2

Error Control Coding for Flash Memory

Haruhiko Kaneko
Tokyo Institute of Technology
Japan

1. Introduction

Error control code (ECC) is extensively used in high-speed wireless/wired communication system, magnetic disk, and optical disc (Lin & Costello, 2004). Also the ECC can efficiently improve data reliability of semiconductor memory system (Fujiwara, 2006). This chapter presents error control coding techniques for flash memory and solid-state drive (SSD). This chapter begins with brief introduction of error sources in the flash memory, and then provides fundamental mathematics of the ECC, followed by constructions of practical ECCs.

2. Errors in flash memory

Efficient ECC should be designed based on the analysis of error characteristics/statistics in the flash memory. This section outlines error sources of the flash memory, and presents a channel model based on a Gaussian-distribution approximation of the threshold voltage.

2.1 Errors sources in flash memory

2.1.1 Physical defect

Similar to general LSI circuits, flash memory suffers from wafer process defect (Muroke, 2006), such as short circuit between drain contact and control gate, adjacent poly lines, metal lines, poly and metal lines, or two metal levels. Other major defects in the flash memory are observed in tunnel oxide and peripheral circuit.
Many of the above defects can be detected by memory chip test (Mohammad et al., 2001), and thus a moderate number of defects can be masked by a redundant hardware design, while faulty chips with an excessive number of defects are discarded. Hence, the above physical defects do not affect ECC design significantly.

2.1.2 Trapping / detrapping in tunnel oxide

Stress on the tunnel oxide by program/erase (P/E) cycles causes generation of traps, such as positive-charge, neutral, and electron traps. The positive-charge and neutral traps induce leakage current, as explained in **2.1.3**, and the electron traps lengthen the charge time in programming phase. Also, detrapping of electrons trapped in the tunnel oxide causes lowered threshold voltage.

2.1.3 Leakage current

Leakage of electrons stored in the floating gate causes alteration of the threshold voltage, which results in errors in readout data. Stress induced leakage current (SILC) is caused by

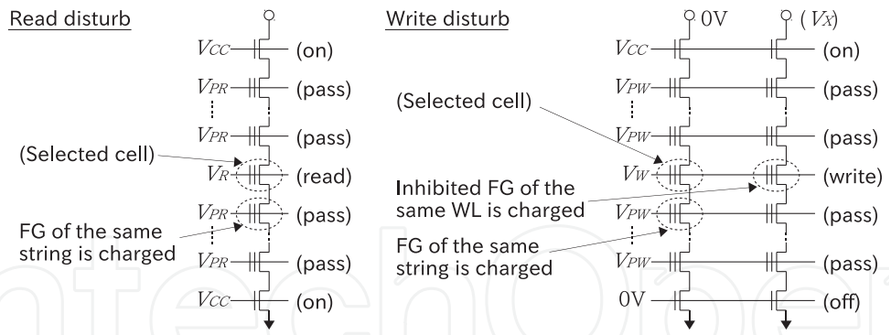


Fig. 1. Read/write disturb in NAND flash memory.

deterioration of the tunnel oxide after many P/E cycles, that is, positive-charge and neutral traps generated in the tunnel oxide causes leakage of electrons from the FG (Mielke et al., 2008). The leakage current significantly increases when multiple traps in the tunnel oxide form a path through which the FG is discharged (Ielmini et al., 2005). In multilevel cell (MLC) memory, the highest level cells are affected by the SILC more severely compared to lower level cells because of the largest electric field in the tunnel oxide.

2.1.4 Read/write disturb

High voltages applied to the CG in the read/write process cause insertion of electrons into the FG. Figure 1 illustrates the following read/write disturb in NAND flash memory.

- Read disturb: FG in the same string of a selected cell is charged.
- Write disturb in selected string: FG in the same string of a selected cell is charged.
- Write disturb in selected WL: Inhibited FG in the same WL of a selected cell is charged.

2.1.5 Overprogramming / overerase

The FG could be excessively charged in the programming phase because of, for example, random telegraph noise in verify step and erratic tunneling caused by positive charges in the tunnel oxide (Mielke et al., 2008). The overprogramming results in error due to an elevated threshold voltage. Some errors caused by overprogramming might be accidentally recovered by detrapping of electrons trapped in the tunnel oxide.

Overerase phenomena is also observed in the flash memory cell (Chimenton et al., 2003), where the threshold voltage is excessively dropped by the erasing. Overerased bits are classified into two classes, that is, tail bit and fast bit. The tail bit has a slightly lower threshold compared to normal bits, while the fast bit has a much lower threshold. It is predicted that the tail and fast bits are caused by statistical fluctuations of cell charges and physical nature of the cell, respectively.

2.1.6 Ionizing radiation

In a radiation harsh environment, such as spacecraft, aircraft, and nuclear plant, errors in the flash memory could be induced by radiation of ionizing particles (e.g., α-particle, β-particle, neutron, and cosmic rays) and high-energy electromagnetic wave (e.g., ultraviolet, X-ray, and γ-ray). The ionizing radiation causes lattice displacement in crystal, which changes the property of the semiconductor junctions, and thus results in errors. Effects of the total ionization dose (TID) on the flash memory have been extensively examined (Claeys et al., 2002; Oldham et al., 2007), and some experiments show that memory cells fail at the TID of

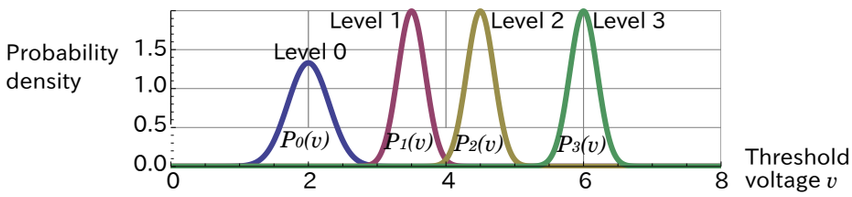


Fig. 2. Example of PDF $P_i(v)$ of threshold voltage.

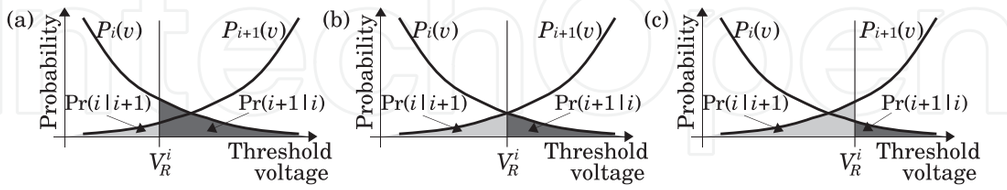


Fig. 3. Probabilities $\Pr(i|i+1)$ and $\Pr(i+1|i)$.

around 100 [Krad]. The TID affects the functions of the charge pump and row decoder, as well as the cell array (Bagatin et al., 2009).

2.2 Channel model

Since error sources of the flash memory are various as described in **2.1**, it is difficult to establish a precise channel model of the flash memory. Therefore, an approximated channel model is derived based on a Gaussian approximation of the threshold voltage distribution. In the following, we assume a b-bit MLC having $Q = 2^b$ charge levels of the FG. Let $P_i(v)$ be the probability density function (PDF) representing the probability that the threshold voltage of level-i cell is equal to v, where $i \in \{0, 1, \ldots, Q-1\}$. The PDF $P_i(v)$ is approximated by the Gaussian distribution as $P_i(v) = \frac{1}{\sqrt{2\pi}\sigma_i} \exp\left(-\frac{(x-\mu_i)^2}{2\sigma_i^2}\right)$, where μ_i and σ_i are the mean and standard deviation of the threshold voltage of level-i cell, respectively. Figure 2 illustrates an example of $P_i(v)$ for 2-bit/cell memory, where $b = 2, Q = 2^b = 4, \mu_0 = 2.0, \mu_1 = 3.5, \mu_2 = 4.5, \mu_3 = 6.0, \sigma_0 = 0.3$, and $\sigma_1 = \sigma_2 = \sigma_3 = 0.2$. In general, the standard deviation of level-0 cell is larger than that of higher level cells.

Let V_R^i be the read (verify) voltage of control gate which discriminates level-i and level-$(i+1)$ cells, where $i \in \{0, 1, \ldots, Q-2\}$ and $\mu_i < V_R^i < \mu_{i+1}$. For a given PDF $P_i(v)$ and a read voltage V_R^i, the probability that level-i cell is identified as level-j is given by

$$\Pr(j|i) = \begin{cases} \int_{-\infty}^{V_R^0} P_i(v)dv & (j=0) \\ \int_{V_R^{j-1}}^{V_R^j} P_i(v)dv & (1 \le j \le Q-2) \\ \int_{V_R^{Q-2}}^{\infty} P_i(v)dv & (j=Q-1) \end{cases}.$$

Figures 3 (a), (b), and (c) illustrate the probabilities $\Pr(i|i+1)$ and $\Pr(i+1|i)$ for three read voltages. Here, the dark and light shaded areas represent the probabilities $\Pr(i+1|i)$ and $\Pr(i+1|i)$, respectively. It is obvious from Fig. 3(b) that error probability between level-i and level-$(i+1)$ cells is minimized when V_R^i is determined such that $P_i(V_R^i) = P_{i+1}(V_R^i)$. For example, for the 4-level cell shown in Fig. 2, the optimum read voltages satisfying this equation are determined as $V_R^0 = 2.288, V_R^1 = 4.000$, and $V_R^2 = 5.250$.

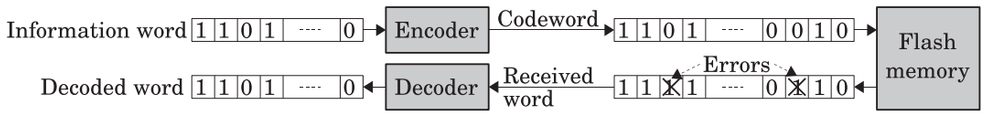


Fig. 4. Error control coding using a block code.

If the spatial and temporal correlations of errors are negligible in the flash memory, the errors can be described by a Q-ary stationary memoryless channel, whose channel matrix is given as

$$P = \begin{bmatrix} p_{0,0} & \cdots & p_{0,Q-1} \\ \vdots & \ddots & \vdots \\ p_{Q-1,0} & \cdots & p_{Q-1,Q-1} \end{bmatrix} = \begin{bmatrix} \Pr(0|0) & \ldots & \Pr(Q-1|0) \\ \vdots & \ddots & \vdots \\ \Pr(0|Q-1) & \ldots & \Pr(Q-1|Q-1) \end{bmatrix},$$

where $p_{i,j} = \Pr(j|i)$.

3. Introduction to linear block code

Figure 4 illustrates an error control coding scheme for the flash memory, where a block code is employed to correct/detect errors. In the write process, an information word is encoded to a codeword by the encoder, and then the codeword is written to the flash memory. In the read process, a received word (i.e., a readout codeword possibly having errors) is decoded by the decoder, wherein the errors are corrected or detected. Since many of practical ECCs are classified into linear block code, this section provides fundamentals of the linear block codes.

3.1 Galois field

Practical linear block codes are usually defined over Galois field. This subsection covers definition and construction of Galois field.

3.1.1 Definition

Galois Field $\mathrm{GF}(q)$ is defined as a finite set having q elements on which two binary operations, namely, addition $(+)$ and multiplication $(\cdot)$, are defined, where q is a prime number or a power of a prime number. Galois field is defined such that the following axioms hold.

Axioms of Galois field

1. Closure under addition: $\forall x, y \in \mathrm{GF}(q),\ x + y \in \mathrm{GF}(q)$.
2. Commutativity of addition: $\forall x, y \in \mathrm{GF}(q),\ x + y = y + x$.
3. Associativity of addition: $\forall x, y, z, \in \mathrm{GF}(q),\ (x + y) + z = x + (y + z)$.
4. Additive identity: $\forall x \in \mathrm{GF}(q), \exists 0 \in \mathrm{GF}(q),\ x + 0 = 0 + x = x$.
5. Additive inverse: $\forall x \in \mathrm{GF}(q), \exists (-x) \in \mathrm{GF}(q),\ x + (-x) = (-x) + x = 0$.
6. Closure under multiplication: $\forall x, y \in \mathrm{GF}(q),\ x \cdot y \in \mathrm{GF}(q)$.
7. Commutativity of multiplication: $\forall x, y \in \mathrm{GF}(q),\ x \cdot y = y \cdot x$.
8. Associativity of multiplication: $\forall x, y, z \in \mathrm{GF}(q),\ (x \cdot y) \cdot z = x \cdot (y \cdot z)$.
9. Multiplicative identity: $\forall x \in \mathrm{GF}(q), \exists 1 \in \mathrm{GF}(q),\ x \cdot 1 = 1 \cdot x = x$.
10. Multiplicative inverse: $\forall x \in \mathrm{GF}(q) - \{0\}, \exists x^{-1},\ x \cdot x^{-1} = x^{-1} \cdot x = 1$.
11. Distributivity: $\forall x, y, z \in \mathrm{GF}(q),\ x \cdot (y + z) = x \cdot y + x \cdot z$.

In the above notation, 0 and 1 are referred to as *zero* and *unity*, respectively. The set of axioms says that the four arithmetic operations (addition, substitution, multiplication, and division) can be applied to elements in $\mathrm{GF}(q)$. There exist two types of Galois field, that is, *prime field* $\mathrm{GF}(q)$ and *extension field* $\mathrm{GF}(q^m)$, where q is a prime number and $m \geq 2$ is an integer.

Addition table

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

Multiplication table

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

Table 1. Addition and multiplication tables of GF(5).

3.1.2 Prime field

Prime field is defined as $\mathrm{GF}(q) = \{0, 1, \ldots, q-1\}$, where q is a prime, and addition and multiplication of elements $x, y \in \mathrm{GF}(q)$ are defined as

$$(x + y) \bmod q \quad \text{and} \quad (x \cdot y) \bmod q,$$

respectively. Here, "a mod q" indicates the remainder of a divided by q. Table 1 presents an example of addition and multiplication tables of GF(5).

