CHAPTER 4

Important Linear Block Codes

This chapter presents several classes of important linear block codes that were discovered in the early days of error-correcting codes. The first class of linear block codes for error correction was discovered by Richard W. Hamming in 1950 [1], two years after Shannon published his landmark paper, which asserted that by proper encoding of information, errors induced by a noisy channel or storage medium can be reduced to any desired level without sacrificing the rate of information transmission or storage. Hamming codes have a minimum distance of 3 and therefore are capable of correcting any single error over the span of the code block length. The weight enumerator of Hamming codes is known. Hamming codes are *perfect codes* and can be decoded easily using a table-lookup scheme. Good codes with a minimum distance of 4 for single-error correction and double-error detection can be obtained by properly shortening the Hamming codes. Hamming codes and their shortened versions have been widely used for error control in digital communication and data storage systems over the years owing to their high rates and decoding simplicity.

The second class of linear block codes constructed in the early days for error correction and detection was the class of Reed–Muller codes. Reed–Muller codes were first discovered by David E. Muller in 1954 [9] for switching-circuit design and error detection. It was Irwin S. Reed, also in 1954 [10], who reformulated these codes for error correction and detection in communication and data storage systems and devised the first decoding algorithm for these codes. Reed–Muller codes form a large class of codes for multiple random error correction. These codes are simple in construction and rich in structural properties. They can be decoded in many ways using either hard-decision or soft-decision decoding algorithms. The Reed decoding algorithm for Reed–Muller codes is a majority-logic decoding algorithm that can easily be implemented. Several soft-decision decoding algorithms for Reed–Muller codes have been devised that achieve very good error performance with low decoding complexity.

Also presented in this chapter is the (24, 12) Golay code with minimum distance 8. This code has many interesting structural properties and has been extensively studied by many coding theorists and mathematicians. It has been used for error control in many communication systems, especially by U.S. space communication programs.

Several additional code construction methods are also presented in this chapter. These construction methods allow us to construct long, powerful codes from short component codes.

4.1 HAMMING CODES

For any positive integer $m \geq 3$, there exists a Hamming code with the following parameters:

Code length:	$n = 2^m - 1$,
Number of information symbols:	$k = 2^m - m - 1$,
Number of parity-check symbols:	$n - k = m$,
Error-correcting capability:	$t = 1$ $(d_{min} = 3)$.

The parity-check matrix $\mathbb{H}$ of this code consists of all the nonzero m-tuples as its columns. In systematic form, the columns of $\mathbb{H}$ are arranged in the following form:

$$\mathbb{H} = [\ \mathbb{I}_m \quad \mathbb{Q}\ ],$$

where $\mathbb{I}_m$ is an $m \times m$ identity matrix, and the submatrix $\mathbb{Q}$ consists of $2^m - m - 1$ columns that are the m-tuples of weight 2 or more. For example, let $m = 3$. The parity-check matrix of a Hamming code of length 7 can be put in the form

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

which is the parity-check matrix of the (7, 4) linear code given in Table 3.1 (see Example 3.3). Hence, the code given in Table 3.1 is a Hamming code. The columns of $\mathbb{Q}$ may be arranged in any order without affecting the distance property and weight distribution of the code. In systematic form, the generator matrix of the code is

$$\mathbb{G} = [\ \mathbb{Q}^T \quad \mathbb{I}_{2^m-m-1}\ ],$$

where $\mathbb{Q}^T$ is the transpose of $\mathbb{Q}$, and $\mathbb{I}_{2^m-m-1}$ is a $(2^m - m - 1) \times (2^m - m - 1)$ identity matrix.

Because the columns of $\mathbb{H}$ are nonzero and distinct, no two columns add to zero. It follows from Corollary 3.2.1 that the minimum distance of a Hamming code is at least 3. Since $\mathbb{H}$ consists of all the nonzero m-tuples as its columns, the vector sum of any two columns, say $\mathbb{h}_i$ and $\mathbb{h}_j$, must also be a column in $\mathbb{H}$, say $\mathbb{h}_l$. Thus,

$$\mathbb{h}_i + \mathbb{h}_j + \mathbb{h}_l = 0.$$

It follows from Corollary 3.2.2 that the minimum distance of a Hamming code is exactly 3. Hence, the code is capable of correcting all the error patterns with a single error or detecting all the error patterns of two or fewer errors.

If we form the standard array for the Hamming code of length $2^m - 1$, all the $(2^m - 1)$-tuples of weight 1 can be used as coset leaders (Theorem 3.5). The number of $(2^m - 1)$-tuples of weight 1 is $2^m - 1$. Because $n - k = m$, the code has 2^m cosets. Thus, the zero vector $\mathbb{0}$ and the $(2^m - 1)$-tuples of weight 1 form all the coset leaders of the standard array. Thus, a Hamming code corrects only error patterns of single error and no others. This is a very interesting structure. A t-error-correcting code is called a *perfect code* if its standard array has all the

error patterns of t or fewer errors and no others as coset leaders. Thus, Hamming codes form a class of single-error-correcting perfect codes [2]. Perfect codes are rare. Besides the Hamming codes, the only other nontrivial binary perfect code is the (23, 12) Golay code (see Section 5.9) [3–5].

Hamming codes can easily be decoded with the table-lookup scheme described in Section 3.5. The decoder for a Hamming code of length $2^m - 1$ can be implemented in the same manner as that for the (7, 4) Hamming code given in Example 3.9.

We may delete any l columns from the parity-check matrix $\mathbb{H}$ of a Hamming code. This deletion results in an $m \times (2^m - l - 1)$ matrix $\mathbb{H}'$. Using $\mathbb{H}'$ as a parity-check matrix, we obtain a shortened Hamming code with the following parameters:

Code length:	$n = 2^m - l - 1$
Number of information symbols:	$k = 2^m - m - l - 1$
Number of parity-check symbols:	$n - k = m$
Error-correcting capability:	$d_{min} \geq 3$.

If we delete columns from $\mathbb{H}$ properly, we may obtain a shortened Hamming code with a minimum distance of 4. For example, if we delete from the submatrix $\mathbb{Q}$ all the columns of even weight, we obtain an $m \times 2^{m-1}$ matrix,

$$\mathbb{H}' = \begin{bmatrix} \mathbb{I}_m & \mathbb{Q}' \end{bmatrix},$$

where $\mathbb{Q}'$ consists of $2^{m-1} - m$ columns of odd weight. Because all the columns of $\mathbb{H}'$ have odd weight, no three columns add to zero; however, for a column $\mathbb{h}_i$ of weight 3 in $\mathbb{Q}'$, there exist three columns $\mathbb{h}_j$, $\mathbb{h}_l$, and $\mathbb{h}_s$ in $\mathbb{I}_m$ such that $\mathbb{h}_i + \mathbb{h}_j + \mathbb{h}_l + \mathbb{h}_s = \mathbb{0}$. Thus, the shortened Hamming code with $\mathbb{H}'$ as a parity-check matrix has a minimum distance of exactly 4.

The distance-4 shortened Hamming code can be used for correcting all error patterns of single error and simultaneously detecting all error patterns of double errors. When a single error occurs during the transmission of a code vector, the resultant syndrome is nonzero, and it contains an odd number of 1's; however, when double errors occur, the syndrome is also nonzero, but it contains an even number of 1's. Based on these facts, decoding can be accomplished as follows:

1. If the syndrome s is zero, we assume that no error occurred.
2. If s is nonzero and it contains an odd number of 1's, we assume that a single error occurred. The error pattern of a single error that corresponds to s is added to the received vector for error correction.
3. If s is nonzero and it contains an even number of 1's, an uncorrectable error pattern has been detected.

The weight distribution of a Hamming code of length $n = 2^m - 1$ is known [2]. The number of code vectors of weight i, A_i, is simply the coefficient of z^i in the expansion of the following polynomial:

$$A(z) = \frac{1}{n+1}\{(1+z)^n + n(1-z)(1-z^2)^{(n-1)/2}\}. \tag{4.1}$$

This polynomial is the weight enumerator for the Hamming codes.

EXAMPLE 4.1

Let $m = 3$. Then, $n = 2^3 - 1 = 7$, and the weight enumerator for the $(7, 4)$ Hamming code is

$$A(z) = \tfrac{1}{8}\{(1+z)^7 + 7(1-z)(1-z^2)^3\} = 1 + 7z^3 + 7z^4 + z^7.$$

Hence, the weight distribution for the $(7, 4)$ Hamming code is $A_0 = 1$, $A_3 = A_4 = 7$, and $A_7 = 1$.

The dual code of a $(2^m - 1, 2^m - m - 1)$ Hamming code is a $(2^m - 1, m)$ linear code. This code has a very simple weight distribution; it consists of the all-zero codeword and $2^m - 1$ codewords of weight 2^{m-1}. Thus, its weight enumerator is

$$B(z) = 1 + (2^m - 1)z^{2^{m-1}}. \tag{4.2}$$

If a Hamming code is used for error detection over a BSC, its probability of an undetected error, $P_u(E)$, can be computed either from (3.35) and (4.1) or from (3.36) and (4.2). Computing $P_u(E)$ from (3.36) and (4.2) is easier. Combining (3.36) and (4.2), we obtain

$$P_u(E) = 2^{-m}\{1 + (2^m - 1)(1-2p)^{2^{m-1}}\} - (1-p)^{2^m - 1}. \tag{4.3}$$

The probability $P_u(E)$ for Hamming codes does satisfy the upper bound $2^{-(n-k)} = 2^{-m}$ for $p \leq \frac{1}{2}$ (i.e., $P_u(E) \leq 2^{-m}$) [6, 7], as can be shown by using the expression of (4.3) (see Problem 4.3). Therefore, Hamming codes are good error-detection codes.

4.2 A CLASS OF SINGLE-ERROR-CORRECTING AND DOUBLE-ERROR-DETECTING CODES

Single-error-correcting and double-error-detecting (SEC–DED) codes have been widely used for error control in communication and computer memory systems. In this section we present a class of SEC–DED codes that are suitable and commonly used for improving computer memory reliability. This class of codes was constructed by Hsiao [8]. The most important feature of this class of codes is their fast encoding and error detection in the decoding process, which are the most critical on-line processes in memory operations.

SEC–DED codes that have the features described can be constructed by shortening Hamming codes presented in the previous section. Construction begins with a Hamming code of length $2^m - 1$ and minimum distance 3. Let H be its parity-check matrix. Delete columns from H such that the new parity-check matrix $\mathbf{H}_0$ satisfies the following requirements:

1. Every column should have an odd number of 1's.
2. The total number of 1's in the $\mathbf{H}_0$ matrix should be a minimum.
3. The number of 1's in each row of $\mathbf{H}_0$ should be made equal, or as close as possible, to the average number (i.e., the total number of 1's in $\mathbf{H}_0$ divided by the number of rows).

TABLE 4.1: Parameters of a list of Hsiao's codes.

n	k	$n-k$	Total number of 1's in H	Average number of 1's per row
12	8	4	16	4
14	9	5	32	6.4
15	10	5	35	7
16	11	5	40	8
22	16	6	54	9
26	20	6	66	11
30	24	6	86	14.3
39	32	7	103	14.7
43	36	7	117	16.7
47	40	7	157	22.4
55	48	7	177	25.3
72	64	8	216	27
80	72	8	256	32
88	80	8	296	37
96	88	8	336	42
104	96	8	376	47
112	104	8	416	52
120	112	8	456	57
128	120	8	512	64
130	121	9	446	49.6
137	128	9	481	53.5

The first requirement guarantees the code generated by $\mathbb{H}_0$ has a minimum distance of at least 4. Therefore, it can be used for single-error correction and double-error detection. The second and third requirements yields minimum logic levels in forming parity or syndrome bits, and less hardware in implementation of the code. Hsiao [8] provided an algorithm to construct $\mathbb{H}_0$ and found some optimal SEC–DED codes. Several parity-check matrices in systematic form for message (or data) lengths of 16, 32, and 64 are given in Figure 4.1. The parameters of a list of Hsiao's codes are given in Table 4.1.

In computer applications, these codes are encoded and decoded in parallel manner. In encoding, the message bits enter the encoding circuit in parallel, and the parity-check bits are formed simultaneously. In decoding, the received bits enter the decoding circuit in parallel, the syndrome bits are formed simultaneously, and the received bits are corrected in parallel. Single-error correction is accomplished by the table-lookup decoding described in Example 3.9. Double-error detection is accomplished by examining the number of 1's in the syndrome vector s. If the syndrome s contains an even number of 1's, then either a double-error pattern or a multiple-even-error pattern has occurred.

