CHAPTER 5

# Cyclic Codes

Cyclic codes form an important subclass of linear block codes. These codes are attractive for two reasons: first, encoding and syndrome computation can be implemented easily by employing shift registers with feedback connections (known as linear sequential circuits); and second, because they have considerable inherent algebraic structure, it is possible to devise various practical methods for decoding them. Cyclic codes are widely used in communication systems for error control. They are particularly efficient for error detection.

Cyclic codes were first studied by Eugene Prange in 1957 [1]. Since then, progress in the study of cyclic codes for both random-error correction and burst-error correction has been spurred by many algebraic coding theorists. Many classes of cyclic codes have been constructed over the years, including BCH codes, Reed–Solomon codes, Euclidean geometry codes, projective geometry codes, quadratic residue codes, and Fire codes, which will be discussed in later chapters. Excellent expositions of cyclic codes can be found in [2–5]. References [6–9] also provide good coverage of cyclic codes.

## 5.1 DESCRIPTION OF CYCLIC CODES

If we cyclically shift the components of an $n$-tuple $\mathbf{v} = (v_0, v_1, \cdots, v_{n-1})$ one place to the right, we obtain another $n$-tuple,

$$\mathbf{v}^{(1)} = (v_{n-1}, v_0, \cdots, v_{n-2}),$$

which is called a *cyclic shift* of $\mathbf{v}$. If the components of $\mathbf{v}$ are cyclically shifted $i$ places to the right, the resultant $n$-tuple is

$$\mathbf{v}^{(i)} = (v_{n-i}, v_{n-i+1}, \cdots, v_{n-1}, v_0, v_1, \cdots, v_{n-i-1}).$$

Clearly, cyclically shifting $\mathbf{v}$ $i$ places to the right is equivalent to cyclically shifting $\mathbf{v}$ $n - i$ places to the left.

**Definition 5.1** An $(n, k)$ linear code $C$ is called a *cyclic code* if every cyclic shift of a codeword in $C$ is also a codeword in $C$.

The (7, 4) linear code given in Table 5.1 is a cyclic code.

To develop the algebraic properties of a cyclic code, we treat the components of a codeword $\mathbf{v} = (v_0, v_1, \cdots, v_{n-1})$ as the coefficients of a polynomial as follows:

$$\mathbf{v}(X) = v_0 + v_1 X + v_2 X^2 + \cdots + v_{n-1} X^{n-1}.$$

Thus, each codeword corresponds to a polynomial of degree $n-1$ or less. If $v_{n-1} \neq 0$, the degree of $\mathbf{v}(X)$ is $n - 1$; if $v_{n-1} = 0$, the degree of $\mathbf{v}(X)$ is less than $n - 1$. The correspondence between the codeword $\mathbf{v}$ and the polynomial $\mathbf{v}(X)$ is one-to-one. We

TABLE 5.1: A (7, 4) cyclic code generated by $g(X) = 1 + X + X^3$.

| Messages | Code vectors | Code polynomials |
|---|---|---|
| (0000) | 0000000 | $0 = 0 \cdot g(X)$ |
| (1000) | 1101000 | $1 + X + X^3 = 1 \cdot g(X)$ |
| (0100) | 0110100 | $X + X^2 + X^4 = X \cdot g(X)$ |
| (1100) | 1011100 | $1 + X^2 + X^3 + X^4 = (1 + X) \cdot g(X)$ |
| (0010) | 0011010 | $X^2 + X^3 + X^5 = X^2 \cdot g(X)$ |
| (1010) | 1110010 | $1 + X + X^2 + X^5 = (1 + X^2) \cdot g(X)$ |
| (0110) | 0101110 | $X + X^3 + X^4 + X^5 = (X + X^2) \cdot g(X)$ |
| (1110) | 1000110 | $1 + X^4 + X^5 = (1 + X + X^2) \cdot g(X)$ |
| (0001) | 0001101 | $X^3 + X^4 + X^6 = X^3 \cdot g(X)$ |
| (1001) | 1100101 | $1 + X + X^4 + X^6 = (1 + X^3) \cdot g(X)$ |
| (0101) | 0111001 | $X + X^2 + X^3 + X^6 = (X + X^3) \cdot g(X)$ |
| (1101) | 1010001 | $1 + X^2 + X^6 = (1 + X + X^3) \cdot g(X)$ |
| (0011) | 0010111 | $X^2 + X^4 + X^5 + X^6 = (X^2 + X^3) \cdot g(X)$ |
| (1011) | 1111111 | $1 + X + X^2 + X^3 + X^4 + X^5 + X^6$ $= (1 + X^2 + X^3) \cdot g(X)$ |
| (0111) | 0100011 | $X + X^5 + X^6 = (X + X^2 + X^3) \cdot g(X)$ |
| (1111) | 1001011 | $1 + X^3 + X^5 + X^6$ $= (1 + X + X^2 + X^3) \cdot g(X)$ |

shall call $v(X)$ the code polynomial of $v$. Hereafter, we shall use the terms *codeword* and *code polynomial* interchangeably. The code polynomial that corresponds to codeword $v^{(i)}$ is

$$v^{(i)}(X) = v_{n-i} + v_{n-i+1}X + \cdots + v_{n-1}X^{i-1} + v_0X^i + v_1X^{i+1} + \cdots + v_{n-i-1}X^{n-1}.$$

There exists an interesting algebraic relationship between $v(X)$ and $v^{(i)}(X)$. Multiplying $v(X)$ by $X^i$, we obtain

$$X^i v(X) = v_0X^i + v_1X^{i+1} + \cdots + v_{n-i-1}X^{n-1} + \cdots + v_{n-1}X^{n+i-1}.$$

The preceding equation can be manipulated into the following form:

$$\begin{aligned} X^i v(X) &= v_{n-i} + v_{n-i+1}X + \cdots + v_{n-1}X^{i-1} + v_0X^i + \cdots + v_{n-i-1}X^{n-1} \\ &\quad + v_{n-i}(X^n + 1) + v_{n-i+1}X(X^n + 1) + \cdots + v_{n-1}X^{i-1}(X^n + 1) \qquad (5.1) \\ &= q(X)(X^n + 1) + v^{(i)}(X), \end{aligned}$$

where $q(X) = v_{n-i} + v_{n-i+1}X + \cdots + v_{n-1}X^{i-1}$. From (5.1) we see that the code polynomial $v^{(i)}(X)$ is simply the remainder resulting from dividing the polynomial $X^i v(X)$ by $X^n + 1$.