3.1.3 Extension field

Extension field $\mathrm{GF}(q^m)$ is constructed using a polynomial defined over $\mathrm{GF}(q)$. Let

$$f(x) = \sum_{i=0}^{m} f_i x^i = f_m x^m + f_{m-1} x^{m-1} + \cdots + f_1 x + f_0$$

be a polynomial over $\mathrm{GF}(q)$ of degree m, where $f_i \in \mathrm{GF}(q)$ for $0 \le i \le m$ and $f_m \neq 0$. Addition and multiplication of polynomials over $\mathrm{GF}(q)$ is defined in the same manner as polynomials over the real number except that addition and multiplication of coefficients are performed according to the definitions of $\mathrm{GF}(q)$. *Period* of a polynomial $f(x)$ is defined as the minimum positive integer e satisfying $f(x)|(x^e - 1)$, where $f(x)|g(x)$ indicates that $g(x)$ is divisible by $f(x)$.

Irreducible polynomial is a polynomial which cannot be factorized to polynomials over $\mathrm{GF}(q)$. For example, $x^2 + 1$ over GF(2) is not irreducible because $x^2 + 1 = (x + 1)(x + 1)$, whereas $x^2 + x + 1$ over GF(2) is irreducible. *Primitive polynomial* $p(x)$ is an irreducible polynomial whose period is $q^m - 1$. List of primitive polynomials is provided in various ECC text books, such as in (Lin & Costello, 2004).

Let $p(x) = \sum_{i=0}^{m} p_i x^i$ be a primitive polynomial of degree m over $\mathrm{GF}(q)$, where $p_m = 1$, and let α be a root of $p(x)$, that is $p(\alpha) = 0$. Since α satisfies $\alpha^m = -\sum_{i=0}^{m-1} p_i \alpha^i$, α^s is expressed as a polynomial of α of degree less than m for any non-negative integer s, that is,

$$\alpha^s = \sum_{i=0}^{m-1} a_i \alpha^i, \tag{1}$$

where $a_i \in \mathrm{GF}(q)$. The left-hand side and the right-hand side of Eq.(1) are referred to as the power and polynomial representations of α^s, respectively, and its coefficient vector

$$\mathrm{vec}(\alpha^s) = (a_{m-1}, a_{m-2}, \ldots, a_0)$$

is referred to as the vector representation of α^s.

Example 1. *Let α be a root of primitive polynomial $p(x) = x^3 + x + 1$ over* GF(2)*. Table 2 shows the polynomial and vector representations of the powers of α. (Note that $-x = x$ in* GF(2)*.)*

Power	Polynomial	Vector	Power	Polynomial	Vector
α^0	1	$(0,0,1)$	α^5	$\alpha^3+\alpha^2=\alpha^2+\alpha+1$	$(1,1,1)$
α^1	α	$(0,1,0)$	α^6	$\alpha^3+\alpha^2+\alpha=\alpha^2+1$	$(1,0,1)$
α^2	α^2	$(1,0,0)$	α^7	$\alpha^3+\alpha=1$	$(0,0,1)$
α^3	$\alpha+1$	$(0,1,1)$	$\vdots$	$\vdots$	$\vdots$
α^4	$\alpha^2+\alpha$	$(1,1,0)$			

Table 2. Polynomial and vector representation of powers of a root α of primitive polynomial $p(x)=x^3+x+1$ over GF(2).

Addition table

$+$	0	α^0	α^1	α^2	α^3	α^4	α^5	α^6
0	0	α^0	α^1	α^2	α^3	α^4	α^5	α^6
α^0	α^0	0	α^3	α^6	α^1	α^5	α^4	α^2
α^1	α^1	α^3	0	α^4	α^0	α^2	α^6	α^5
α^2	α^2	α^6	α^4	0	α^5	α^1	α^3	α^0
α^3	α^3	α^1	α^0	α^5	0	α^6	α^2	α^4
α^4	α^4	α^5	α^2	α^1	α^6	0	α^0	α^3
α^5	α^5	α^4	α^6	α^3	α^2	α^0	0	α^1
α^6	α^6	α^2	α^5	α^0	α^4	α^3	α^1	0

Multiplication table

$\cdot$	0	α^0	α^1	α^2	α^3	α^4	α^5	α^6
0	0	0	0	0	0	0	0	0
α^0	0	α^0	α^1	α^2	α^3	α^4	α^5	α^6
α^1	0	α^1	α^2	α^3	α^4	α^5	α^6	α^0
α^2	0	α^2	α^3	α^4	α^5	α^6	α^0	α^1
α^3	0	α^3	α^4	α^5	α^6	α^0	α^1	α^2
α^4	0	α^4	α^5	α^6	α^0	α^1	α^2	α^3
α^5	0	α^5	α^6	α^0	α^1	α^2	α^3	α^4
α^6	0	α^6	α^0	α^1	α^2	α^3	α^4	α^5

Table 3. Addition and multiplication tables of GF(2^3).

Theorem 1. *Let α be a root of primitive polynomial $p(x)$ of degree m over* GF(q). *The following relation holds for non-negative integers i and j:*

$$\alpha^i=\alpha^j \quad \Leftrightarrow \quad i\equiv j \pmod{q^m-1}.$$

An extension field GF(q^m) of degree m is generated from a ground field GF(q) as GF(q^m) $=\{0,\alpha^0,\alpha^1,\ldots,\alpha^{q^m-2}\}$, where α is a root of primitive polynomial of degree m over GF(q), and the vector representation of 0 is $(0,\ldots,0)$. Here, the additive and multiplicative identities are 0 and $\alpha^0=1$, respectively, and the addition and multiplication of α^i and α^j are defined as

$$\alpha^i+\alpha^j=\alpha^k$$

and

$$\alpha^i\cdot\alpha^j=\alpha^{(i+j) \bmod q^m-1},$$

where $\mathrm{vec}(\alpha^i)+\mathrm{vec}(\alpha^j)=\mathrm{vec}(\alpha^k)$ over GF(q). In short, the addition is defined on the vector representation, and the multiplication on the power representation.

Example 2. *Let α be a root of primitive polynomial $p(x)=x^3+x+1$ over* GF(2). *Extension field* GF(2^3) *is defined as* GF(2^3) $=\{0,\alpha^0,\alpha^1,\ldots,\alpha^6\}$, *where the non-zero elements are listed in Table 2. The addition and multiplication tables of* GF(2^3) *are presented in Table 3.*

3.2 Linear space

3.2.1 Definition

Let $\mathbf{u}=(u_0,u_1,\ldots,u_{n-1})$ and $\mathbf{v}=(v_0,v_1,\ldots,v_{n-1})$ be vectors of length n over GF(q), where $u_i,v_i\in$ GF(q) for $0\le i\le n-1$ and GF(q) is either prime field or extension field. For the vectors over GF(q), addition, inner product, and scalar multiplication are defined as

$$\begin{aligned}\mathbf{u}+\mathbf{v}&=(u_0+v_0,u_1+v_1,\ldots,v_{n-1}+u_{n-1}),\\ \mathbf{u}\cdot\mathbf{v}&=u_0\cdot v_0+u_1\cdot v_1+\cdots+u_{n-1}\cdot v_{n-1}, \quad \text{and}\\ a\cdot\mathbf{v}&=(a\cdot v_0,a\cdot v_1,\ldots,a\cdot v_{n-1}),\end{aligned}$$

respectively, where the addition and multiplication of vector components are defined on GF(q). For simplicity, the multiplication operator '$\cdot$' will be omitted hereafter. Let $\mathbf{V}$ be a set of vectors of length n over GF(q). The set $\mathbf{V}$ is a linear space if the following conditions hold.
1. Closure under addition: $\forall \mathbf{u}, \mathbf{v} \in \mathbf{V},\ \mathbf{u}+\mathbf{v} \in \mathbf{V}$.
2. Commutativity of addition: $\forall \mathbf{u}, \mathbf{v} \in \mathbf{V},\ \mathbf{u}+\mathbf{v} = \mathbf{v}+\mathbf{u}$.
3. Associativity of addition: $\forall \mathbf{u}, \mathbf{v}, \mathbf{w} \in \mathbf{V},\ (\mathbf{u}+\mathbf{v})+\mathbf{w} = \mathbf{u}+(\mathbf{v}+\mathbf{w})$.
4. Zero vector: $\forall \mathbf{u} \in \mathbf{V}, \exists \mathbf{o} \in \mathbf{v},\ \mathbf{u}+\mathbf{o} = \mathbf{o}+\mathbf{u} = \mathbf{u}$.
5. Inverse vector: $\forall \mathbf{u} \in \mathbf{V}, \exists -\mathbf{u} \in \mathbf{V},\ \mathbf{u}+(-\mathbf{u}) = \mathbf{o}$.
6. Closure under scalar multiplication: $\forall \mathbf{u} \in \mathbf{V}, \forall a \in \mathrm{GF}(q),\ a\mathbf{u} \in \mathbf{V}$.
7. Distributivity (scalar addition): $\forall \mathbf{u} \in \mathbf{V}, \forall a, b \in \mathrm{GF}(q),\ (a+b)\mathbf{u} = a\mathbf{u}+b\mathbf{u}$.
8. Distributivity (vector addition): $\forall \mathbf{u}, \mathbf{v} \in \mathbf{V}, \forall a \in \mathrm{GF}(q),\ a(\mathbf{u}+\mathbf{v}) = a\mathbf{u}+a\mathbf{v}$.
9. Associativity of scalar multiplication: $\forall \mathbf{u} \in \mathbf{V}, \forall a, b \in \mathrm{GF}(q),\ (ab)\mathbf{u} = a(b\mathbf{u})$.
10. Identity of scalar multiplication: $\forall \mathbf{u} \in \mathbf{V},\ 1\mathbf{u} = \mathbf{u}$.

Example 3. *The following set of all vectors over* GF(2) *of length* 4 *is a linear space:*

$$\mathbf{V} = \{(0,0,0,0),(0,0,0,1),(0,0,1,0),(0,0,1,1),(0,1,0,0),(0,1,0,1),(0,1,1,0),(0,1,1,1),$$
$$(1,0,0,0),(1,0,0,1),(1,0,1,0),(1,0,1,1),(1,1,0,0),(1,1,0,1),(1,1,1,0),(1,1,1,1)\}.$$

A subset $\mathbf{S}$ of a linear space $\mathbf{V}$ is a linear subspace if the following conditions hold:
1. Closure under addition: $\forall \mathbf{u}, \mathbf{v} \in \mathbf{S}, \quad \mathbf{u}+\mathbf{v} \in \mathbf{S}$.
2. Closure under scalar multiplication: $\forall \mathbf{u} \in \mathbf{S}, \forall a \in \mathrm{GF}(q), \quad a\mathbf{u} \in \mathbf{S}$.

Example 4. *One example of the linear subspace of* $\mathbf{V}$ *in Example 3 is* $\mathbf{S} = \{(0,0,0,0),(0,0,1,1),(1,1,0,0),(1,1,1,1)\}$.

3.2.2 Basis vector and dimension

Linear space $\mathbf{V}$ can be specified by a set of *basis vectors* as follows:

$$\mathbf{V} = \{\mathbf{v} = a_0\mathbf{v}_0 + a_1\mathbf{v}_1 + \cdots + a_{k-1}\mathbf{v}_{k-1} \mid a_i \in \mathrm{GF}(q), \mathbf{v}_i\text{: basis vector}\},$$

where $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_{k-1}$ are linearly independent vectors of length n over GF(q). Here, the basis vectors satisfy the following condition:

$$a_0\mathbf{v}_0 + a_1\mathbf{v}_1 + \cdots + a_{k-1}\mathbf{v}_{k-1} = \mathbf{o} \quad \Leftrightarrow \quad a_0 = a_1 = \cdots = a_{k-1} = 0,$$

where $\mathbf{o}$ is the zero-vector.

Example 5. *The followings are examples of basis vectors of* $\mathbf{V}$ *and* $\mathbf{S}$ *of Examples 3 and 4, respectively.*
One example of basis vectors of $\mathbf{V}$*:* $\{(0,0,0,1),(0,0,1,0),(0,1,0,0),(1,0,0,0)\}$.
Another example of basis vectors of $\mathbf{V}$*:* $\{(0,0,1,1),(1,1,0,0),(0,0,0,1),(0,1,1,0)\}$.
An example of basis vectors of $\mathbf{S}$*:* $\{(0,0,1,1),(1,1,0,0)\}$.

For a given vector space $\mathbf{V}$, there exists a number of combinations of basis vectors, while the number of basis vectors in each combination is identical. The *dimension* k of vector space $\mathbf{V}$ is defined as the number of basis vectors of $\mathbf{V}$.

Example 6. *The dimensions of* $\mathbf{V}$ *and* $\mathbf{S}$ *in Examples 3 and 4 are* $k = 4$ *and* 2*, respectively.*

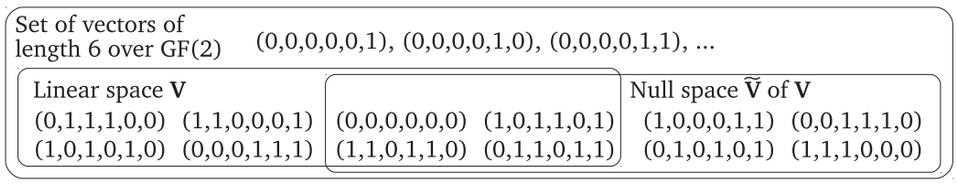


Fig. 5. Example of null space over GF(2).