Consider the parity-check matrix of the (72, 64) SEC–DED code given in Figure 4.1(c). Each row contains 27 ones. If three-input X-OR gates are used to form syndrome bits, each syndrome bit is formed by a three-level X-OR tree with

$$\mathbf{H}_0 = \begin{bmatrix} 1000001001100100111100 \\ 0100000011111010001010 \\ 0010001110111001100000 \\ 0001001110000111010001 \\ 0000100001001111000111 \\ 0000010100010000111111 \end{bmatrix}$$

(a)

$$\mathbf{H}_0 = \begin{bmatrix} 100000010001010100000100000111100011011 \\ 010000000001000000011111011100010110000 1 \\ 001000000010110111100001001001010100110 \\ 000100011111111000000011010010001000100 \\ 000010001101100111111110000100000001000 \\ 000001000100001001001001111111110010000 \\ 000000111000001010010000100000011111111 \end{bmatrix}$$

(b)

$$\mathbf{H}_0 = \left[\begin{array}{c|c|c|c|c|c|c|c|c} 01000000 & 01100100 & 11111111 & 00000111 & 00111000 & 11001000 & 00001000 & 00001001 & 10010010 \\ 00100000 & 10010010 & 01100100 & 11111111 & 00000111 & 00111000 & 11001000 & 00001000 & 00001001 \\ 00010000 & 00001001 & 10010010 & 01100100 & 11111111 & 00000111 & 00111000 & 11001000 & 00001000 \\ 00001000 & 00001000 & 00001001 & 10010010 & 01100100 & 11111111 & 00000111 & 00111000 & 00001000 \\ 00000100 & 11001000 & 00001000 & 00001001 & 10010010 & 01100100 & 11111111 & 00000111 & 11001000 \\ 00000010 & 00111000 & 11001000 & 00001000 & 00001001 & 10010010 & 01100100 & 11111111 & 00000111 \\ 00000001 & 00000111 & 00111000 & 11001000 & 00001000 & 00001001 & 10010010 & 01100100 & 11111111 \end{array}\right]$$

(c)

$$\mathbf{H}_0 = \left[\begin{array}{c|c|c|c|c|c|c|c|c} 10000000 & 11111111 & 00001111 & 00001111 & 00001100 & 01101000 & 10001000 & 10001000 & 10000000 \\ 01000000 & 11110000 & 11111111 & 00000000 & 11110011 & 01100100 & 01000100 & 01000100 & 01000000 \\ 00100000 & 00110000 & 11110000 & 11111111 & 00001111 & 00000010 & 00100010 & 00100010 & 00100110 \\ 00010000 & 11001111 & 00000000 & 11110000 & 11111111 & 00000001 & 00010001 & 00010001 & 00010110 \\ 00001000 & 01101000 & 10001000 & 10001000 & 10000000 & 11111111 & 00001111 & 00000000 & 11110011 \\ 00000100 & 01100100 & 01000100 & 01000100 & 01000000 & 11110000 & 11111111 & 00001111 & 00001100 \\ 00000010 & 00000010 & 00100010 & 00100010 & 00100110 & 11001111 & 00000000 & 11111111 & 00001111 \\ 00000001 & 00000001 & 00010001 & 00010001 & 00010110 & 00110000 & 11110000 & 11110000 & 11111111 \end{array}\right]$$

(d)

FIGURE 4.1: (a) Parity-check matrix of an optimal (22, 16) SEC–DED code; (b) parity-check matrix of an optimal (39, 32) SEC–DED code; (c) parity-check matrix of an optimal (72, 64) SEC–DED code; (d) parity-check matrix of another optimal (72, 64) SEC–DED code.

13 gates. The eight X-OR trees for generating the eight syndrome bits are identical. These provide uniform and minimum delay in the error-correction process.

4.3 REED–MULLER CODES

Reed–Muller (RM) codes form a class of multiple-error-correction codes. These codes were discovered by Muller in 1954 [9], but the first decoding algorithm for these codes was devised by Reed, also in 1954 [10]. They are simple in construction and rich in structural properties. They can be decoded easily in several ways using either hard- or soft-decision algorithms.

For any integers m and r with $0 \leq r \leq m$, there exists a binary rth-order RM code, denoted by RM(r, m), with the following parameters:

Code length: $n = 2^m$,
Dimension: $k(r, m) = 1 + \binom{m}{1} + \binom{m}{2} + \cdots + \binom{m}{r}$,
Minimum distance: $d_{min} = 2^{m-r}$,

where $\binom{m}{i} = \frac{m!}{(m-i)!i!}$ is the binomial coefficient. For example, let $m = 5$ and $r = 2$. Then, $n = 32$, $k(2, 5) = 16$, and $d_{min} = 8$. There exists a (32, 16) RM code with a minimum distance of 8.

For $1 \leq i \leq m$, let $\mathbb{v}_i$ be a 2^m-tuple over $GF(2)$ of the following form:

$$\mathbb{v}_i = (\underbrace{0 \cdots 0}_{2^{i-1}}, \underbrace{1 \cdots 1}_{2^{i-1}}, \underbrace{0 \cdots 0}_{2^{i-1}}, \cdots\cdots, \underbrace{1 \cdots 1}_{2^{i-1}}) \tag{4.4}$$

which consists of 2^{m-i+1} alternating all-zero and all-one 2^{i-1}-tuples. For $m = 4$, we have the following four 16-tuples:

$$\mathbb{v}_4 = (0\,0\,0\,0\,0\,0\,0\,0\,1\,1\,1\,1\,1\,1\,1\,1),$$
$$\mathbb{v}_3 = (0\,0\,0\,0\,1\,1\,1\,1\,0\,0\,0\,0\,1\,1\,1\,1),$$
$$\mathbb{v}_2 = (0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1\,0\,0\,1\,1),$$
$$\mathbb{v}_1 = (0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1\,0\,1).$$

Let $\mathbb{a} = (a_0, a_1, \cdots, a_{n-1})$ and $\mathbb{b} = (b_0, b_1, \cdots, b_{n-1})$ be two binary n-tuples. We define the following logic (Boolean) product of $\mathbb{a}$ and $\mathbb{b}$:

$$\mathbb{a} \cdot \mathbb{b} \triangleq (a_0 \cdot b_0, a_1 \cdot b_1, \cdots, a_{n-1} \cdot b_{n-1}),$$

where "$\cdot$" denotes the logic product (or AND operation), i.e., $a_i \cdot b_i = 1$ if and only if $a_i = b_1 = 1$. For $m = 4$,

$$\mathbb{v}_3 \cdot \mathbb{v}_2 = (0\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,0\,0\,1\,1).$$

For simplicity, we use $\mathbb{ab}$ for $\mathbb{a} \cdot \mathbb{b}$.

Let $\mathbb{v}_0$ denote the all-one 2^m-tuple, $\mathbb{v}_0 = (1, 1, \cdots, 1)$. For $1 \leq i_1 < i_2 < \cdots < i_l \leq m$, the product vector

$$\mathbb{v}_{i_1}\mathbb{v}_{i_2}\cdots\mathbb{v}_{i_l}$$

is said to have degree l. Because the weights of $\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_m$ are even and powers of 2, the weight of the product $\mathbf{v}_{i_1}\mathbf{v}_{i_2}\cdots\mathbf{v}_{i_l}$ is also even and a power of 2, in fact, 2^{m-l}.

The rth-order RM code, RM(r, m), of length 2^m is generated (or spanned) by the following set of independent vectors:

$$G_{\mathrm{RM}}(r, m) = \{\mathbf{v}_0, \mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_m, \mathbf{v}_1\mathbf{v}_2, \mathbf{v}_1\mathbf{v}_3, \cdots, \mathbf{v}_{m-1}\mathbf{v}_m, \cdots, \text{up to products of degree } r\}. \tag{4.5}$$

There are

$$k(r, m) = 1 + \binom{m}{1} + \binom{m}{2} + \cdots + \binom{m}{r}$$

vectors in $G_{\mathrm{RM}}(r, m)$. Therefore, the dimension of the code is $k(r, m)$.

If the vectors in $G_{\mathrm{RM}}(r, m)$ are arranged as rows of a matrix, then the matrix is a generator matrix of the RM(r, m) code. Hereafter, we use $G_{\mathrm{RM}}(r, m)$ as the generator matrix. For $0 \le l \le r$, there are exactly $\binom{m}{l}$ rows in $G_{\mathrm{RM}}(r, m)$ of weight 2^{m-l}. Because all the vectors in $G_{\mathrm{RM}}(r, m)$ are of even weight, all the codewords in the RM(r, m) code have even weight. From the code construction we readily see that the RM$(r-1, m)$ code is a proper subcode of the RM(r, m) code. Hence, we have the following inclusion chain:

$$\mathrm{RM}(0, m) \subset \mathrm{RM}(1, m) \subset \cdots \subset \mathrm{RM}(r, m). \tag{4.6}$$

Furthermore, RM codes have the following structural property: the $(m-r-1)$th-order RM code, RM$(m-r-1, m)$, is the dual code of the rth-order RM code, RM(r, m) (see Problem 4.9). The zeroth-order RM code is a repetition code and the $(m-1)$th-order RM code is a single-parity-check code.

EXAMPLE 4.2

Let $m = 4$ and $r = 2$. The second-order RM code of length $n = 16$ is generated by the following 11 vectors:

$\mathbf{v}_0$	1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
$\mathbf{v}_4$	0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
$\mathbf{v}_3$	0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1
$\mathbf{v}_2$	0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1
$\mathbf{v}_1$	0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1
$\mathbf{v}_3\mathbf{v}_4$	0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1
$\mathbf{v}_2\mathbf{v}_4$	0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 1
$\mathbf{v}_1\mathbf{v}_4$	0 0 0 0 0 0 0 0 0 1 0 1 0 1 0 1
$\mathbf{v}_2\mathbf{v}_3$	0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1
$\mathbf{v}_1\mathbf{v}_3$	0 0 0 0 0 1 0 1 0 0 0 0 0 1 0 1
$\mathbf{v}_1\mathbf{v}_2$	0 0 0 1 0 0 0 1 0 0 0 1 0 0 0 1

This is a (16, 11) code with a minimum distance of 4.

With the preceding construction, the generator matrix $G_{\mathrm{RM}}(r, m)$ of the RM(r, m) code is not in systematic form. It can be put in systematic form with

elementary row and column operations; however, RM codes in nonsystematic form have many interesting and useful structures that reduce decoding complexity. This topic will be discussed in a later chapter.

The Reed decoding algorithm for RM codes is best explained by an example. Consider the second-order RM code RM(2, 4) of length 16 given in Example 4.2. Suppose

$$(a_0, a_4, a_3, a_2, a_1, a_{34}, a_{24}, a_{14}, a_{23}, a_{13}, a_{12})$$

is the message to be encoded. Then, the corresponding codeword is

$$\begin{aligned}(b_0, b_1, b_2, \cdots, b_{15}) = {} & a_0\mathbb{v}_0 + a_4\mathbb{v}_4 + a_3\mathbb{v}_3 + a_2\mathbb{v}_2 + a_1\mathbb{v}_1 \\ & + a_{34}\mathbb{v}_3\mathbb{v}_4 + a_{24}\mathbb{v}_2\mathbb{v}_4 + a_{14}\mathbb{v}_1\mathbb{v}_4 \\ & + a_{23}\mathbb{v}_2\mathbb{v}_3 + a_{13}\mathbb{v}_1\mathbb{v}_3 + a_{12}\mathbb{v}_1\mathbb{v}_2.\end{aligned}$$

Note that the sum of the first four components of each generator vector is zero except the vector $\mathbb{v}_1\mathbb{v}_2$. The same is true for the other three groups of four consecutive components. As a result, we have the following four sums that relate the information bit a_{12} to the code bits:

$$\begin{aligned}a_{12} &= b_0 + b_1 + b_2 + b_3, \\ a_{12} &= b_4 + b_5 + b_6 + b_7, \\ a_{12} &= b_8 + b_9 + b_{10} + b_{11}, \\ a_{12} &= b_{12} + b_{13} + b_{14} + b_{15}.\end{aligned}$$

These four sums give four independent determinations (or reconstructions) of the information bit a_{12} from the code bits. If the codeword $(b_0, b_1, \cdots, b_{15})$ is transmitted and there is a single transmission error in the received vector, the error can affect only one determination of a_{12}. As a result, the other three (*majority*) determinations of a_{12} will give the correct value of a_{12}. This is the basis for decoding RM codes.

Let $\mathbb{r} = (r_0, r_1, \cdots, r_{15})$ be the received vector. In decoding a_{12}, we form the following sums:

$$\begin{aligned}A_1 &= r_0 + r_1 + r_2 + r_3, \\ A_2 &= r_4 + r_5 + r_6 + r_7, \\ A_3 &= r_8 + r_9 + r_{10} + r_{11}, \\ A_4 &= r_{12} + r_{13} + r_{14} + r_{15}.\end{aligned}$$

which are obtained by replacing the code bits with the corresponding received bits in the four independent determinations of a_{12}. These sums are called *check-sums*, which are simply the estimates of a_{12}. Then, a_{12} is decoded based on the following *majority-logic decision rule*: a_{12} is taken to be equal to the value assumed by the majority in $\{A_1, A_2, A_3, A_4\}$. If there is a tie, a random choice of the value of a_{12} is made. It is clear that if there is only one error in the received vector, a_{12} is always decoded correctly.

Similar independent determinations of information bits $a_{13}, a_{23}, a_{14}, a_{24}$, and a_{34} can be made from the code bits. For example, the four independent determinations of a_{13} are:

$$a_{13} = b_0 + b_1 + b_4 + b_5,$$
$$a_{13} = b_2 + b_3 + b_6 + b_7,$$
$$a_{13} = b_8 + b_9 + b_{12} + b_{13},$$
$$a_{13} = b_{10} + b_{11} + b_{14} + b_{15}.$$

At the decoder, we decode a_{13} by forming the following four check-sums from the received bits using the preceding four independent determinations of a_{13}:

$$A_1 = r_0 + r_1 + r_4 + r_5,$$
$$A_2 = r_2 + r_3 + r_6 + r_7,$$
$$A_3 = r_8 + r_9 + r_{12} + r_{13},$$
$$A_4 = r_{10} + r_{11} + r_{14} + r_{15}.$$

From these four check-sums we use the majority-logic decision rule to decode a_{13}. If there is a single transmission error in the received sequence, the information bits $a_{12}, a_{13}, a_{23}, a_{14}, a_{24}$, and a_{34} will be decoded correctly.

