IET Computers & Digital Techniques

Research Article

IET Journals
The Institution of Engineering and Technology

Design of reliable storage and compute systems with lightweight group testing based non-binary error correction codes

ISSN 1751-8601
Received on 15th March 2018
Accepted on 30th October 2018
doi: 10.1049/iet-cdt.2018.5008
www.ietdl.org

Lake Bu[1] ✉, *Mark G. Karpovsky*[2], *Michel A. Kinsy*[1]

[1]*Adaptive and Secure Computing Systems Laboratory, Department of Electrical and Computer Engineering, Boston University, Boston, USA*
[2]*Reliable Computing Laboratory, Department of Electrical and Computer Engineering, Boston University, Boston, USA*
✉ *E-mail: bulake@bu.edu*

Abstract: In this study, the authors propose a new group testing based (GTB) error control codes (ECCs) approach for improving the reliability of memory structures in computing systems. Compared with conventional single- and double-bit error correcting codes, the GTB codes provide higher reliability at the multi-byte error correction granularity. The proposed codes are cost-efficient in their encoding and decoding procedures. Instead of requiring multiplication or inversion over Galois finite field like most multi-byte ECC schemes, the proposed technique only involves bitwise XOR operations, therefore, significantly reducing the computation complexity and latency. For instance, to correct m errors in a Q-ary codeword of length N, where $Q \geq 2$, the compute complexity is mere $O(mN\log Q)$. The GTB codes trade redundancy for encoding and decoding simplicity, and are able to achieve better code rate than other ECCs of the same trade-off. The proposed GTB codes lend themselves well to designs with high reliability and low computation complexity requirements, such as storage systems with strong fault tolerance, or compute systems with straggler tolerance, and so on.

1 Introduction

The very-large-scale integration industry has been growing exponentially in the last two decades. The semiconductor products now have much higher integration and speed by the increasing density of transistors on chip and clock frequency. Their sizes are reducing and the performance is still improving. However, the growth of integration and speed in electronic components will cause instability in compute and storage systems, which leads to the increase of probability of errors. Thus, it is crucial to provide higher reliability and faster correction with relatively low overhead to those systems.

To equip the electronic systems with reliability on data level, error control codes (ECCs) are commonly used, which are capable of recovering the system from random errors [1]. Even the widely-adopted reliability techniques such as duplication for single error detection and triplication for single error correction are variations of ECCs. For memories, bit-error correcting codes are mostly adapted, among which single- and double-bit ECCs are the majority. However, nowadays most memories are byte-organised or word-organised. Meaning each byte contains b bits, and a single error can affect the whole byte. Moreover, for large-capacity and high-speed memories, due to their high-density nature, they can be very vulnerable to the impact of particles [2]. It is not rare that even multiple b-bit bytes can be distorted in this case [3, 4]. In addition, in distributed systems, errors appear in the form of stragglers, where multiple data blocks sized 32-, 64-bit, or larger can be missing. Bit-level error correction will thus be insufficient.

In the domain of byte-error correction, Reed–Solomon (RS) codes and non-binary Hamming codes are known for having the smallest code redundancy (minimal number of redundant bytes). However, their decoding complexity is relatively high by requiring multiplications and inversions over finite fields.

Generally speaking, the decoding complexity of a byte-level (or non-binary) Q-ary ECC is determined by three factors: the error correcting capability m (number of erroneous bytes to be located and corrected), the size of each codeword byte b, where usually for a Q-ary code, $b = \log Q$, and N the number of bytes in each codeword. For most byte-level ECCs, the encoding and decoding complexity grows proportionally to b^2. Particularly for RS codes, when $b \geq 32$, the complexity cannot be neglected and the implementation becomes impractical in both hardware and software [5, 6].

Another drawback of conventional non-binary ECCs is the decoding latency. For example, the decoding of RS codes based on Berlekamp–Massey algorithm and Chien search requires $(2m^2 + 9m + 3 + N)$ clock cycles for m-error correction in a N-byte RS codeword [7–9]. Even the single-byte ECCs such as non-binary Hamming codes cost considerable latency on finite field multiplications to correct single byte errors. As modern systems and smart devices are all working at high speed, such latency is hardly acceptable.

Hence, interleaved codes are invented as an alternative solution. Interleaved Hamming codes [10] can be used for single-byte error correction, and interleaved orthogonal latin square codes (OLSCs) [11, 12] usually for multi-byte error correction. Other types of interleaved codes also exist. Briefly, they all are to interleave b binary ECC codewords to form one non-binary codeword consisting of b-bit bytes. In the decoding procedure, error correction is performed on each of the binary codewords first. Then they are de-interleaved to restore the correct non-binary codeword. The decoding of this technique is in low complexity by avoiding multiplications and inversions over GF(2^b) fields. However, the simplicity is achieved at the cost of b identical decoders for each code. When b is large, the hardware cost on decoders is non-negligible.

In response to these disadvantages, we propose a new class of group testing based (GTB) ECC. The major properties of the proposed codes are:

1. The GTB codes function over GF(Q) field, where $Q = 2^b$ and $b \geq 1$;
2. High reliability: GTB codes feature multi-byte error correction capability;
3. Low encoding and decoding complexity: bitwise XORs and integer additions are the only operations performed in the encoding and decoding procedures of GTB codes. No finite field multiplications or inversions are needed;
4. Parallelism: the encoding and decoding procedures of GTB codes can be parallelised to 1-step encoding, 1-step syndrome computing, and 1-step error correction;

5. Code rate: GTB codes trade redundancy for computation complexity. Nevertheless, they still achieve better code rates than ECCs of the same trade-off, such as OLSCs.

In short, GTB codes are a new class of ECC that is high in error tolerance, low in computation complexity and latency. The price paid for these advantages is smaller code rate than that of Hamming and RS codes. However, it is managed in an acceptable range.

The rest of the paper is organised as follows. Section 2 is on the construction of the check matrices for GTB codes. Section 3 introduces the mathematical definition and optimal parameters of GTB codes. The encoding and decoding algorithms of GTB codes are explained in Sections 4 and 5. In Section 6, a new concept of 1-step threshold decoding is introduced to simplify and generalise the procedure. In Section 7, single- and double-byte error correction, the two most common and important cases from the practical point of view, are discussed.

In terms of evaluation, the GTB codes are compared in various aspects with the classical codes, such as non-binary Hamming, RS, and interleaved codes. Section 8 is on the code rate comparison, and Section 9 investigates their potential beyond the designed error detection and correction capability. Section 10 is on the implementation of the GTB decoder, as well as the hardware comparison with other ECCs' decoders. In this section, we show that the hardware cost of GTB decoder is linear to its codeword size and error tolerance capacity.

Section 11 illustrates several possible applications of the GTB codes. Finally, Section 12 concludes the paper.

2 Check matrices of the GTB codes

Before we formally define the GTB codes, the construction of their check matrices is discussed in this section. Most ECCs' definitions are closely related to their check matrices, which can contain critical information on length, code rate, and error tolerance capability of a code. GTB codes are no exception.

More importantly, for both binary and non-binary GTB codes, the check matrices remain in the binary form. The simplicity of check matrices ensures the low decoding complexity for GTB codes, even under non-binary scenarios. In our proposal, the superimposed codes will be used for the construction of check matrices for GTB codes.

2.1 Definition of superimposed codes

First, the original definition of superimposed codes is given below.

Definition 1: Let $M_{i,j} \in \{0, 1\}$ be the element in row i and column j in a binary matrix $\boldsymbol{M}$ of size $A \times N$. The set of columns of $\boldsymbol{M}$ is called an m-superimposed code, if for any set T of up to m columns and any single column $h \notin T$, there exists a row k in the matrix $\boldsymbol{M}$, for which $M_{k,h} = 1$ for column h, and $M_{k,j} = 0$ for all $j \in T$ [13, 14].

The above property is called zero-false-drop of order m. Next, two properties of superimposed codes are defined.

Definition 2: For any two A-bit binary vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, we say that $\boldsymbol{u}$ covers $\boldsymbol{v}$ if $\boldsymbol{u} \cdot \boldsymbol{v} = \boldsymbol{u}$, where $\cdot$ denotes the dot product of the two vectors. An $(A \times N)$ binary matrix $\boldsymbol{M}$ is m-disjunct if the bitwise OR of any set of no more than m columns does not cover any other single column that is not in the set. The columns of an m-disjunct matrix compose an m-superimposed code [15].

For example, the columns of the following matrix are 1-disjunct superimposed code

$$\boldsymbol{M} = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 \end{vmatrix}$$

It follows that, the superimposed codes are uniquely decodable of order m, as defined below.

Definition 3: An $(A \times N)$ binary matrix $\boldsymbol{M}$ is m-separable, if the bitwise OR of any up to m columns are all different.

It has been proved that the matrix $\boldsymbol{M}$ constructed from superimposed codes is not only m-separable, but also m-disjunct. Due to its zero-false-drop and uniquely decodable properties, superimposed codes are often used in non-adaptive group testings. For an m-superimposed code matrix sized $A \times N$, A is the number of tests needed to locate m particular (erroneous, defective, or poisonous etc.) objects, and N the total number of objects in the tests. The rows of $\boldsymbol{M}$ are test patterns, and the 1's in each column indicate which tests an object will participate.

A positive test result is represented as 1 (meaning at least one or more targeted objects located in the test), and negative result as 0. Therefore, the A-bit test syndrome is essentially the bitwise OR of all the m columns corresponding to the m targeted objects. By Definition 3, for each possible combination of m (or less) columns, the resulted A-bit test syndrome is unique and thus the targeted objects are decodable.

2.2 Lower and upper bounds on minimal length of superimposed codes

The bounds on lengths of superimposed codes are actually the bounds on the minimum numbers of tests for non-adaptive group testing. Adaptive testing may result in a smaller number of tests (length of the syndrome) than non-adaptive testing. However, the decoding for the corresponding GTB codes will be much more complicated. According to D'yachkov and Singleton's research [16], the lower and upper bounds on the shortest superimposed codes' length $A_{\min}$ are very tight by a factor of $1/\log m$

$$\Omega\left(\frac{m^2 \log N}{\log m}\right) \leq A_{\min} \leq O(m^2 \log N), \tag{1}$$

where $A_{\min}$ is the length of the shortest m-superimposed codes with N codewords.

It is notable that when $m = 1$, (1) becomes the most commonly seen case where $A_{\min} = \lceil \log N \rceil$.

2.3 Notations

Before describing the construction of superimposed codes, which will be used for the construction of check matrices for GTB codes, we introduce the following notations:

- n_q: the total number of bytes in codewords of a q-ary $(n_q, k_q, d_q)_q$ code C_q;
- k_q: the number of information bytes in C_q;
- $r_q = n_q - k_q$: the number of redundant bytes in C_q;
- d_q: the minimum Hamming distance in C_q;
- A: the length of codewords in a superimposed code C_{SI};
- $N = |C_{\text{SI}}|$: the number of codewords in C_{SI};
- $\boldsymbol{M}$: the binary matrix of size $A \times N$ whose columns are codewords in C_{SI} as its columns;
- l: the maximum Hamming weight of rows in $\boldsymbol{M}$;
- d_{SI}: the distance between codewords of C_{SI};
- m: the number of errors to be corrected by a GTB code.