3.2.3 Null space

Let $\mathbf{V}$ be a linear space of dimension k defined as

$$\mathbf{V} = \{\mathbf{v} = a_0\mathbf{v}_0 + a_1\mathbf{v}_1 + \cdots + a_{k-1}\mathbf{v}_{k-1} \mid a_i \in \mathrm{GF}(q), \mathbf{v}_i \text{ is basis vector}\},$$

where $\mathbf{v}_i$ is a vector of length n over GF(q). The *null space* of $\mathbf{V}$ is defined as

$$\widetilde{\mathbf{V}} = \{\widetilde{\mathbf{v}} = (\widetilde{v}_0, \widetilde{v}_1, \ldots, \widetilde{v}_{n-1}) \mid \forall \mathbf{v} \in \mathbf{V}, \mathbf{v} \cdot \widetilde{\mathbf{v}} = 0, \widetilde{v}_i \in \mathrm{GF}(q)\}.$$

Equivalently, the null space $\widetilde{\mathbf{V}}$ is defined using the basis vectors of $\mathbf{V}$ as

$$\widetilde{\mathbf{V}} = \{\widetilde{\mathbf{v}} = (\widetilde{v}_0, \widetilde{v}_1, \ldots, \widetilde{v}_{n-1}) \mid \forall i \in \{0, 1, \ldots, k-1\}, \mathbf{v}_i \cdot \widetilde{\mathbf{v}} = 0, \widetilde{v}_i \in \mathrm{GF}(q)\}.$$

It can be proved that the null space is a linear space.

Example 7. *Figure 5 presents an example of linear space* $\mathbf{V}$ *and its null space* $\widetilde{\mathbf{V}}$ *over* GF(2).

3.3 Linear block code

3.3.1 Definition

Let $\mathbf{F}^n$ be a linear space defined as a set of all vectors of length n over GF(q), that is,

$$\mathbf{F}^n = \{\mathbf{u} = (u_0, u_1, \ldots, u_{n-1}) \mid u_i \in \mathrm{GF}(q)\}.$$

A *block code* of length n over GF(q) is defined as a subset of $\mathbf{F}^n$, and a *linear block code* $\mathbf{C}$ of length n over GF(q) is defined as a linear subspace of $\mathbf{F}^n$. A code $\mathbf{C}$ of length n with dimension k is denoted as (n, k) code. Encoding by a linear block code $\mathbf{C}$ is defined as a bijective mapping from $\mathbf{F}^k$ to $\mathbf{C}$. Vectors in $\mathbf{C}$ and $\mathbf{F}^k$ are referred to as codeword and information word, respectively.

Example 8. *From the linear subspace* $\mathbf{V}$ *shown in Fig. 5, (6,3) linear block code* $\mathbf{C}$ *over* $\mathbf{F} = \mathrm{GF}(2)$ *is generated as*

$$\mathbf{C} = \{(0,1,1,1,0,0), (1,0,1,0,1,0), (1,1,0,0,0,1), (0,0,0,1,1,1),$$
$$(0,0,0,0,0,0), (1,1,0,1,1,0), (1,0,1,1,0,1), (0,1,1,0,1,1)\}.$$

Encoding by $\mathbf{C}$ *is defined as an arbitrarily bijective mapping from* $\mathbf{F}^3$ *to* $\mathbf{C}$*. The following is one example of the bijective mapping:*

$$(0,0,0) \to (0,0,0,0,0,0) \quad (0,0,1) \to (1,1,0,0,0,1) \quad (0,1,0) \to (1,0,1,0,1,0) \quad (0,1,1) \to (0,1,1,0,1,1)$$
$$(1,0,0) \to (0,1,1,1,0,0) \quad (1,0,1) \to (1,0,1,1,0,1) \quad (1,1,0) \to (1,1,0,1,1,0) \quad (1,1,1) \to (0,0,0,1,1,1)$$

Systematic encoding is an encoding in which each bit in the information word appears in a fixed position of the codeword. The above encoding example is a systematic encoding because an information word $\mathbf{d} = (d_0, d_1, d_2) \in \mathbf{F}^3$ is mapped to a codeword $\mathbf{u} = (u_0, u_1, u_2, u_3, u_4, u_5) \in \mathbf{C}$, where $d_0 = u_3, d_1 = u_4$, and $d_2 = u_5$.

3.3.2 Generator matrix

Since linear block code $\mathbf{C}$ of length n over GF(q) is a linear subspace, $\mathbf{C}$ can be specified by a set of basis vectors $\{\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k-1}\}$, where $\mathbf{g}_i$ is a row vector of length n over GF(q). Generator matrix $\mathbf{G}$ of an (n, k) linear block code $\mathbf{C}$ is defined as a $k \times n$ matrix over GF(q) whose row vectors are basis vectors of $\mathbf{C}$, that is,

$$\mathbf{G} = \begin{bmatrix} g_{0,0} & \cdots & g_{0,n-1} \\ \vdots & \ddots & \vdots \\ g_{k-1,0} & \cdots & g_{k-1,n-1} \end{bmatrix} = \begin{bmatrix} \mathbf{g}_0 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix},$$

where $g_{i,j} \in \mathrm{GF}(q)$ for $0 \leq i \leq k-1$ and $0 \leq j \leq n-1$. The code $\mathbf{C}$ is defined using the generator matrix $\mathbf{G}$ as follows:

$$\mathbf{C} = \{\mathbf{u} = (u_0, u_1, \ldots, u_{n-1}) = \mathbf{dG} \mid \mathbf{d} = (d_0, d_1, \ldots, d_{k-1}), d_i \in \mathrm{GF}(q)\}.$$

From this definition, linear code $\mathbf{C}$ is equivalent to the row space of $\mathbf{G}$.

Example 9. *A generator matrix of the linear block code* $\mathbf{C}$ *shown in Example 8 is given as*

$$\mathbf{G} = \begin{bmatrix} 0\,1\,1\,1\,0\,0 \\ 1\,0\,1\,0\,1\,0 \\ 1\,1\,0\,0\,0\,1 \end{bmatrix}.$$

3.3.3 Parity-check matrix

Let $\widetilde{\mathbf{C}}$ be the null space of linear code $\mathbf{C}$, and let $\{\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{r-1}\}$ be the set of basis vectors of $\widetilde{\mathbf{C}}$, where $\mathbf{h}_i$ is a row vector of length n over GF(q), and $r = n - k$. Parity-check matrix $\mathbf{H}$ of the linear code $\mathbf{C}$ is defined as the following $r \times n$ matrix over GF(q):

$$\mathbf{H} = \begin{bmatrix} h_{0,0} & \cdots & h_{0,n-1} \\ \vdots & \ddots & \vdots \\ h_{r-1,0} & \cdots & h_{r-1,n-1} \end{bmatrix} = \begin{bmatrix} \mathbf{h}_0 \\ \vdots \\ \mathbf{h}_{r-1} \end{bmatrix},$$

where $h_{i,j} \in \mathrm{GF}(q)$ for $0 \leq i \leq r-1$ and $0 \leq j \leq n-1$. The code $\mathbf{C}$ is defined by the parity-check matrix as follows:

$$\mathbf{C} = \{\mathbf{u} = (u_0, u_1, \ldots, u_{n-1}) \mid \mathbf{H}\mathbf{u}^T = \mathbf{o}^T = (0, \ldots, 0)^T, u_i \in \mathrm{GF}(q)\}.$$

In this definition, the code $\mathbf{C}$ is equivalent to the null space of the row space of $\mathbf{H}$.

Example 10. *An example parity-check matrix of the code* $\mathbf{C}$ *shown in Example 8 is given as*

$$\mathbf{H} = \begin{bmatrix} 1\,0\,0\,0\,1\,1 \\ 0\,1\,0\,1\,0\,1 \\ 0\,0\,1\,1\,1\,0 \end{bmatrix}.$$

3.3.4 Minimum distance

Let $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1})$ and $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ be vectors of length n over GF(q). Hamming weight of $\mathbf{u}$ is defined as

$$w(\mathbf{u}) = \text{ (the number of nonzero elements in } \mathbf{u}),$$

and Hamming distance between $\mathbf{u}$ and $\mathbf{v}$ is defined as $d(\mathbf{u},\mathbf{v}) = w(\mathbf{u}-\mathbf{v})$. In other words, Hamming distance $d(\mathbf{u},\mathbf{v})$ is the number of component positions in which the two vectors differ. Minimum distance of a block code $\mathbf{C}$ is defined as

$$d_{\min}(\mathbf{C}) = \min_{\mathbf{u},\mathbf{v}\in\mathbf{C},\mathbf{u}\neq\mathbf{v}} d(\mathbf{u},\mathbf{v}).$$

In general, it is hard to determine the minimum distance from the above definition if the cardinality of $\mathbf{C}$ is large. The following theorems are useful to determine the minimum distance of a linear block code.

Theorem 2. *For a linear block code* $\mathbf{C}$*, it holds that* $d_{\min}(\mathbf{C}) = \min_{\mathbf{u}\in\mathbf{C},\mathbf{u}\neq\mathbf{o}} w(\mathbf{u})$.

Proof. Since $\mathbf{C}$ is a linear block code, the following relation holds:

$$d_{\min}(\mathbf{C}) = \min_{\mathbf{u},\mathbf{v}\in\mathbf{C},\mathbf{u}\neq\mathbf{v}} d(\mathbf{u},\mathbf{v}) = \min_{\mathbf{u},\mathbf{v}\in\mathbf{C},\mathbf{u}\neq\mathbf{v}} w(\mathbf{u}-\mathbf{v}) = \min_{\mathbf{w}\in\mathbf{C},\mathbf{w}\neq\mathbf{o}} w(\mathbf{w}).$$

□

The above theorem says that the minimum distance of a linear block code $\mathbf{C}$ is equal to the minimum Hamming weight of non-zero codeword of $\mathbf{C}$. The minimum Hamming weight of non-zero codeword can be determined from the parity-check matrix as follows.

Theorem 3. *Let* $\mathbf{H}$ *be a parity-check matrix of linear block code* $\mathbf{C}$*. If there exist* d *column vectors in* $\mathbf{H}$ *which are linearly dependent, and also* $d-1$ *or fewer column vectors in* $\mathbf{H}$ *are linearly independent, then the minimum Hamming weight of non-zero codeword of* $\mathbf{C}$ *is* d*.*

Proof. Since $\mathbf{H}$ has d column vectors which are linearly dependent, there exists a codeword $\mathbf{u}$ of Hamming weight d satisfying $\mathbf{H}\mathbf{u}^T = \mathbf{o}^T$. Also, since $d-1$ or fewer column vectors in $\mathbf{H}$ are linearly independent, any vector $\mathbf{x}$ of Hamming weight $1 \le w(\mathbf{x}) \le d-1$ does not satisfy $\mathbf{H}\mathbf{x}^T = \mathbf{o}$, which means that $\mathbf{C}$ does not have non-zero codeword of Hamming weight less than or equal to $d-1$. Therefore, the minimum Hamming weight of non-zero codeword of $\mathbf{C}$ is d. □

Combining Theorems 2 and 3, the following theorem is obtained.

Theorem 4. *Let* $\mathbf{H}$ *be a parity-check matrix of linear block code* $\mathbf{C}$*. If there exist* d *column vectors in* $\mathbf{H}$ *which are linearly dependent, and also* $d-1$ *or fewer column vectors in* $\mathbf{H}$ *are linearly independent, then the minimum distance of* $\mathbf{C}$ *is* d*.*

3.3.5 Error control capability of bounded distance decoding

Let $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1})$ be a codeword over GF(q), and let

$$\mathbf{r} = (r_0, r_1, \ldots, r_{n-1}) = \mathbf{u} + \mathbf{e} = (u_0, u_1, \ldots, u_{n-1}) + (e_0, e_1, \ldots, e_{n-1})$$

be a received word, that is, a readout word from flash memory, where $\mathbf{e}$ is an error vector over GF(q). The number of errors in $\mathbf{r}$ is defined as $d(\mathbf{u},\mathbf{r}) = w(\mathbf{u}-\mathbf{r}) = w(\mathbf{e})$.

Theorem 5. *A block code of minimum distance* d *is capable of correcting* t *errors and detecting* s *errors by the bounded distance decoding, where* $d \ge t+s+1$ *and* $t \le s$*.*

Figure 6 illustrates the relation between t, s and d under the bounded distance decoding.

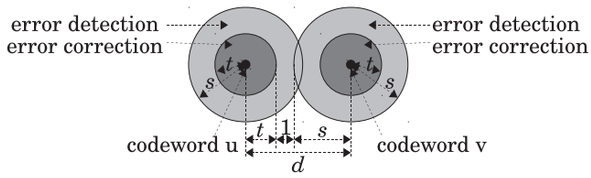


Fig. 6. Relation between the minimum distance and error correction/detection capabilities under the bounded distance decoding.

3.4 Cyclic code

3.4.1 Construction

Cyclic code $\mathbf{C}$ is a subclass of linear block code satisfying the following condition:

$$\mathbf{u} = (u_0, u_1, \ldots, u_{n-2}, u_{n-1}) \in \mathbf{C} \quad \Rightarrow \quad \mathbf{u}' = (u_{n-1}, u_0, \ldots, u_{n-3}, u_{n-2}) \in \mathbf{C}.$$

That is, if $\mathbf{u}$ is a codeword of $\mathbf{C}$, then a cyclic shift of $\mathbf{u}$ is also a codeword. Cyclic code is usually defined using polynomials over GF(q). Let $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1})$ and $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ be codewords of $\mathbf{C}$. Polynomial representations of $\mathbf{u} \in \mathbf{C}$ and $\mathbf{v} \in \mathbf{C}$ are given as follows:

$$\mathbf{u}(x) = u_0 + u_1 x + u_2 x^2 + \ldots + u_{n-1} x^{n-1}, \quad \mathbf{v}(x) = v_0 + v_1 x + v_2 x^2 + \ldots + v_{n-1} x^{n-1},$$

where $u_i, v_i \in$ GF(q). Addition of two codewords $\mathbf{u}(x) \in \mathbf{C}$ and $\mathbf{v}(x) \in \mathbf{C}$ is expressed as

$$\mathbf{u}(x) + \mathbf{v}(x) = (u_0 + v_0) + (u_1 + v_1)x + (u_2 + v_2)x^2 + \cdots + (u_{n-1} + v_{n-1})x^{n-1},$$

and right cyclic shift of $\mathbf{u}(x) \in \mathbf{C}$ as

$$x\mathbf{u}(x) \bmod (x^n - 1) = u_{n-1} + u_0 x + u_1 x^2 + \cdots + u_{n-2} x^{n-1} \in \mathbf{C}. \tag{2}$$

By recursively applying the relation of Eq.(2), we obtain $x^i \mathbf{u}(x) \bmod (x^n - 1) \in \mathbf{C}$ for any non-negative integer i. Since $\mathbf{C}$ is a linear block code, $(x^i + x^j)\mathbf{u}(x) \bmod (x^n - 1) \in \mathbf{C}$ for any pair of non-negative integers i and j. This implies that, if $\mathbf{u}(x)$ is a codeword of $\mathbf{C}$, then $\mathbf{f}(x)\mathbf{u}(x) \bmod (x^n - 1)$ is also a codeword of $\mathbf{C}$, where $\mathbf{f}(x)$ is a polynomial over GF(q).