After the decoding of $a_{12}, a_{13}, a_{23}, a_{14}, a_{24}$, and a_{34}, the vector

$$a_{34}\mathbf{v}_3\mathbf{v}_4 + a_{24}\mathbf{v}_2\mathbf{v}_4 + a_{14}\mathbf{v}_1\mathbf{v}_4 + a_{23}\mathbf{v}_2\mathbf{v}_3 + a_{13}\mathbf{v}_1\mathbf{v}_3 + a_{12}\mathbf{v}_1\mathbf{v}_2$$

is subtracted from $\mathbf{r}$. The result is a modified received vector:

$$\begin{aligned}\mathbf{r}^{(1)} &= \left(r_0^{(1)}, r_1^{(1)}, \cdots, r_{15}^{(1)}\right) \\ &= \mathbf{r} - a_{34}\mathbf{v}_3\mathbf{v}_4 - a_{24}\mathbf{v}_2\mathbf{v}_4 - a_{14}\mathbf{v}_1\mathbf{v}_4 - a_{23}\mathbf{v}_2\mathbf{v}_3 - a_{13}\mathbf{v}_1\mathbf{v}_3 - a_{12}\mathbf{v}_1\mathbf{v}_2.\end{aligned}$$

In the absence of errors, $\mathbf{r}^{(1)}$ is simply the following codeword:

$$a_0\mathbf{v}_0 + a_4\mathbf{v}_4 + a_3\mathbf{v}_3 + a_2\mathbf{v}_2 + a_1\mathbf{v}_1 = (b_0^{(1)}, b_1^{(1)}, \cdots, b_{15}^{(1)}).$$

We note that starting from the first component, the sums of every two consecutive components in $\mathbf{v}_0, \mathbf{v}_4, \mathbf{v}_3$, and $\mathbf{v}_2$ are zero; however, the sum of every two consecutive components of $\mathbf{v}_1$ is equal to 1. As a consequence, we can form the following eight independent determinations of the information bit a_1 from the code bits $b_0^{(1)}$ through $b_{15}^{(1)}$:

$$\begin{aligned} a_1 &= b_0^{(1)} + b_1^{(1)}, \quad a_1 = b_8^{(1)} + b_9^{(1)}, \\ a_1 &= b_2^{(1)} + b_3^{(1)}, \quad a_1 = b_{10}^{(1)} + b_{11}^{(1)}, \\ a_1 &= b_4^{(1)} + b_5^{(1)}, \quad a_1 = b_{12}^{(1)} + b_{13}^{(1)}, \\ a_1 &= b_6^{(1)} + b_7^{(1)}, \quad a_1 = b_{14}^{(1)} + b_{15}^{(1)}. \end{aligned}$$

Similar independent determinations of a_2, a_3, and a_4 can be formed. In decoding a_1, we form the following check-sums from the bits of the modified received vector

$\mathbb{r}^{(1)}$ and the preceding eight independent determinations of a_1:

$$\begin{aligned} A_1^{(1)} &= r_0^{(1)} + r_1^{(1)}, \quad A_5^{(1)} = r_8^{(1)} + r_9^{(1)}, \\ A_2^{(1)} &= r_2^{(1)} + r_3^{(1)}, \quad A_6^{(1)} = r_{10}^{(1)} + r_{11}^{(1)}, \\ A_3^{(1)} &= r_4^{(1)} + r_5^{(1)}, \quad A_7^{(1)} = r_{12}^{(1)} + r_{13}^{(1)}, \\ A_4^{(1)} &= r_6^{(1)} + r_7^{(1)}, \quad A_8^{(1)} = r_{14}^{(1)} + r_{15}^{(1)}. \end{aligned}$$

From these check-sums we decode a_1 by using the majority-logic decision rule. Similarly, we can decode the information bits a_2, a_3, and a_4.

After the decoding of a_1, a_2, a_3, and a_4, we remove the effect of a_1, a_2, a_3, and a_4 from $\mathbb{r}^{(1)}$ and form the following modified received vector:

$$\begin{aligned} \mathbb{r}^{(2)} &= (r_0^{(2)}, r_1^{(2)}, \cdots, r_{15}^{(2)}) \\ &= \mathbb{r}^{(1)} - a_4\mathbb{v}_4 - a_3\mathbb{v}_3 - a_2\mathbb{v}_2 - a_1\mathbb{v}_1. \end{aligned}$$

In the absence of errors, $\mathbb{r}^{(2)}$ is the following codeword:

$$a_0\mathbb{v}_0 = (a_0, a_0, \cdots, a_0).$$

This result gives 16 independent determinations of a_0. In decoding a_0, we simply set a_0 to the value taken by the majority of the bits in $\mathbb{r}^{(2)}$. This step completes the entire decoding.

The demonstrated decoding is referred to as *majority-logic decoding*. Because it consists of three steps (or levels) of decoding, it is called three-step majority-logic decoding. It can easily be implemented using majority-logic elements.

If there is only one error in the received vector $\mathbb{r}$, the information bits a_{12}, a_{13}, a_{14}, a_{23}, a_{24}, and a_{34} will be correctly decoded. Then, the modified received vector $\mathbb{r}^{(1)}$ will still contain a single error at the same location. This single error affects only one of the eight check-sums for a_i, with $1 \leq i \leq 4$. The other seven check-sums give the correct value of a_i. Therefore, the information bits a_1, a_2, a_3, and a_4 will be correctly decoded. As a result, the next modified received vector $\mathbb{r}^{(2)}$ will contain only one error (still at the same location), and the information bit a_0 will be correctly decoded.

If there are two transmission errors in $\mathbb{r}$, these two errors may affect two check-sums for information bit a_{ij}. In this case, there is no majority in the four check-sums; two check-sums take 0, and two other check-sums take 1. A choice between these two values as the decoded value of a_{ij} may result in an incorrect decoding of a_{ij}. This incorrect decoding affects the subsequent levels of decoding and results in *error propagation*. Consequently, the decodings of a_1, a_2, a_3, a_4, and a_0 are very likely to be incorrect. Because the code guarantees correcting any single error but not two errors, its minimum distance is at least 3 but less than 5. Since all the codewords have even weight, the minimum distance of the code must be 4.

We have used an example to introduce the concepts of majority-logic decoding and the multiple-step decoding process of the Reed algorithm. Now we are ready to present the general algorithm for decoding RM codes. The major part of the decoding is to form check-sums at each decoding step.

Consider the rth-order RM code, RM(r, m). Let

$$\mathbb{a} = (a_0, a_1, \cdots, a_m, a_{1,2}, \cdots, a_{m-1,m}, \cdots, a_{1,2,\cdots,r}, \cdots, a_{m-r+1,m-r+2,\cdots,m})$$

be the message to be encoded. The corresponding codeword is

$$\begin{aligned}\mathbb{b} &= (b_0, b_1, \cdots, b_{n-1}) \\ &= a_0\mathbb{v}_0 + \sum_{1\le i_1\le m} a_{i_1}\mathbb{v}_{i_1} + \sum_{1\le i_1<i_2\le m} a_{i_1 i_2}\mathbb{v}_{i_1}\mathbb{v}_{i_2} \\ &\quad + \cdots + \sum_{1\le i_1<i_2<\cdots<i_r\le m} a_{i_1 i_2\cdots i_r}\mathbb{v}_{i_1}\mathbb{v}_{i_2}\cdots\mathbb{v}_{i_r}.\end{aligned} \tag{4.7}$$

Let $\mathbb{r} = (r_0, r_1, \cdots, r_{n-1})$ be the received vector. Decoding of RM(r, m) code consists of $r + 1$ steps. At the first step of decoding, the information bits $a_{i_1 i_2\cdots i_r}$ corresponding to the product vectors $\mathbb{v}_{i_1}\mathbb{v}_{i_2}\cdots\mathbb{v}_{i_r}$ of degree r in (4.7) are decoded based on their check-sums formed from the received bits in $\mathbb{r}$. Based on these decoded information bits, the received vector $\mathbb{r} = (r_0, r_1, \cdots, r_{n-1})$ is modified. Let $\mathbb{r}^{(1)} = (r_0^{(1)}, r_1^{(1)}, \cdots, r_{n-1}^{(1)})$ denote the modified received vector. At the second step of decoding, the bits in the modified received vector $\mathbb{r}^{(1)}$ are used to form the check-sums for decoding the information bits $a_{i_1 i_2\cdots i_{r-1}}$ that correspond to the product vectors $\mathbb{v}_{i_1}\mathbb{v}_{i_2}\cdots\mathbb{v}_{i_{r-1}}$ of degree $r-1$ in (4.7). Then, the decoded information bits at the second step of decoding are used to modify $\mathbb{r}^{(1)}$. The modification results in the next modified received vector $\mathbb{r}^{(2)} = (r_0^{(2)}, r_1^{(2)}, \cdots, r_{n-1}^{(2)})$ for the third step of decoding. This step-by-step decoding process continues until the last information bit a_0 that corresponds to the all-one vector $\mathbb{v}_0$ in (4.7) is decoded. This decoding process is called $(r + 1)$-step majority-logic decoding [2, 11].

Now, we need to know how to form check-sums for decoding at each step. For $1 \le i_1 < i_2 < \cdots < i_{r-l} \le m$ with $0 \le l \le r$, we form the following index set:

$$S \triangleq \{c_{i_1-1}2^{i_1-1} + c_{i_2-1}2^{i_2-1} + \cdots + c_{i_{r-l}-1}2^{i_{r-l}-1} : c_{i_j-1} \in \{0, 1\} \text{ for } 1 \le j \le r-l\} \tag{4.8}$$

which is a set of 2^{r-l} nonnegative integers less than 2^m in binary form. The exponent set $\{i_1 - 1, i_2 - 1, \cdots, i_{r-l} - 1\}$ is a subset of $\{0, 1, \cdots, m-1\}$. Let E be the set of integers in $\{0, 1, \cdots, m-1\}$ but not in $\{i_1 - 1, i_2 - 1, \cdots, i_{r-l} - 1\}$; that is,

$$\begin{aligned}E &\triangleq \{0, 1, \cdots, m-1\} \setminus \{i_1 - 1, i_2 - 1, \cdots, i_{r-l} - 1\} \\ &= \{j_1, j_2, \cdots, j_{m-r+l}\},\end{aligned} \tag{4.9}$$

where $0 \le j_1 < j_2 < \cdots < j_{m-r+l} \le m - 1$. We form the following set of integers:

$$S^c \triangleq \{d_{j_1}2^{j_1} + d_{j_2}2^{j_2} + \cdots + d_{j_{m-r+l}}2^{j_{m-r+l}} : d_{j_t} \in \{0, 1\} \text{ for } 1 \le t \le m - r + l\}. \tag{4.10}$$

Note that there are 2^{m-r+l} nonnegative integers in S^c, and

$$S \cap S^c = \{0\}.$$

For $l = r$, $S = \{0\}$, and $S^c = \{0, 1, \cdots, 2^m - 1\}$.

For $0 \le l \le r$, suppose we have just completed the lth step of decoding. The decoded information bits are $a^*_{i_1 i_2 \cdots i_{r-l+1}}$. We form the following modified received vector:

$$\mathbb{r}^{(l)} \triangleq \mathbb{r}^{(l-1)} - \sum_{1 \le i_1 < \cdots < i_{r-l+1} \le m} a^*_{i_1 i_2 \cdots i_{r-l+1}} \mathbb{v}_{i_1} \mathbb{v}_{i_2} \cdots \mathbb{v}_{i_{r-l+1}} \tag{4.11}$$

where $\mathbb{r}^{(l-1)}$ is the modified received vector for the lth step of decoding, and $\mathbb{r}^{(0)} = \mathbb{r}$. For each integer $q \in S^c$, we form the following set of integers (called *indices*):

$$\begin{aligned} B &\triangleq q + S \\ &= \{q + s : s \in S\}. \end{aligned} \tag{4.12}$$

Then, the check-sums for decoding the information bits $a_{i_1 i_2 \cdots i_{r-l}}$ are

$$A^{(l)} = \sum_{t \in B} r_t^{(l)} \tag{4.13}$$

for $q \in S^c$. Because there are 2^{m-r+l} integers q in S^c, we can form 2^{m-r+l} check-sums for decoding each information bit $a_{i_1 i_2 \cdots i_{r-l}}$.

At the first step of decoding, $l = 0$ and 2^{m-r} check-sums can be formed for decoding each information bit $a_{i_1 i_2 \cdots i_r}$. If there are $2^{m-r-1} - 1$ or fewer errors in the received vector $\mathbb{r}$, then more than half (majority) of the check-sums assume the value of $a_{i_1 i_2 \cdots i_r}$ and hence the decoding of $a_{i_1 i_2 \cdots i_r}$ is correct, using the majority-logic decision rule; however, if $\mathbb{r}$ contains 2^{m-r-1} or more errors, there is no guarantee that a majority of the check-sums will assume the value of $a_{i_1 i_2 \cdots i_r}$. In this case, majority-logic decision may result in an incorrect decoding. For example, consider an error pattern with 2^{m-r-1} errors such that each error appears in a different check-sum. In this case, half of the 2^{m-r} check-sums assume the value 0, and the other half assume the value 1. There is no clear majority. A random choice of the two values may result in an incorrect decoding of $a_{i_1 i_2 \cdots i_r}$. Now, consider another error pattern of $2^{m-r-1} + 1$ errors such that each error appears in a different check-sum. In this case, $2^{m-r-1} + 1$ (majority) of the check-sums assume the opposite value of $a_{i_1 i_2 \cdots i_r}$ and majority-logic decision based on the check-sums results in incorrect decoding of $a_{i_1 i_2 \cdots i_r}$. Note that the number of check-sums is doubled at each subsequent decoding step. If there are $2^{m-r-1} - 1$ or fewer errors in the received vector $\mathbb{r}$, then majority-logic decision based on check-sums results in correct decoding at each step. This implies that the minimum distance of the RM(r, m) code is at least $2 \cdot (2^{m-r-1} - 1) + 1 = 2^{m-r} - 1$. Because the codewords in RM(r, m) have even weights, the minimum distance is at least 2^{m-r}; however, each product vector of degree r in the generator matrix $G_{\text{RM}}(r, m)$ has weight 2^{m-r} and is a codeword. Therefore, the minimum distance of the code is exactly 2^{m-r}.

The Reed decoding algorithm for RM codes is simply a multistage decoding algorithm in which the decoded information at each stage of decoding is passed down for the next stage of decoding.

