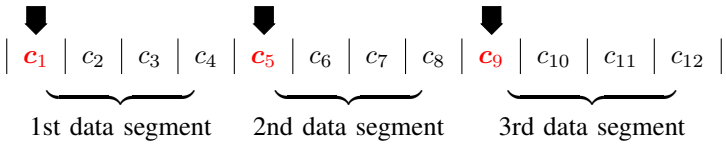

Fig. 1: Racetrack memory with twelve data cells and three heads

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, $\boldsymbol{u} = (u_1, u_2, \ldots, u_\ell) \in \mathbb{F}_q^\ell$.

Under this 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. [3] 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 a bit between two over-shifts[1]. To correct these errors, we use a q-ary deletion-correcting code. In particular, in Section III-A, we construct 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*. The techniques that we use are in the same spirit of the ones in [17], [18].

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 [4], [5] 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 [6].

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. In Section III-B, we 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*. The rates of these codes are shown to be close to optimality.

Furthermore, we also propose several schemes which add extra heads that can be used to correct errors and thus construct codes to correct limited-shift errors in racetrack memory for this model. These results are presented in Section IV. This paper is the first to propose and study these schemes.

[1]Throughout this paper, we always assume that any two bursts of deletions are not adjacent.

A. Our Contributions

Our first contribution is a construction of a q-ary b-limited t_1-burst-deletion-correcting code. Then we analyse the rate of the above code to obtain the following result.

Theorem 1. *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, in Theorem 10, we show how to use the above code to correct a combination of sticky-insertions and bursts of deletions. We obtain the following result.

Theorem 2. *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*

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

It is clear that an upper bound on the maximal rate of the above codes is $1 - \delta$, that is $R_1, R_2 \leqslant 1 - \delta$. Since ϵ is arbitrarily small, when b is small and $q = 2^m$ is big, the rates of the above codes are close to the upper bound. Hence, these codes are asymptotically optimal.

Our next contributions are several schemes to add extra heads and construct codes to correct limited-shift errors. In particular, we obtain the following results.

Theorem 3. *Consider a racetrack memory comprising $n = m \cdot \ell$ bits and m heads which are distributed uniformly. Using $(\lceil \log(b+2) \rceil - 1)$ extra heads, it is possible to construct a code correcting a combination of any number of sticky-insertions and t_1 bursts of deletions whose length is at most b such that its rate satisfies*

$$R_3 \geqslant \frac{m-1}{m} \cdot (1 - \delta - \epsilon)$$

where $t_1 \cdot b = \delta \cdot \ell$ and $1 > \delta, \epsilon > 0$.

Theorem 4. *Consider a racetrack memory comprising m heads which are distributed uniformly. Using m extra heads, we can construct a q-ary code correcting any number of sticky-insertions and any number of deletions where any two deletions are not adjacent such that its rate $R_4 \geqslant \log_q \frac{q-2+\sqrt{q^2-4}}{2} - \epsilon$ for any $\epsilon > 0$.*

III. Constructions without Extra Heads

In this section, we present constructions of codes correcting limited-shift errors without using any extra heads.

A. q-ary b-Limited t_1-Burst-Deletion-Correcting Codes

In this subsection, we assume that only limited-over-shift errors occur in the memory. Our main goal is to study q-ary b-limited t-burst-deletion-correcting codes. Before we construct these codes, we introduce some necessary definitions.

Definition 5.

- *A cyclic sequence $\sigma = (\sigma_1, \ldots, \sigma_\ell)$ is called a* de Bruijn sequence *of length ℓ and strength h over an alphabet of size q if all ℓ possible substrings of length h are distinct. It is known that $\ell \leqslant q^h$.*
- *A q-ary sequence $\pi = (\pi_1, \ldots, \pi_\ell)$ 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.*

Example 1. Let $\ell = 2q$ and $b \leqslant q$ then $\boldsymbol{u} = (0, 1, \ldots, q-1, 0, 1, \ldots, q-1)$ is a b-bounded de Bruijn sequence of length ℓ, strength $h = 1$ over alphabet size q for any $1 \leqslant n \leqslant \ell$ and $\boldsymbol{v} = (0, 0, 1, 1, \ldots, q-1, q-1)$ is a de Bruijn sequence of length ℓ, strength $h = 2$ over alphabet size q. □

We note that de Bruijn sequences have been well studied in the literature and have several known constructions [19], [20]. For the completeness of the results in the paper, we present the following construction of b-bounded de Bruijn sequences.