2.4 Construction of superimposed codes

Construction 1: Let C_q be a q-ary ($q = p^s$ is a power of prime and $p \neq 2$) conventional error correcting code with parameters $(n_q, k_q, d_q)_q$. Each byte of C_q in GF (q) can be represented by a q-bit binary vector with Hamming weight one. A superimposed code C_{SI} can be constructed by substituting every q-ary digit of codewords

in C_q by its corresponding binary vector. The resulting m-superimposed code C_{SI} will have the following parameters [17]:

$$\begin{aligned} A &= qn_q, \\ N &= q^{k_q}, \\ l &= q^{k_q-1}, \\ d_{\mathrm{SI}} &= 2d_q, \\ m &= \left\lfloor \frac{n_q-1}{n_q-d_q} \right\rfloor. \end{aligned} \tag{2}$$

If C_q is a maximal-distance separable (MDS) q-nary code, for which $d_q = r_q + 1$ and $n_q \le q$, such as RS codes, then m can be written as [15]

$$m = \left\lfloor \frac{n_q-1}{k_q-1} \right\rfloor. \tag{3}$$

The codewords of the m-superimposed code C_{SI} form the columns of an $A \times N$ matrix $\boldsymbol{M}$. In every row there are exactly l 1's, and every column exactly n_q 1's.

Example 1: An extended ternary RS code has its parameters $(n_q, k_q, d_q)_q = (3, 2, 2)_3$. The codewords are

$$\begin{aligned} C_q = \{&(0,0,0),(0,1,2),(0,2,1),(1,0,2),\\ &(1,1,1),(1,2,0),(2,0,1),(2,1,0),(2,2,2)\}. \end{aligned}$$

Suppose 0, 1, 2 are substituted by 3-bit binary vectors (100), (010), (001), respectively. According to (2), $N = 9$, $A = 9$, $d_{\mathrm{SI}} = 4$, and $m = 2$. The code C_{SI} is a 2-superimposed code.

Then the code C_{SI} consisting of the following $N = 9$ codewords can be listed as the columns of a 9×9 matrix $\boldsymbol{M}$

$$\boldsymbol{M} = \left|\begin{array}{ccccccccc} 1&1&1&0&0&0&0&0&0\\ 0&0&0&1&1&1&0&0&0\\ 0&0&0&0&0&0&1&1&1\\ \hline 1&0&0&1&0&0&1&0&0\\ 0&1&0&0&1&0&0&1&0\\ 0&0&1&0&0&1&0&0&1\\ \hline 1&0&0&0&0&1&0&1&0\\ 0&0&1&0&1&0&1&0&0\\ 0&1&0&1&0&0&0&0&1 \end{array}\right|$$

The superimposed codeword length A generated from Construction 1 is usually not the minimal according to the lower bound in Section 2.2. Other constructions can achieve smaller A (e.g. the approaches presented in [13, 14]), but will result in a higher decoding complexity for GTB codes.

As Section 5 will show, with the low-density binary matrices built for GTB codes by Construction 1, only a small number of binary operations are needed for multi-byte error correction. The construction makes GTB codes low computation complexity and high speed in contrary to most popular non-binary ECCs.

3 GTB codes

The definition of GTB codes is based on the binary check matrices $\boldsymbol{M}$ constructed by superimposed codes in Section 2. In order to facilitate the introduction of GTB codes and their properties, we define the notations below:

- $M_{i,*}$: the ith row of $\boldsymbol{M}$;
- $M_{*,j}$: the jth column of $\boldsymbol{M}$;
- N: the length of a $(N, K, D)_Q$ GTB codeword V;
- K: the number of information bytes in a GTB codeword;
- R: the number of redundant bytes in a GTB codeword;
- b: the number of bits in each byte of a Q-ary GTB codeword, where $Q = 2^b$ and $b \ge 1$;
- λ: the maximal number of 1's in common between any two columns in $\boldsymbol{M}$ [18]. Then we have $\left|M_{*,i} \cdot M_{*,j}\right| \le \lambda$ for $i, j \in \{1, 2, \ldots, N\}$, $i \ne j$, where $\cdot$ is bitwise AND;
- $\oplus$: the bitwise XOR operator;
- Block: $\boldsymbol{M}$ can be partitioned into n_q sub-matrices, such that each one has exactly q rows. Each sub-matrix is called a block B_t – a set of q rows where $B_t = \{M_{i,*} \mid \lceil i/q \rceil = t\}$, $t \in \{1, 2, \ldots, n_q\}$. Each row in a block has exactly l 1's and each column has exactly one 1.

3.1 Definition of GTB codes

We now define GTB codes based on the binary check matrix constructed in Section 2.

Definition 4: Let $\boldsymbol{M}$ be an $A \times N$ binary matrix whose columns are the codewords of an m-superimposed code constructed in Construction 1. V is called a GTB code if

$$V = \{v \mid \boldsymbol{M} \cdot v = 0\},\ v \in GF(Q^N), \tag{4}$$

where v is any codeword of the GTB code V.

Remark 1: Since the check matrix $\boldsymbol{M}$ of a Q-ary GTB code is a binary matrix regardless of the value of Q, the syndrome computation $\boldsymbol{M} \cdot \boldsymbol{v}$ will be as simple as a number of additions in GF(2^b), namely bitwise XORs (denoted by $\oplus$). No multiplications or inversions in finite filed GF(2^b) are involved in the decoding procedures. Moreover, the number of XORs performed in $\boldsymbol{M} \cdot \boldsymbol{v}$ is determined by the number of 1's in $\boldsymbol{M}$, which is always a low-density matrix by Construction 1.

For the optimal construction of GTB codes to be introduced in the following sub-section, the fraction of 1's in $\boldsymbol{M}$ can be as low as $N^{-0.5}$.

3.2 Optimal construction of GTB codes

As discussed in Remark 1, the complexity of GTB codes' decoding procedure is closely related to the number of 1's in $\boldsymbol{M}$, which is also the number of bitwise XORs required to compute the syndrome.

Definition 5: A GTB code is optimal if the number of 1's in its check matrix $\boldsymbol{M}$ is minimum. Since there are A rows in $\boldsymbol{M}$ and each row has l 1's, the total number of 1's in $\boldsymbol{M}$ can be calculated by Al.

Construction 1 shows that $\boldsymbol{M}$ can be constructed from any conventional q-ary error correcting codes. One of the most popular multi-byte ECCs is the Bose–Chaudhuri–Hocquenghem (BCH) code and it owns many fine properties. RS code as another choice is a special case of BCH code. We compare the two codes in terms of Al to suggest a proper choice for the optimal GTB codes.

Given N and m, the properties of BCH code [19] are

$$\begin{aligned} n_q &= k_q + r_q; \\ k_q &= \log_q N; \\ r_q &= (d_q - 2)i + 1, \end{aligned} \tag{5}$$

where $n_q = q^i - 1$. With (2), the parameters of the BCH code-based GTB code are

$$\begin{aligned} m &= \left\lfloor \frac{n_q-1}{n_q-d_q} \right\rfloor; \\ l &= q^{k_q-1}; \\ A &= qn_q = (k_q + (d_q-2)i + 1)q; \\ Al &= (k_q + (d_q-2)i + 1)q^{k_q}. \end{aligned} \tag{6}$$

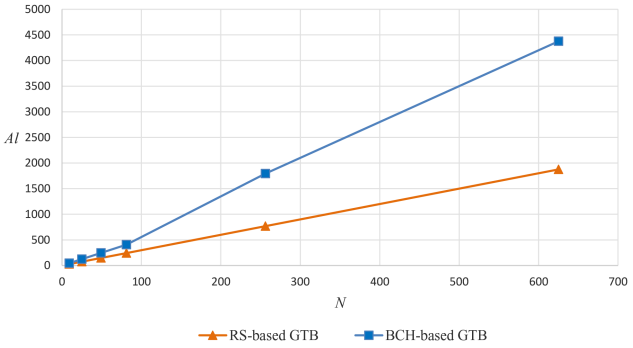

Fig. 1 *Al comparison between GTB codes constructed from RS codes and BCH codes under the same N and m. As N grows larger, the RS code-based GTB has larger advantage over the BCH-based*

RS code is a special case of BCH codes when $i = 1$. The RS code is a MDS code where $d_q = r_q + 1$. Therefore, for RS codes the above equations can be rewritten as

$$\begin{aligned} n_q &= k_q + r_q; \\ k_q &= \log_q N; \\ r_q &= r_q = d_q - 1. \end{aligned} \tag{7}$$

With (2), the parameters of the RS code-based GTB code are

$$\begin{aligned} m &= \left\lfloor \frac{n_q - 1}{k_q - 1} \right\rfloor; \\ l &= q^{k_q - 1}; \\ A &= qn_q = (k_q + d_q - 1)q; \\ Al &= (k_q + d_q - 1)q^{k_q}. \end{aligned} \tag{8}$$

By (6) and (8), to correct the same number of errors in codewords of the same length, RS code-based GTB codes have smaller A and Al the BCH code-based, as shown in Fig. 1. Therefore, we will use RS as the base code to construct the check matrices for GTB codes.

Remark 2: Since RS codes are MDS codes, it has maximal d_q. Under the same n_q, it is able to generate larger m than other non-MDS codes by [20]

$$m = \left\lfloor \frac{n_q - 1}{n_q - d_q} \right\rfloor.$$

We now specify the parameters of the RS codes achieving the minimal Al.

Theorem 1: Given N the length of GTB codewords and m the number of errors to be corrected, the optimal Q-ary GTB code and its check matrix with minimal Al can be constructed by the RS code with the following parameters:

$$(n_q, k_q, d_q)_q = (m + 1, 2, m)_q; \tag{9}$$

Then by (2), we have

$$\begin{aligned} N &= q^2; \\ A &= q(m + 1); \\ l &= q. \end{aligned} \tag{10}$$

The minimal syndrome computation complexity is then

$$Al = (m + 1)q^2 = (m + 1)N. \tag{11}$$

The proof of Theorem 1 is given in Appendix 1.

There are also cases when a GTB code needs to be designed by given K and m. In this scenario based on Theorem 1, we are able to compute the optimal q to construct its check matrix.

Corollary 1: If K and m are given, then the optimal q_{opt} minimising Al will be [21]

$$q_{\text{opt}} = \left\lceil \frac{(m + 1) + \sqrt{(m + 1)^2 + 4(K - m)}}{2} \right\rceil_{p^s}, \tag{12}$$

where p^s stands for the nearest power of prime that is larger than the value inside $\lceil\ \rceil$.

The proof of the corollary is given in Appendix 2.