EXAMPLE 4.3

Consider the second-order RM code of length 16 with $m = 4$ given in Example 4.2. Suppose we want to construct the check-sums for the information bit a_{12}. Because

$i_1 = 1$ and $i_2 = 2$, we obtain the following sets:

$$\begin{aligned} S &= \{c_0 + c_1 2^1 : c_0, c_1 \in \{0, 1\}\} \\ &= \{0, 1, 2, 3\}, \\ E &= \{0, 1, 2, 3\} \setminus \{0, 1\} \\ &= \{2, 3\}, \\ S^c &= \{d_2 2^2 + d_3 2^3 : d_2, d_3 \in \{0, 1\}\} \\ &= \{0, 4, 8, 12\}. \end{aligned}$$

Then, the index sets for forming the check-sums for a_{12} are

$$\begin{aligned} B_1 &= 0 + S = \{0, 1, 2, 3\}, \\ B_2 &= 4 + S = \{4, 5, 6, 7\}, \\ B_3 &= 8 + S = \{8, 9, 10, 11\}, \\ B_4 &= 12 + S = \{12, 13, 14, 15\}. \end{aligned}$$

It follows from (4.13) with $l = 0$ that the four check-sums for a_{12} are

$$\begin{aligned} A_1^{(0)} &= r_0 + r_1 + r_2 + r_3, \\ A_2^{(0)} &= r_4 + r_5 + r_6 + r_7, \\ A_3^{(0)} &= r_8 + r_9 + r_{10} + r_{11}, \\ A_4^{(0)} &= r_{12} + r_{13} + r_{14} + r_{15}. \end{aligned}$$

Now, consider the check-sums for a_{13}. Because $i_1 = 1$ and $i_2 = 3$, we obtain the following sets:

$$\begin{aligned} S &= \{c_0 + c_2 2^2 : c_0, c_2 \in \{0, 1\}\} \\ &= \{0, 1, 4, 5\}, \\ E &= \{0, 1, 2, 3\} \setminus \{0, 2\} \\ &= \{1, 3\}, \\ S^c &= \{d_1 2 + d_3 2^3 : d_1, d_3 \in \{0, 1\}\} \\ &= \{0, 2, 8, 10\}. \end{aligned}$$

The index sets for constructing the check-sums for a_{13} are

$$\begin{aligned} B_1 &= 0 + S = \{0, 1, 4, 5\}, \\ B_2 &= 2 + S = \{2, 3, 6, 7\}, \end{aligned}$$

$$B_3 = 8 + S = \{8, 9, 12, 13\},$$
$$B_4 = 10 + S = \{10, 11, 14, 15\}.$$

From these index sets and (4.13), we obtain the following check-sums for a_{13}:

$$A_1^{(0)} = r_0 + r_1 + r_4 + r_5,$$
$$A_2^{(0)} = r_2 + r_3 + r_6 + r_7,$$
$$A_3^{(0)} = r_8 + r_9 + r_{12} + r_{13},$$
$$A_4^{(0)} = r_{10} + r_{11} + r_{14} + r_{15}.$$

Using the same procedure, we can form the check-sums for information bits a_{14}, a_{23}, a_{24}, and a_{34}.

To find the check-sums for a_1, a_2, a_3, and a_4, we first form the modified received vector $\mathbb{r}^{(1)}$ based on (4.11):

$$\mathbb{r}^{(1)} = \mathbb{r} - \sum_{1 \le i_1 < i_2 \le 4} a^*_{i_1 i_2} \mathbb{v}_{i_1} \mathbb{v}_{i_2}.$$

Suppose we want to form the check-sums for a_3. Because $i_1 = 3$, we obtain the following sets:

$$S = \{c_2 2^2 : c_2 \in \{0, 1\}\} = \{0, 4\},$$
$$E = \{0, 1, 2, 3\} \setminus \{2\} = \{0, 1, 3\},$$
$$S^c = \{d_0 + d_1 2 + d_3 2^3 : d_0, d_1, d_3 \in \{0, 1\}\}$$
$$= \{0, 1, 2, 3, 8, 9, 10, 11\}.$$

Then, the index sets for forming the check-sums of a_3 are

$$B_1 = \{0, 4\}, \quad B_5 = \{8, 12\},$$
$$B_2 = \{1, 5\}, \quad B_6 = \{9, 13\},$$
$$B_3 = \{2, 6\}, \quad B_7 = \{10, 14\},$$
$$B_4 = \{3, 7\}, \quad B_8 = \{11, 15\}.$$

It follows from (4.13) with $l = 1$ that we obtain the following eight check-sums:

$$A_1^{(1)} = r_0^{(1)} + r_4^{(1)}, \quad A_5^{(1)} = r_8^{(1)} + r_{12}^{(1)},$$
$$A_2^{(1)} = r_1^{(1)} + r_5^{(1)}, \quad A_6^{(1)} = r_9^{(1)} + r_{13}^{(1)},$$
$$A_3^{(1)} = r_2^{(1)} + r_6^{(1)}, \quad A_7^{(1)} = r_{10}^{(1)} + r_{14}^{(1)},$$
$$A_4^{(1)} = r_3^{(1)} + r_7^{(1)}, \quad A_8^{(1)} = r_{11}^{(1)} + r_{15}^{(1)}.$$

Similarly, we can form the check-sums for a_1, a_2, and a_4.

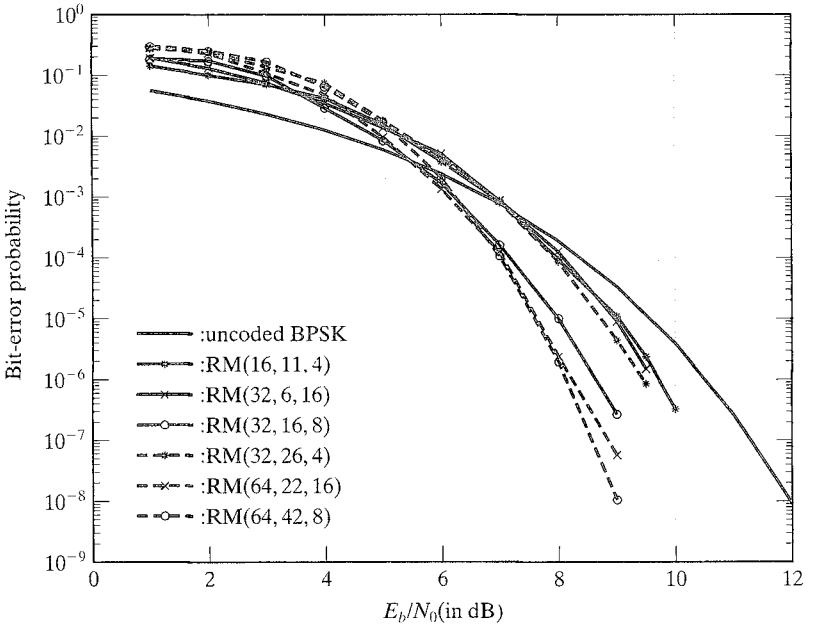

FIGURE 4.2: Bit-error performances of some RM codes with majority-logic decoding.

Error performances of some RM codes of lengths up to 64 using majority-logic decoding are shown in Figure 4.2.

4.4 OTHER CONSTRUCTIONS FOR REED–MULLER CODES

Besides the construction method presented in Section 4.3, there are other methods for constructing RM codes. We present three such methods in this section. These methods reveal more structures of RM code that are useful in constructing trellis diagrams and soft-decision decodings.

Let $A = [a_{ij}]$ be an $m \times m$ matrix and $B = [b_{ij}]$ be an $n \times n$ matrix over $GF(2)$. The *Kronecker product* of A and B, denoted by $A \otimes B$, is the $mn \times mn$ matrix obtained from A by replacing every entry a_{ij} with the matrix $a_{ij}B$. Note that for $a_{ij} = 1$, $a_{ij}B = B$, and for $a_{ij} = 0$, $a_{ij}B$ is an $n \times n$ zero matrix. Let

$$G_{(2,2)} = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \tag{4.14}$$

be a 2×2 matrix over $GF(2)$. The *twofold Kronecker product* of $G_{(2,2)}$ is defined as

$$G_{(2^2,2^2)} \triangleq \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.15}$$

The three-fold Kronecker product of $G_{(2,2)}$ is defined as

$$
\begin{aligned}
G_{(2^3,2^3)} &\triangleq \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix} \otimes \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.
\end{aligned} \tag{4.16}
$$

Similarly, we can define the m-fold Kronecker product of $G_{(2,2)}$. Let $n = 2^m$. We use $G_{(n,n)}$ to denote the m-fold Kronecker product of $G_{(2,2)}$. $G_{(n,n)}$ is a $2^m \times 2^m$ matrix over $GF(2)$. The rows of $G_{(n,n)}$ have weights $2^0, 2^1, 2^2, \ldots, 2^m$, and the number of rows with weight 2^{m-l} is $\binom{m}{l}$ for $0 \leq l \leq m$.

The generator matrix $G_{\mathrm{RM}}(r, m)$ of the rth-order RM code $\mathrm{RM}(r, m)$ of length $n = 2^m$ consists of those rows of $G_{(n,n)}$ with weights equal to or greater than 2^{m-r}. These rows are the same vectors given by (4.5), except they are a different permutation.

EXAMPLE 4.4

Let $m = 4$. The fourfold Kronecker product of $G_{(2,2)}$ is

$$
G_{(2^4,2^4)} = \begin{bmatrix}
1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{bmatrix}.
$$

The generator matrix $G_{\mathrm{RM}}(2, 4)$ of the second-order RM code, RM(2, 4), of length 16 consists of the rows in $G_{(16,16)}$ with weights 2^2, 2^3, and 2^4. Thus, we obtain

$$G_{\mathrm{RM}}(2,4) = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix},$$

which is exactly the same matrix given in Example 4.2, except for the ordering of the rows.

Let $\mathbb{u} = (u_0, u_1, \ldots, u_{n-1})$ and $\mathbb{v} = (v_0, v_1, \ldots, v_{n-1})$ be two n-tuples over $GF(2)$. From $\mathbb{u}$ and $\mathbb{v}$ we form the following $2n$-tuple:

$$|\mathbb{u}|\mathbb{u} + \mathbb{v}| \triangleq (u_0, u_1, \ldots, u_{n-1}, u_0 + v_0, u_1 + v_1, \ldots, u_{n-1} + v_{n-1}). \tag{4.17}$$

For $i = 1, 2$, let C_i be a binary (n, k_i) linear code with generator matrix G_i and minimum distance d_i, respectively. Assume that $d_2 > d_1$. We form the following linear code of length $2n$:

$$\begin{aligned} C &= |C_1|C_1 + C_2| \\ &= \{|\mathbb{u}|\mathbb{u} + \mathbb{v}| : \mathbb{u} \in C_1 \text{ and } \mathbb{v} \in C_2\}. \end{aligned} \tag{4.18}$$

C is a binary $(2n, k_1 + k_2)$ linear code with generator matrix

$$G = \begin{bmatrix} G_1 & G_1 \\ \mathbb{0} & G_2, \end{bmatrix} \tag{4.19}$$

where $\mathbb{0}$ is a $k_2 \times n$ zero matrix. The minimum distance $d_{\min}(C)$ of C is

$$d_{\min}(C) = \min\{2d_1, d_2\}. \tag{4.20}$$

To prove this, let $\mathbb{x} = |\mathbb{u}|\mathbb{u} + \mathbb{v}|$ and $\mathbb{y} = |\mathbb{u}'|\mathbb{u}' + \mathbb{v}'|$ be two distinct codewords in C. The Hamming distance between $\mathbb{x}$ and $\mathbb{y}$ can be expressed in terms of Hamming weights as follows:

$$d(\mathbb{x}, \mathbb{y}) = w(\mathbb{u} + \mathbb{u}') + w(\mathbb{u} + \mathbb{u}' + \mathbb{v} + \mathbb{v}'), \tag{4.21}$$

where $w(\mathbb{z})$ denotes the Hamming weight of $\mathbb{z}$. There are two cases to be considered, $\mathbb{v} = \mathbb{v}'$ and $\mathbb{v} \neq \mathbb{v}'$. If $\mathbb{v} = \mathbb{v}'$, since $\mathbb{x} \neq \mathbb{y}$, we must have $\mathbb{u} \neq \mathbb{u}'$. In this case,

$$d(\mathbb{x}, \mathbb{y}) = w(\mathbb{u} + \mathbb{u}') + w(\mathbb{u} + \mathbb{u}'). \tag{4.22}$$

Because $\mathbb{u} + \mathbb{u}'$ is a nonzero codeword in C_1, $w(\mathbb{u} + \mathbb{u}') \geq d_1$. Then, it follows from (4.22) that

$$d(\mathbb{x}, \mathbb{y}) \geq 2d_1. \tag{4.23}$$

If $\mathbb{v} \neq \mathbb{v}'$, we have

$$\begin{aligned} d(\mathbb{x}, \mathbb{y}) &\geq w(\mathbb{u} + \mathbb{u}') + w(\mathbb{v} + \mathbb{v}') - w(\mathbb{u} + \mathbb{u}') \\ &= w(\mathbb{v} + \mathbb{v}'). \end{aligned} \tag{4.24}$$

Since $\mathbb{v} + \mathbb{v}'$ is a nonzero codeword in C_2, $w(\mathbb{v} + \mathbb{v}') \geq d_2$. From (4.24) we have

$$d(\mathbb{x}, \mathbb{y}) \geq d_2. \tag{4.25}$$

Inequalities (4.23) and (4.25) imply that

$$d(\mathbb{x}, \mathbb{y}) \geq \min\{2d_1, d_2\}. \tag{4.26}$$

Because $\mathbb{x}$ and $\mathbb{y}$ are two arbitrary different codewords in C, the minimum distance $d_{min}(C)$ must be lower bounded as follows:

$$d_{min}(C) \geq \min\{2d_1, d_2\}. \tag{4.27}$$

Let $\mathbb{u}_0$ and $\mathbb{v}_0$ be two minimum-weight codewords in C_1 and C_2, respectively. Then, $w(\mathbb{u}_0) = d_1$ and $w(\mathbb{v}_0) = d_2$. The vector $|\mathbb{u}_0|\mathbb{u}_0|$ is a codeword in C with weight $w(|\mathbb{u}_0|\mathbb{u}_0|) = 2d_1$. The vector $|\mathbb{0}|\mathbb{v}_0|$ is also a codeword in C with weight $w(|\mathbb{0}|\mathbb{v}_0|) = d_2$. From (4.27) we see that $d_{min}(C)$ must be either $2d_1$ or d_2. Therefore, we conclude that

$$d_{min}(C) = \min\{2d_1, d_2\}. \tag{4.28}$$

The preceding construction of a code from two component codes is called the $|\mathbb{u}|\mathbb{u} + \mathbb{v}|$-*construction* [12, 13] which is a powerful technique for constructing powerful long codes from short codes.