Construction 6. *Let $\boldsymbol{u} = (u_1, \ldots, u_b)$ be a de Bruijn sequence of strength h, length b over an alphabet of size q. For any n, we construct a q-ary word of length n, $\boldsymbol{c} = (c_1, \ldots, c_n) \in \mathbb{F}_q^n$, such that $c_i = u_j$ if $i \equiv j \pmod b$.*

We can verify that $\boldsymbol{c}$ is a b-bounded de Bruijn sequence of strength h over an alphabet of size q. Although the length of the b-bounded de Bruijn sequence $\boldsymbol{c}$ is not limited, we need to require that $b \leqslant q^h$.

Lemma 7. *Let $\boldsymbol{c}$ be a b-bounded de Bruijn sequence of strength one over an alphabet of size q. Let $\boldsymbol{c}(\Delta^-)$ be the vector obtained from $\boldsymbol{c}$ after deleting all symbols specified by the locations in the set Δ^- such that the maximum number of consecutive deletions is at most $b-1$. The set Δ^- is uniquely determined from $\boldsymbol{c}$ and $\boldsymbol{c}(\Delta^-)$.*

Proof: Assume that $\Delta^- = \{i_1, i_2, \ldots, i_t\}$ where $i_1 < i_2 < \cdots < i_t$ is the set of t locations of all t deleted bits. Since i_1 is the leftmost index that a deletion occurs, $\boldsymbol{c}[1, i_1 - 1] = \boldsymbol{c}(\Delta^-)[1, i_1 - 1]$. Furthermore, since $\boldsymbol{c}$ is a b-bounded de Bruijn sequence of strength one, the symbols $\boldsymbol{c}[i]$ for $i_1 \leqslant i \leqslant i_1 + b - 1$ are distinct. Since $\boldsymbol{c}[i_1]$ is deleted and the maximum number of consecutive deletions is at most $b-1$, $\boldsymbol{c}(\Delta^-)[i_1] \neq \boldsymbol{c}[i_1]$. So, i_1 is the leftmost index that $\boldsymbol{c}$ and $\boldsymbol{c}(\Delta^-)$ differ. Therefore, we can determine i_1 from the two vectors $\boldsymbol{c}$ and $\boldsymbol{c}(\Delta^-)$. Let $\Delta_1^- = \Delta^- \setminus \{i_1\} = \{i_2, \ldots, i_t\}$. To correct the first error, we insert the symbol $\boldsymbol{c}[i_1]$ into the i_1-th position in $\boldsymbol{c}(\Delta^-)$ and obtain the vector $\boldsymbol{c}(\Delta_1^-)$. Similarly, we can determine the value of i_2 as the leftmost index that $\boldsymbol{c}$ and $\boldsymbol{c}(\Delta_1^-)$ differ and correct the deleted symbol. We repeat this procedure until we determine all the locations of the deleted symbols in $\boldsymbol{c}$. Therefore, the set Δ^- is determined and the lemma is proven. ■

We are now ready to present a construction of q-ary b-limited t_1-burst-deletion-correcting codes.

Construction 8. *Let $\pi = (\pi_1, \pi_2, \ldots, \pi_\ell)$ 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 $\boldsymbol{c} = (c_1, c_2, \ldots, c_\ell) \in \mathbb{C}_{q_2}(\ell, t)$, we define $\boldsymbol{f}(\boldsymbol{c}, \pi) = (f_1, f_2, \ldots, f_\ell)$ 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, $\mathbb{C}_q(b, \ell, t) = \{\boldsymbol{f}(\boldsymbol{c}, \pi) : \boldsymbol{c} \in \mathbb{C}_{q_2}(\ell, t)\}$.*

The following theorem proves the correctness of Construction 8.