Definition 1. *Let $\mathbf{g}(x)$ be a factor of $(x^n - 1)$, that is, $(x^n - 1) = \mathbf{g}(x)\mathbf{h}(x)$, where the degree of $\mathbf{g}(x)$ is r. Cyclic code $\mathbf{C}$ over* GF(q) *of length n is defined using $\mathbf{g}(x)$ as the generator polynomial, that is,*

$$\begin{aligned}\mathbf{C} &= \{\mathbf{f}(x)\mathbf{g}(x) \bmod (x^n - 1) \mid \mathbf{f}(x)\text{: polynomial over GF}(q)\} \\ &= \{\mathbf{d}(x)\mathbf{g}(x) \mid \mathbf{d}(x)\text{: polynomial over GF}(q) \text{ of degree less than } n - r\}.\end{aligned}$$

The following theorem is obvious from the above definition.

Theorem 6. *Let $\mathbf{C}$ be a cyclic code generated by $\mathbf{g}(x)$. Polynomial $\mathbf{u}(x)$ is a codeword of $\mathbf{C}$ if and only if*

$$\mathbf{u}(x) \bmod \mathbf{g}(x) = 0.$$

Example 11. *Polynomial $(x^7 - 1)$ over* GF(2) *is factorized as*

$$(x^7 - 1) = (x^3 + x + 1)(x^3 + x^2 + 1)(x + 1).$$

Let $\mathbf{g}(x) = x^3 + x + 1$ be the generator polynomial of cyclic code of length $n = 7$. Table 4 lists codewords of the cyclic code generated by $\mathbf{g}(x)$. This code is referred to as $(7, 4)$ cyclic Hamming code.

Example 12. *Cyclic redundancy check (CRC) codes are used as error detecting code. Table 5 shows generator polynomials of standardized CRC codes (Lin & Costello, 2004; Witzke & Leung, 1985).*

$\mathbf{d}(x)$	$\mathbf{u}(x) = \mathbf{d}(x)\mathbf{g}(x)$	(vector)	$\mathbf{d}(x)$	$\mathbf{u}(x) = \mathbf{d}(x)\mathbf{g}(x)$	(vector)
0	0	(0000000)	x^3	$x^3 + x^4 + x^6$	(0001101)
1	$1 + x + x^3$	(1101000)	$1 + x^3$	$1 + x + x^4 + x^6$	(1100101)
x	$x + x^2 + x^4$	(0110100)	$x + x^3$	$x + x^2 + x^3 + x^6$	(0111001)
$1 + x$	$1 + x^2 + x^3 + x^4$	(1011100)	$1 + x + x^3$	$1 + x^2 + x^6$	(1010001)
x^2	$x^2 + x^3 + x^5$	(0011010)	$x^2 + x^3$	$x^2 + x^4 + x^5 + x^6$	(0010111)
$1 + x^2$	$1 + x + x^2 + x^5$	(1110010)	$1 + x^2 + x^3$	$1+x+x^2+x^3+x^4+x^5+x^6$	(1111111)
$x + x^2$	$x + x^3 + x^4 + x^5$	(0101110)	$x + x^2 + x^3$	$x + x^5 + x^6$	(0100011)
$1+x+x^2$	$1 + x^4 + x^5$	(1000110)	$1+x+x^2+x^3$	$1 + x^3 + x^5 + x^6$	(1001011)

Table 4. $(7, 4)$ Cyclic Hamming code generated by $g(x) = x^3 + x + 1$.

Standard	Generator polynomial	Standard	Generator polynomial
CRC-12	$x^{12} + x^{11} + x^3 + x^2 + x + 1$	CRC-16	$x^{16} + x^{15} + x^2 + 1$
IBM-SDLC	$x^{16} + x^{15} + x^{13} + x^7 + x^4 + x^2 + x + 1$	CCITT X-25	$x^{16} + x^{12} + x^5 + 1$
CD-ROM	$x^{32} + x^{31} + x^{16} + x^{15} + x^4 + x^3 + x + 1$	DVD-ROM	$x^{32} + x^{31} + x^4 + 1$
IEC TC57	$x^{16} + x^{14} + x^{11} + x^8 + x^6 + x^5 + x^4 + 1$		
IEEE 802.3	$x^{32}+x^{26}+x^{23}+x^{22}+x^{16}+x^{12}+x^{11}+x^{10}+x^8+x^7+x^5+x^4+x^2+x+1$		

Table 5. Example of standardized CRC code.

3.4.2 Systematic encoding

Let $\mathbf{C}$ be a cyclic code of length n, and let $\mathbf{g}(x)$ be a degree-r generator polynomial of $\mathbf{C}$. Systematic encoding of an information word $\mathbf{d}(x)$ of degree $k = n - r$ by $\mathbf{C}$ is defined as

$$\mathbf{u}(x) = x^r \mathbf{d}(x) - \rho(x),$$

where $\rho(x) = x^r \mathbf{d}(x) \bmod \mathbf{g}(x)$. It can be easily verified that $\mathbf{u}(x)$ is a codeword of $\mathbf{C}$ because the following relation holds.

$$\begin{aligned} \mathbf{u}(x) \bmod \mathbf{g}(x) &= (x^r \mathbf{d}(x) - \rho(x)) \bmod \mathbf{g}(x) \\ &= (x^r \mathbf{d}(x) - x^r \mathbf{d}(x) \bmod \mathbf{g}(x)) \bmod \mathbf{g}(x) \\ &= (x^r \mathbf{d}(x) - x^r \mathbf{d}(x)) \bmod \mathbf{g}(x) = 0. \end{aligned}$$

This systematic encoding generates a codeword

$$\mathbf{u}(x) = u_0 + u_1 x + \cdots + u_{r-1} x^{r-1} + u_r x^r + u_{r+1} x^{r+1} + \ldots u_{n-1} x^{n-1},$$

where the first r terms correspond to the check part $\rho(x)$, and the remaining $k = n - r$ terms to the information word $\mathbf{d}(x)$.

Example 13. *Table 6 presents the systematic encoding of the (7,4) cyclic Hamming code generated by* $g(x) = x^3 + x + 1$. *Although this code is identical to the code shown in Table 4, the mapping from* $\mathbf{d}(x)$ *to* $\mathbf{u}(x)$ *is systematic, that is,* $\mathbf{d}(x) = d_0 + d_1 x + d_2 x^2 + d_3 x^3$ *corresponds to* $u_3 x^3 + u_4 x^4 + u_5 x^5 + u_6 x^6$.

4. Basic error control codes

This section introduces basic error control codes which have been applied to various digital systems. All codes presented in this section are linear codes of length n defined over GF(2^m), where m is a positive integer. That is, a codeword is expressed as $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1})$, where $u_i \in$ GF(2^m) for $0 \leq i \leq n - 1$. Here, u_i is referred to as *bit* (for $m = 1$) or *symbol* (for $m \geq 2$). Example of codeword structure of a systematic code is shown in Fig. 7(a), where the

$\mathbf{d}(x)$	$\mathbf{u}(x)$	(vector)	$\mathbf{d}(x)$	$\mathbf{u}(x)$	(vector)
0	0	(0000000)	x^3	$1+x^2+x^6$	(1010001)
1	$1+x+x^3$	(1101000)	$1+x^3$	$x+x^2+x^3+x^6$	(0111001)
x	$x+x^2+x^4$	(0110100)	$x+x^3$	$1+x+x^4+x^6$	(1100101)
$1+x$	$1+x^2+x^3+x^4$	(1011100)	$1+x+x^3$	$x^3+x^4+x^6$	(0001101)
x^2	$1+x+x^2+x^5$	(1110010)	x^2+x^3	$x+x^5+x^6$	(0100011)
$1+x^2$	$x^2+x^3+x^5$	(0011010)	$1+x^2+x^3$	$1+x^3+x^5+x^6$	(1001011)
$x+x^2$	$1+x^4+x^5$	(1000110)	$x+x^2+x^3$	$x^2+x^4+x^5+x^6$	(0010111)
$1+x+x^2$	$x+x^3+x^4+x^5$	(0101110)	$1+x+x^2+x^3$	$1+x+x^2+x^3+x^4+x^5+x^6$	(1111111)

Table 6. Systematic encoding of $(7,4)$ cyclic Hamming code generated by $g(x)=x^3+x+1$.

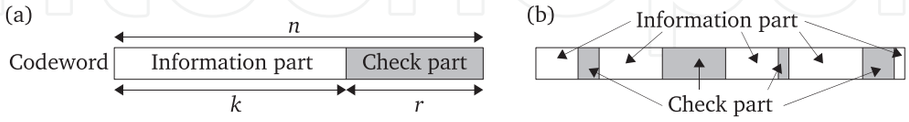


Fig. 7. Example of systematic codeword structure.

codeword consists of information and check parts of length k and r, respectively. Note that the check part can be divided and distributed over the codeword, as illustrated in Fig, 7(b).

4.1 Parity-check code

4.1.1 Definition

Parity-check code is a single-bit error detecting code over GF(2) whose codeword is expressed as $\mathbf{u}=(u_0,u_1,\ldots,u_{n-1})$, where $u_i \in \mathrm{GF}(2)$ for $0 \le i \le n-1$ and $\sum_{i=0}^{n-1} u_i = 0$. Here, the information length is $k=n-r=n-1$ and the minimum distance is $d_{\min}=2$. The parity-check and generator matrices are given as

$$\mathbf{H}=\begin{bmatrix}1\ 1\ \ldots\ 1\end{bmatrix}_{1\times n} \quad \text{and} \quad \mathbf{G}=\left[\begin{array}{c|c}\mathbf{I}_k & \begin{matrix}1\\ \vdots\\ 1\end{matrix}\end{array}\right]=\left[\begin{array}{ccc|c}1 & & 0 & 1\\ & \ddots & & \vdots\\ 0 & & 1 & 1\end{array}\right]_{k\times n},$$

respectively, where $\mathbf{I}_k$ is the $k\times k$ identity matrix.

4.1.2 Encoding/decoding

Encoding: Let $\mathbf{d}=(d_0,d_1,\ldots,d_{k-1})$ be an information word of length k. Codeword $\mathbf{u}$ for the information word $\mathbf{d}$ is determined as $\mathbf{u}=(u_0,u_1,\ldots,u_{n-2},u_{n-1})=(d_0,d_1,\ldots,d_{k-1},p)$, where $n=k+1$ and $p=\sum_{i=0}^{k-1} d_i$.
Decoding: Let $\mathbf{r}=(r_0,r_1,\ldots,r_{n-1})$ be a received word, where $r_i \in \mathrm{GF}(2)$ for $0 \le i \le n-1$. Syndrome s is calculated as $s=\sum_{i=0}^{n-1} r_i$. If $s=0$ holds, $\mathbf{r}$ is assumed to have no error; otherwise $\mathbf{r}$ has an odd number of errors.

4.2 Hamming SEC code

4.2.1 Definition

Hamming code is a single-bit error correcting (SEC) code over GF(2) defined by a parity-check matrix $\mathbf{H}$ whose column vectors are nonzero and distinct. The code length n of the Hamming code is upper bounded by $n \le 2^r-1$, where r is the number of check bits. Table 7 presents the maximum code length and information length of the Hamming code for $2 \le r \le 10$. Systematic parity-check matrix $\mathbf{H}$ of $(n, n-r)$ Hamming SEC code is expressed as $\mathbf{H}=\left[\mathbf{Q}\middle|\mathbf{I}_r\right]$, where $\mathbf{Q}$ is an $r\times(n-r)$ matrix whose column vectors have Hamming weight ≥ 2,

Check length: r	2	3	4	5	6	7	8	9	10
Maximum code length: $n = 2^r - 1$	3	7	15	31	63	127	255	511	1023
Information length: $k = n - r$	1	4	11	26	57	120	247	502	1013

Table 7. Maximum code length n and information length k of Hamming code.

and $\mathbf{I}_r$ is the $r \times r$ identity matrix. Generator matrix of the code is $\mathbf{G} = \left[\, \mathbf{I}_k \middle| \mathbf{Q}^T \,\right]$, where $\mathbf{Q}^T$ indicates the transpose of $\mathbf{Q}$.

Example 14. *The following shows the parity-check and generator matrices of* $(7,4)$ *Hamming code:*

$$\mathbf{H} = \left[\, \mathbf{Q} \middle| \mathbf{I}_3 \,\right] = \left[\begin{array}{cccc|ccc} 0 & 1 & 1 & 1 & 1 & 0 & 0 \\ 1 & 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right] \text{ and } \mathbf{G} = \left[\, \mathbf{I}_4 \middle| \mathbf{Q}^T \,\right] = \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right].$$

4.2.2 Encoding

Encoding using generator matrix

Let $\mathbf{G}$ be a $k \times n$ generator matrix of (n,k) Hamming code. Information word $\mathbf{d} = (d_0, d_1, \ldots, d_{k-1})$ of length k is encoded as $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1}) = \mathbf{dG}$.