EXAMPLE 4.5

Let C_1 be the binary (8, 4) linear code of minimum distance 4 generated by

$$G_1 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \end{bmatrix}.$$

Let C_2 be the (8, 1) repetition code of minimum distance 8 generated by

$$G_2 = \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

Using $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$-construction, we obtain a $(16, 5)$ binary linear code of minimum distance 8 with the following generator matrix,

$$G = \begin{bmatrix} G_1 & G_1 \\ 0 & G_2 \end{bmatrix}$$

$$= \begin{bmatrix} 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix}.$$

RM codes of length 2^m can be constructed from RM codes of length 2^{m-1} using the $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$-construction [12]. For $m \geq 2$, the rth-order RM code in $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$-construction is given as follows:

$$\begin{aligned} \mathrm{RM}(r, m) = \{|\mathbf{u}|\mathbf{u}+\mathbf{v}| : \mathbf{u} &\in \mathrm{RM}(r, m-1) \text{ and} \\ \mathbf{v} &\in \mathrm{RM}(r-1, m-1)\} \end{aligned} \tag{4.29}$$

with generator matrix

$$G_{\mathrm{RM}}(r, m) = \begin{bmatrix} G_{\mathrm{RM}}(r, m-1) & G_{\mathrm{RM}}(r, m-1) \\ 0 & G_{\mathrm{RM}}(r-1, m-1) \end{bmatrix}. \tag{4.30}$$

The matrix of (4.30) shows that a RM code can be constructed recursively from short RM codes by a sequence of $|\mathbf{u}|\mathbf{u}+\mathbf{v}|$-constructions. For example, the rth-order RM code $\mathrm{RM}(r, m)$ of length 2^m can be constructed from RM codes $\mathrm{RM}(r, m-2)$, $\mathrm{RM}(r-1, m-2)$, and $\mathrm{RM}(r-2, m-2)$ of length 2^{m-2}. The generator matrix in terms of component codes is given as follows:

$$G = \begin{bmatrix} G_{\mathrm{RM}}(r, m-2) & G_{\mathrm{RM}}(r, m-2) & G_{\mathrm{RM}}(r, m-2) & G_{\mathrm{RM}}(r, m-2) \\ 0 & G_{\mathrm{RM}}(r-1, m-2) & 0 & G_{\mathrm{RM}}(r-1, m-2) \\ 0 & 0 & G_{\mathrm{RM}}(r-1, m-2) & G_{\mathrm{RM}}(r-1, m-2) \\ 0 & 0 & 0 & G_{\mathrm{RM}}(r-2, m-2) \end{bmatrix}. \tag{4.31}$$

The recursive structure of RM codes is very useful in analyzing and constructing their trellises [15, 16]. This structure also allows us to devise multistage soft-decision decoding schemes for RM codes that achieve good error performance with reduced decoding complexity. This topic will also be discussed in a later chapter.

Consider a Boolean function $f(X_1, X_2, \ldots, X_m)$ of m variables, $X_1, X_2, \ldots, X_m$, that take values of 0 or 1 [12, 14]. For each combination of values of $X_1, X_2, \ldots,$ and X_m, the function f takes a truth value of either 0 or 1. For the 2^m combinations of values of $X_1, X_2, \ldots, X_m$, the truth values of f form a 2^m-tuple over $GF(2)$.

For a nonnegative integer l less than 2^m, let $(b_{l1}, b_{l2}, \ldots, b_{lm})$ be the standard binary representation of l, such that $l = b_{l1} + b_{l2}2 + b_{l3}2^2 + \cdots + b_{lm}2^{m-1}$. For a

given Boolean function $f(X_1, X_2, \ldots, X_m)$, we form the following 2^m-tuple (truth vector):

$$\mathbb{v}(f) = (v_0, v_1, \ldots, v_l, \ldots, v_{2^m-1}) \tag{4.32}$$

where

$$v_l \triangleq f(b_{l1}, b_{l2}, \ldots, b_{lm}) \tag{4.33}$$

and $(b_{l1}, b_{l2}, \ldots, b_{lm})$ is the standard binary representation of the index integer l. We say that the Boolean function $f(X_1, X_2, \ldots, X_m)$ represents the vector $\mathbb{v}$. We use the notation $\mathbb{v}(f)$ for the vector represented by $f(X_1, X_2, \ldots, X_m)$. For $1 \leq i \leq m$, consider the Boolean function

$$f(X_1, X_2, \ldots, X_m) = X_i. \tag{4.34}$$

It is easy to see that this Boolean function represents the vector $\mathbb{v}_i$ defined by (4.4). For $1 \leq i, j \leq m$, the function

$$f(X_1, X_2, \ldots, X_m) = X_i X_j \tag{4.35}$$

represents the logic product of $\mathbb{v}_i$ and $\mathbb{v}_j$, represented by $g(X_1, X_2, \ldots, X_m) = X_i$ and $h(X_1, X_2, \ldots, X_m) = X_j$, respectively. For $1 \leq i_1 < i_2 < \cdots < i_r \leq m$, the Boolean function

$$f(X_1, X_2, \ldots, X_m) = X_{i_1} X_{i_2} \cdots X_{i_r} \tag{4.36}$$

represents the logic product of $\mathbb{v}_{i_1}, \mathbb{v}_{i_2}, \ldots,$ and $\mathbb{v}_{i_r}$. Therefore, the generator vectors of the rth-order RM code of length $n = 2^m$ (the rows in $G_{\mathrm{RM}}(r, m)$) are represented by the Boolean functions in the following set:

$$\begin{aligned} B(r, m) = \{&1, X_1, X_2, \ldots, X_m, X_1X_2, X_1X_3, \ldots, X_{m-1}X_m, \\ &\ldots \text{ up to all products of } r \text{ variables}\}. \end{aligned} \tag{4.37}$$

Let $P(r, m)$ denote the set of all Boolean functions (or polynomials) of degree r or less with m variables. Then, $\mathrm{RM}(r, m)$ is given by the following set of vectors [12]:

$$\mathrm{RM}(r, m) = \{\mathbb{v}(f) : f \in P(r, m)\}. \tag{4.38}$$

The Boolean representation is very useful in studying the weight distribution of RM codes [18, 20].

4.5 THE SQUARING CONSTRUCTION OF CODES

Consider a binary (n, k) linear code C with generator matrix G. For $0 \leq k_1 \leq k$, let C_1 be an (n, k_1) linear subcode of C that is spanned by k_1 rows of G. Partition C into 2^{k-k_1} cosets of C_1. This partition of C with respect to C_1 is denoted by C/C_1. As shown in Section 3.5, each coset of C_1 is of the following form:

$$\mathbb{v}_l \oplus C_1 = \{\mathbb{v}_l + \mathbb{u} : \mathbb{u} \in C_1\} \tag{4.39}$$

with $1 \leq l \leq 2^{k-k_1}$, where for $\mathbb{v}_l \neq \mathbb{0}$, $\mathbb{v}_l$ is in C but not in C_1, and for $\mathbb{v}_l = \mathbb{0}$, the coset $\mathbb{0} \oplus C_1$ is just the subcode C_1 itself. The codeword $\mathbb{v}_l$ is called the *leader* (or *representative*) of the coset $\mathbb{v}_l \oplus C_1$. We also showed in Section 3.5 that any codeword

in a coset can be used as the coset representative without changing the composition of the coset. The all-zero codeword $\mathbb{0}$ is always used as the representative for C_1. The set of representatives for the cosets in the partition C/C_1 is denoted by $[C/C_1]$, which is called the *coset representative space* for the partition C/C_1. Because all the cosets in C/C_1 are disjoint, C_1 and $[C/C_1]$ have only the all-zero codeword $\mathbb{0}$ in common; that is, $C_1 \cap [C/C_1] = \{\mathbb{0}\}$. Then, we can express C as the sum of the coset representatives in $[C/C_1]$ and the codewords in C_1 as follows:

$$C = [C/C_1] \oplus C_1 \triangleq \{\mathbb{v} + \mathbb{u} : \mathbb{v} \in [C/C_1], \mathbb{u} \in C_1\}. \tag{4.40}$$

The preceding sum is called the *direct-sum* of $[C/C_1]$ and C_1.

Let G_1 be the subset of k_1 rows of the generator matrix G that generates the linear subcode C_1. Then, the 2^{k-k_1} codewords generated by the $k - k_1$ rows in the set $G \setminus G_1$ can be used as the representatives for the cosets in C/C_1. These 2^{k-k_1} codewords form an $(n, k - k_1)$ linear subcode of C.

Let C_2 be an (n, k_2) linear subcode of C_1 with $0 \leq k_2 \leq k_1$. We can further partition each coset $\mathbb{v}_l \oplus C_1$ in the partition C/C_1 based on C_2 into $2^{k_1-k_2}$ cosets of C_2; each coset consists of the following codewords in C:

$$\mathbb{v}_l \oplus (\mathbb{w}_q \oplus C_2) \triangleq \{\mathbb{v}_l + \mathbb{w}_q + \mathbb{u} : \mathbb{u} \in C_2\} \tag{4.41}$$

with $1 \leq l \leq 2^{k-k_1}$ and $1 \leq q \leq 2^{k_1-k_2}$, where for $\mathbb{w}_q \neq \mathbb{0}$, $\mathbb{w}_q$ is a codeword in C_1 but not in C_2. We denote this partition $C/C_1/C_2$. This partition consists of 2^{k-k_2} cosets of C_2. Now, we can express C as the following direct-sum:

$$C = [C/C_1] \oplus [C_1/C_2] \oplus C_2. \tag{4.42}$$

Let $C_1, C_2, \ldots, C_m$ be a sequence of linear subcodes of C with dimensions $k_1, k_2, \ldots, k_m$, respectively, such that

$$C \supseteq C_1 \supseteq C_2 \supseteq \cdots \supseteq C_m \tag{4.43}$$

and

$$k \geq k_1 \geq k_2 \geq \cdots \geq k_m \geq 0. \tag{4.44}$$

Then, we can form a chain of partitions,

$$C/C_1, C/C_1/C_2, \ldots, C/C_1/C_2/\cdots/C_m, \tag{4.45}$$

and can express C as the following direct-sum:

$$C = [C/C_1] \oplus [C_1/C_2] \oplus \cdots \oplus [C_{m-1}/C_m] \oplus C_m. \tag{4.46}$$

We now present another method for constructing long codes from a sequence of subcodes of a given short code. This method is known as the *squaring construction* [15].

Let C_0 be a binary (n, k_0) linear block code with minimum Hamming distance d_0. Let $C_1, C_2, \ldots, C_m$ be a sequence of subcodes of C_0 such that

$$C_0 \supset C_1 \supset C_2 \supset \cdots \supset C_m. \tag{4.47}$$

For $0 \leq i \leq m$, let G_i, k_i, and d_i be the generator matrix, the dimension, and the minimum distance of the subcode C_i, respectively. We form a chain of partitions as follows:

$$C_0/C_1, C_0/C_1/C_2, \ldots, C_0/C_1/\cdots/C_m. \tag{4.48}$$

For $0 \leq i < m$, let $G_{i/i+1}$ denote the generator matrix for the coset representative space $[C_i/C_{i+1}]$. The rank of $G_{i/i+1}$ is

$$\text{Rank}(G_{i/i+1}) = \text{Rank}(G_i) - \text{Rank}(G_{i+1}). \tag{4.49}$$

Without loss of generality, we assume that $G_0 \supset G_1 \supset G_2 \supset \cdots \supset G_m$. Then, for $0 \leq i < m$,

$$G_{i/i+1} = G_i \backslash G_{i+1}. \tag{4.50}$$

One-level squaring construction is based on C_1 and the partition C_0/C_1. Let $\mathbb{a} = (a_0, a_1, \ldots, a_{n-1})$ and $\mathbb{b} = (b_0, b_1, \ldots, b_{n-1})$ be two binary n-tuples, and let $(\mathbb{a}, \mathbb{b})$ denote the $2n$-tuple $(a_0, a_1, \ldots, a_{n-1}, b_0, b_1, \ldots, b_{n-1})$. We form the following set of $2n$-tuples:

$$|C_0/C_1|^2 \triangleq \{(\mathbb{a} + \mathbb{x}, \mathbb{b} + \mathbb{x}) : \mathbb{a}, \mathbb{b} \in C_1 \text{ and } \mathbb{x} \in [C_0/C_1]\}. \tag{4.51}$$

Then, $|C_0/C_1|^2$ is a $(2n, k_0 + k_1)$ linear block code with minimum Hamming distance

$$D_1 \triangleq \min\{2d_0, d_1\}. \tag{4.52}$$

The generator matrix for $|C_0/C_1|^2$ is given by

$$G = \begin{bmatrix} G_1 & 0 \\ 0 & G_1 \\ G_{0/1} & G_{0/1} \end{bmatrix}. \tag{4.53}$$

Let M_1 and M_2 be two matrices with the same number of columns. The matrix

$$M = \begin{bmatrix} M_1 \\ M_2 \end{bmatrix}$$

is called the direct-sum of M_1 and M_2, denoted by $M = M_1 \oplus M_2$. Then, we can express the generator matrix for the one-level squaring construction code $|C_0/C_1|^2$ in the following form:

$$G = I_2 \otimes G_1 \oplus (1, 1) \otimes G_{0/1}, \tag{4.54}$$

where $\otimes$ denotes the Kronecker product, $\oplus$ the direct-sum, I_2 the identity matrix of dimension 2,

$$I_2 \otimes G_1 = \begin{bmatrix} G_1 & 0 \\ 0 & G_1 \end{bmatrix}, \tag{4.55}$$

and

$$(1, 1) \otimes G_{0/1} = [G_{0/1} \;\; G_{0/1}]. \tag{4.56}$$

Now, we extend the one-level squaring construction to a two-level squaring construction. First, we form two codes, $U \triangleq |C_0/C_1|^2$ and $V \triangleq |C_1/C_2|^2$, using one-level squaring construction. It is easy to see that V is a subcode of U. The two-level squaring construction based on the partitions C_0/C_1, C_1/C_2, and $C_0/C_1/C_2$ gives the following code:

$$\begin{aligned} |C_0/C_1/C_2|^4 &\triangleq \{(\mathbb{a}+\mathbb{x}, \mathbb{b}+\mathbb{x}) : \mathbb{a}, \mathbb{b} \in V \text{ and } \mathbb{x} \in [\mathrm{U/V}]\} \\ &= \{(\mathbb{a}+\mathbb{x}, \mathbb{b}+\mathbb{x}) : \mathbb{a}, \mathbb{b} \in |C_1/C_2|^2 \\ &\qquad \text{and } \mathbb{x} \in [|C_0/C_1|^2 / |C_1/C_2|^2]\}, \end{aligned} \tag{4.57}$$

which is simply the code obtained by one-level squaring construction based on V and U/V. This code is a $(4n, k_0 + 2k_1 + k_2)$ linear block code with minimum Hamming distance

$$D_2 \triangleq \min\{4d_0, 2d_1, d_2\}. \tag{4.58}$$

Let G_U, G_V, and $G_{U/V}$ denote the generator matrices for U, V, and $[U/V]$, respectively. Then, the generator matrix for $|U/V|^2$ is

$$G = \begin{bmatrix} G_V & 0 \\ 0 & G_V \\ G_{U/V} & G_{U/V} \end{bmatrix}. \tag{4.59}$$

We put G_V and G_U in the form of (4.53) and note that $G_{U/V} = G_U \backslash G_V$. Then, we can put the generator matrix G of $|C_0/C_1/C_2|^4 = |U/V|^2$ given by (4.59) in the following form:

$$G = \begin{bmatrix} G_2 & 0 & 0 & 0 \\ 0 & G_2 & 0 & 0 \\ 0 & 0 & G_2 & 0 \\ 0 & 0 & 0 & G_2 \\ G_{0/1} & G_{0/1} & G_{0/1} & G_{0/1} \\ G_{1/2} & G_{1/2} & G_{1/2} & G_{1/2} \\ 0 & 0 & G_{1/2} & G_{1/2} \\ 0 & G_{1/2} & 0 & G_{1/2} \end{bmatrix}. \tag{4.60}$$

We can express this matrix in the following compact form:

$$G = I_4 \otimes G_2 \oplus (1111) \otimes G_{0/1} \oplus \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix} \otimes G_{1/2}. \tag{4.61}$$

Note that

$$(1111) \text{ and } \begin{bmatrix} 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{bmatrix}$$

are the generator matrices of the zeroth- and first-order RM codes of length 4.