Theorem 9. *The code $\mathbb{C}_q(b+1, \ell, t)$ from Construction 8 is a q-ary b-limited t_1-burst-deletion-correcting code where $t = t_1 \cdot b$.*

Proof: Let $\boldsymbol{f} = (f_1, \ldots, f_\ell) \in \mathbb{C}_q(b+1, \ell, t)$ be a stored codeword of length ℓ. Assume that $\Delta^- = \{i_1, \ldots, i_t\}$ is the set of locations of all deleted bits such that $i_1 < \cdots < i_t$. Hence, $\boldsymbol{f}(\Delta^-)$ is the received word. Since there are at most t_1 bursts of deletions whose length is at most b, $|\Delta^-| = t \leqslant t_1 \cdot b$ and there are at most b consecutive numbers in Δ^-. From the received vector $\boldsymbol{f}(\Delta^-)$, we can extract the unique pair of vectors $(\pi(\Delta^-), \boldsymbol{c}(\Delta^-))$ where $\boldsymbol{c}(\Delta^-)$ and $\pi(\Delta^-)$ are two vectors obtained from $\boldsymbol{c}$ and π respectively after deleting all the symbols specified by the locations in the set Δ^-. Using Lemma 7, we can determine the set Δ^- from the vectors $\pi(\Delta^-)$ and π since π is a $(b+1)$-bounded de Bruijn sequence of strength one. Moreover, $\mathbb{C}_{q_2}(\ell, t)$ can correct up to t erasures. Hence, using the decoder of $\mathbb{C}_{q_2}(\ell, t)$, we can recover the vector $\boldsymbol{c}$ provided $\boldsymbol{c}(\Delta^-)$ and Δ^-. From the two vectors $\boldsymbol{c}$ and π, we obtain the stored codeword $\boldsymbol{f} = (f_1, \ldots, f_\ell)$ such that $f_i = (\pi_i, c_i)$ for $1 \leqslant i \leqslant \ell$. In conclusion, Theorem 9 is proven and a simple decoding algorithm to recover $\boldsymbol{f}$ follows from the proof. ■

Let

$$R(\mathbb{C}_{q_2}(\ell, t)) = \frac{\log_{q_2} |\mathbb{C}_{q_2}(\ell, t)|}{\ell}$$

denote the rate of the code $\mathbb{C}_{q_2}(\ell, t)$. From Construction 6, there exists a $(b+1)$-bounded de Bruijn sequence π of strength one over an alphabet of size q_1 if $b+1 \leqslant q_1$. From Construction 8, if π exists and there exists a q_2-ary t-erasure-correcting code of length ℓ, then $|\mathbb{C}_q(b+1, \ell, t)| = |\mathbb{C}_{q_2}(\ell, t)|$. Let $q_1 = q/q_2 = b+1$. The rate of the q-ary code $\mathbb{C}_q(b+1, \ell, t)$ is

$$R = \frac{\log_q |\mathbb{C}_q(b+1, \ell, t)|}{\ell} = \frac{\log_q q_2 \cdot \log_{q_2} |\mathbb{C}_{q_2}(\ell, t)|}{\ell}$$
$$= \log_q(q/q_1) \cdot R(\mathbb{C}_{q_2}(\ell, t)) = (1 - \log_q(b+1)) \cdot R(\mathbb{C}_{q_2}(\ell, t)).$$

Moreover, it is known that for any $0 < \epsilon, \delta < 1$, there exists a q_2-ary code of length ℓ correcting $t = \delta \cdot \ell$ erasures with rate $R(\mathbb{C}_{q_2}(\ell, t)) \geqslant 1 - \delta - \epsilon$. Therefore, Theorem 1 holds.