3.3 Parameters of optimal GTB codes

With the RS parameters presented above, we show that the GTB codes have the following properties.

Theorem 2: If a Q-ary GTB code V of length N is defined by $V = \{v | \boldsymbol{M} \cdot v = 0\}$, $v \in \text{GF}(Q^N)$, where $\boldsymbol{M}$ is generated by Construction 1 and Theorem 1, then V has the following parameters:

$$(N, K, D)_Q = (q^2, q^2 - q(m + 1) + m, 2m + 2)_Q,$$

where N is the length of any GTB codeword, K the number of information bytes, and D the distance of the code.

Such a code is able to detect up to $2m + 1$ errors and correct up to m errors.

The proof of Theorem 2 is given in Appendix 3.

Remark 3: By Theorem 2, there are two advantages of GTBs over most other ECCs:

1. The check matrix and optimal parameters of an $(N, K, D)_Q$ m-error correcting GTB codes do not depend on Q. It indicates that the increase of Q will not negatively affect the decoding complexity or code rate of a GTB code;
2. The distance of an m-error correcting GTB code is $D = 2m + 2$, while for most other ECCs it is $2m + 1$. The larger distance provided by GTB code enables it to have higher error detection capability.

Corollary 2: The rate of GTB codes is

$$\frac{K}{N} = 1 - \frac{A - m}{N} = 1 - \frac{(m + 1)q - m}{q^2}. \tag{13}$$

It is obvious that when

$$N = q^2 \to \infty \quad \text{and} \quad \frac{m}{q} \to 0,$$

we have

$$\frac{K}{N} \to 1.$$

Remark 4: We note that Construction 1 also provides the trade-offs between syndrome computation complexity (Al) and code rates. For a GTB code based on a $(n_q, k_q, d_q)_q = (k_q + m - 1, k_q, m)_q$ RS code, if $k_q > 2$, then this GTB code will have parameters $(N, K, D)_Q = (q^{k_q}, q^{k_q} - (k_q + m - 1)q + k_q + m - 2, 2m + 2)_Q$. Although its Al will be larger than what has been given in Theorem 1, its code rate will be better.

4 Encoding

Since the rows in the check matrix of a GTB code are not all linear independent, to acquire the encoding matrix, the check matrix $\boldsymbol{M}$ needs to be transformed into the row canonical form. The row canonical form and the standard encoding matrix are very similar,

except the order of a few columns. Similar to its decoding, the encoding of GTB codes requires no finite field computations but bitwise XORs.

Definition 6: A matrix $\boldsymbol{M}$ is said to be in row canonical form or reduced row echelon form [22] if the following conditions hold:

- All zero rows, if any, are at the bottom of the matrix;
- Each first non-zero entry in a row is to the right of the first non-zero entry in the preceding row;
- Each pivot (the first non-zero entry) is equal to 1;
- Each pivot is the only non-zero entry in its column.

Corollary 3: If an $A \times N$ matrix $\boldsymbol{M}$ is a binary check matrix generated from an m-superimposed code for a $(N, K, D)_Q$ GTB code V, then it can be transformed to a row canonical form M', with $A - m$ non-zero rows. In M' all the $A - m$ columns with the pivots represent the locations of redundant bytes, and the remaining $N - A + m$ columns the information bytes.

We now show an example of GTB code encoding, which is very similar to conventional binary ECC encoding.

Example 2: A GTB code V over GF(Q), $Q = 2^3$, has $N = 9$ information digits and is able to correct single-digit errors. According to Theorem 1 its check matrix $\boldsymbol{M}$ can be constructed from the $(n_q, k_q, d_q)_q = (2, 2, 1)_3$ RS code. Then code V has the parameters of $(N, K, D)_Q = (9, 4, 4)_{2^3}$ and for this code

$$\boldsymbol{M} = \begin{vmatrix} 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \end{vmatrix}$$

By Corollary 3, after transforming $\boldsymbol{M}$ into row canonical form, we have

$$M' = \begin{vmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{vmatrix}$$

M' indicates that in a codeword (message) $\boldsymbol{v} = (v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9)$, the redundant bytes are v_1, v_2, v_3, v_4, v_7, and the information byte v_5, v_6, v_8, v_9.

If $v_5 = (011)$, $v_6 = (101)$, $v_8 = (110)$, $v_9 = (111)$, then the codeword can be encoded by M' as

$$\begin{aligned} &v_1 = v_5 \oplus v_6 \oplus v_8 \oplus v_9 = (111); \\ &v_2 = v_5 \oplus v_8 = (101);\ v_3 = v_6 \oplus v_9 = (010); \\ &v_4 = v_5 \oplus v_6 = (110);\ v_7 = v_8 \oplus v_9 = (001). \end{aligned}$$

So $\boldsymbol{v} = (111, 101, 010, 110, 011, 101, 001, 110, 111)$.

5 Decoding: error locating and correction

The decoding procedure of GTB codes consists of three parts: syndrome computation, error locating, and error correction.

5.1 Syndrome computation for GTB codes

Definition 7: For a GTB code $V = \{v | \boldsymbol{M} \cdot v = 0\}$ over GF(Q^N), where $\boldsymbol{M}$ is an $A \times N$ binary matrix, if a codeword $\boldsymbol{v} = (v_1, v_2, \ldots, v_N)$ is distorted by an error e to $\tilde{v} = \boldsymbol{v} \oplus e$, $\tilde{v}, \boldsymbol{v}, e \in GF(Q)$, then the syndrome

$$S = \boldsymbol{M} \cdot \tilde{v} = \boldsymbol{M} \cdot e.$$

There are A digits in $S = (S(1), S(2), \ldots, S(A))$, and $S(i) \in GF(Q)$. Then the support of syndrome $S_{\text{sup}} = (S_{\text{sup}}(1), S_{\text{sup}}(2), \ldots, S_{\text{sup}}(A))$ is defined as

$$S_{\text{sup}}(i) = \begin{cases} 0, & S(i) = 0; \\ 1, & S(i) \neq 0. \end{cases}$$

The A-bit binary syndrome S_{sup} is used for error locating, and the A-digit Q-ary syndrome S for error correction.

The syndrome computation of GTB codes only involve bitwise XORs, and its complexity can be minimised as shown in Theorem 1.

5.2 Error locating

The GTB codes' m-error locating algorithm is generated by the following algorithm.

Algorithm 1: Let the columns of an $A \times N$ binary matrix $\boldsymbol{M}$ be the set of all codewords of an m-superimposed code constructed by Construction 2.1 from a $(n_q, k_q, d_d)_q$ RS code C_q. Let $S_{\text{sup}} = (S_{\text{sup}}(1), S_{\text{sup}}(1), \ldots, S_{\text{sup}}(A))$ be the A-bit binary vector representing the support of syndrome $S = \boldsymbol{M} \cdot \tilde{v} = \boldsymbol{M} \cdot e$, and $\tilde{v} = (\tilde{v}_1, \tilde{v}_2, \ldots, \tilde{v}_N)$, $\tilde{v}_j = v_j \oplus e_j$. Let $\boldsymbol{u} = (u_1, u_2, \ldots, u_N)$ be the N-bit error locating vector for GTB codes such that

$$u_j = \left(\sum_{\{i | M_{i,j} = 1\}} S_{\text{sup}}(i) = m + 1 \right) ? 1 : 0.$$

If $u_j = 1$, then $\tilde{v}_j = v_j \oplus e_j$, and $|e_j| \neq 0$.

Note: through this paper, $c = (a = b) ? 1 : 0$ denotes

$$c = \begin{cases} 1, & \text{if } a = b; \\ 0, & \text{otherwise}. \end{cases}$$

The proof of Algorithm 1 is given in Appendix 4.

Remark 5: It is notable that if $M_{i,*} \cdot e = 0$ and $e \neq 0$, then the corresponding $S_{\text{sup}}(i) = 0$. This can result in $u_j = 0$ even if $e_j \neq 0$. We call the mis-detection error masking, as multiple errors can mask each other from being revealed, whose probability is Q^{-1}. Therefore, Algorithm 1 provides an error locating and correction probability close to 1 as Q is relatively large. This algorithm can be modified to achieve error locating and correction probability of 1 in important practical cases, such as $m = 1$ and $m = 2$ in Section 7.

In Section 6, we develop a *1-step threshold decoding* algorithm, which for any m it provides a trade-off between the error correction probability and redundancy R. We show that by increasing R to at most a factor of 2, the error correction probability achieves exactly 1.

5.3 Error correction

For any located m-error, it always can be corrected by the algorithm presented by the following algorithm.

Algorithm 2: Let a codeword $\boldsymbol{v}$ over GF(Q) be distorted by an m-digit error to $\tilde{v} = v \oplus e$. e is located by the error locating vector $\boldsymbol{u} = (u_1, u_2, \ldots, u_j, \ldots, u_N)$, where if $u_j = 1$, $|e_j| \neq 0$. Also let S be the A-digit syndrome where $S = \boldsymbol{M} \cdot \tilde{v} = \boldsymbol{M} \cdot e$ and $S = (S(1), S(2), \ldots, S(i), \ldots, S(A))$, $S(i) \in GF(Q)$. For any non-zero error digit e_j in e, there must exist at least one row $M_{i,*}$ in $\boldsymbol{M}$, such that

$$\sum_{\{j | M_{i,j} = 1\}} u_j = 1.$$

Then $S(i) = M_{i,*} \cdot \tilde{v} = M_{i,*} \cdot e = e_j$. So $\tilde{v}_j$ can be corrected by

$$v_j = \tilde{v}_j \oplus e_j = \tilde{v}_j \oplus S(i).$$

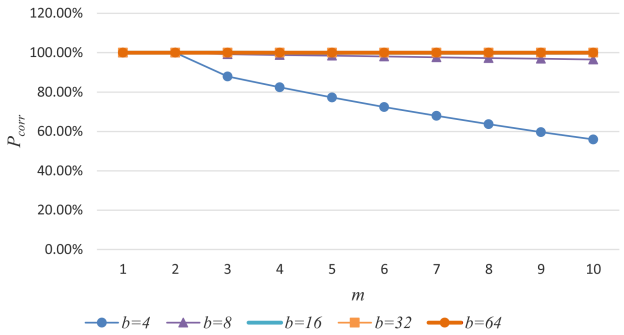

Fig. 2 *As b increases over 8, the error correction probability P_{corr} always remains close to 1. The increase of m has little negative impact when $b \geq 16$*

The proof of Algorithm 2 is given in Appendix 5.

To summarise, the decoding procedure consists of the following fine-grained steps: calculating the syndrome, converting it to the support of the syndrome, error locating, finding the $m \times m$ identity sub-matrix corresponding to the m-digit error, and error correction. With parallelisation, the procedure can be combined to two steps: error locating and error correction.

Example 3: A Q-ary GTB code has $Q = 2^3 = 8$ and parameters $(N, K, D)_Q = (9, 2, 6)_8$. The distorted codeword $\tilde{v} = (1, 2, 3, 6, 6, 2, 2, 3, 1)_8$.