Higher-level squaring construction can be carried out recursively in a similar manner. For $m \geq 2$, let

$$U_m \triangleq |C_0/C_1/\cdots/C_{m-1}|^{2^{m-1}}$$

and

$$V_m \triangleq |C_1/C_2/\cdots/C_m|^{2^{m-1}}$$

denote the two codes obtained by $(m-1)$-level squaring construction. The code obtained by m-level squaring construction is given by [15]:

$$|C_0/C_1/\cdots/C_m|^{2^m} \triangleq \{(\mathbb{a}+\mathbb{x}, \mathbb{b}+\mathbb{x}) : \mathbb{a}, \mathbb{b} \in V_m \text{ and } \mathbb{x} \in [U_m/V_m]\}. \tag{4.62}$$

The generator matrix is given by [15, 16]:

$$G = I_{2^m} \otimes G_m \oplus \sum_{0 \le r < m} G_{\mathrm{RM}}(r, m) \otimes G_{r/r+1}, \tag{4.63}$$

where I_{2^m} denotes the identity matrix of dimension 2^m, and $G_{\mathrm{RM}}(r, m)$ is the generator matrix of the rth-order RM code, $\mathrm{RM}(r, m)$, of length 2^m.

RM codes are good examples of the squaring construction. Long RM codes can be constructed from short RM codes iteratively using the squaring construction [15, 16]. From the construction of RM codes given in Section 4.2, we find that for $0 < i \le r$,

$$\mathrm{RM}(r, m) \supset \mathrm{RM}(r-1, m) \supset \cdots \supset \mathrm{RM}(r-i, m). \tag{4.64}$$

Consider the RM code $\mathrm{RM}(r, m)$. As shown in Section 4.4, this code can be obtained from $\mathrm{RM}(r, m-1)$ and $\mathrm{RM}(r-1, m-1)$ codes using the $|\mathbb{u}|\mathbb{u}+\mathbb{v}|$-construction. It follows from (4.30) that the generator matrix for the $\mathrm{RM}(r, m)$ code can be expressed as follows:

$$G_{\mathrm{RM}}(r, m) = \begin{bmatrix} G_{\mathrm{RM}}(r, m-1) & G_{\mathrm{RM}}(r, m-1) \\ 0 & G_{\mathrm{RM}}(r-1, m-1) \end{bmatrix}. \tag{4.65}$$

We define

$$\Delta_{\mathrm{RM}}(r/r-1, m-1) \triangleq G_{\mathrm{RM}}(r, m-1) \backslash G_{\mathrm{RM}}(r-1, m-1). \tag{4.66}$$

Note that $\Delta_{\mathrm{RM}}(r/r-1, m-1)$ consists of those rows in $G_{\mathrm{RM}}(r, m-1)$ but not in $G_{\mathrm{RM}}(r-1, m-1)$ and it spans the coset representative space $[\mathrm{RM}(r, m-1)/\mathrm{RM}(r-1, m-1)]$. Now, we can put $G_{\mathrm{RM}}(r, m-1)$ in the following form:

$$G_{\mathrm{RM}}(r, m-1) = \begin{bmatrix} G_{\mathrm{RM}}(r-1, m-1) \\ \Delta_{\mathrm{RM}}(r/r-1, m-1) \end{bmatrix}. \tag{4.67}$$

Replacing $G_{\mathrm{RM}}(r, m-1)$ in (4.65) with the expression of (4.67) and performing row operations, we put $G_{\mathrm{RM}}(r, m)$ in the following form:

$$G_{\mathrm{RM}}(r, m) = \begin{bmatrix} G_{\mathrm{RM}}(r-1, m-1) & 0 \\ 0 & G_{\mathrm{RM}}(r-1, m-1) \\ \Delta_{\mathrm{RM}}(r/r-1, m-1) & \Delta_{\mathrm{RM}}(r/r-1, m-1) \end{bmatrix}. \tag{4.68}$$

This is exactly the generator matrix form of one-level squaring construction. Therefore, the rth-order RM code of length 2^m can be constructed from the rth-order and $(r-1)$th order RM codes of length 2^{m-1}; that is,

$$\mathrm{RM}(r, m) = |\mathrm{RM}(r, m-1)/\mathrm{RM}(r-1, m-1)|^2. \tag{4.69}$$

Because

$$\mathrm{RM}(r, m-1) = |\mathrm{RM}(r, m-2)/\mathrm{RM}(r-1, m-2)|^2$$

and

$$\mathrm{RM}(r-1, m-1) = |\mathrm{RM}(r-1, m-2)/\mathrm{RM}(r-2, m-2)|^2,$$

then we can construct $\mathrm{RM}(r, m)$ from $\mathrm{RM}(r, m-2)$, $\mathrm{RM}(r-1, m-2)$, and $\mathrm{RM}(r-2, m-2)$ using two-level squaring construction; that is,

$$\mathrm{RM}(r, m) = |\mathrm{RM}(r, m-2)/\mathrm{RM}(r-1, m-2)/\mathrm{RM}(r-2, m-2)|^{2^2}. \tag{4.70}$$

Repeating the preceding process, we find that for $1 \leq \mu \leq r$, we can express the $\mathrm{RM}(r, m)$ code as a μ-level squaring construction code as follows:

$$\mathrm{RM}(r, m) = |\mathrm{RM}(r, m-\mu)/\mathrm{RM}(r-1, m-\mu)/\cdots/\mathrm{RM}(r-\mu, m-\mu)|^{2^\mu}. \tag{4.71}$$

A problem related to the construction of codes from component codes is code *decomposition*. A code is said to be *decomposable* if it can be expressed in terms of component codes. A code is said to be μ-level decomposable if it can be expressed as a μ-level squaring construction code from a sequence of subcodes of a given code, as shown in (4.62). From (4.71) we see that a RM code is μ-level decomposable.

A μ-level decomposable code can be decoded in multiple stages: component codes are decoded sequentially one at a time, and decoded information is passed from one stage to the next stage. This multistage decoding provides a good trade-off between error performance and decoding complexity, especially for long codes.

RM codes also can be constructed from Euclidean geometry, which is discussed in Chapter 8. This construction reveals more algebraic and geometric structures of these codes, especially the structures and the number of minimum-weight codewords. There is a one-to-one correspondence between a minimum-weight codeword of the rth-order RM code, $\mathrm{RM}(r, m)$, and an $(m-r)$-dimensional flat in m-dimensional Euclidean geometry, $\mathrm{EG}(m, 2)$, over $GF(2)$. This correspondence gives the number of minimum-weight codewords of the $\mathrm{RM}(r, m)$ code [12, 17]

$$A_{2^{m-r}} = 2^r \prod_{i=0}^{m-r-1} \left(\frac{2^{m-i} - 1}{2^{m-r-i} - 1} \right). \tag{4.72}$$

In fact, these minimum-weight codewords span (or generate) the code; that is the linear combinations of these minimum-weight codewords produce all the codewords of the $\mathrm{RM}(r, m)$ code.

The weight distribution of several subclasses of RM codes and all RM codes of lengths up to 512 have been enumerated [12, 18–21]. The first-order RM code, $\mathrm{RM}(1, m)$, has only three weights, 1, 2^{m-1}, and 2^m. The number of codewords of these weights are

$$\begin{aligned} A_0 &= A_{2^m} = 1, \\ A_{2^{m-1}} &= 2^{m+1} - 2. \end{aligned} \tag{4.73}$$

The second-order RM code, RM(2, m) has the following weight distribution:

$$A_0 = A_{2^m} = 1,$$

$$A_{2^{m-1}\pm 2^{m-1-l}} = 2^{l(l+1)} \frac{\prod_{i=m-2l+1}^{m}(2^i - 1)}{\prod_{i=1}^{l}(2^{2i} - 1)}, \quad \text{for } 1 \le l \le \lfloor \frac{m}{2} \rfloor \tag{4.74}$$

$$A_{2^{m-l}} = 2^{(m^2+m+2)/2} - 2 - 2\sum_{l=1}^{\lfloor \frac{m}{2} \rfloor} 2^{l(l+1)} \frac{\prod_{i=m-2l+1}^{m}(2^i - 1)}{\prod_{i=1}^{l}(2^{2i} - 1)}.$$

Because the $(m - r - 1)$th-order RM code, RM$(m - r - 1, m)$, is the dual code of the rth-order RM code, RM(r, m), the weight distributions of the RM$(m - 2, m)$ and RM$(m - 3, m)$ codes can be derived from (4.73), (4.74), and the MacWilliams identity of (3.32).

EXAMPLE 4.6

The weight distribution of the (32, 16) RM code, RM(2, 5), is

i	0	8	12	16	20	24	32
A_i	1	620	13888	36518	13888	620	1

This code is a self-dual code.

RM codes form a remarkable class of linear block codes. Their rich structural properties make them very easy to decode by either hard- or soft-decision decoding. Various soft-decision decoding algorithms for these codes have been devised, and some will be discussed in later chapters. Other classes of codes are more powerful than RM codes—for the same minimum distance, these codes have higher rates; however, the low decoding complexity of RM codes makes them very attractive in practical applications. In fact, in terms of both error performance and decoding complexity, RM codes often outperform their corresponding more powerful codes.

The $(m - 2)$th-order RM code of length 2^m is actually the distance-4 extended Hamming code obtained by adding an overall parity bit to the Hamming code of length $2^m - 1$.

4.6 THE (24, 12) GOLAY CODE

Besides the Hamming codes, the only other nontrivial binary perfect code is the (23, 12) Golay code constructed by M. J. E. Golay in 1949 [3]. This code has a minimum distance of 7 and is capable of correcting any combination of three or fewer random errors in a block of 23 digits. The code has abundant and beautiful algebraic structure, and it has become a subject of study by many coding theorists and mathematicians; many research papers have been published on its structure and decoding. The Golay code is the most extensively studied single code. In addition to having beautiful structure, this code has been used in many real communication systems for error control. This code in its cyclic form will be studied in Chapter 5.

The (23, 12) Golay code can be extended by adding an overall parity-check bit to each codeword. This extension results in a (24, 12) code with a minimum distance

of 8. This code is capable of correcting all error patterns of three or fewer errors and detecting all error patterns of four errors. It is not a perfect code anymore; however, it has many interesting structural properties and has been widely used for error control in many communication systems, especially in the U.S. space program. It served as the primary *Voyager* error-control system, providing clear color pictures of Jupiter and Saturn between 1979 and 1981.

In this section we study the (24, 12) Golay code and its decoding. A generator matrix in systematic form for this code is as follows [12, 23, 24]:

$$\mathbb{G} = [\ \mathbb{P} \quad \mathbb{I}_{12}\ ],$$

where $\mathbb{I}_{12}$ is the identity matrix of dimension 12 and

$$\mathbb{P} = \begin{bmatrix} 1\,0\,0\,0\,1\,1\,1\,0\,1\,1\,0\,1 \\ 0\,0\,0\,1\,1\,1\,0\,1\,1\,0\,1\,1 \\ 0\,0\,1\,1\,1\,0\,1\,1\,0\,1\,0\,1 \\ 0\,1\,1\,1\,0\,1\,1\,0\,1\,0\,0\,1 \\ 1\,1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,1 \\ 1\,1\,0\,1\,1\,0\,1\,0\,0\,0\,1\,1 \\ 1\,0\,1\,1\,0\,1\,0\,0\,0\,1\,1\,1 \\ 0\,1\,1\,0\,1\,0\,0\,0\,1\,1\,1\,1 \\ 1\,1\,0\,1\,0\,0\,0\,1\,1\,1\,0\,1 \\ 1\,0\,1\,0\,0\,0\,1\,1\,1\,0\,1\,1 \\ 0\,1\,0\,0\,0\,1\,1\,1\,0\,1\,1\,1 \\ 1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,0 \end{bmatrix}. \tag{4.75}$$

The $\mathbb{P}$ matrix has the following properties: (1) it is symmetrical with respect to its diagonal; (2) the ith column is the transpose of the ith row; (3) $\mathbb{P} \cdot \mathbb{P}^T = \mathbb{I}_{12}$, where $\mathbb{P}^T$ is the transpose of $\mathbb{P}$; and (4) the submatrix obtained by deleting the last row and last column is formed by cyclically shifting the first row to the left 11 times (or cyclically shifting the first column upward 11 times). It follows from the second property that

$$\mathbb{P}^T = \mathbb{P}.$$

Consequently, the parity-check matrix in systematic form for the (24, 12) extended Golay code is given by

$$\begin{aligned} \mathbb{H} &= [\ \mathbb{I}_{12} \quad \mathbb{P}^T\ ] \\ &= [\ \mathbb{I}_{12} \quad \mathbb{P}\ ]. \end{aligned} \tag{4.76}$$

It can be proved that the code is self-dual [see Problem 4.18].