B. q-ary b-Limited t_1-Burst-Deletion t_2-Sticky-Insertion-Correcting Codes

The main goal in this subsection is to study error-correcting-codes to combat both kinds of errors: under-shift and limited-over-shift errors. In particular, we investigate q-ary b-limited t_1-burst-deletion t_2-sticky-insertion-correcting codes. Our first contribution in this subsection is the following theorem.

Theorem 10. *The code $\mathbb{C}_q(b+2, \ell, t)$ from Construction 8 is a q-ary b-limited t_1-burst-deletion t_2-sticky-insertion-correcting code for any integer t_2 and $t = t_1 \cdot b$.*

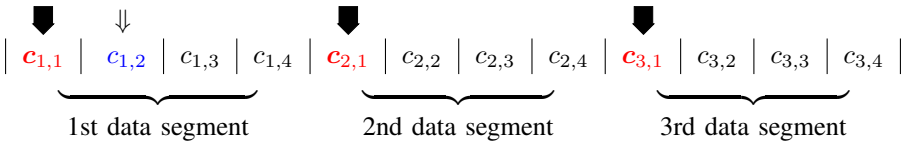

Fig. 2: Racetrack memory with three main heads and one extra head

To prove Theorem 10, we simply mimic the proof of Theorem 9 with the following lemma replacing the role of Lemma 7. We skip the details due to the lack of space.

Lemma 11. *Let π be a $(b+2)$-bounded de Bruijn sequence of strength one over an alphabet of size q. Let $\Delta^- = \{i_1, \ldots, i_t\}$ be such that $i_1 < \cdots < i_t$ and there are at most b consecutive numbers. Let $\Delta^+ = \{j_1, \ldots, j_{t_2}\}$ be such that $j_1 \leqslant \cdots \leqslant j_{t_2}$. Assume that we receive the word $\pi(\Delta^-, \Delta^+)$ which is a vector obtained from π after deleting the symbols π_i for all $i \in \Delta^-$ and repeating the symbols π_j for all $j \in \Delta^+$. Then, we can determine the two sets Δ^- and Δ^+.*

Proof: Since π is a $(b+2)$-bounded de Bruijn sequence of strength one, $\pi[i] \neq \pi[j]$ for all i, j satisfy $|i-j| \leqslant b+1$. Finally, using Lemma 7, we can determine the set Δ^-. ∎

The size and the rate of the code $\mathbb{C}_q(b+2, \ell, t)$ have been analysed in Subsection III-A. Therefore, Theorem 2 holds.

Remark 1.

- In classical insertion/deletion channel, we always know the difference between the numbers of deletions and insertions. However, in racetrack memories, this information may not come for free. We note that in our construction, we can decode without knowing this information.
- Recall that $b = 2$ is the most practical case according to experimental results [3]. In this case, a 2^m-ary two-limited-burst-deletion t_2-sticky-insertion-correcting code $\mathbb{C}_{2^m}(4, \ell, t)$ has rate $R(\mathbb{C}_{2^m}(4, \ell, t)) \geqslant \frac{m-2}{m} \cdot (1 - \delta - \epsilon)$, for any $0 < \epsilon, \delta < 1$ and $2t_1 = \delta \cdot \ell$.

IV. Constructions with Extra Heads

In this section, we also consider a racetrack memory which comprises of n cells and m heads, and each head reads the information in a data segment of length ℓ such that $n = m \cdot \ell$. However, the goal of this section is to present schemes that add extra heads for error-correction and construct codes to correct limited-shift errors in a racetrack memory. We assume that both kinds of errors, under-shift and limited-over-shift, could occur but when a limited-over-shift occurs, at most b consecutive bits are not read. Our contributions in this section are the following two schemes.

A. Adding Extra Heads in One Data Segment

In this subsection, we present a scheme to add one extra head and construct a code correcting any number of under-shifts and t_1 two-limited-over-shifts. We describe the scheme in details as follows.