Example 15. *Let* $\mathbf{G}$ *be the generator matrix of Example 14. Information word* $\mathbf{d} = (0,1,1,0)$ *is encoded as*

$$\mathbf{u} = \mathbf{dG} = (0,1,1,0) \left[\begin{array}{cccc|ccc} 1 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{array}\right] = (0,1,1,0,0,1,1).$$

Encoding using systematic parity-check matrix

Let $\mathbf{H} = \left[\, \mathbf{Q} \middle| \mathbf{I}_r \,\right]$ be an $r \times n$ systematic parity-check matrix of $(n, n-r)$ Hamming code, and let $\mathbf{d} = (d_0, d_1, \ldots, d_{k-1})$ be an information word of length $k = n - r$. This information word $\mathbf{d}$ is encoded as $\mathbf{u} = (\mathbf{d}, \mathbf{p}) = (d_0, d_1, \ldots, d_{k-1}, p_0, p_1, \ldots, p_{r-1})$, where the check part $\mathbf{p}$ is determined as $\mathbf{p} = (p_0, p_1, \ldots, p_{r-1}) = \mathbf{dQ}^T$.

Example 16. *Let* $\mathbf{H} = \left[\, \mathbf{Q} \middle| \mathbf{I}_3 \,\right]$ *be the systematic parity-check matrix of Example 14. The check part* $\mathbf{p}$ *for information word* $\mathbf{d} = (0,1,1,0)$ *is calculated as*

$$\mathbf{p} = \mathbf{dQ}^T = (0,1,1,0) \left[\begin{array}{ccc} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \\ 1 & 1 & 1 \end{array}\right] = (0,1,1).$$

Thus, codeword is generated as $\mathbf{u} = (\mathbf{d}, \mathbf{p}) = (0,1,1,0,\ 0,1,1)$.

4.2.3 Decoding

Let $\mathbf{u}' = (u'_0, u'_1, \ldots, u'_{n-1})$ be a received word expressed as

$$\mathbf{u}' = \mathbf{u} + \mathbf{e} = (u_0, u_1, \ldots, u_{n-1}) + (e_0, e_1, \ldots, e_{n-1}),$$

where $\mathbf{u}$ is an original codeword and $\mathbf{e}$ is an error vector. If $\mathbf{e}$ is the zero-vector, $\mathbf{u}'$ has no error. Hamming SEC code can recover $\mathbf{u}$ from $\mathbf{u}'$ when $w(\mathbf{e}) \leq 1$. Decoding is based on the following relation:

$$\mathbf{s} = \mathbf{H}\mathbf{u'}^T = \mathbf{H}(\mathbf{u} + \mathbf{e})^T = \mathbf{H}\mathbf{u}^T + \mathbf{H}\mathbf{e}^T = \mathbf{H}\mathbf{e}^T.$$

Check length: r	3	4	5	6	7	8	9	10	11
Maximum code length: $n = 2^{r-1}$	4	8	16	32	64	128	256	512	1024
Information length: $k = n - r$	1	4	11	26	57	120	247	502	1013

Table 8. Maximum code length n and information length k of OWC SEC-DED code.

This relation says that, if $\mathbf{u}'$ has a single bit error in the i-th bit, the syndrome $\mathbf{s}$ is equal to the i-th column vector of $\mathbf{H}$, where $0 \leq i \leq n-1$. The received word $\mathbf{u}'$ is decoded as follows:

1. The syndrome $\mathbf{s}$ is calculated as $\mathbf{s} = \mathbf{H}\mathbf{u}'^T$.
2. If $\mathbf{s}$ is the zero-vector, then $\mathbf{u}'$ is assumed to have no error, and thus decoded word is determined as $\widetilde{\mathbf{u}} = \mathbf{u}'$.
3. If $\mathbf{s}$ is equal to the i-th column vector of $\mathbf{H}$, the received word $\mathbf{u}'$ is assumed to have an error in the i-th bit. Decoded word is determined as $\widetilde{\mathbf{u}} = \mathbf{u}' + \mathbf{i}_i$, where $\mathbf{i}_i$ is a binary vector whose i-th element is 1 and the other elements are 0.
4. If $\mathbf{s}$ is nonzero and is not equal to any column vector of $\mathbf{H}$, the received word $\mathbf{u}'$ has multiple-bit error. Decoding result of this case is *uncorrectable error detection*.

Example 17. *Let* $\mathbf{u} = (0,1,1,0,0,1,1)$ *be the codeword generated in Example 16, and let* $\mathbf{e} = (0,0,1,0,0,0,0)$ *be the error vector. Received word is given as*

$$\mathbf{u}' = (0,1,1,0,0,1,1) + (0,0,1,0,0,0,0) = (0,1,0,0,0,1,1).$$

The syndrome of $\mathbf{u}'$ *is calculated as*

$$\mathbf{s} = \mathbf{H}\mathbf{u}'^T = \left[\begin{array}{ccc|ccc} 0&1&1&1&0&0 \\ 1&0&1&1&0&1&0 \\ 1&1&0&1&0&0&1 \end{array}\right] (0,1,0,0,0,1,1)^T = (1,1,0)^T.$$

Since the calculated syndrome is equal to the 2nd column of $\mathbf{H}$*, decoded word is determined as*

$$\widetilde{\mathbf{u}} = \mathbf{u}' + \mathbf{i}_2 = (0,1,0,0,0,1,1) + (0,0,1,0,0,0,0) = (0,1,1,0,0,1,1) = \mathbf{u}.$$

4.3 Odd-weight column (OWC) SEC-DED code

4.3.1 Definition

OWC code is a liner code over GF(2) having minimum distance $d_{\min} = 4$, and thus this code can be used as either single-bit error correcting - double-bit error detecting (SEC-DED) code or triple-bit error detecting (TED) code. The code length n of the OWC code is upper bounded by $n \leq 2^{r-1}$, where r is the number of check bits. Table 8 presents the maximum code length and information length of the OWC SEC-DED code for $3 \leq r \leq 11$.

Parity-check matrix of this code is generated from odd-weight column vectors. Systematic parity-check matrix of $(n, n-r)$ OWC code is expressed as $\mathbf{H} = [\,\mathbf{Q}_O \big| \mathbf{I}_r\,]$, where $\mathbf{Q}_O$ is an $r \times (n-r)$ matrix whose column vectors have odd Hamming weight $w \geq 3$, and $\mathbf{I}_r$ is the $r \times r$ identity matrix. Generator matrix of the code is given as $\mathbf{G} = [\,\mathbf{I}_k \big| \mathbf{Q}_O^T\,]$.

Example 18. *The following shows the parity-check and generator matrices of* $(8,4)$ *OWC code:*

$$\mathbf{H} = [\,\mathbf{Q}_O \,\big|\, \mathbf{I}_4\,] = \left[\begin{array}{cccc|cccc} 0&1&1&1&1&0&0&0 \\ 1&0&1&1&0&1&0&0 \\ 1&1&0&1&0&0&1&0 \\ 1&1&1&0&0&0&0&1 \end{array}\right] \text{ and } \mathbf{G} = [\,\mathbf{I}_4 \,\big|\, \mathbf{Q}_O^T\,] = \left[\begin{array}{cccc|cccc} 1&0&0&0&0&1&1&1 \\ 0&1&0&0&1&0&1&1 \\ 0&0&1&0&1&1&0&1 \\ 0&0&0&1&1&1&1&0 \end{array}\right].$$

m		3	4	5	6	7	8	9	10
n		7	15	31	63	127	255	511	1023
	$(t=1)$	4	11	26	57	120	247	502	1013
	$(t=2)$	-	7	21	51	113	239	493	1003
k	$(t=3)$	-	5	16	45	106	231	484	993
	$(t=4)$	-	-	11	39	99	223	475	983
	$(t=5)$	-	-	6	36	92	215	466	973

Table 9. Maximum code length n and information length k of t-bit error correcting BCH code.

4.3.2 Encoding

Information word $\mathbf{d} = (d_0, d_1, \ldots, d_{k-1})$ is encoded by either generator matrix $\mathbf{G}$ or systematic parity-check matrix $\mathbf{H} = \left[\, \mathbf{Q}_O \middle| \mathbf{I}_r \,\right]$ in the same way as the Hamming SEC code.

Example 19. *Information word* $\mathbf{d} = (0,1,0,1)$ *is encoded by the generator matrix* $\mathbf{G}$ *of Example 18 as*

$$\mathbf{u} = \mathbf{dG} = (0,1,0,1) \left[\begin{array}{cccc|cccc} 1&0&0&0&0&1&1&1 \\ 0&1&0&0&1&0&1&1 \\ 0&0&1&0&1&1&0&1 \\ 0&0&0&1&1&1&1&0 \end{array}\right] = (0,1,0,1,0,1,0,1).$$

4.3.3 Decoding

Decoding is based on the Hamming weight of syndrome $\mathbf{s} = \mathbf{Hu'}^T = \mathbf{He'}^T$. That is, since every column vector of $\mathbf{H}$ has an odd Hamming weight, syndrome $\mathbf{s}$ of a single-bit error has an odd weight, while that of a double-bit error has an even weight $w \geq 2$. This means that the syndromes of double-bit errors are distinct from those of single-bit errors. Hence, the OWC SEC-DED code can be decoded in the same way as Hamming code shown in **4.2.3**.

Example 20. *Let* $\mathbf{u} = (0,1,0,1,0,1,0,1)$ *be the codeword generated in Example 19, and let* $\mathbf{e} = (0,1,0,0,0,0,1,0)$ *be the error vector. Received word is given as*

$$\mathbf{u'} = \mathbf{u} + \mathbf{e} = (0,1,0,1,0,1,0,1) + (0,1,0,0,0,0,1,0) = (0,0,0,1,0,1,1,1).$$

The syndrome of $\mathbf{u'}$ *is calculated as*

$$\mathbf{s} = \mathbf{Hu'}^T = \left[\begin{array}{cccc|cccc} 0&1&1&1&1&0&0&0 \\ 1&0&1&1&0&1&0&0 \\ 1&1&0&1&0&0&1&0 \\ 1&1&1&0&0&0&0&1 \end{array}\right] (0,0,0,1,0,1,1,1)^T = (1,0,0,1)^T.$$

Since the calculated syndrome has even Hamming weight $w(\mathbf{s}) = 2$*, the syndrome is not equal to any column vector of* $\mathbf{H}$*. Hence, the decoding algorithm indicates that* $\mathbf{u'}$ *has uncorrectable errors.*

4.4 BCH code

4.4.1 Definition

Binary primitive BCH code is a t-bit error correcting cyclic code of length $n = 2^m - 1$ with $r \leq mt$ check bits, where $t \geq 1$ and $m \geq 2$. In most cases, $r = mt$ holds, and thus information length is $k = 2^m - 1 - mt$. Table 9 presents the maximum code length n and information length k of the BCH code for $3 \leq m \leq 10$ and $1 \leq t \leq 5$ (Lin & Costello, 2004).

Let α be a primitive element of GF(2^m). The parity-check matrix of t-bit error correcting BCH code is defined as

$$\mathbf{H} = \begin{bmatrix} 1 & \alpha^1 & \alpha^2 & \alpha^3 & \dots & \alpha^i & \dots & \alpha^{n-1} \\ 1 & \alpha^3 & (\alpha^3)^2 & (\alpha^3)^3 & \dots & (\alpha^3)^i & \dots & (\alpha^3)^{n-1} \\ 1 & \alpha^5 & (\alpha^5)^2 & (\alpha^5)^3 & \dots & (\alpha^5)^i & \dots & (\alpha^5)^{n-1} \\ \vdots & \vdots & \vdots & \vdots & & \vdots & & \vdots \\ 1 & \alpha^{2t-1} & (\alpha^{2t-1})^2 & (\alpha^{2t-1})^3 & \dots & (\alpha^{2t-1})^i & \dots & (\alpha^{2t-1})^{n-1} \end{bmatrix},$$

where α^i is expressed as a column vector of length m.
Generator matrix of the binary primitive BCH code is derived from the generator polynomial as follows. Let $\mathrm{cnj}_m(i)$ be a set of exponents of *conjugates* of $\alpha^i \in \mathrm{GF}(2^m)$, defined as follows:

$$\mathrm{cnj}_m(i) = \{\, i \cdot 2^j \bmod (2^m - 1) \mid j \in \{0, 1, 2, \dots\} \,\}.$$

Minimal polynomial of $\alpha^i \in \mathrm{GF}(2^m)$ is defined as

$$\phi_i(x) = \prod_{j \in \mathrm{cnj}_m(i)} (x - \alpha^j).$$

The generator polynomial of t-bit error correcting BCH code is given as

$$\mathbf{g}(x) = g_0 + g_1 x + g_2 x^2 + \cdots + g_r x^r = \mathrm{LCM}\{\phi_1(x), \phi_3(x), \dots, \phi_{2t-1}(x)\},$$

where LCM is the least common multiple of polynomials, and $g_i \in \mathrm{GF}(2)$. Then, generator matrix is determined as

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_r & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & g_2 & \cdots & g_r & 0 & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \cdots & g_r & 0 & \cdots & 0 \\ & & & & \vdots & & & & & \\ 0 & \cdots & 0 & g_0 & g_1 & g_2 & \cdots & g_r \end{bmatrix}.$$

4.4.2 Encoding/decoding

Information word is encoded using either generator matrix $\mathbf{G}$ or generator polynomial $\mathbf{g}(x)$ as described in **3.4.2**. From the definition of generator polynomial $\mathbf{g}(x)$, a codeword polynomial $\mathbf{u}(x) = u_0 + u_1 x + u_2 x^2 + \cdots + u_{n-1} x^{n-1}$ has $2t$ consecutive roots, $\alpha^1, \alpha^2, \dots, \alpha^{2t}$, that is $\mathbf{u}(\alpha^i) = 0$ for $i \in \{1, 2, \dots, 2t\}$. Using this relation, a received word $\mathbf{u}'(x) = u'_0 + u'_1 x + u'_2 x^2 + \cdots + u'_{n-1} x^{n-1}$ is decoded as follows.