A simple decoding algorithm for the (24, 12) Golay code can be devised using the properties of the $\mathbb{P}$ matrix [23]. For $0 \le i \le 11$, let $\mathbb{p}_i$ denote the ith row of $\mathbb{P}$ and $\mathbb{u}^{(i)}$ the 12-tuple in which only the ith component is nonzero. For example, $\mathbb{u}^{(5)} = (0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0)$. We readily see that

$$\mathbb{p}_i = \mathbb{u}^{(i)} \cdot \mathbb{P}. \tag{4.77}$$

Let $\mathbb{e} = (\mathbb{x}, \mathbb{y})$ be an error vector, where $\mathbb{x}$ and $\mathbb{y}$ are binary 12-tuples. Suppose a codeword $\mathbb{v}$ is transmitted, and a correctable error pattern $\mathbb{e} = (\mathbb{x}, \mathbb{y})$ occurs. Then,

the received vector is $\mathbb{r} = \mathbb{v} + \mathbb{e}$. The syndrome of $\mathbb{r}$ is

$$\mathbb{s} = \mathbb{r} \cdot \mathbb{H}^T = (\mathbb{v} + \mathbb{e}) \cdot \mathbb{H}^T = \mathbb{e} \cdot \mathbb{H}^T.$$

It follows from (4.76) and $\mathbb{P}^T = \mathbb{P}$ that

$$\begin{aligned} \mathbb{s} &= (\mathbb{x}, \mathbb{y}) \cdot \begin{bmatrix} \mathbb{I}_{12} \\ \mathbb{P} \end{bmatrix} \\ &= \mathbb{x} \cdot \mathbb{I}_{12} + \mathbb{y} \cdot \mathbb{P} \\ &= \mathbb{x} + \mathbb{y} \cdot \mathbb{P}. \end{aligned} \tag{4.78}$$

Using the property $\mathbb{P} \cdot \mathbb{P}^T = \mathbb{I}_{12}$, we can express $\mathbb{y}$ in terms of $\mathbb{x}$, $\mathbb{s}$, and $\mathbb{P}$ as follows:

$$\mathbb{y} = (\mathbb{s} + \mathbb{x}) \cdot \mathbb{P}. \tag{4.79}$$

In the following, we first show that a correctable error pattern for the (24, 12) Golay code can be expressed in terms of $\mathbb{P}$, $\mathbb{p}_i$, $\mathbb{u}^{(i)}$, and $\mathbb{s}$. We then present a decoding algorithm for the code.

For any correctable error pattern with weight $w(\mathbb{e}) \leq 3$, we have the following four possibilities:

$$\begin{aligned} &(1)\ w(\mathbb{x}) \leq 3 \text{ and } w(\mathbb{y}) = 0, \\ &(2)\ w(\mathbb{x}) \leq 2 \text{ and } w(\mathbb{y}) = 1, \\ &(3)\ w(\mathbb{x}) \leq 1 \text{ and } w(\mathbb{y}) = 2, \\ &(4)\ w(\mathbb{x}) = 0 \text{ and } w(\mathbb{y}) = 3. \end{aligned}$$

Theses four possibilities define four different types of correctable error patterns. For $0 \leq j \leq 3$, let $\mathbb{e}^{(j)} = (\mathbb{x}, \mathbb{y})$, for which $w(\mathbb{y}) = j$, and $w(\mathbb{x}) \leq 3 - j$. Suppose $\mathbb{e} = \mathbb{e}^{(0)}$. It follows from (4.78) that $\mathbb{s} = \mathbb{x}$ and $w(\mathbb{s}) = w(\mathbb{x}) \leq 3$. In this case,

$$\mathbb{e} = (\mathbb{s}, \mathbb{0}),$$

where $\mathbb{0}$ is the all-zero 12-tuple. Suppose $\mathbb{e} = \mathbb{e}^{(1)}$ and $\mathbb{y} = \mathbb{u}^{(i)}$. Then, it follows from (4.78) that

$$\mathbb{s} = \mathbb{x} + \mathbb{u}^{(i)} \cdot \mathbb{P} = \mathbb{x} + \mathbb{p}_i.$$

Hence, $\mathbb{x} = \mathbb{s} + \mathbb{p}_i$, and $w(\mathbb{s} + \mathbb{p}_i) = w(\mathbb{x}) \leq 2$. In this case,

$$\mathbb{e} = (\mathbb{s} + \mathbb{p}_i,\ \mathbb{u}^{(i)}).$$

Suppose $\mathbb{e} = \mathbb{e}^{(2)}$ or $\mathbb{e}^{(3)}$, and $w(\mathbb{x}) = 0$. It follows from (4.79) that

$$\mathbb{y} = \mathbb{s} \cdot \mathbb{P},$$

and $w(\mathbb{s} \cdot \mathbb{P}) = w(\mathbb{y}) = 2$ or 3. For this case, we can express $\mathbb{e}$ as follows:

$$\mathbb{e} = (\mathbb{0},\ \mathbb{s} \cdot \mathbb{P}).$$

Now, suppose $\mathbb{e} = \mathbb{e}^{(2)}$, and $w(\mathbb{x}) = 1$. If the nonzero component of $\mathbb{x}$ is at the ith position, then $\mathbb{x} = \mathbb{u}^{(i)}$. It follows from (4.79) that

$$\begin{aligned} \mathbb{y} &= (\mathbb{s} + \mathbb{u}^{(i)}) \cdot \mathbb{P} \\ &= \mathbb{s} \cdot \mathbb{P} + \mathbb{u}^{(i)} \cdot \mathbb{P} \\ &= \mathbb{s} \cdot \mathbb{P} + \mathbb{p}_i \end{aligned}$$

and $w(\mathbf{s} \cdot \mathbb{P} + \mathbf{p}_i) = w(\mathbf{y}) = 2$. Consequently, we can express $\mathbf{e}$ as follows:

$$\mathbf{e} = (\mathbf{u}^{(i)},\ \mathbf{s} \cdot \mathbb{P} + \mathbf{p}_i).$$

A decoding algorithm can be devised for the (24, 12) Golay code based on the preceding analysis and expressions of correctable error patterns. The decoding consists of the following steps:

Step 1. Compute the syndrome $\mathbf{s}$ of the received sequence $\mathbf{r}$.
Step 2. If $w(\mathbf{s}) \leq 3$, then set $\mathbf{e} = (\mathbf{s}, \mathbf{0})$ and go to step 8.
Step 3. If $w(\mathbf{s} + \mathbf{p}_i) \leq 2$ for some row $\mathbf{p}_i$ in $\mathbb{P}$, then set $\mathbf{e} = (\mathbf{s} + \mathbf{p}_i, \mathbf{u}^{(i)})$ and go to step 8.
Step 4. Compute $\mathbf{s} \cdot \mathbb{P}$.
Step 5. If $w(\mathbf{s} \cdot \mathbb{P}) = 2$ or 3, then set $\mathbf{e} = (\mathbf{0}, \mathbf{s} \cdot \mathbb{P})$ and go to step 8.
Step 6. If $w(\mathbf{s} \cdot \mathbb{P} + \mathbf{p}_i) = 2$ for some row $\mathbf{p}_i$ in $\mathbb{P}$, then set $\mathbf{e} = (\mathbf{u}^{(i)}, \mathbf{s} \cdot \mathbb{P} + \mathbf{p}_i)$ and go to step 8.
Step 7. If the syndrome does not correspond to a correctable error pattern, stop the decoding process, or request a retransmission. (This represents a decoding failure.)
Step 8. Set the decoded codeword $\mathbf{v}^* = \mathbf{r} + \mathbf{e}$ and stop.

EXAMPLE 4.7

Suppose the (24, 12) Golay code is used for error control. Let $\mathbf{r} = (1\,0\,0\,0\,0\,0\,1\,1\,0\,1\,0\ 0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,1)$ be the received sequence. To decode $\mathbf{r}$, we first compute the syndrome $\mathbf{s}$ of $\mathbf{r}$:

$$\mathbf{s} = \mathbf{r} \cdot \mathbb{H}^T = (1\,1\,1\,0\,1\,1\,1\,1\,1\,1\,0\,0).$$

Because $w(\mathbf{s}) > 3$, we go to decoding step 3. We find that

$$\begin{aligned}\mathbf{s} + \mathbf{p}_{11} &= (1\,1\,1\,0\,1\,1\,1\,1\,1\,1\,0\,0) + (1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,1\,0)\\ &= (0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0),\end{aligned}$$

and $w(\mathbf{s} + \mathbf{p}_{11}) = 2$. So we set

$$\begin{aligned}\mathbf{e} &= (\mathbf{s} + \mathbf{p}_{11},\ \mathbf{u}^{(11)})\\ &= (0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1)\end{aligned}$$

and decode $\mathbf{r}$ into

$$\begin{aligned}\mathbf{v}^* &= \mathbf{r} + \mathbf{e}\\ &= (1\,0\,0\,1\,0\,0\,1\,1\,0\,1\,1\,0\,1\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0).\end{aligned}$$

4.7 PRODUCT CODES

Besides the $|\mathbf{u}|\mathbf{u} + \mathbf{v}|$ and squaring constructions of codes, another technique for constructing long, powerful codes from short component codes is the *product coding* technique.

Let C_1 be an (n_1, k_1) linear code, and let C_2 be an (n_2, k_2) linear code. Then, an $(n_1 n_2, k_1 k_2)$ linear code can be formed such that each codeword is a rectangular array of n_1 columns and n_2 rows in which every row is a codeword in C_1, and every column is a codeword in C_2, as shown in Figure 4.3. This two-dimensional code is called the *direct product* (or simply the *product*) of C_1, and C_2 [25]. The $k_1 k_2$ digits in the upper right corner of the array are information symbols. The digits in the upper left corner of this array are computed from the parity-check rules for C_1 on rows, and the digits in the lower right corner are computed from the parity-check rules for C_2 on columns. Now, should we compute the check digits in the lower left corner by using the parity-check rules for C_2 on columns or the parity-check rules for C_1 on rows? It turns out that either way yields the same $(n_1 - k_1) \times (n_2 - k_2)$ check digits (see Problem 4.21), and it is possible to have all row codewords in C_1 and all column codewords in C_2 simultaneously.

The product code $C_1 \times C_2$ is encoded in two steps. A message of $k_1 k_2$ information symbols is first arranged as shown in the upper right corner of Figure 4.3. At the first step of encoding, each row of the information array is encoded into a codeword in C_1. This row encoding results in an array of k_2 rows and n_1 columns, as shown in the upper part of Figure 4.3. At the second step of encoding each of the n_1 columns of the array formed at the first encoding step is encoded into a codeword in C_2. This results in a code array of n_2 rows and n_1 columns, as shown in Figure 4.3. This code array also can be formed by first performing the column encoding and then the row encoding. Transmission can be carried out either column by column or row by row.

If code C_1 has minimum weight d_1 and code C_2 has minimum weight d_2, the minimum weight of the product code is exactly $d_1 d_2$. A minimum-weight codeword in the product code is formed by (1) choosing a minimum-weight codeword in C_1 and a minimum-weight codeword in C_2 and (2) forming an array in which all columns corresponding to zeros in the codeword from C_1 are zeros, and all columns corresponding to ones in the codeword from C_1 are the minimum-weight codeword chosen from C_2.

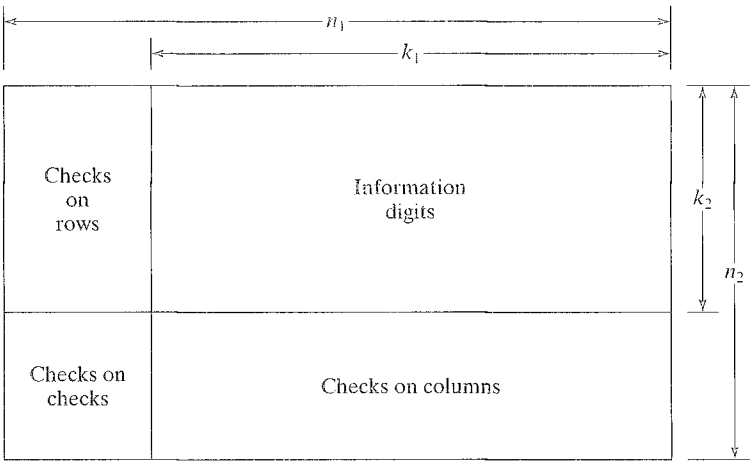

FIGURE 4.3: Code array for the product code $C_1 \times C_2$.

It is not easy to characterize the correctable error patterns for the product code; this depends on how the correction is done. One method involves using a two-step decoding. The decoding is first performed on rows and then on columns. In this case a pattern will be correctable if and only if the uncorrectable patterns on rows after row correction leave correctable patterns on the columns. It generally improves the correction to decode by rows, then columns, then columns and rows again. This method, of course, increases the decoding delay. This type of decoding is called *iterative decoding* [25].

The product code is capable of correcting any combination of $[(d_1d_2 - 1)/2]$ errors, but the method described will not achieve this. For example, consider the product code of two Hamming single-error-correcting codes. The minimum distance of each is 3, so the minimum distance of the product is 9. A pattern of four errors at the corners of a rectangle gives two errors in each of the two rows and two columns and is therefore not correctable by simple correction on rows and columns. Nevertheless, simple correction on rows and columns, although nonoptimum, can be very effective. The complexity of the two-step decoding is roughly the sum of the complexities of the two component code decodings.

EXAMPLE 4.8

Consider the product of the (5, 4) SPC code C_1 with itself. The product code $C_1 \times C_1$ is a (25, 16) linear block code C with a minimum distance of 4. Suppose the message $\mathbf{u} = (1\ 0\ 1\ 1\ 0\ 0\ 0\ 1\ 0\ 1\ 0\ 1\ 1\ 1\ 0\ 1)$ is to be encoded. This message is read into a storage buffer and arranged into a 4×4 information array, as shown in Figure 4.4. The first four information symbols form the first row of the information array, the second four information symbols form the second row, and so on. At the first step of encoding, a single (even) parity-check symbol is added to each row of the information array. This results in a 4×5 array, as shown in Figure 4.4. In the second step of encoding a single (even) parity-check symbol is added to each of the five columns of the array, as shown in Figure 4.4. Suppose this code array is transmitted column by column. At the received end, the received sequence is rearranged into a 5×5 code array column by column, called the *received array*. Suppose a single error occurs at the intersection of a row and a column. The erroneous row and column containing this single error are indicated by parity-check failures, then the error is corrected by complementing the received symbol (i.e., 0 to 1, and 1 to 0) at the intersection. All the single-error patterns can be corrected in this manner. Checking the row and column parity failures cannot correct any double-error pattern, but it can detect all the double-error patterns. When a double-error pattern occurs, there

1	1	0	1	1
1	0	0	0	1
0	0	1	0	1
1	1	1	0	1
1	0	0	1	0

FIGURE 4.4: A code array of the product of the (5, 4) SPC code with itself.