Scheme I: Add one extra head right after the first head as in Fig. 2. In case there is no error, the output matrix from the $m+1$ heads is:

$$\begin{pmatrix} \boldsymbol{c}^1 \\ \boldsymbol{c}^* \\ \boldsymbol{c}^2 \\ \vdots \\ \boldsymbol{c}^m \end{pmatrix} = \begin{pmatrix} c_{1,1} & c_{1,2} & \cdots & c_{1,\ell-1} & c_{1,\ell} \\ c_{1,2} & c_{1,3} & \cdots & c_{1,\ell} & c_{2,1} \\ c_{2,1} & c_{2,2} & \cdots & c_{2,\ell-1} & c_{2,\ell} \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ c_{m,1} & c_{m,2} & \cdots & c_{m,\ell-1} & c_{m,\ell} \end{pmatrix}$$

where $\boldsymbol{c}^i = (c_{i,1}, \ldots, c_{i,\ell})$ is the output in the i-th main head for all $1 \leqslant i \leqslant m$ and $\boldsymbol{c}^* = (c_{1,2}, \ldots, c_{1,\ell}, c_{2,1})$ is the output in the extra head. In case there are sticky-insertions and bursts of deletions, we need to construct a code that corrects these errors. We now present a code correcting any number of under-shift errors and t_1 two-limited-over-shift errors in this model.

Construction 12. *Let $\boldsymbol{c}^1 = (c_{1,1}, c_{1,2}, \ldots, c_{1,\ell}, c_{2,1})$ be a four-bounded de Bruijn sequence of strength two. Let $\mathbb{C}_{q_2}(\ell, t)$ be a q_2-ary t-erasure-correcting code of length ℓ, where $t = 2 \cdot t_1$ and $q_2 = 2^{m-1}$. For each $1 \leqslant j \leqslant \ell$, let $\Phi_{m-1}(\widehat{\boldsymbol{c}}_j) = v_j \in \mathbb{F}_{q_2}$ where $\widehat{\boldsymbol{c}}_j = (c_{2,j}, \ldots, c_{m,j})$. Let $(\boldsymbol{c}^2, \ldots, \boldsymbol{c}^m)$ be such that $\boldsymbol{v} = (v_1, \ldots, v_\ell) \in \mathbb{C}_{q_2}(\ell, t) \subset \mathbb{F}_{q_2}^\ell$. Then, it is possible to correct any number of under-shifts and t_1 two-limited-over-shifts.*

To show the correctness of Construction 12, we first show that using the outputs from the first head $\boldsymbol{c}^1$ and the extra head $\boldsymbol{c}^*$, we can determine the locations of all sticky-insertions and deletions provided that there are at most two consecutive deletions. For $1 \leqslant j \leqslant \ell$, let $\Phi_2(\boldsymbol{c}^1[j], \boldsymbol{c}^*[j]) = \pi_j \in \mathbb{F}_4$. Since $(c_{1,1}, c_{1,2}, \ldots, c_{1,\ell})$ is a four-bounded de Bruijn sequence of strength two, the vector $\pi = (\pi_1, \ldots, \pi_\ell)$ is a four-bounded de Bruijn sequence of strength one. From Lemma 11, we can determine the set of locations of all errors. Now, we can correct any number of sticky-insertions and t_1 bursts of deletions of length at most two since $\mathbb{C}_{q_2}(\ell, t)$ can correct $t = 2 \cdot t_1$ erasures.

Since we know the rate of the code $\mathbb{C}_{q_2}(\ell, t)$, we can obtain the following result.

Theorem 13. *Using one extra head, we can construct a code correcting any number of sticky-insertions and t_1 bursts of deletions of length at most two such that its rate $R \geqslant \frac{m-1}{m} \cdot (1 - \delta - \epsilon)$ where $2 \cdot t_1 = \delta \cdot \ell$ and any $\epsilon > 0$.*

It is also possible to generalize this result to obtain the result in Theorem 3. Due to the lack of space, we skip the details in this version.