1. *Syndrome calculation:* Syndrome $\mathbf{s}$ is determined as $\mathbf{s} = (s_1, s_2, \dots, s_{2t})$, where $s_i = \mathbf{u}'(\alpha^i)$. If $\mathbf{s} = (0, 0, \dots, 0)$, then $\mathbf{u}'(x)$ is assumed to have no error.
2. *Generation of error-location polynomial $\sigma(x)$:* Error-location polynomial

   $$\sigma(x) = \sigma_0 + \sigma_1 x + \cdots + \sigma_{t'} x^{t'}$$

   is calculated from the syndrome $\mathbf{s}$. This polynomial has t' roots, $\alpha^{-j_1}, \alpha^{-j_2}, \dots, \alpha^{-j_{t'}}$, where $t' \leq t$ is the number of errors in $\mathbf{u}'(x)$, and $j_i \in \{0, 1, \dots, n-1\}$ is the location of i-th error. The following shows Berlekamp's algorithm to derive $\sigma(x)$ from $\mathbf{s}$.

   (Berlekamp's algorithm) Error-location polynomial $\sigma(x)$ is derived using Table 10, where the rows of $\mu = -1$ and $\mu = 0$ are given initial values. In the following, μ is referred to as the

μ	ρ_μ	$\sigma^\mu(x)$	l_μ	$\mu - l_\mu$	d_μ
-1	-	1	0	-1	1
0	-	1	0	0	S_1
$\vdots$					
$2t$					

Table 10. Berlekamp's algorithm.

row number. Values in the rows from $\mu = 1$ to $\mu = 2t$ are determined as follows:
Step 1 Current row number is set as $\mu = 0$.
Step 2 If $d_\mu \neq 0$, $\rho_{\mu+1}$ is determined as

$$\rho_{\mu+1} = \arg \max_{\substack{-1 \le \mu' < \mu \\ d_{\mu'} \neq 0}} (\mu' - l_{\mu'}).$$

That is, $\rho_{\mu+1}$ is a row number μ' which is prior to the current row μ, where $d_{\mu'} \neq 0$, and $\mu' - l_{\mu'}$ has the largest value among the prior rows.
Step 3 Polynomial $\sigma^{\mu+1}(x)$ is determined as follows:

$$\sigma^{\mu+1}(x) = \begin{cases} \sigma^\mu(x) & (d_\mu = 0) \\ \sigma^\mu(x) + d_\mu d_\rho^{-1} x^{\mu-\rho} \sigma^\rho(x) & (d_\mu \neq 0) \end{cases},$$

where ρ is $\rho_{\mu+1}$ determined in **Step 2**.
Step 4 If $\mu = 2t - 1$, then $\sigma^{\mu+1}(x) = \sigma^{2t}(x)$ gives the error-location polynomial.
Step 5 $l_{\mu+1}$ is determined as the degree of $\sigma^{\mu+1}(x)$.
Step 6 $d_{\mu+1}$ is determined as follows:

$$d_{\mu+1} = S_{\mu+2} + \sigma_1^{\mu+1} S_{\mu+1} + \sigma_2^{\mu+1} S_\mu + \cdots + \sigma_{l_{\mu+1}}^{\mu+1} S_{\mu+2-l_{\mu+1}},$$

where $\sigma_i^{\mu+1}$ is the coefficient of degree-i term of $\sigma^{\mu+1}(x)$.
Step 7 The current row number is incremented as $\mu = \mu + 1$, and go to **Step 2**.

3. *Search of the roots of $\sigma(x)$:* The roots of the error-location polynomial $\sigma(x)$ are determined by *Chien search*, which finds a set of integers, $\{l'_1, l'_2, \ldots, l'_\tau\}$, satisfying $\sigma(\alpha^{l'_i}) = 0$, where $0 \le l'_i \le n - 1$.
4. *Error correction:* Error pattern is determined as $\mathbf{e}(x) = x^{l_1} + x^{l_2} + \cdots + x^{l_\tau}$, where $l_i = (n - l'_i) \bmod n$ for $1 \le i \le \tau$. Finally, errors in $\mathbf{u}'(x)$ are corrected as $\widetilde{\mathbf{u}}(x) = \mathbf{u}'(x) + \mathbf{e}(x)$.

4.5 Reed-Solomon code

RS code is a linear cyclic code over GF(q) of length $n = q - 1$ with r check symbols, where the minimum distance is $d_{\min} = r + 1$. Practically, q is a power of 2, such as $q = 2^8$, and thus the following considers the RS codes over GF(2^m). Let α be a primitive element of GF(2^m). The parity-check matrix of RS code over GF(2^m) is given as

$$\mathbf{H} = \begin{bmatrix} 1 & \alpha^1 & \alpha^2 & \ldots & \alpha^{n-2} & \alpha^{n-1} \\ 1 & \alpha^2 & \alpha^4 & \ldots & \alpha^{2(n-2)} & \alpha^{2(n-1)} \\ \vdots & \vdots & \vdots & & \vdots & \vdots \\ 1 & \alpha^{r-1} & \alpha^{(r-1)2} & \ldots & \alpha^{(r-1)(n-2)} & \alpha^{(r-1)(n-1)} \\ 1 & \alpha^r & \alpha^{r \cdot 2} & \ldots & \alpha^{r(n-2)} & \alpha^{r(n-1)} \end{bmatrix},$$

and the generator polynomial of RS code is defined as $\mathbf{g}(x) = (x - \alpha)(x - \alpha^2) \ldots (x - \alpha^r)$.

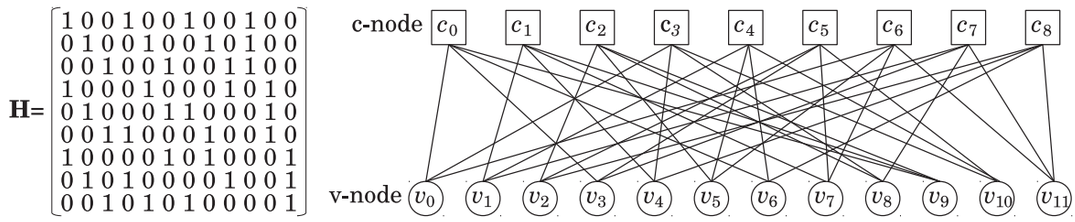


Fig. 8. Example of Tanner graph.

5. Low-Density Parity-Check (LDPC) code

LDPC code is a linear block code defined by a sparse parity-check matrix (Gallager, 1962), that is, the number of non-zero element in an $m \times n$ parity-check matrix is $O(n)$. The LDPC codes are employed in recent high-speed communication systems because appropriately designed LDPC codes have high error correction capability. The LDPC codes will be applicable to high-density MLC flash memory suffering from high BER.

5.1 Tanner graph

An LDPC matrix $\mathbf{H} = [h_{i,j}]_{m \times n}$ is expressed by a *Tanner graph*, which is a bipartite graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V} = V \cup C$ is a set of nodes, and $\mathcal{E}$ is a set of edges. Here, $V = \{v_0, v_1, \ldots, v_{n-1}\}$ is a set of variable-nodes (v-nodes) corresponding to column vectors of $\mathbf{H}$, and $C = \{c_0, c_1, \ldots, c_{m-1}\}$ is a set of check-nodes (c-nodes) corresponding to row vectors of $\mathbf{H}$. The edge set is defined as $\mathcal{E} = \{(c_i, v_j) | h_{i,j} \neq 0\}$. That is, c-node c_i and v-node v_j are connected by an edge (c_i, v_j) if and only if $h_{i,j} \neq 0$. *Girth* of $\mathcal{G}$ is defined as the length of shortest cycle in $\mathcal{G}$. The girth affects the error correction capability of LDPC code, that is, a code with a small girth l, e.g., $l = 4$, will have poor error correction capability compared to codes with a large girth.

Example 21. *Figure 8 presents a parity-check matrix* $\mathbf{H}$ *and corresponding Tanner graph* $\mathcal{G}$*.*

5.2 Regular/irregular LDPC code

5.2.1 Regular LDPC code

Regular LDPC code is defined by a parity-check matrix whose columns have a constant weight $\lambda \ll m$ and rows have almost constant weight. More precisely, Hamming weight $w_c(\mathbf{H}_{*,j})$ of the j-th column in $\mathbf{H}$ satisfies $w_c(\mathbf{H}_{*,j}) = \lambda$ for $0 \leq j \leq n-1$, and Hamming weight $w_r(\mathbf{H}_{i,*})$ of the i-th row in $\mathbf{H}$ satisfies $\lfloor n\lambda/m \rfloor \leq w_c(\mathbf{H}_{i,*}) \leq \lceil n\lambda/m \rceil$ for $0 \leq i \leq m-1$. Note that the total number of nonzero elements in $\mathbf{H}$ is $n\lambda$. The regular LDPC matrix is constructed as follows (Lin & Costello, 2004; Moreira & Farrell, 2006).

- *Random construction:* LDPC matrix $\mathbf{H}$ is randomly generated by computer search under the following constraints:
  - Every column of $\mathbf{H}$ has a constant weight λ.
  - Every row of $\mathbf{H}$ has weight either $\lfloor n\lambda/m \rfloor$ or $\lceil n\lambda/m \rceil$.
  - Overlapping of nonzero element in every pair of columns in $\mathbf{H}$ is at most one.

  The last constraint guarantees that the girth of generated $\mathbf{H}$ is at least six.
- *Geometric construction:* LDPC matrix can be constructed using geometric structure, such as, Euclidean geometry and projective geometry.

5.2.2 Irregular LDPC code

Irregular LDPC code is defined by an LDPC matrix having unequal column weight. The codes with appropriate column weight distribution have higher error correction capability compared to the regular LDPC codes (Richardson et al., 2001).

Column no.	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Locations	0	32	64	8	31	63	14	30	17	28	22	27	7	19	6
of 1s.	1	34	70	18	42	76	45	47	62	48	60	49	53	44	46
(Row no.)	4	39	78	95	54	91	94	83	80	82	81	84	77	85	75

Table 11. Position of 1s in the base matrix $\mathbf{H}_0$ of IEEE 802.15.3c.

5.3 Example

5.3.1 WLAN (IEEE 802.11n, 2009)

(1296,1080) LDPC code is defined by the following parity-check matrix:

$$\mathbf{H} = \begin{bmatrix} 48 & 29 & 37 & 52 & 2 & 16 & 6 & 14 & 53 & 31 & 34 & 5 & 18 & 42 & 53 & 31 & 45 & - & 46 & 52 & 1 & 0 & - & - \\ 17 & 4 & 30 & 7 & 43 & 11 & 24 & 6 & 14 & 21 & 6 & 39 & 17 & 40 & 47 & 7 & 15 & 41 & 19 & - & - & 0 & 0 & - \\ 7 & 2 & 51 & 31 & 46 & 23 & 16 & 11 & 53 & 40 & 10 & 7 & 46 & 53 & 33 & 35 & - & 25 & 35 & 38 & 0 & - & 0 & 0 \\ 19 & 48 & 41 & 1 & 10 & 7 & 36 & 47 & 5 & 29 & 52 & 52 & 31 & 10 & 26 & 6 & 3 & 2 & - & 51 & 1 & - & - & 0 \end{bmatrix},$$

where "$-$" indicates the 54×54 zero matrix, and integer i indicates a 54×54 matrix generated from the 54×54 identity matrix by cyclically shifting the columns to the right by i elements.

5.3.2 WiMAX (IEEE 802.16e, 2009)

(1248,1040) LDPC code is defined by the following parity-check matrix:

$$\mathbf{H} = \begin{bmatrix} 0 & 13 & 29 & - & 25 & 2 & - & 49 & 45 & 4 & 46 & 28 & 44 & 17 & 2 & 0 & 19 & 10 & 2 & 41 & 43 & 0 & - & - \\ - & 3 & - & 19 & 21 & 25 & 6 & 42 & 25 & - & 22 & 11 & 6 & 38 & 7 & 39 & 0 & 23 & 26 & 0 & 0 & 0 & 0 & - \\ 27 & 43 & 44 & 2 & 36 & - & 11 & - & 16 & 13 & 49 & 33 & 43 & 4 & 46 & 42 & 32 & 47 & 36 & 8 & - & - & 0 & 0 \\ 36 & - & 27 & 8 & - & 19 & 7 & 5 & 5 & 10 & 28 & 48 & 15 & 49 & 30 & 16 & 45 & 49 & 5 & 35 & 43 & - & - & 0 \end{bmatrix},$$

where "$-$" indicates the 52×52 zero matrix, and integer i indicates a 52×52 matrix generated from the 52×52 identity matrix by cyclically shifting the columns to the right by i elements.

5.3.3 WPAN (IEEE 802.15.3c, 2009)

Let $\mathbf{H}_0$ be a 96×15 matrix whose elements are all-zero expect the elements listed in Table 11. (1440,1344) Quasi-cyclic LDPC code is defined by the following parity-check matrix:

$$\mathbf{H} = \left[\mathbf{H}_0 \middle| \mathbf{H}_1 \middle| \mathbf{H}_2 \middle| \ldots \middle| \mathbf{H}_{94} \middle| \mathbf{H}_{95} \right],$$

where $\mathbf{H}_i$ is obtained by cyclically i-row upward shifting of the base matrix $\mathbf{H}_0$.

5.4 Soft input decoding algorithm of binary LDPC code

Let $\mathbf{u} = (u_0, u_1, \ldots, u_{n-1})$ be a codeword of binary LDPC code defined by an $m \times n$ LDPC matrix $\mathbf{H}$. To retrieve a codeword $\mathbf{u}$ stored in the flash memory, the posteriori probability $f_i(x)$ is determined from readout values $(v_0, v_1, \ldots, v_{n-1})$, where $f_i(x)$ denotes the probability that the value of i-th bit of the codeword is $x \in \{0, 1\}$. For example, if a binary input asymmetric channel with channel matrix $\mathbf{P} = [p_{i,j}]_{2\times 2}$ is assumed, then the posteriori probability is given as $f_i(x) = p_{x,v_i}/(p_{0,v_i} + p_{1,v_i})$, where it is assumed that $\Pr(u_i = 0) = \Pr(u_i = 1) = 1/2$. The sum-product algorithm (SPA) determines a decoded word $\tilde{\mathbf{u}} = (\tilde{u}_0, \tilde{u}_1, \ldots, \tilde{u}_{n-1})$ from the posteriori probabilities $(f_0(x), f_1(x), \ldots, f_{n-1}(x))$. The SPA is an iterative belief propagation algorithm performed on the Tanner graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where each edge $e_{i,j} = (c_i, v_j) \in \mathcal{E}$ is assigned two probabilities $Q_{i,j}(x)$ and $R_{i,j}(x)$, where $x \in \{0, 1\}$. The following notations are used in the SPA.