Since $D = 6$, we can do double-error correction with this code. First by Theorem 3.1, we have

$$q = \sqrt{N} = 3;\ d_q = m = \frac{D-2}{2} = 2;$$
$$k_q = 2;\ n_q = m + 1 = 3.$$

Thus, the check matrix can be constructed by a $(n_q, k_q, d_q)_q = (3, 2, 2)_3$ RS code

$$\boldsymbol{M} = \left|\begin{array}{ccccccccc} 1&1&1&0&0&0&0&0&0\\ 0&0&0&1&1&1&0&0&0\\ 0&0&0&0&0&0&1&1&1\\ \hline 1&0&0&1&0&0&1&0&0\\ 0&1&0&0&1&0&0&1&0\\ 0&0&1&0&0&1&0&0&1\\ \hline 1&0&0&0&0&1&0&1&0\\ 0&0&1&0&1&0&1&0&0\\ 0&1&0&1&0&0&0&0&1 \end{array}\right|$$

The syndrome and the support of the syndrome are

$$S = \boldsymbol{M} \cdot \tilde{v} = (0, 2, 0, 5, 7, 0, 0, 7, 5);$$
$$S_{\text{sup}} = (0, 1, 0, 1, 1, 0, 0, 1, 1).$$

By Theorem 5.1, the double error is located as

$$\boldsymbol{u} = (0, 0, 0, 1, 1, 0, 0, 0, 0).$$

Knowing that $e = (0, 0, 0, e_4, e_5, 0, 0, 0, 0)$, $(e_4 \neq 0,\ e_5 \neq 0)$, by Theorem 5.2 the identity sub-matrix corresponding to e_4 and e_5 is

$$\begin{vmatrix} M_{8,4} & M_{8,5} \\ M_{9,4} & M_{9,5} \end{vmatrix} = \begin{vmatrix} 0 & 1 \\ 1 & 0 \end{vmatrix}.$$

Hence

$$v_4 = \tilde{v}_4 \oplus S(8) = 3;$$
$$v_5 = \tilde{v}_4 \oplus S(9) = 1;$$
$$v = (1, 2, 3, 3, 1, 2, 2, 3, 1)_8.$$

6 Generalised 1-step threshold decoding

The error correcting algorithm introduced in the previous section requires that for any $e_j \in e, e_j \neq 0$, there are n_q digits of the syndrome affected by it. However, if there exists one or more than one row $M_{i,*}$ in $\boldsymbol{M}$ such that $S(i) = M_{i,*} \cdot \tilde{v} = M_{i,*} \cdot e = e_j \oplus e_k \oplus \cdots \oplus e_z = 0$, and $e_j, e_k, \ldots, e_z \in e$, then $u_j = 0$ and so e_j cannot be located. We will refer to this case as error masking in position i.

For a Q-ary GTB code, $Q = 2^b$, it is obvious that the upper bound on the probability of having at least one error masking is Q^{-1}. Also for any e_j in an m-digit error e, it will at most affect $n_q = m + 1$ digits of the syndrome, out of which there can be at most $m - 1$ error maskings since $\lambda = 1$. Therefore, we have the probability P_{corr} of no error masking for $e_j \neq 0$ lower bounded by

$$P_{\text{corr}} \geq (1 - Q^{-1})^{m-1}. \tag{14}$$

Given m, the larger the Q (or $b = \log Q$) is, the greater the P_{corr} is as shown in Fig. 2. It is expected that as m grows, the error correcting probability decreases. However, if b is large enough, (14) can still provide a satisfying P_{corr}. Without loss of generality, we examine the error correction probability under $1 \leq m \leq 10$ and $b = \{4, 8, 16, 32, 64\}$ in GTB code $(N, K, D)_Q = (625, 360, 22)_Q$.

This means for a GTB codeword consisting of 16-bit bytes or larger, it can always locate and correct up to m errors with probability close to 100%.

Moreover, P_{corr} can be enhanced by increasing the redundancy of the GTB code, as shown in the following algorithm.

Algorithm 3: Let C_{RS} be a $(n_q, k_q, d_q)_q = (m + 1 + \Delta, 2, m + \Delta)_q$ RS code and $\boldsymbol{M}$ is the check matrix of a GTB code V_Δ of $(N, K, D)_Q = (q^2, q^2 - q(m + 1 + \Delta) + m + \Delta, 2(m + \Delta) + 2)_Q$, then for V_Δ we have

$$P_{\text{corr}} \geq (1 - Q^{-1})^{m-1-\Delta}. \tag{15}$$

And

$$P_{\text{corr}} \to 1 \quad \text{when } \Delta \to (m - 1).$$

So the error locating vector $\boldsymbol{u} = (u_1, u_2, \ldots, u_N)$ can be re-written as

$$u_j = \left(\sum_{\{i \mid M_{i,j} = 1\}} S_{\text{sup}}(i) \geq m + 1 \right) ?1 : 0.$$

If $u_j = 1$, then $\tilde{v}_j = v_j \oplus e_j$, and $e_j \neq 0$.

The proof of Algorithm 3 is given in Appendix 6.

As an example, Algorithm 3 we can improve the error correcting probability in Fig. 2 through increasing Δ. Taking $m = 10$ as an example, if $0 \leq \Delta \leq 9$, the updated P_{corr} are graphed in Fig. 3.

It can be found that under different b, the P_{corr} goes to 1 in different velocity. When $\Delta = m - 1$, no matter what b is, the P_{corr} is always 100%. At this point, the GTB code can be decoded by simple majority voting through threshold gates, and Algorithm 3 becomes 1-step threshold decoding.

However, it is notable that as Δ increases, the code rate decreases. For instance, when $\Delta = m - 1 = 9$ and $P_{\text{corr}} = 1$, the original $(N, K, D)_Q = (625, 360, 22)_Q$ code becomes $(625, 144, 40)_Q$.

Example 4: By (14), a $(N, K, D)_Q = (25, 12, 6)_{2^8}$ GTB code V_0 is able to correct double errors at a probability of 99.6%. The columns of its check matrix are generated by all the codewords of a $(n_q, k_q, d_q)_q = (3, 2, 2)_5$ RS code (see equation below)

To make it capable of correcting all double errors at a probability 1, we select $\Delta = m - 1 = 1$ and so the new matrix will

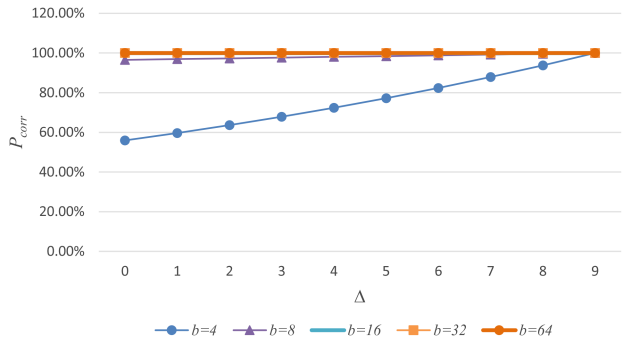

Fig. 3 *Probability of successfully correcting 10 errors in a* $(N, K, D)_Q = (625, 360, 22)_Q$ *GTB code when* Δ *increases*

be generated from the codewords of a $(n_q, k_q, d_q)_q = (4, 2, 3)_5$ RS code (see equation below) By substituting 0, 1, 2, 3, 4 with (10000), (01000), (00100), (00010), (00001), the check matrix $\boldsymbol{M}$ can be constructed. So the parameters of the new GTB code V_1 will be a $(N, K, D)_Q = (25, 8, 10)_{2^8}$ code and it can correct all double errors with probability 100% by the 1-step threshold decoding introduced in Algorithm 3.

Remark 6: For 1-step threshold decoding, if $m = 1$, then by (27) we always have $P_{\text{corr}} = 1$. When $m \geq 2$, by increasing at most $m - 1$ blocks the GTB codes can achieve $P_{\text{corr}} = 1$ at the cost of decreasing the code rate. For this case, the code parameters will change from $(N, K, D)_Q = (q^2, q^2 - 3q + 2, 6)_Q$ to $(q^2, q^2 - 4q + 3, 8)_Q$.

In the following section, we will introduce a procedure to achieve $P_{\text{corr}} = 1$ for $m = 2$ GTB codes without reducing the code rate. For the rest of the paper we always assume $\Delta = 0$.

7 Single- and double-byte error correcting GTB codes

Single and double errors are most commonly seen in error corrections. The GTB codes can always achieve 100% error correcting probability in both cases.

7.1 Single-byte error correction

According to Construction 1, single-byte error correcting GTB codes can be generated from $(n_q, k_q, d_q)_q = (2, 2, 1)_q$ RS codes, which will result in $(N, K, D)_Q = (q^2, q^2 - 2q + 1, 4)_Q$ GTB codes for $m = 1$ with $A \cdot l = 2q^2 = 2N$. A check matrix of this code with $q = 3$ is given in Example 2. For single errors, Algorithms 1 and 2 always detects and corrects the errors with probability of 1.

Another way to generate single-error correcting GTB codes is to construct it based on a binary Hamming check matrix.

Definition 8: A Q-ary GTB code Y with following parameters $(N, K, D)_Q = (N, N - \lceil \log_2(N+1) \rceil, 3)_Q$ is defined by $y \in Y$ if $y = \{y | \boldsymbol{M} \cdot y = 0\}$, where $\boldsymbol{M}$ is a binary Hamming check matrix. If a codeword y is distorted by a single error to $\tilde{y} = y \oplus e$ and the syndrome $S = \boldsymbol{M} \cdot \tilde{y}$, then the support of the syndrome S_{sup} is the error location, and $e = S(i)$, if $S(i) \neq 0$.

From the above definition it is obvious that this Hamming based GTB code Y has better code rate, namely smaller redundancy ($R = \log_2(N+1)$) than that of RS based GTB code V ($R = 2\sqrt{N} - 1$).

However, code Y has larger syndrome computation complexity by

$$Al = \frac{(N+1)}{2} \log_2(N+1);$$

while for a regular GTB code V

$$Al = 2N.$$

7.2 Double-byte error correction

Double-byte errors usually cost much more time and space to be located and corrected than single-byte errors. However, for GTB codes it is still very time and cost efficient to correct double errors.

An example of correcting double error has been given in Example 3. However, if there is an error masking, meaning $e_4 = e_5$, then $u_4 = u_5 = 0$. In this case, the double error cannot be guaranteed to be located and corrected. Therefore, we propose the customised algorithm below to achieve 100% double error correction probability.

Algorithm 4: Let the columns of matrix $\boldsymbol{M}$ in size $A \times N$ be the set of all non-zero codewords of a 2-superimposed code constructed by Construction 1 from a $(n_q, k_q, d_d)_q$ RS code C_q. Let S_{sup} be the A-bit binary vector representing the support of syndrome $S = \boldsymbol{M} \cdot \tilde{v} = \boldsymbol{M} \cdot e$, where e is a double error and causes one error masking in S. Let $w = (w_1, w_2, \ldots, w_N)$ be the error locating vector for GTB codes such that

$$w_j = \sum_{\{i | M_{i,j} = 1\}} S_{\text{sup}}(i). \tag{16}$$

If $w_j = m + 1 = 3$, then $e_j \neq 0$.