B. Adding Extra Heads in All Data Segments

In this subsection, we present a scheme that adds m extra heads and construct a code correcting any number of sticky-insertions and any number of deletions such that any two deletions are not adjacent.

Scheme II: Recall that a racetrack memory has m heads distributed uniformly. Right after each head, we add an extra head as in Figure 3. In total, we add m extra heads.

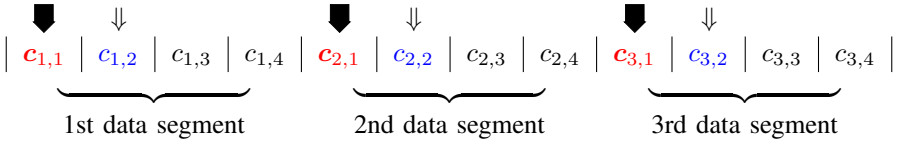

Fig. 3: Racetrack memory with three main heads and three extra heads

In the model without extra heads, we can view the received word as a q-ary word $\boldsymbol{u} = (u_1, \ldots, u_\ell)$ of length ℓ where $q = 2^m$. Hence, in this model, we can view the received word as a symbol-pair word $\pi(\boldsymbol{u}) = ((u_1, u_2), (u_2, u_3), \ldots, (u_{\ell-1}, u_\ell))$. We note that this model is the same as the symbol-pair read channel [21]–[23]. However, this paper is the first to study codes correcting sticky-insertions and deletions in symbol-pair read channel. We now present a construction of a code correcting any number of sticky-insertions and any number of deletions such that any two deletions are not adjacent.

Construction 14. *Let $\mathcal{I} = \{(a, a), (a, b, a, b) : 0 \leqslant a, b \leqslant q - 1\}$. A word $\boldsymbol{u}$ is said to avoid the set $\mathcal{I}$ if it does not contain any pattern from $\mathcal{I}$ as a substring. Let $\mathbb{C}_q(\ell, \mathcal{I})$ be a set of all q-ary words of length ℓ avoiding the set $\mathcal{I}$.*

Theorem 15. *Under the set up in Scheme II, the code $\mathbb{C}_q(\ell, \mathcal{I})$ can correct any number of sticky-insertions and any number of deletions such that any two deletions are not adjacent.*

Proof: Let $\boldsymbol{u} \in \mathbb{C}_q(\ell, \mathcal{I})$ be a stored codeword. To show the correctness of Construction 14, we show how to recover the word $\boldsymbol{u}$. Consider a shifting operation at position i. After reading a pair (u_i, u_{i+1}), if there is no error, that is all cells are shifted successfully by one position, then the output is the next pair (u_{i+1}, u_{i+2}). However, if an under-shift occurs, that is, all cells are not shifted successfully, then the output is the same pair (u_i, u_{i+1}). Otherwise, if an over-shift occurs, assuming that all cells are over-shifted by a single location, then the output is the pair (u_{i+2}, u_{i+3}). Since $\boldsymbol{u} \in \mathbb{C}_q(\ell, \mathcal{I})$, $\boldsymbol{u}$ avoids all patterns (a, a) for all $0 \leqslant a \leqslant q - 1$. Hence, $u_i \neq u_{i+1}$ and $u_{i+1} \neq u_{i+2}$. Thus, we can easily detect whether an error occurs. Furthermore, $(u_i, u_{i+1}) \neq (u_{i+2}, u_{i+3})$ since $\boldsymbol{u}$ avoids all patterns (a, b, a, b) for all $0 \leqslant a, b \leqslant q - 1$. So, we can determine whether the error is an under-shift or an over-shift. In both cases, we receive all bits and know exactly what are the bits. Therefore, we can recover the stored codeword $\boldsymbol{u}$. ∎

A simple decoding algorithm can be found in the above proof while an encoding algorithm has been studied recently [24]. In [24] , Chee et al. showed that the rate of the above code $R_4 \geqslant \log_q \frac{q-2+\sqrt{q^2-4}}{2} - \epsilon$ for any $\epsilon > 0$. Therefore, Theorem 4 holds.