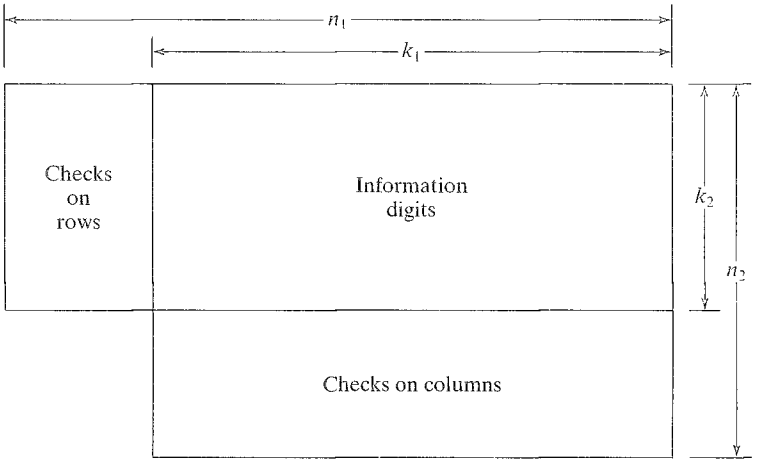

FIGURE 4.5: Incomplete product of two codes.

are three possible distributions of the two errors: (1) they are in the same row; (2) they are in the same column; or (3) they are in different rows and different columns. In the first case, there are two column parity failures but no row parity failure. Hence, errors are detected but they cannot be located. In the second case, there are two row parity failures but no column parity failure. Again, errors are detected but cannot be located. In the third case, there are two row parity failures and two column parity failures, so there are four intersecting locations. The two errors are situated at two opposite diagonal positions, but we cannot determine the positions.

In constructing a two-dimensional product code, if we do not form the $(n_1 - k_1) \times (n_2 - k_2)$ checks on checks in the lower left corner of Figure 4.3, we obtain an incomplete code array, as shown in Figure 4.5. This incomplete product of two codes results in a $(k_1 n_2 + k_2 n_1 - k_1 k_2, k_1 k_2)$ linear block code with a minimum distance of $d_1 + d_2 - 1$ (see Problem 4.22). The code has a higher rate but smaller minimum distance than the complete product code.

4.8 INTERLEAVED CODES

Given an (n, k) linear block code C, it is possible to construct a $(\lambda n, \lambda k)$ linear block code (i.e., a code λ times as long with λ times as many information symbols) by interleaving, that is, simply by arranging λ codewords in C into λ rows of a rectangular array and then transmitting the array column by column, as shown in Figure 4.6. The resulting code, denoted by C^{λ}, is called an *interleaved code*. The parameter λ is referred to as the *interleaving depth* (or *degree*). If the minimum distance of the base code C is d_{min}, the minimum distance of the interleaved code is also d_{min}.

The obvious way to implement an interleaved code is to set up the code array and operate on rows in encoding and decoding. In this way, a pattern of errors can be corrected for the whole array if and only if the pattern of errors in each row

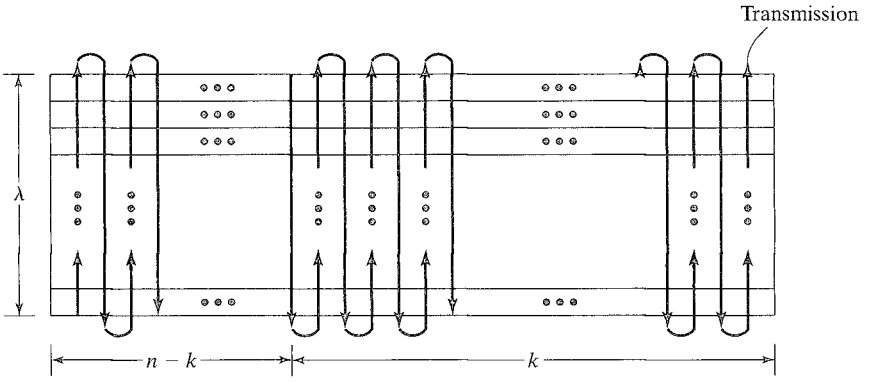

FIGURE 4.6: Transmission of an interleaved code.

is a correctable pattern for the original code C. The interleaving technique is very effective for deriving long, powerful codes for correcting errors that cluster to form *bursts*. This topic will be discussed in a later chapter.

Interleaving a single code can easily be generalized to interleaving several different codes of the same length. For $1 \leq i \leq \lambda$, let C_i be an (n, k_i) linear block code. Take λ codewords, one from each code, and arrange them as λ rows of a rectangular array as follows:

$$\begin{bmatrix} v_{1,0}, & v_{1,1}, & \cdots, & v_{1,n-1} \\ v_{2,0}, & v_{2,1}, & \cdots, & v_{2,n-1} \\ \vdots & & & \\ v_{\lambda,0}, & v_{\lambda,1}, & \cdots, & v_{\lambda,n-1} \end{bmatrix}. \tag{4.80}$$

Then, transmit this array column by column. This interleaving of λ codes results in an $(\lambda n, k_1 + k_2 + \cdots + k_\lambda)$ linear block code, denoted by $C^\lambda = C_1 * C_2 * \cdots * C_\lambda$. Each column of the array given in (4.80) is a binary λ-tuple. If each column of (4.80) is regarded as an element in Galois field $GF(2^\lambda)$, then C^λ may be regarded as a linear block code with symbols from $GF(2^\lambda)$.

The interleaving technique presented here is called *block interleaving*. Other types of interleaving will be discussed in later chapters and can be found in [26].

PROBLEMS

4.1 Form a parity-check matrix for the (15, 11) Hamming code. Devise a decoder for the code.

4.2 Show that Hamming codes achieve the Hamming bound (see Problem 3.15).

4.3 Show that the probability of an undetected error for Hamming codes of length $2^m - 1$ on a BSC with transition probability p satisfies the upper bound 2^{-m} for $p \leq 1/2$. (*Hint:* Use the inequality $(1 - 2p) \leq (1 - p)^2$.)

4.4 Compute the probability of an undetected error for the (15, 11) code on a BSC with transition probability $p = 10^{-2}$.

4.5 Devise a decoder for the (22, 16) SEC–DED code whose parity-check matrix is given in Figure 4.1(a).

4.6 Form the generator matrix of the first-order RM code RM(1, 3) of length 8. What is the minimum distance of the code? Determine its parity-check sums and devise a majority-logic decoder for the code. Decode the received vector $\mathbb{r} = (0\,1\,0\,0\,0\,1\,0\,1)$.

4.7 Form the generator matrix of the first-order RM code RM(1, 4) of length 16. What is the minimum distance of the code? Determine its parity-check sums and devise a majority-logic decoder for the code. Decode the received vector $\mathbb{r} = (0\,0\,1\,1\,0\,0\,1\,0\,0\,1\,1\,1\,0\,0\,1\,1)$.

4.8 Find the parity-check sums for the second-order RM code RM(2, 5) of length 32. What is the minimum distance of the code? Form the parity-check sums for the code. Describe the decoding steps.

4.9 Prove that the $(m - r - 1)$th-order RM code, RM$(m - r - 1, m)$, is the dual code of the rth-order RM code, RM(r, m).

4.10 Show that the RM(1, 3) and RM(2, 5) codes are self-dual.

4.11 Find a parity-check matrix for the RM(1, 4) code.

4.12 Construct the RM(2, 5) code of length 32 from RM codes of length 8 using $|\mathbb{u}|\mathbb{u} + \mathbb{v}|$-construction.

4.13 Using the $|\mathbb{u}|\mathbb{u} + \mathbb{v}|$-construction, decompose the RM(2, 5) code into component codes that are either repetition codes of dimension 1 or even parity-check codes of minimum distance 2.

4.14 Determine the Boolean polynomials that give the codewords of the RM(1, 3) code.

4.15 Use Boolean representation to show that the RM(r, m) code can be constructed from RM$(r, m - 1)$ and RM$(r - 1, m - 1)$ codes.

4.16 Construct the RM(2, 4) code from the RM(2, 3) and RM(1, 3) codes using one-level squaring construction. Find its generator matrix in the form of (4.53) or (4.68).

4.17 Using two-level squaring construction, express the generator matrix of the RM(2, 4) code in the forms of (4.60) and (4.61).

4.18 Prove that the (24, 12) Golay code is self-dual. (*Hint:* Show that $\mathbb{G} \cdot \mathbb{G}^T = 0$.)

4.19 Design an encoding circuit for the (24, 12) Golay code.

4.20 Suppose that the (24, 12) Golay code is used for error correction. Decode the following received sequences:

a. $\mathbb{r} = (1\,0\,1\,1\,0\,1\,1\,1\,0\,0\,1\,0\,0\,0\,0\,0\,1\,1\,0\,0\,0\,0\,1\,1)$,

b. $\mathbb{r} = (0\,0\,1\,1\,1\,1\,1\,1\,0\,0\,1\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,0\,1)$.

4.21 Show that the digits for checking the parity-check digits of a product code array shown in Figure 4.3 are the same no matter whether they are formed by using the parity-check rules for C_2 on columns or the parity-check rules for C_1 on rows.

4.22 Prove that the minimum distance of the incomplete product of an (n_1, k_1, d_1) linear code and an (n_2, k_2, d_2) linear code is $d_1 + d_2 - 1$.

4.23 The incomplete product of the $(n_1, n_1 - 1, 2)$ and the $(n_2, n_2 - 1, 2)$ even parity-check codes has a minimum distance of 3. Devise a decoding algorithm for correcting a single error in the information part of a code array.

BIBLIOGRAPHY

1. R. W. Hamming, "Error Detecting and Error Correcting Codes," *Bell Syst. Tech. J.*, 29: 147–60, April 1950.

2. W. W. Peterson and E. J. Weldon, Jr., *Error Correcting Codes*, 2d ed., MIT Press, Cambridge, 1972.

3. M. J. E. Golay, "Notes on Digital Coding," *Proc. IEEE*, 37: 657, June 1949.

4. A. Tietavainen, "On the Nonexistence of Perfect Codes over Finite Fields," *SIAM J. Appl. Math*, 24: 88–96, 1973.

5. V. Pless, *Introduction to the Theory of Error Correcting Codes*, 2d ed., John Wiley, New York, 1989.

6. S. K. Leung-Yan-Cheong and M. E. Hellman, "Concerning a Bound on Undetected Error Probability," *IEEE Trans. Inform. Theory*, IT-22: 235–37, March 1976.

7. T. Klove, *Error Detecting Codes*, Kluwer Academic, Boston, Mass., 1995.

8. M. Y. Hsiao, "A Class of Optimal Minimum Odd-Weight-Column SEC–DED Codes," *IBM J. Res. Dev.*, 14, July 1970.

9. D. E. Muller, "Applications of Boolean Algebra to Switching Circuits Design and to Error Detection," *IRE Trans.*, EC-3: 6–12, September 1954.

10. I. S. Reed, "A Class of Multiple-Error-Correcting Codes and the Decoding Scheme," *IRE Trans.*, IT-4: 38–49, September 1954.

11. J. L. Massey, *Threshold Decoding*, MIT Press, Cambridge, 1963.

12. F. J. MacWilliams and N. J. A. Sloane, *The Theory of Error-Correcting Codes*, North Holland, Amsterdam, 1977.

13. M. Plotkin, "Binary Codes with Specific Minimum Distance," *IEEE Trans. Inform. Theory*, IT-6: 445–50, November 1960.

14. V. P. Nelson, H. T. Nagle, B. D. Carroll, and J. D. Irwin, *Digital Logic Circuit Analysis & Design*, Prentice Hall, Englewood Cliffs, N.J., 1995.

15. G. D. Forney Jr. "Coset Codes II: Binary Lattices and Related Codes," *IEEE Trans. Inform. Theory*, IT-34: 1152–1187, September 1988.

16. S. Lin, T. Kasami, T. Fujiwara, and M. Fossorier, *Trellises and Trellis-Based Decoding Algorithms for Linear Block Codes*, Kluwer Academic, Boston, Mass., 1998.

17. S. Lin, "Some Codes Which Are Invariant under a Transitive Permutation Group and Their Connection with Balanced Incomplete Block Designs," chap. 24 in *Combinatorial Mathematics and Its Applications*, ed. R. C. Bose and T. A. Dowling, University of North Carolina Press, Chapel Hill, 1969.

18. T. Kasami, "The Weight Enumerators for Several Classes of Subcodes of the Second-Order Binary Reed–Muller Codes," *Inform. and Control*, 18: 369–94, 1971.

19. M. Sugino, Y. Ienaga, N. Tokura, and T. Kasami, "Weight Distribution of (128, 64) Reed–Muller Code," *IEEE Trans. Inform. Theory*, IT-17: 627–28, September 1971.

20. T. Kasami, N. Tokura, and S. Azumi, "On the Weight Enumeration of Weight Less than $2.5d$ of Reed–Muller Codes," *Inform. and Control*, 30: 380–95, April 1976.

21. T. Sugita, T. Kasami, and T. Fujiwara, "The Weight Distributions of the Third-Order Reed–Muller Code of Length 512," *IEEE Trans. Inform. Theory*, IT-42: 1622–25, September 1996.

22. R. G. Gallager, *Low-Density Parity-Check Codes*, MIT Press, Cambridge, 1960.

23. S. A. Vanstone and P. C. van Oorschot, *An Introduction to Error Correcting Codes with Applications*, Kluwer Academic, Boston, Mass., 1989.

24. S. B. Wicker, *Error Control Systems for Digital Communication and Storage*, Prentice Hall, Englewood Cliffs, N.J., 1995.

25. P. Elias, "Error-Free Coding," *IRE Trans. Inform. Theory*, PGIT-4: 29–37, September 1954.

26. G. C. Clark Jr. and J. B. Cain, *Error-Correction Coding for Digital Communications,* Plenum, New York, 1981.