If there is no $w_j = m + 1 = 3$, there will be some j such that $w_j = m = 2$ which indicate error location candidates.

Denote $W = \{w_j | w_j = m\}$ as the set of error location candidates. If there is a row $M_{i,*}$ such that

$$(w_j = 2) \wedge (S(i) = 0) = 1, \tag{17}$$

then $e_j = 0$, and the remaining items in set W are error locations.

The proof of Algorithm 4 is given in Appendix 7.

Example 5: A GTB code V with the same parameters and the same legal codeword v as Example 3. It is now distorted by a double error at digits 4 and 5 that causes error masking

$$e = (0, 0, 0, 7, 7, 0, 0, 0, 0).$$

Then

$$S = \boldsymbol{M} \cdot \tilde{v} = (0, 0, 0, 7, 7, 0, 0, 7, 7);$$
$$S_{\text{sup}} = (0, 0, 0, 1, 1, 0, 0, 1, 1);$$
$$w = (1, 2, 1, 2, 2, 1, 2, 1, 1);$$
$$W = \{w_2, w_4, w_5, w_7\}.$$

0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4	4	4	4	4
0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4
0	4	3	2	1	4	3	2	1	0	3	2	1	0	4	2	1	0	4	3	1	0	4	3	2

0	0	0	0	0	1	1	1	1	1	2	2	2	2	2	3	3	3	3	3	4	4	4	4	4
0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4	0	1	2	3	4
0	1	2	3	4	2	3	4	0	1	4	0	1	2	3	1	2	3	4	0	3	4	0	1	2
0	3	1	4	2	2	0	3	1	4	4	2	0	3	1	1	4	2	0	3	3	1	4	2	0

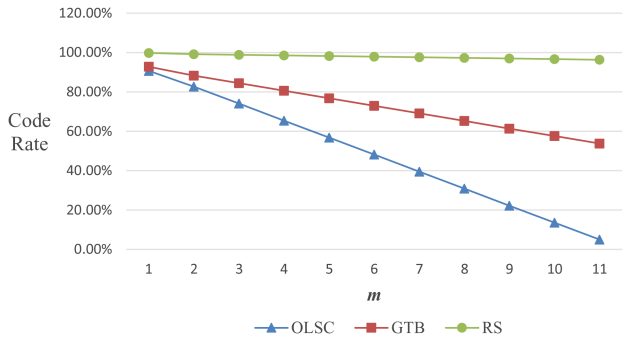

Fig. 4 *Code rate comparison among RS, interleaved OLSC, and GTB codes as m increases*

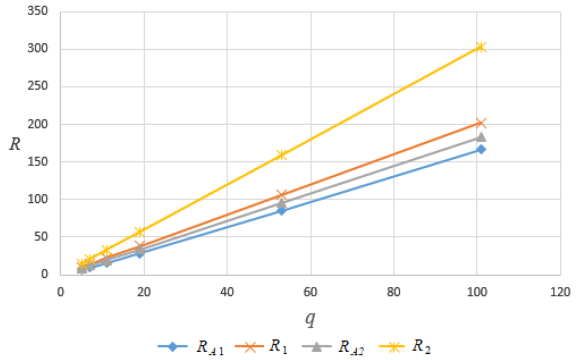

Fig. 5 *Comparison between the actual GTB code's redundancy R_1 and R_2 for $m = 1$ and $m = 2$, and the theoretical lower bound R_{A1} and R_2. Particularly, R_1 is very close to R_{A1}*

The sub-matrix of $\boldsymbol{M}$ consisting of columns indexed by W and the syndrome vector are

$$\left[\begin{array}{cccc} \boxed{1} & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & \boxed{1} \\ \hline 0 & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ \hline 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & 1 & 0 & 0 \end{array}\right] \quad S = \left[\begin{array}{c} \boxed{0} \\ 0 \\ \boxed{0} \\ \hline 7 \\ 7 \\ 0 \\ \hline 0 \\ 7 \\ 7 \end{array}\right]$$

It is not hard to find out that for $M_{1,*}$, $(w_2 = 2) \wedge (S(1) = 0) = 1$, and for $M_{3,*}$, $(w_7 = 2) \wedge (S(3) = 0) = 1$. Then $e_2 = e_7 = 0$.

Therefore, the double error is $e = (0, 0, 0, e_4, e_5, 0, 0, 0, 0)$ where $e_4 = 7$ and $e_5 = 7$. Similarly to Example 4, they can be corrected by Theorem 2.

8 Code rate comparison

Sections 8, 9, and 10 are evaluation sections. The GTB codes are assessed by comparing with conventional popular ECCs such as RS code known by its minimum redundancy, and interleaved OLSC known by its low decoding complexity. The comparisons are made in the important aspect of ECCs: code rate, error detection and correction capability, and hardware cost.

8.1 Code rate comparison with RS and interleaved OLSC codes

Code rate, calculated by K/N, is a metric often used for evaluating the transmission efficiency of an ECC.

By Theorem 1, given N and m, the redundancy of a GTB code is

$$\begin{aligned} q_{\mathrm{GTB}} &= \sqrt{N}; \\ R_{\mathrm{GTB}} &= (m+1)q_{\mathrm{GTB}} - m. \end{aligned} \tag{18}$$

With the same N and m, for interleaved OLSCs the redundancy is

$$\begin{aligned} q_{\mathrm{OLSC}} &= \sqrt{K}; \\ R_{\mathrm{OLSC}} &= 2mq_{\mathrm{OLSC}}. \end{aligned} \tag{19}$$

For RS codes

$$R_{\mathrm{RS}} = 2m. \tag{20}$$

It is obvious that RS codes always achieve the best code rate. GTB codes have better code rate than interleaved OLSCs. Without loss of generality, we examine the code rates of the three ECCs under $1 \geq m \leq 11$ with fixed codeword length $N = 625$, as shown in Fig. 4.

As expected, RS codes as MDS codes always have the best code rate over all others. When m is relatively small, the code rates of GTB and OLSCs codes are similar. As m grows larger, GTB codes have increasingly better rate than OLSCs. When $m = 1$, both code rates of GTB and OLSC are around 90%. When $m = 11$, GTB codes is still able to have the rate at 53.8%, while OLSCs only in its 10% at 5.0%.

8.2 Code rate comparison with the low-density codes' lower bound

The check matrix by Construction 1 indicates that the GTB code also belongs to the family of low-density codes. We now examine its performance versus the theoretical code rate lower bound of low-density codes.

For a class of error locating $Q-$ary codes with row density q^{-1} in the binary check matrix (the fraction of 1's in a row), denote the number of total error locations as L_e, the minimum number of bits in S_{sup} as R, and the probability of one or more errors revealed in one non-zero syndrome bit as r_e. For example, for $m = 1$, $r_e = q/(q^2+1) \simeq q^{-1}$, and for $m = 2$

$$r_e = \frac{q + (q^2 - q)q}{1 + q^2 + \binom{q^2}{2}} \simeq \frac{2}{q+1}.$$

Then we have the following lower bound for R for any low-density $Q-$ary code

$$R \geq \frac{\lceil \log_2 L_e \rceil}{H(r_e)}, \tag{21}$$

where $H()$ is the binary entropy function.

Fig. 5 shows the comparison made at $m = 1$ and $m = 2$ between GTB code rate and the theoretical lower bound.

9 Potentials of GTB codes' error detection and correction capability

For any m-error correction ECC, it is able to correct up to m and detect up to $2m$ random errors. However, it is also known that for such an ECC, at some probability it has certain potential to correct and detect errors beyond the m and $2m$ limitation. This potential is determined by the weight distribution and uniqueness of the syndromes. In this section, we compare such potentials among the GTB and interleaved OLSCs because of their similar weight distributions.

9.1 $2m + 1$ error detection probability comparison

Denoting p as the probability of a byte being distorted, the number of codewords with Hamming weight i as A_i, the more precise error detection probability using codeword weight distribution is [23]

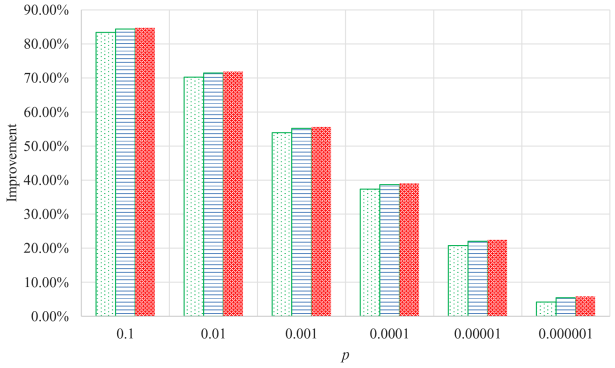

Fig. 6 *Error detection improvement of GTB codes over OLSCs under the same N and m. For every p the three bars are, respectively, Q = 2, 4, and 8. Both codes are attempting to detect 2m + 1 errors*

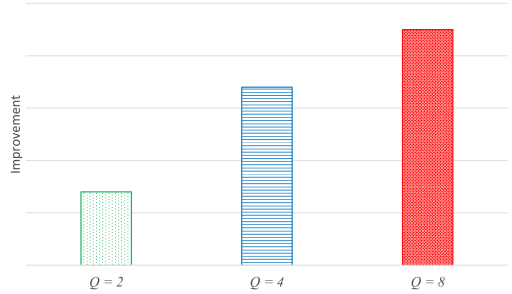

Fig. 7 *Zoom-in of Fig. 6, which shows the trend of improvement with the increase of Q under all p*

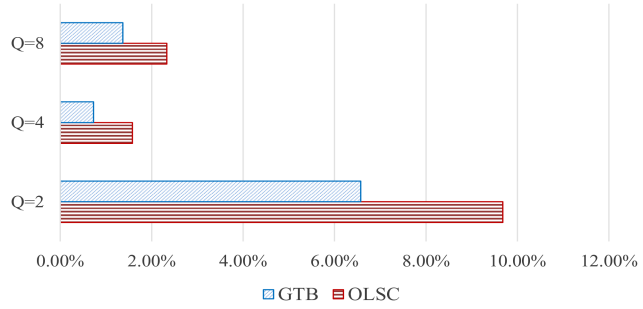

Fig. 8 *Although this figure does not reveal any trend or error mis-correction as Q increases, it does show that GTB codes have less missed errors in all cases*

$$P_{\text{det}} = 1 - \sum_{i=1}^{N} A_i p^i (1 - (Q-1)p)^{N-i}. \tag{22}$$

Under the same m, it is fair to compare among codes of similar length. If the code length differs largely, then the weight distribution will also differ largely. Therefore, under the same N, m, and p, we select GTB and OLSCs for comparison. The error detection potential of GTB codes over OLSCs is defined as follows:

$$\text{Improvement} = \frac{P_{\text{GTB}} - P_{\text{OLSC}}}{P_{\text{OLSC}}}. \tag{23}$$

Fig. 6 shows the improvement between the error detection potential P_{GTB} of GTB codes and P_{OLSC} of OLSCs, when both of them attempt to detect beyond $2m$ errors.