- $d_i^c = |\{j \mid e_{i,j} \in \mathcal{E}\}|$: degree of c-node c_i.

- $\{J_0^{i,j}, J_1^{i,j}, \ldots, J_{d_i^c-2}^{i,j}\} = \{\, J \mid e_{i,J} \in \mathcal{E}, J \neq j\}$: set of indices of v-nodes adjacent to c-node c_i excluding v_j.

Sum-product algorithm

1. Initialize $R_{i,j}(x)$ as $R_{i,j}(0) = R_{i,j}(1) = 1/2$ for each $e_{i,j} \in \mathcal{E}$.
2. Calculate $Q_{i,j}(x)$ for each $e_{i,j} \in \mathcal{E}$:

$$Q_{i,j}(x) = \eta \times f_j(x) \times \prod_{I \in \{I \mid e_{I,j} \in \mathcal{E}\} \setminus \{i\}} R_{I,j}(x)\,,$$

   where $x \in \{0,1\}$ and η is determined such that $Q_{i,j}(0) + Q_{i,j}(1) = 1$.
3. Calculate $R_{i,j}(x)$ for each $e_{i,j} \in \mathcal{E}$:

$$R_{i,j}(0) = \sum_{(x_0,\ldots,x_{d_i^c-2}) \in X_{d_i^c-1}} \left(\prod_{k=0}^{d_i^c-2} Q_{i,J_k^{i,j}}(x_k) \right), \quad R_{i,j}(1) = 1 - R_{i,j}(0),$$

   where $X_l = \left\{ (x_0, \ldots, x_{l-1}) \,\middle|\, \sum_{i=0}^{l-1} x_i = 0 \right\}$.
4. Generate a temporary decoded word $\widetilde{\mathbf{u}} = (\widetilde{u}_0, \widetilde{u}_1, \ldots, \widetilde{u}_{n-1})$ from

$$Q_j(x) = f_j(x) \times \prod_{I \in \{I \mid e_{I,j} \in \mathcal{E}\}} R_{I,j}(x)\,,$$

   where $x \in \{0,1\}$ and

$$\widetilde{u}_j = \begin{cases} 0 & (Q_j(0) > Q_j(1)) \\ 1 & (\text{otherwise}) \end{cases}.$$

5. Calculate syndrome $\mathbf{s} = \mathbf{H}\widetilde{\mathbf{u}}^T$. If $\mathbf{s} = \mathbf{o}$, then output $\widetilde{\mathbf{u}}$ as a decoded word, and terminate.
6. If the number of iterations is greater than a predetermined threshold, then terminate with uncorrectable error detection; otherwise go to step 2.

There exist variations of the SPA, such as Log domain SPA and log-likelihood ratio (LLR) SPA. Also, there are some reduced-complexity decoding algorithms, such as bit-flipping decoding algorithm and min-sum algorithm (Lin & Costello, 2004).

5.5 Nonbinary LDPC code

5.5.1 Construction

Nonbinary LDPC code is a linear block code over GF(q) defined by an LDPC matrix $\mathbf{H} = [h_{i,j}]_{m \times n}$, where $h_{i,j} \in$ GF(q). The nonbinary LDPC codes generally have higher error correction capability compared to the binary codes (Davey & MacKay, 1998). Several construction methods of the nonbinary LDPC matrix have been proposed. For example, high performance quasi-cyclic LDPC codes are constructed using Euclidean geometry (Zhou et al., 2009). It is shown in (Li et al., 2009) that, under a Gaussian approximation of the probability density, optimum column weight of $\mathbf{H}$ over GF(q) decreases and converges to two with increasing q. For example, the optimum column weight of rate-1/2 LDPC code on the AWGN channel is 2.6 for $q = 2$, while that is 2.1 for $q = 64$.

5.5.2 Decoding

The SPA for the binary LDPC code can be extended to the one for nonbinary codes straightforwardly, in which probabilities $Q_{i,j}(x)$ and $R_{i,j}(x)$ are iteratively calculated for $x \in \mathrm{GF}(q)$. However, the computational complexity of $R_{i,j}(x)$ is $O(q^2)$, and thus the SPA is impractical for a large q. For practical cases of $q = 2^b$, a reduced complexity SPA for nonbinary LDPC code has been proposed using the fast Fourier transform (FFT) (Song & Cruz, 2003).

Definition 2. *Let* $(X(0), X(\alpha^0), X(\alpha^1), \ldots, X(\alpha^{q-2}))$ *be a vector of real numbers of length* $q = 2^p$, *where* α *is a primitive element of* $\mathrm{GF}(q)$. *Function* f_k *is defined as follows:*

$$f_k(X(0), X(\alpha^0), X(\alpha^1), \ldots, X(\alpha^{q-2})) = (Y(0), Y(\alpha^0), Y(\alpha^1), \ldots, Y(\alpha^{q-2})),$$

where

$$Y(\beta_0) = \frac{1}{\sqrt{2}}(X(\beta_0) + X(\beta_1)) \text{ and } Y(\beta_1) = \frac{1}{\sqrt{2}}(X(\beta_0) - X(\beta_1)).$$

Here, $\beta_0 \in \mathrm{GF}(2^p)$ *and* $\beta_1 \in \mathrm{GF}(2^p)$ *are expressed as*

$$\mathrm{vec}(\beta_0) = (i_{p-1}, i_{p-2}, \ldots, i_{k+1}, 0, i_{k-1}, \ldots, i_0) \text{ and}$$
$$\mathrm{vec}(\beta_1) = (i_{p-1}, i_{p-2}, \ldots, i_{k+1}, 1, i_{k-1}, \ldots, i_0).$$

The FFT of $(X(0), X(\alpha^0), \ldots, X(\alpha^{q-2}))$ *is defined as*

$$\mathcal{F}(X(0), X(\alpha^0), \ldots, X(\alpha^{q-2})) = f_{p-1}(f_{p-2}(\ldots f_1(f_0(X(0), X(\alpha^0), \ldots, X(\alpha^{q-2})))\ldots)).$$

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be the Tanner graph of LDPC matrix $\mathbf{H} = [h_{i,j}]_{m \times n}$ over $\mathrm{GF}(q)$, where each edge $e_{i,j} \in \mathcal{E}$ is assigned a nonzero value $h_{i,j} \in \mathrm{GF}(q)$. The following shows the outline of the FFT-based SPA for given posteriori probability $f_j(x)$, that is, the probability of the i-th symbol being x, where $x \in \mathrm{GF}(q)$ and $0 \le i \le n-1$.

FFT-based Sum-product algorithm for nonbinary LDPC code

1. Initialize $R_{i,j}(x)$ as $R_{i,j}(x) = 1/q$ for each $e_{i,j} \in \mathcal{E}$ and $x \in \mathrm{GF}(q)$.
2. Calculate $Q_{i,j}(x)$ for each $e_{i,j} \in \mathcal{E}$ and $x \in \mathrm{GF}(q)$:

$$Q_{i,j}(x) = \eta \times f_j(x) \times \prod_{I \in \{I | e_{I,j} \in \mathcal{E}\} \setminus \{i\}} R_{I,j}(x)\,,$$

   where η is determined such that $\sum_{x \in \mathrm{GF}(q)} Q_{i,j}(x) = 1$.
3. Calculate $R_{i,j}(x)$ for each $e_{i,j} \in \mathcal{E}$ and $x \in \mathrm{GF}(q)$ as follows:
   (a) Generate the probability distribution permuted by $h_{i,j}$, that is, $Q'_{i,j}(x \cdot h_{i,j}) = Q_{i,j}(x)$.
   (b) Apply the FFT to $Q'_{i,j}(x)$ as

$$(\widetilde{Q}_{i,j}(0), \widetilde{Q}_{i,j}(\alpha^0), \ldots, \widetilde{Q}_{i,j}(\alpha^{q-2})) = \mathcal{F}(Q'_{i,j}(0), Q'_{i,j}(\alpha^0), \ldots, Q'_{i,j}(\alpha^{q-2})).$$

   (c) Calculate the product of $\widetilde{Q}_{i,j}(x)$ for each $e_{i,j} \in \mathcal{E}$ as $\widetilde{R}_{i,j}(x) = \prod_{k=0}^{d_i^c - 2} \widetilde{Q}_{i,J_k^{i,j}}(x)$.
   (d) Apply the FFT to $\widetilde{R}_{i,j}(x)$ as

$$(R'_{i,j}(0), R'_{i,j}(\alpha^0), \ldots, R'_{i,j}(\alpha^{q-2})) = \mathcal{F}(\widetilde{R}_{i,j}(0), \widetilde{R}_{i,j}(\alpha^0), \ldots, \widetilde{R}_{i,j}(\alpha^{q-2})).$$

   (e) Generate the probability distribution permuted by $h_{i,j}^{-1}$, that is, $R_{i,j}(x) = R'_{i,j}(x \cdot h_{i,j})$.

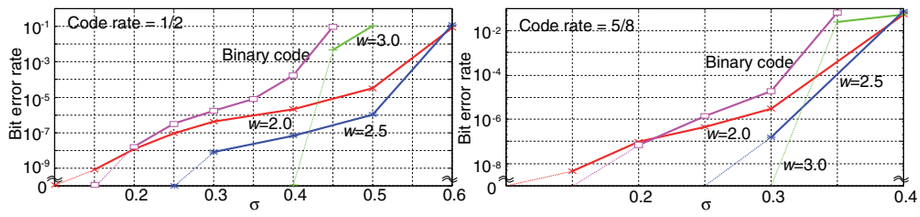


Fig. 9. Decoded BER of LDPC code over GF(8).

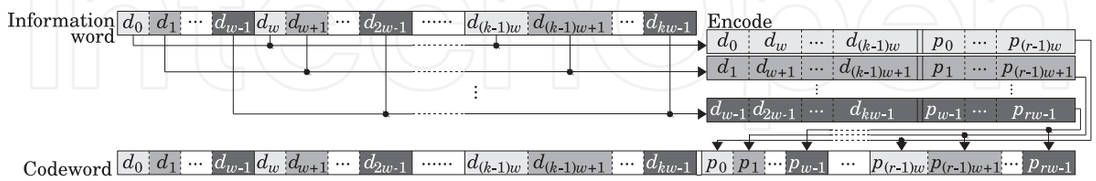


Fig. 10. w-Way interleave of $(k+r,k)$ systematic code.

4. Generate a temporary decoded word $\widetilde{\mathbf{u}} = (\widetilde{u}_0, \widetilde{u}_1, \ldots, \widetilde{u}_{n-1})$ using

$$Q_j(x) = f_j^x \times \prod_{I \in \{I | e_{I,j} \in \mathcal{E}\}} R_{I,j}(x)\,,$$

where $x \in \mathrm{GF}(q)$ and $\widetilde{u}_j = \arg\max_{x \in \mathrm{GF}(q)} Q_j(x)$.

5. Calculate syndrome $\mathbf{s} = \mathbf{H}\widetilde{\mathbf{u}}^T$. If $\mathbf{s} = \mathbf{o}$, then output $\widetilde{\mathbf{u}}$ as a decoded word, and terminate.
6. If the number of iterations is greater than a predetermined threshold, then terminate with uncorrectable error detection; otherwise go to step 2.

5.6 Nonbinary LDPC code for flash memory

The following evaluates the decoded BER of the nonbinary LDPC codes for a channel model of 8-level cell flash memory (Maeda & Kaneko, 2009), where the threshold voltages are hypothesized as $\mu_0 = -3.0000$, $\mu_1 = -2.0945$, $\mu_2 = -1.2795$, $\mu_3 = -0.4645$, $\mu_4 = 0.3505$, $\mu_5 = 1.1655$, $\mu_6 = 1.9805$, and $\mu_7 = 3.0000$. These threshold voltages are determined to minimize the raw BER under the condition that $\mu_0 = -3.0000$, $\mu_{Q-1} = 3.0000$, and the standard deviation σ_i of $P_i(v)$ is given as $\sigma_i = \sigma$ for $i \in \{1, 2, \ldots, Q-2\}$, $\sigma_0 = 1.2\sigma$, and $\sigma_{Q-1} = 1.5\sigma$.
The decoded BER is calculated by decoding 100,000 words, where the maximum number of iterations in the SPA is 200. Figure 9 illustrates the relation between the standard deviation σ and the decoded BER of nonbinary LDPC codes over GF(8) having code rates $1/2$ and $5/8$. The decoded BER is evaluated for the code length 8000, where the column weights of the parity-check matrix are 2, 3, and 2.5. This figure also shows the decoded BER of binary irregular LDPC code. This figure says that the nonbinary LDPC codes have lower BER than binary irregular LDPC codes, and the nonbinary codes with column weight $w = 2.5$ give the lowest BER in many cases.