Acknowledgment

The research of Y. M. Chee was supported in part by the Singapore Ministry of Education under Grants MOE2017-T3-1-007 and MOE2015-T2-2-086. The research of H. M. Kiah was supported in part by the Singapore Ministry of Education under Research Grants MOE2015-T2-2-086 and MOE2016-T1-001-156. The research of Alexander Vardy was supported in part by the National Science Foundation under Grants CCF-1405119 and CCF-1719139. The research of Eitan Yaakobi was supported in part by the Israel Science Foundation (ISF) under Grant 1624/14.

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," *Design Automation Conference (DAC)*, 2013, pp.1–6.
[3] 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.
[4] 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.
[5] 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.
[6] 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.
[7] 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.
[8] A. S. J. Helberg and H. C. Ferreira, "On multiple insertion/deletion correcting codes," *IEEE Trans. Inform. Theory*, vol. 48, pp. 305–308, 2002.
[9] J. Brakensiek, V. Guruswami and S. Zbarsky, "Efficient low-redundancy codes for correcting multiple deletions", *IEEE Trans. Inf. Theory*, 2017.
[10] 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.
[11] 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.
[12] 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.
[13] A. Vahid, G. Mappouras, D. J. Sorin, R. Calderbank, "Correcting two deletions and insertions in racetrack memory", 2017, arXiv:1701.06478.
[14] Y. M. Chee, H. M. Kiah, A. Vardy, V. K. Vu, and E. Yaakobi, "Coding for racetrack memories", in *Proc. IEEE Int. Symp. on Inform. Theory,* 2017, pp. 619–623.
[15] Y. M. Chee, H. M. Kiah, A. Vardy, V. K. Vu, and E. Yaakobi, "Codes correcting position errors in racetrack memories", in *Proc. IEEE Inform. Theory Workshop,* 2017, pp. 161–165.
[16] Y. M. Chee, H. M. Kiah, A. Vardy, V. K. Vu, and E. Yaakobi, "Coding for racetrack memories", *IEEE Trans. Inform. Theory,* accepted in Feb 2017.
[17] B. Haeupler and A. Shahrasbi, "Synchronization strings: codes for insertions and deletions approaching the singleton bound", in *Proc. ACM SIGACT Symp. Theory of Computing, STOC 2017*, pp. 33–46.
[18] Y. M. Chee, S. Ling, T. T. Nguyen, V. K. Vu, and H. Wei, "Permutation codes correcting a single burst deletion II: Stable deletions", *Proc. IEEE Int. Symp. on Inform. Theory,* 2017, pp. 2688–2692.
[19] N. G. De Bruijn, "A combinatorial problem", *Koninklijke Nederlandse Akademie v. Wetenschappen* vol. 49, pp. 758-764, 1946.
[20] C. J. Mitchell, T. Etzion, and K. G. Paterson, "A method for constructing decodable de Bruijn sequences", *IEEE Trans. Inform. Theory*, vol. 42, no. 5, pp. 1472–1478, 1996.
[21] Y. Cassuto and M. Blaum, "Codes for symbol-pair read channels", *IEEE Trans. Inform. Theory*, vol. 57, no. 12, pp. 8011–8020, 2011.
[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. Inform. Theory*, vol. 59, no. 11, pp. 7259–7267, 2013.
[23] E. Yaakobi, J. Bruck, and P. H. Siegel, "Constructions and decoding of cyclic codes over b-symbol read channels", *IEEE Trans. Inform. Theory*, vol. 62, no. 4, pp. 1541–1551, 2016.
[24] Y. M. Chee, J. Chrisnata, H. M. Kiah, and T. T. Nguyen, "Efficient encoding/decoding of irreducible words for codes correcting tandem duplications", to present in *IEEE Int. Symp. on Inform. Theory,* Vail, CO, USA, Jun. 2018, arXiv:1801.02310v1.