The reason why GTB performs drastically better than the OLSCs in its detection beyond $2m$ errors, is because its larger codeword distance under the same N and m. As explained in Remark 3 and proven in Appendix 3, an m-error correcting GTB code has distance $D = 2m + 2$, while most conventional m-error correcting ECCs only have $D = 2m + 1$.

As expected, the larger the bit distortion rate p is, the larger the error detection improvement will be. When $p = 10^{-1}$, GTB codes' error detection probability is as much as over 80% more than OLSCs. Even as p decreases to nearly 0, and both error detection probabilities increase to almost 100%, GTB codes still perform better.

Fig. 7 is a zoom-in of Fig. 6, indicating the larger Q is, the better improvement of error detection potential will be.

9.2 $m + 1$ error correction probability comparison

For a code with distance $D = 2m + 1$ or $D = 2m + 2$, it is able to correct all m-byte errors. However, it usually is also capable of correcting more than m errors with a certain probability. Particularly, for an m-error correcting code, it is very likely to be able to correct most $m + 1$ errors.

As long as the syndromes $S = \boldsymbol{M} \cdot \tilde{v}$ of different error patterns are unique, it is possible to decode those errors, at least theoretically.

Taking the same parameters as in Section 9.1, when both GTB and OLSC have the same N and m, the probability of correcting $m + 1$ errors are all over 90%. To make the comparison more obvious, we compare the error missing probability defined as follows:

$$P_{\text{miss}} = 1 - \frac{\text{number of unique syndromes}}{\text{number of total syndromes}} \tag{24}$$

Fig. 8 compares the error mis-correction probability under the same setting of Fig. 6.

It is also notable that OLSCs only provide error correction on the information bytes, not redundant bytes. In contrary, GTB codes correct errors on both.

10 Hardware implementation and complexity comparison

In Section 5, the algorithms of error locating and correction are introduced in Algorithms 1 and 2. The hardware decoder module consists of five components: syndrome computation, support of syndrome conversion, error locating, finding the identity matrix for error magnitude, and error correction. Since all the five components are combinatorial networks, their latency altogether is negligible.

10.1 GTB codes' decoding complexity estimation

In this subsection, the hardware cost of the decoder is estimated in the number of equivalent 2-input gates.

1. Syndrome computation:

$$S = \boldsymbol{M} \cdot \tilde{v} = \boldsymbol{M} \cdot e;$$

Hardware cost:

$$bA(l-1)$$

2. Support of syndrome conversion:

$$S_{\text{sup}}(i) = \begin{cases} 0, & S(i) = 0 \\ 1, & S(i) \neq 0 \end{cases}$$

Hardware cost:

$$A(b-1)$$

3. Error locating:

$$u_j = \left(\sum_{\{i \mid M_{i,j}=1\}} S_{\text{sup}}(i) = m + 1 \right) ? 1 : 0;$$

Hardware cost:

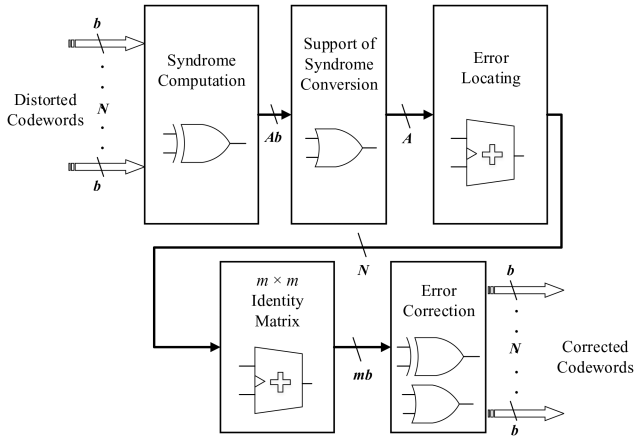

Fig. 9 *Five-stage decoder of GTB codes. The bit width of each bus is labelled*

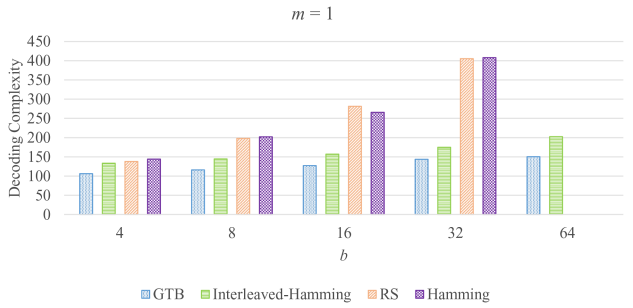

Fig. 10 *Hardware cost comparison among four codes when $m = 1$. As b increases, the GTB codes demonstrate more savings than the non-binary Hamming and RS codes. Another cost-efficient choice for $m = 1$ is the interleaved/parallel Hamming codes. When $b = 64$, RS or non-binary Hamming decoders implementation becomes impractical due to large finite field size*

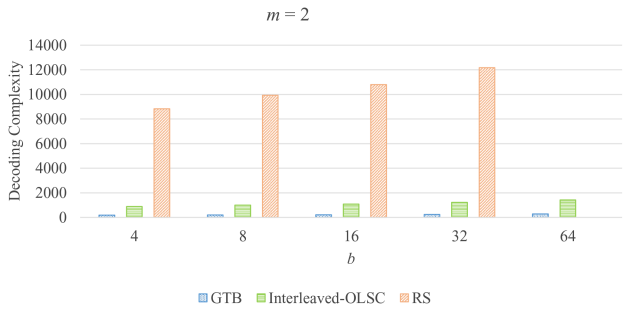

Fig. 11 *Hardware cost of the three codes' decoding procedure for $m = 2$. When $b = 64$, RS or non-binary Hamming decoders implementation becomes impractical due to large finite field size*

$$mN$$

4. Finding the identity matrix for error magnitude:

$$\text{Row}(i) = \left(\sum_{\{j \mid M_{i,j} = 1\}} u_j = 1 \right) ?1:0;$$

Hardware cost:

$$A(2.5l + \log l - 1)$$

5. Error correction:

$$e_j = \bigvee_{\{i \mid M_{i,j} = 1\}} \text{Row}(i) \cdot S(i),$$

where $\vee$ is the bitwise OR for all elements in its subscript.

Hardware cost:

$$Nb(m+1) + b(A+N)$$

Summing all the five hardware cost estimation together and denoting the overall decoding complexity as L, we have

$$\begin{aligned} L &= bA(l-1) + A(b-1) + mN \\ &\quad + A(2.5l + \log l - 1) + Nb(m+1) + b(A+N) \\ &\simeq b(Al + mN) \end{aligned}$$

Since $Al \simeq mN$

$$L \simeq b(Al + mN) \simeq 2mNb\,. \tag{25}$$

Thus from (25), it follows that the decoding complexity of GTB codes is linear to its codeword length N, byte size $b = \log Q$, and the number of errors to be corrected m.

The schematic diagram of this decoding system is shown in Fig. 9.

All five components are simple circuits: bitwise XOR gates, bitwise OR gates, n_q-bit and m-bit adders. Comparing with other non-binary error correcting codes which require finite field multipliers or even inverters, it is of higher cost-efficiency.

Moreover, the circuit in Fig. 9 is a combinational network. It takes almost no time in decoding. In contrast, many other popular ECCs require decoders working in serial under a relatively large latency, which is proportional to the codeword length N and number of errors to be corrected m.

10.2 Hardware decoder module comparison

In this subsection, we verify the hardware cost of GTB codes by FPGA implementation.

Since single and double errors are most common cases in hardware distortions [24], the comparison is made among decoders of $m = 1$, and $m = 2$, with codeword length $b = 512$ bits.

10.2.1 Single-byte error decoder comparison: The major competitors of GTB codes when $m = 1$ are non-binary Hamming codes, RS codes [25], and interleaved/parallel binary Hamming codes [10, 26].

For the experiment of protecting a 512-bit data vector, we select the parameters $m = 1$, $b = \{4, 8, 16, 32, 64\}$, $K = 512/b$ for GTB codes. The decoders of four different codes are implemented for comparison on a Xilinx Virtex4 XC4VFX60 FPGA board. The decoding complexity is the sum of both decoder's and redundancy's hardware costs in terms of CLBs on FPGA.

From Fig. 10, the GTB decoding costs the least amount of FPGA resources, and then the interleaved/parallel Hamming codes. Their decoding complexity is almost the same. GTB codes achieve more saving in syndrome computing by the optimal construction in Section 3.2, and interleaved Hamming codes are better in redundancy. Other decoders consume 70–150% more resources than them.

10.2.2 Double-byte error decoding complexity comparison: The major competitors of GTB codes when $m = 2$ are RS codes and interleaved OLSC [27].

Similar as the previous implementation, the codes' parameters are $m = 2$, $b = \{4, 8, 16, 32, 64\}$, $K = 512/b$. The decoders of three different codes are implemented for comparison. Their decoding complexity including both the redundancy and decoder hardware costs are shown in Fig. 11.

When $m = 2$, the interleaved OLSC decoder costs five times more than GTB codes, and RS costs almost 50 times more. As b increases, GTB codes achieve more savings from the RS codes, which need to operate over larger finite fields.

11 Application examples

In this section, we introduce three possible applications of the GTB codes. While reliable memory design is a natural area for GTB

codes, we also propose two novel applications that GTB codes may contribute to.

11.1 GTB codes for memories

The use of GTB codes on error correction for memories is straightforward. As shown in Section 10.2, it is able to achieve a lower decoding complexity at the cost of redundancy (memory storage). Therefore at the design stage, the redundancy equation (18) and decoding complexity estimation equation (25) have to be evaluated and compared with the actual memory parameters plugged in.

11.2 GTB codes for coded computation

Another application for GTB codes is on coded computation which aims to restore missing data or tolerate stragglers. The general assumption is that part (one or a few bytes) of the information is missing, instead of being faulty. Therefore, data regeneration, rather than error correction, becomes the goal. The situation is often seen in a distributed system, where each node is carrying out a share of the task. A few slower nodes (stragglers) can have negative impact on the entire system by making all others wait for them to finish before everyone can proceed [28]. Such a scenario is often seen in heterogeneous clusters [29], storage systems [30], machine learning acceleration [31], and so on.

An ECC capable of data regeneration is called an erasure code. Among most erasure codes, the MDS codes such as RS codes are generally adopted for their minimum redundancy. On the other hand, codes such as OLSC cannot function as erasure codes because of their incapability of error correction on redundant bytes. An erasure ECC with distance D is able to correct up to $\lfloor (D-1)/2 \rfloor$ errors, and regenerate up to $D-1$ missing bytes. Unlike error correction which uses the check matrix to restore the codeword integrity, data regeneration usually leverages the encoding matrix.