6. Combination of error control codes

6.1 Fundamental techniques

Interleaving: Interleaving is an effective technique to correct burst errors. Figure 10 illustrates the w-way interleave of a $(k+r,k)$ systematic code. Here, information word of length wk is interleaved to generate w information subwords of length k, which are

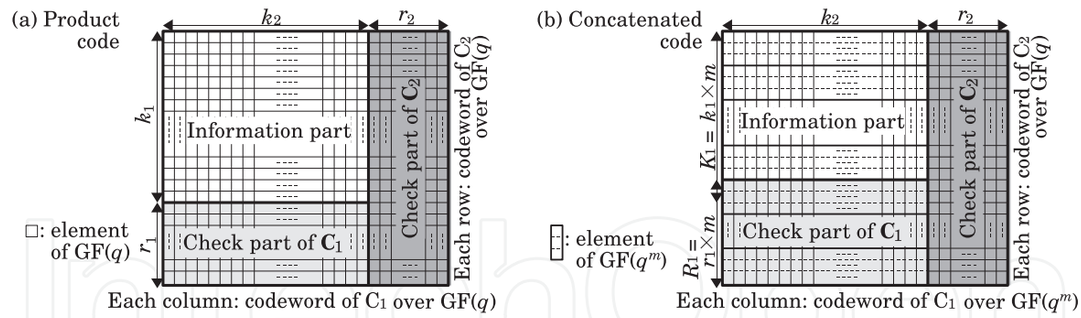


Fig. 11. Product/concatenated code using systematic block codes.

independently encoded by the $(k+r,k)$ systematic code. Then, the generated check bits are interleaved and appended to the information word. If the $(k+r,k)$ code can correct burst l-bit errors, then the interleaved code can correct burst wl-bit errors.

Product code: Product code is defined using two block codes over GF(q), that is, $(k_1 + r_1, k_1)$ code $\mathbf{C}_1$ and $(k_2 + r_2, k_2)$ code $\mathbf{C}_2$, as illustrated in Fig. 11(a). Information part is expressed as a $k_1 \times k_2$ matrix over GF(q). Each column of the information part is encoded by $\mathbf{C}_1$, and then each row of the obtained $(k_1 + r_1) \times k_2$ matrix is encoded by $\mathbf{C}_2$. The minimum distance of the product code is $d = d_1 \times d_2$, where d_1 and d_2 are the minimum distances of $\mathbf{C}_1$ and $\mathbf{C}_2$, respectively.

Concatenated code: Concatenated code is defined using two block codes $\mathbf{C}_1$ and $\mathbf{C}_2$, where $\mathbf{C}_1$ is a $(k_1 + r_1, k_1)$ code over GF(q^m), and $\mathbf{C}_2$ is a $(k_2 + r_2, k_2)$ code over GF(q), as shown in Fig. 11(b). Information part is expressed as a $K_1 \times k_2$ matrix, where $K_1 = k_1 \times m$. Each column of the information part, which is regarded as a vector of length k_1 over GF(q^m), is encoded by $\mathbf{C}_1$, and then each row of the obtained $(K_1 + R_1) \times k_2$ matrix over GF(q) is encoded by $\mathbf{C}_2$, where $R_1 = r_1 \times m$. For example, we can construct the concatenated code using a RS code over GF(2^8) as $\mathbf{C}_1$ and a binary LDPC code as $\mathbf{C}_2$, by which bursty decoding failure of the LDPC code $\mathbf{C}_2$ can be corrected using the RS code $\mathbf{C}_1$.

6.2 Three-level coding for solid-state drive

The following outlines a three-level error control coding suitable for the SSD (Kaneko et al., 2008), where the SSD is assumed to have N memory chips accessed in parallel. A *cluster* is defined as a group of N pages stored in the N memory chips, where the pages have same memory address, and is read or stored simultaneously. Let $(\mathbf{D}_0, \mathbf{D}_1, \ldots, \mathbf{D}_{N-2})$ be the information word, where $\mathbf{D}_i$ is a binary $k \times b$ matrix. This information word is encoded as follows.

1. *First level coding:* Generate a parity-check segment as $\mathbf{P} = \mathbf{D}_0 \oplus \mathbf{D}_1 \oplus \cdots \oplus \mathbf{D}_{N-2}$, where $\mathbf{P}$ is a binary $k \times b$ matrix and $\oplus$ denotes matrix addition over GF(2).
2. *Second level coding:* Let $\mathbf{d} = (\mathbf{d}_0, \mathbf{d}_1, \ldots, \mathbf{d}_{N-2}, \mathbf{p})$ be a binary row vector with length kN, where $\mathbf{d}_i = (\mathbf{d}_{i,0} \oplus \mathbf{d}_{i,1} \oplus \cdots \oplus \mathbf{d}_{i,b-1})^T$ and $\mathbf{p} = (\mathbf{p}_0 \oplus \mathbf{p}_1 \oplus \cdots \oplus \mathbf{p}_{b-1})^T$. Encode $\mathbf{d}$ by the code C_{CL} to generate the shared-check segment $\mathbf{Q} = (\mathbf{Q}_0, \mathbf{Q}_1, \ldots, \mathbf{Q}_{N-1})$ having $r_0 bN$ bits, where $\mathbf{Q}_i = [\ \mathbf{q}_{i,0}\mathbf{q}_{i,1}\ldots\mathbf{q}_{i,b-1}\ ]$ is a binary $r_0 \times b$ matrix for $i \in \{0, 1, \ldots, N-1\}$. Here, the check bits of C_{CL} are expressed as a row vector with length $r_0 bN$ bits, that is, $(\mathbf{q}_{0,0}^T, \mathbf{q}_{0,1}^T, \ldots, \mathbf{q}_{0,b-1}^T, \mathbf{q}_{1,0}^T, \ldots, \mathbf{q}_{N-1,b-1}^T)$. Then, for $i \in \{0, 1, \ldots, N-2\}$, append $\mathbf{Q}_i$ to the bottom of $\mathbf{D}_i$, and also append $\mathbf{Q}_{N-1}$ to the bottom of $\mathbf{P}$.

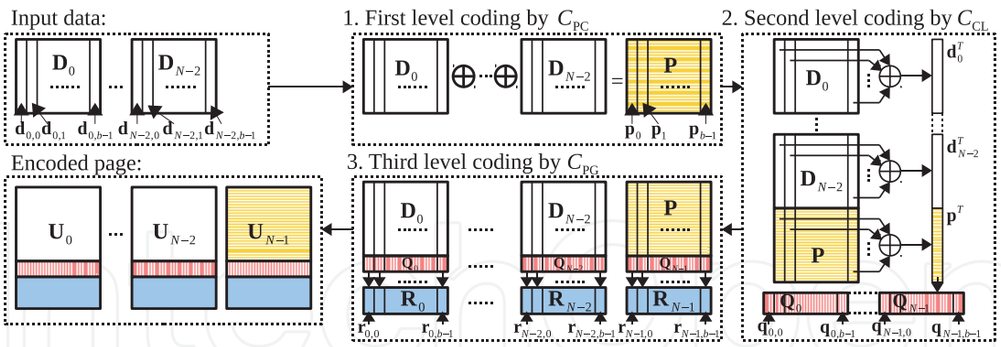


Fig. 12. Encoding process of three level ECC for SSD.

3. *Third level coding:* For $i \in \{0, 1, \ldots, N-2\}$ and $j \in \{0, 1, \ldots, b-1\}$, encode $\begin{pmatrix} \mathbf{d}_{i,j} \\ \mathbf{q}_{i,j} \end{pmatrix}$ by code C_{PG} to generate check bits $\mathbf{r}_{i,j}$, where $\mathbf{d}_{i,j}$, $\mathbf{q}_{i,j}$, and $\mathbf{r}_{i,j}$ are binary column vectors with lengths k, r_0, and r_1, respectively. Similarly, for $j \in \{0, 1, \ldots, b-1\}$, encode $\begin{pmatrix} \mathbf{p}_j \\ \mathbf{q}_{N-1,j} \end{pmatrix}$ by the code C_{PG} to generate check bits $\mathbf{r}_{N-1,j}$, where $\mathbf{p}_j$, $\mathbf{q}_{N-1,j}$, and $\mathbf{r}_{N-1,j}$ are binary column vectors with lengths k, r_0, and r_1, respectively.

The above encoding process generates encoded page $\mathbf{U}_i$ as shown in Fig. 12.

7. References

Lin, S. & Costello, D. J. Jr. (2004). *Error Control Coding*, Pearson Prentice Hall, 0-13-042672-5, New Jersey.

Fujiwara, E. (2006). *Code Design for Dependable Systems –Theory and Practical Applications–*, Wiley-Interscience, 0-471-75618-0, New Jersey.

Muroke, P. (2006). Flash Memory Field Failure Mechanisms, *Proc. 44th Annual International Reliability Physics Symposium*, pp. 313–316, San Jose, March 2006, IEEE, New Jersey.

Mohammad, M. G.; Saluja, K. K. & Yap, A. S. (2001). Fault Models and Test Procedures for Flash Memory Disturbances, *Journal of Electronic Testing: Theory and Applications*, Vol. 17, pp. 495–508, 2001.

Mielke, N.; Marquart, T.; Wu, N.; Kessenich, J.; Belgal, H.; Schares, E.; Trivedi, F.; Goodness, E. & Nevill, L. R. (2008). Bit Error Rate in NAND Flash Memories, *Proc. 46th Annual International Reliability Physics Symposium*, pp. 9–19, Phenix, 2008, IEEE, New Jersey.

Ielmini, D.; Spinelli, A. S. & Lacaita, A. L. (2005). Recent Developments on Flash Memory Reliability, *Microelectronic Engineering*, Vol. 80, pp. 321–328, 2005.

Chimenton, A.; Pellati, P. & Olivo, P. (2003). Overerase Phenomena: An Insight Into Flash Memory Reliability, *Proceedings of the IEEE*, Vol. 91, no. 4, pp. 617–626, April 2003.

Claeys, C.; Ohyama, H.; Simoen, E.; Nakabayashi, M. and Kobayashi, K, (2002). Radiation Damage in Flash Memory Cells, *Nuclear Instruments and Methods in Physics Research B*, Vol. 186, pp. 392–400, Jan. 2002.

Oldham, T. R.; Friendlich, M.; Howard, Jr., J. W.; Berg, M. D.; Kim, H. S.; Irwin, T. L. & LaBel, K. A. (2007). TID and SER Response of an Advanced Samsung 4Gb NAND Flash Memory, *Proc. IEEE Radiation Effects Data Workshop on Nuclear and Space Radiation Effect Conf*, pp. 221–225, July 2007.

Bagatin, M.; Cellere, G.; Gerardin, S.; Paccagnella, A.; Visconti, A. & Beltrami, S. (2009). TID Sensitivity of NAND Flash Memory Building Blocks, *IEEE Trans. Nuclear Science*, Vol. 56, No. 4, pp. 1909–1913, Aug. 2009.

Witzke, K. A. & Leung, C. (1985). A Comparison of Some Error Detecting CRC Code Standards, *IEEE Trans. Communications*, Vol. 33, No. 9, pp. 996–998, Sept. 1985.

Gallager, R. G (1962). Low Density Parity Check Codes, *IRE Trans. Information Theory*, Vol. 8, pp. 21–28, Jan. 1962.

Moreira, J. C. & Farrell, P. G. (2006). *Essentials of Error-Control Coding,* Wiley, 0-470-02920-X, West Sussex.

Richardson, T. J.; Shokrollahi, M. A. & Urbanke, R. L. (2001). Design of Capacity-Approaching Irregular Low-Density Parity-Check Codes, *IEEE Trans. Information Theory,* Vol. 47, No. 2, pp.619–637, Feb. 2001.

IEEE Std 802.11n-2009, Oct. 2009.

IEEE Std 802.16-2009, May 2009.

IEEE Std 802.15.3c-2009, Oct. 2009.

Davey, M. C. & MacKay, D. (1998). Low-Density Parity-Check Codes over GF(q), *IEEE Communications Letters,* Vol. 2, No. 6, pp. 165–167, June 1998.

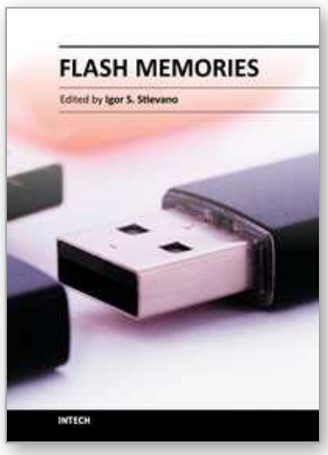
Zhou, B.; Kang, J.; Tai, Y. Y.; Lin, S. & Ding, Z. (2009) High Performance Non-Binary Quasi-Cyclic LDPC Codes on Euclidean Geometry, *IEEE Trans. Communications,* Vol. 57, No. 5, pp. 1298–1311, May 2009.

Li, G.; Fair, I, J. & Krzymien, W. A. (2009). Density Evolution for Nonbinary LDPC Codes Under Gaussian Approximation, *IEEE Trans. Information Theory,* Vol. 55, No. 3, pp. 997–1015, March 2009.

Song, H. & Cruz, J. R. (2003). Reduced-Complexity Decoding of *Q*-Ary LDPC codes for Magnetic Decoding, *IEEE Trans. Magnetics,* Vol. 39, No. 3, pp. 1081–1087, March 2003.

Maeda, Y. & Kaneko, H. (2009). Error Control Coding for Multilevel Cell Flash Memories Using Nonbinary Low-Density Parity-Check Codes, *Proc. IEEE Int. Symp. Defect and Fault Tolerance in VLSI Systems,* pp. 367–375, Oct. 2009.

Kaneko, H.; Matsuzaka, T. & Fujiwara, E. (2008). Three-Level Error Control Coding for Dependable Solid-State Drives. *Proc. IEEE Pacific Rim International Symposium on Dependable Computing,* pp. 281–288, Dec. 2008.



Flash Memories

Edited by Prof. Igor Stievano

ISBN 978-953-307-272-2
Hard cover, 262 pages
Publisher InTech
Published online 06, September, 2011
Published in print edition September, 2011

Flash memories and memory systems are key resources for the development of electronic products implementing converging technologies or exploiting solid-state memory disks. This book illustrates state-of-the-art technologies and research studies on Flash memories. Topics in modeling, design, programming, and materials for memories are covered along with real application examples.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Haruhiko Kaneko (2011). Error Control Coding for Flash Memory, Flash Memories, Prof. Igor Stievano (Ed.), ISBN: 978-953-307-272-2, InTech, Available from: http://www.intechopen.com/books/flash-memories/error-control-coding-for-flash-memory

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821

IntechOpen