In this subsection, we show by an example that the GTB codes can also be used as for coded computation. In this application, GTB codes will require more redundancy than MDS codes, but its data regeneration procedure has considerably less complexity and latency. Similar as other erasure codes, for a $(N, K, D)_Q$ GTB code where $D = 2m + 2$, it is able to regenerate up to $D - 1 = 2m + 1$ missing bytes.

Example 6: A distributed machine learning training system has nine nodes for parallel matrix multiplication. Among the nine nodes there can be no more than 30%stragglers. A straggler-tolerance scheme can be designed by the framework of GTB codes.

First, a $(N, K, D)_Q = (16, 9, 4)_Q$ GTB code is selected, which is able to regenerate up to $D - 1 = 3$ bytes of missing or straggling data. Among the $N = 16$ nodes, 9 of them are the original matrix multiplication nodes, and 7 the redundant nodes to support data regeneration. The decoding matrix can be generated by Construction 1, and its encoding matrix can be derived by Corollary 3 (see equation below) For a codeword $\boldsymbol{v}$ encoded by $\boldsymbol{M}'$ where $\boldsymbol{v} = \{v_1, v_2, \ldots, v_i, \ldots, v_{16}\}$, each v_i stands for the output of a matrix multiplication node. By $\boldsymbol{M}'$ we have nodes {6, 7, 8, 10, 11, 12, 14, 15, 16} as the original matrix multiplication nodes, and the rest can be coded as redundant. Assume that nodes 14, 15, 16 are stragglers in a specific round of computation, and so $\boldsymbol{v} = \{v_1, v_2, \ldots, v_{13}, ?, ?, ?\}$.

By $\boldsymbol{M}'$ we can simply derive that

$$v_{14} = v_2 \oplus v_6 \oplus v_{10};$$
$$v_{15} = v_3 \oplus v_7 \oplus v_{11};$$
$$v_{16} = v_4 \oplus v_8 \oplus v_{12}.$$

Therefore, the 3 bytes of missing data are regenerated.

Remark 7: For a similar straggler-tolerance system built by a RS code, it will have $(N_{\mathrm{RS}}, K_{\mathrm{RS}}, D_{\mathrm{RS}})_Q = (12, 9, 4)_Q$ as the system parameters. Although its required redundant nodes will be less than the GTB-based system (only 3 needed for the RS code, while 7 for the GTB code), its data regeneration has to involve matrix inversion and matrix multiplication in finite fields. As Q can be large in modern distributed systems (32-bit or 64-bit), those operations will have to be carried out in a large field (GF(2^{32}) or GF(2^{64})), resulting in a non-negligible or even impractical computation complexity and delay. As for the GTB codes, since its encoding matrix is binary, only bitwise XOR operations are necessary for data regeneration, making it more applicable to realistic distributed systems.

11.3 GTB codes for group testing with neutralisation

Section 2 introduced the check matrix for GTB codes originated from group testing. A conventional group testing scheme is able to identify up to m targeted objects with a m-disjunct matrix, whose rows are the group test patterns. Each column indexes an object in the group, where the 1's in a column indicate which tests that object is to participate. The OR of the m columns corresponding to the targeted objects will form a syndrome vector. A 1 in the syndrome vector is a positive test result (at least one targeted object has participated in this test), and a 0 means negative (no targeted objects in this test). By decoding the syndrome vector all targeted objects can be identified. This technique is widely adopted in applications such as locating the erroneous items in a system, or identifying the poisonous solutions in a group of chemistry liquid.

However, such scheme is not applicable to the cases with object neutralisation. For example, high level of acid or alkali in solutions can both be considered as harmful (poisonous) to humans. For a conventional group testing designed to identify the harmful solutions in a group, it will return syndrome 0 at the tests which both the acid and alkali solutions participate, since their mix will be non-harmful salt and water.

The conventional group testing techniques are not able to handle such cases when some targeted objects can neutralise each other and result in negative test syndromes. Essentially, this is because the conventional group testing syndromes are computed by bitwise OR of m columns, which has no equivalence to neutralisation. However, the syndrome computation of GTB codes is based on XOR of the columns, enabling it to symbolise the neutralisation phenomenon. We show this new feature by the following example.

Example 7: There are nine solutions indexed by $\{1, 2, \ldots, 9\}$ among which two are harmful to humans. One of the two poisonous solutions is hydrochloric acid (HCl), and another sodium hydroxide (NaOH). However, the indexes of the two poisonous solutions are unknown. A group testing matrix to identify the two harmful solutions is constructed as

$$M' = \begin{array}{|cccccccccccccccc|}
\hline
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\
\hline
1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 0 & 1 & 1 & 1 \\
0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
\hline
\end{array}$$

$$M = \begin{array}{c|ccccccccc} & 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 \\ \hline a & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ b & 0 & 0 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ c & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \\ d & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ e & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\ f & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 1 \\ g & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 \\ h & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 \\ i & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \end{array},$$

where the column indexes of $\{1, 2, \ldots, 9\}$ correspond to the nine solutions. Each row indexed by $\{a, b, \ldots, i\}$ is a group test pattern, where the 1's in a row indicate which solutions are to be mixed for a test.

Suppose after tests $\{a, b, \ldots, i\}$, the syndrome vector is returned as $S = \{0, 0, 0, 1, 1, 0, 0, 1, 1\}$. Therefore, we know that each of the tests in $\{d, e, h, i\}$ has involved at least one poisonous solution.

By the conventional group testing technique, it is not possible to locate the two targeted solutions because there exist no i, j such that

$$M_{*,i} \vee M_{*,j} = S,$$

where $\vee$ is the bitwise OR operator.

However, by GTB decoding Algorithm 4, it can be found that for $i = 4$, $j = 5$, we have

$$M_{*,i} \oplus M_{*,j} = S,$$

where $\oplus$ is the bitwise XOR operator.

Therefore, the two poisonous solutions are identified by their indexes 4 and 5. It can also be deduced that their neutralisation has happened at test a, where the non-harmful salt and water are produced.

Remark 8: For more complicated cases, such as multiple types of neutralisations among multiple objects, non-binary GTB codes can be leveraged to encode each neutralisation type accordingly. However, this is beyond the scope of this paper.

With the proposed scheme, group testing can be made more efficient, that within one group of tests, objects of multiple types can be identified, even if there are neutralisations or maskings among them.

12 Conclusion

As multi-byte errors become more probable with newer and faster storage and compute systems, stronger protection against byte-level distortions is highly demanded. Therefore, we propose a new GTB multi-byte error correcting codes to address this issue. For codewords with large bytes in Galois field $GF(Q)$ where $Q = 2^b$ and $b \geq 1$, the proposed new code's decoding does not require any multiplications or inversions in Galois fields. Instead, only bitwise XORs and integer additions are necessary. The GTB codes achieve much lower encoding and decoding complexity than other known codes such as Hamming and RS codes. Comparing with the competitors of low decoding complexity, such as bit-interleaved codes, GTB codes have the advantage of better code rate.

The check matrices of GTB codes are generated from binary superimposed codes, which enables low-complexity decoding with mere binary or integer operations. The decoding complexity is as low as $O(mNb)$. As $Q = 2^b$ increases, the complexity only increases proportionally to b. In contrast, popular codes relying on finite field operations have complexity proportional to at least b^2. These characters make GTB codes a promising low-cost and high-reliability ECC for the design of reliable systems.

Based on the GTB codes' fast and low-complexity decoding, we suggest that it can serve systems requiring high reliability, low latency, and less demanding in redundancy [32].

13 Acknowledgments

This research is partially supported by the NSF grant nos. CNS-1745808 and CNS-1012910. A bulk of this work was done when Lake Bu was working at the Reliable Computing Laboratory.

14 References

[1] Wang, Z., Karpovsky, M.: 'Reliable and secure memories based on algebraic manipulation detection codes and robust error correction'. Proc. Int. Depend Symp, Barcelona, Spain, 2013

[2] Fujiwara, E.: '*Code design for dependable systems: theory and practical applications*' (John Wiley & Sons, Marblehead, MA, USA, 2006), p. 264

[3] Umanesan, G., Fujiwara, E.: 'A class of codes for correcting single spotty byte errors', *IEICE Trans. Fundam. Electron. IEEE*, 2003, **86**, (3), pp. 704–714

[4] Zhen, W., Karpovsky, M., Kulikowski, K.J.: 'Replacing linear hamming codes by robust nonlinear codes results in a reliability improvement of memories'. IEEE/IFIP Int. Conf. Dependable Systems & Networks, DSN'09, Lisbon, Portugal, 2009

[5] Schifra: 'Schifra Reed-Solomon error correcting code library', 2017. Available at http://www.schifra.com/index.html

[6] Xilinx, Reed-Solomon Decoder v9.0, 2015

[7] Wu, Y.: 'New list decoding algorithms for Reed-Solomon and bch codes'. IEEE Int. Symp. Information Theory, 2007 (ISIT 2007), Nice, France, 2007

[8] Jeng, J., Truong, T.: 'On decoding of both errors and erasures of a Reed-Solomon code using an inverse-free Berlekamp-Massey algorithm', *IEEE Trans. Commun.*, 1999, **47**, (10), pp. 1488–1494

[9] Xilinx, LogiCORE IP Reed-Solomon Decoder, v8.0, ds862 ed., October 19, 2011

[10] Cui, Y., Zhang, X.: 'Research and implementation of interleaving grouping hamming code algorithm'. 2013 IEEE Int. Conf. Signal Processing Communication and Computing (ICSPCC), Kunming, China, 2013, pp. 1–4

[11] Laendner, S., Milenkovic, O.: 'LDPC codes based on Latin squares: cycle structure, stopping set, and trapping set analysis', *IEEE Trans. Commun.*, 2007, **55**, (2), pp. 303–312

[12] Datta, R., Touba, N.A.: 'Generating burst-error correcting codes from orthogonal Latin square codes–a graph theoretic approach'. 2011 IEEE Int. Symp. Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT), Vancouver, Canada, 2011, pp. 367–373

[13] D'yachkov, A.G., Macula, A.J., Rykov, V.V.: 'On optimal parameters of a class of superimposed codes and designs'. IEEE Int. Symp. Information Theory, Cambridge, MA, USA, 1998

[14] D'yachkov, A.G., Macula, A.J., Rykov, V.V.: 'New applications and results of superimposed code theory arising from the potentialities of molecular biology', in '*Numbers, information and complexity*' (Springer, New York, NY, USA, 2000), pp. 265–282

[15] Kautz, W., Singleton, R.: 'Nonrandom binary superimposed codes', *IEEE Trans. Inf. Theory*, 1964, **10**, (4), pp. 363–377

[16] D'yachkov, A.G., Rykov, V.V.: 'Bounds for the length of disjunctive codes', *Probl. Inf. Transm.*, 1982, **18**, (3), pp. 7–13

[17] Luo, P., Lin, A., Zhen, W., *et al.*: 'Hardware implementation of secure Shamir's secret sharing scheme'. IEEE 15th Int. Symp. High-Assurance Systems Engineering (HASE), Miami, FL, USA, 2014

[18] D'yachkov, A.G., Rykov, V.V.: 'Optimal superimposed codes and designs for Renyi's search model', *J. Stat. Plan. Inference*, 2002, **100**, (2), pp. 281–302

[19] Ling, S., Xing, C.: '*Coding theory: a first course*' (Cambridge University Press, Cambridge, UK, 2004)

[20] Wang, Z., Karpovsky, M.G., Bu, L.: 'Design of reliable and secure devices realizing Shamir's secret sharing', *IEEE Trans. Comput.*, 2015, pp. 2443–2455

[21] Bu, L., Karpovsky, M.G., Wang, Z.: 'New byte error correcting codes with simple decoding for reliable cache design'. 21st IEEE On-Line Testing Symp. (IOLTS), Halkidiki, Greece, 2015

[22] Meyer, C.D.: '*Matrix analysis and applied linear algebra*' (Siam, Philadelphia, PA, USA, 2000)

[23] Moreira, J.C., Farrell, P.G.: '*Essentials of error-control coding*' (John Wiley & Sons, Marblehead, MA, USA, 2006)

[24] Fog, A.: '*The microarchitecture of Intel, AMD and VIA CPUs – an optimization guide for assembly programmers and compiler makers*' (Technical University of Denmark, Lyngby, Denmark, 2014)

[25] Pontarelli, S., Reviriego, P., Ottavi, M., *et al.*: 'Low delay single symbol error correction codes based on Reed Solomon codes', *IEEE Trans. Comput.*, 2015, **64**, (5), pp. 1497–1501

[26] Namba, K., Lombardi, F.: 'High-speed parallel decodable nonbinary single-error correcting (sec) codes', *IEEE Trans. Device Mater. Reliab.*, 2016, **16**, (1), pp. 30–37

[27] Yalcin, G., Islek, E., Tozlu, O., *et al.*: 'Exploiting a fast and simple ECC for scaling supply voltage in level-1 caches'. IEEE On-Line Testing Symp. (IOLTS), Platja d'Aro, Spain, 2014

[28] Dean, J., Barroso, L.A.: 'The tail at scale', *Commun. ACM*, 2013, **56**, (2), pp. 74–80

[29] Reisizadeh, A., Prakash, S., Pedarsani, R., *et al.*: 'Coded computation over heterogeneous clusters', arXiv preprint arXiv:1701.05973, 2017

[30] Lee, K.W.: 'Speeding up distributed storage and computing systems using codes'. Ph.D. dissertation, UC Berkeley, 2016
[31] Lee, K., Lam, M., Pedarsani, R., *et al.*: 'Speeding up distributed machine learning using codes', *IEEE Trans. Inf. Theory*, 2018, **64**, (3), pp. 1514–1529
[32] Ge, S., Wang, Z., Luo, P., *et al.*: 'Secure memories resistant to both random errors and fault injection attacks using nonlinear error correction codes'. ACM Proc. 2nd Int. Workshop on Hardware and Architectural Support for Security and Privacy, Tel-Aviv, Israel, 2013

15 Appendix

15.1 Appendix 1: Proof of the RS code parameters to construct the check matrix for GTB codes

Proof: According to Construction 1 and (2), we have

$$A \cdot l = q \cdot n_q \cdot q^{k_q - 1} = n_q q^{k_q} = n_q N$$

Since N is given, it comes down to find the minimal n_q. For RS codes, we have

$$m = \left\lfloor \frac{n_q - 1}{k_q - 1} \right\rfloor$$

When m is given, the problem comes down to find the minimal k_q, which is obviously $k_q = 2$.

Then by substituting it to all other equations

$$\begin{aligned} N &= q^2; \\ A &= q(m+1); \\ l &= q\,. \end{aligned}$$

Hence

$$A \cdot l = (m+1)N\,.$$

□

15.2 Appendix 2: Proof of the optimal q

Proof: Since $N = K + R = q^{k_q}$, $R = A - m$, and $A = q \cdot n_q$, by substituting R and A into N, we have

$$q^{k_q} - q \cdot n_q - K + m = 0.$$

From Theorem 1 the optimal k_q and n_q are given as

$$k_q = 2;\ n_q = m + 1.$$

By solving the first quadratic equation, the optimal q_{opt} when K and m are given is (12). □

15.3 Appendix 3: Proof of the parameters of GTB codes

Proof: The redundancy R of a code is equal to the number of linearly independent rows in its check matrix. By Construction 1 and the definition of blocks, $\boldsymbol{M}$ has n_q blocks and the sum of all rows in each block is always a vector of all 1's.

Therefore, we only need to remove any one row each of $n_q - 1$ blocks to make the rest of the rows all linearly independent. Also from (9), $n_q - 1 = m$, so that

$$\begin{aligned} R &= A - (n_q - 1) = qn_q - m = q(m+1) - m; \\ K &= N - R = q^2 - q(m+1) + m\,. \end{aligned}$$

If $\boldsymbol{M}$ is constructed from an m-superimposed code, in every column there are $n_q = m + 1$ number of *ones*. In Section 3.1, λ is defined as the maximal number of *ones* in common between any two columns. From Construction 1

$$\lambda = n_q - d_q\,.$$

Since for RS codes ..

$$\lambda = n_q - r_q - 1 = k_q - 1.$$

From Theorem 1 the optimal $k_q = 2$, thus

$$\lambda = 1. \tag{26}$$

For any two columns of $\boldsymbol{M}$, since $\lambda = 1$, the bitwise XOR of them can generate at most one 0 from two 1's in the same location of these two columns. We call this a cancellation. For x columns, the maximal number of cancellations is $\binom{x}{2}$.

On the other hand, since each of these x columns has $m + 1$ 1's, the maximal number of 1's in all x columns is $x(m + 1)$. For each block there needs to be at least 1 cancellation to make their bitwise XOR equal to zero. Therefore, the sum can have at most $x(m + 1) - (m + 1)$ 1's.

To make the number of cancellations greater than or equal to the number of 1's in the sum of x columns, we have

$$\begin{aligned} &\binom{x}{2} \geq (m+1)x - (m+1); \\ &\Rightarrow x^2 - (2m+3)x + 2(m+1) \geq 0. \end{aligned}$$

Solving this quadratic equation, we have $x \geq 2m + 2$. This shows that for a matrix $\boldsymbol{M}$ constructed by an m-superimposed code with the parameters from (9), it takes at least $2m + 2$ columns to make their bitwise XOR sum a vector of all *zeros*. Meaning for GTB codes

$$D = 2m + 2.$$

Therefore, given N and m, the parameters of the corresponding optimal GTB code are

$$(N, K, D)_Q = (q^2, q^2 - q(m+1) + m, 2m+2)_Q\,.$$

□

15.4 Appendix 4: Proof of the GTB code error locating algorithm

Proof: Construction 1 indicates that for any column j in $\boldsymbol{M}$, there are exactly $n_q = m + 1$ *ones* in $M_{*,j}$. Therefore, if the error e_j affects all the n_q bits of S_{sup} where v_j participates in computation, then reversely the location of e_j can be found by summing up all the affected support of the syndromes and comparing it with $m + 1$. □

15.5 Appendix 5: Proof of the GTB code error correction algorithm

Proof: According to Construction 1, in a matrix $\boldsymbol{M}$ whose columns are codewords of an m-superimposed code, within any set T of columns, $|T| \leq m + 1$, for any column h, $h \in T$, there must exist a row k in $\boldsymbol{M}$, where $M_{k,h} = 1$ in column h, and $M_{k,j} = 0$ for all $j \in T$, $j \neq h$. Since this is true for all columns in T, there exists an $(m + 1) \times (m + 1)$ identity sub-matrix in any given $m + 1$ columns.

The column indexes of m errors are given by $\boldsymbol{u}$ as in Algorithm 1. The rows $M_{i,*}$ for the identity sub-matrix can be easily identified by checking if only one error participates in the computing of $S(i)$.

Meaning for any one out of m errors, where $T = \{j \big| e_j \neq 0\}$ is the set of error locations, there exits at least one row $M_{i,*}$, where only $M_{i,j} = 1$ and $M_{i,h} = 0$ for all $h \in T, h \neq j$.

This $m \times m$ identity sub-matrix will provide the indexes of the digits in syndrome S which are affected by each single error digit e_j only, such that

$$S(i) = M_{i,*} \cdot \tilde{v} = M_{i,*} \cdot e = e_j .$$

Therefore, $v_j = \tilde{v}_j \oplus e_j = \tilde{v}_j \oplus S(i)$. □

15.6 Appendix 6: Proof of 1-step decoding

Proof: If the number of digits in C_{RS} codewords increases by Δ, meaning the number of blocks in $\boldsymbol{M}$ increasing to $n_q' = n_q + \Delta$, then the number of blocks without error masking will be increased by Δ. So the number of digits without error masking in the syndrome will increase by Δ. In this way, the lower bound in (14) can be re-written as

$$P_{\mathrm{corr}} \geq (1 - Q^{-1})^{m-1-\Delta}.$$

If $\Delta = 0$, then (15) is equivalent to (14).

If $\Delta = m - 1$, then $P_{\mathrm{corr}} = 1$. Meaning for any e_j, among $2m$ digits of the syndrome it affects, there are always at least $n_q = m + 1$ digits indicating that it is an error. This will be the same as majority voting. Meaning all m errors can be located and corrected with exactly the probability of 100%.

Since $(\sum S_{\mathrm{sup}}(i) \geq m + 1)?1:0$ can be simply represented as a threshold gate, where $m + 1$ or more *ones* in the input produce a binary 1 in the output, the procedure presented in Algorithm 3 can be called 1-step threshold decoding. □

15.7 Appendix 7: Proof of double error correction algorithm

Proof: When $m = 2$, we have $\lambda = 1$. So there can be at most one error masking in syndrome S. Therefore, if

$$w_j = \sum_{\{i \big| M_{i,j} = 1\}} S_{\mathrm{sup}}(i) = m;$$

Then it is possible that $e_j \neq 0$.

However, there can be more than two $u_j = 1$. From Definition 2, the bitwise OR of any two columns cannot cover another column. Therefore, for any $w_j = m = 2$, there must exit a row $M_{i,*}$ such that

$$\sum_{\{j \big| M_{i,j} = 1\}} w_j = 1.$$

However, since for double error correcting GTB codes $D = 2m + 2 = 6$, any two syndromes of two different double errors must be different. Therefore, for this row $M_{i,*}$, if

$$S(i) = 0;$$

Then in v_j there cannot be an error. Because if there is an error in v_j, and $\sum w_j = 1$, then $S(i) = e_j \neq 0$. □