Next, we prove a number of important algebraic properties of a cyclic code that make possible the simple implementation of encoding and syndrome computation.

THEOREM 5.1 The nonzero code polynomial of minimum degree in a cyclic code $C$ is unique.

*Proof.* Let $\mathbf{g}(X) = g_0 + g_1X + \cdots + g_{r-1}X^{r-1} + X^r$ be a nonzero code polynomial of minimum degree in $C$. Suppose that $\mathbf{g}(X)$ is not unique. Then, there exists another code polynomial of degree $r$, say $\mathbf{g}'(X) = g'_0 + g'_1X + \cdots + g'_{r-1}X^{r-1} + X^r$. Because $C$ is linear, $\mathbf{g}(X) + \mathbf{g}'(X) = (g_0 + g'_0) + (g_1 + g'_1)X + \cdots + (g_{r-1} + g'_{r-1})X^{r-1}$ is a code polynomial of degree less than $r$. If $\mathbf{g}(X)+\mathbf{g}'(X) \neq 0$, then $\mathbf{g}(X)+\mathbf{g}'(X)$ is a nonzero code polynomial of degree less than the minimum degree $r$. This is impossible. Therefore, $\mathbf{g}(X) + \mathbf{g}'(X) = 0$. This implies that $\mathbf{g}'(X) = \mathbf{g}(X)$. Hence, $\mathbf{g}(X)$ is unique. Q.E.D.

THEOREM 5.2 Let $\mathbf{g}(X) = g_0 + g_1X + \cdots + g_{r-1}X^{r-1} + X^r$ be the nonzero code polynomial of minimum degree in an $(n, k)$ cyclic code $C$. Then, the constant term $g_0$ must be equal to 1.

*Proof.* Suppose that $g_0 = 0$. Then,

$$\begin{aligned}\mathbf{g}(X) &= g_1X + g_2X^2 + \cdots + g_{r-1}X^{r-1} + X^r \\ &= X(g_1 + g_2X + \cdots + g_{r-1}X^{r-2} + X^{r-1}).\end{aligned}$$

If we shift $\mathbf{g}(X)$ cyclically $n-1$ places to the right (or one place to the left), we obtain a nonzero code polynomial, $g_1 + g_2X + \cdots + g_{r-1}X^{r-2} + X^{r-1}$, of degree less than $r$. This is a contradiction to the assumption that $\mathbf{g}(X)$ is the nonzero code polynomial with minimum degree. Thus, $g_0 \neq 0$. Q.E.D.

It follows from Theorem 5.2 that the nonzero code polynomial of minimum degree in an $(n, k)$ cyclic code $C$ is of the following form:

$$\mathbf{g}(X) = 1 + g_1X + g_2X^2 + \cdots + g_{r-1}X^{r-1} + X^r. \tag{5.2}$$

Consider the (7, 4) cyclic code given in Table 5.1. The nonzero code polynomial of minimum degree is $\mathbf{g}(X) = 1 + X + X^3$.

Consider the polynomials $X\mathbf{g}(X), X^2\mathbf{g}(X), \cdots, X^{n-r-1}\mathbf{g}(X)$, of degrees $r+1, r+2, \cdots, n-1$, respectively. It follows from (5.1) that $X\mathbf{g}(X) = \mathbf{g}^{(1)}(X)$, $X^2\mathbf{g}(X) = \mathbf{g}^{(2)}(X), \cdots, X^{n-r-1}\mathbf{g}(X) = \mathbf{g}^{(n-r-1)}(X)$; that is, they are cyclic shifts of the code polynomial $\mathbf{g}(X)$. Therefore, they are code polynomials in $C$. Since $C$ is linear, a linear combination of $\mathbf{g}(X), X\mathbf{g}(X), \cdots, X^{n-r-1}\mathbf{g}(X)$,

$$\begin{aligned}\mathbf{v}(X) &= u_0\mathbf{g}(X) + u_1X\mathbf{g}(X) + \cdots + u_{n-r-1}X^{n-r-1}\mathbf{g}(X) \\ &= (u_0 + u_1X + \cdots + u_{n-r-1}X^{n-r-1})\mathbf{g}(X),\end{aligned} \tag{5.3}$$

is also a code polynomial, where $u_i = 0$ or 1. The following theorem characterizes an important property of a cyclic code.

THEOREM 5.3 Let $\mathbf{g}(X) = 1 + \mathbf{g}_1X + \cdots + \mathbf{g}_{r-1}X^{r-1} + X^r$ be the nonzero code polynomial of minimum degree in an $(n, k)$ cyclic code $C$. A binary polynomial of degree $n-1$ or less is a code polynomial if and only if it is a multiple of $\mathbf{g}(X)$.

*Proof.* Let $\mathbf{v}(X)$ be a binary polynomial of degree $n-1$ or less. Suppose that $\mathbf{v}(X)$ is a multiple of $\mathbf{g}(X)$. Then,

$$\begin{aligned}\mathbf{v}(X) &= (a_0 + a_1X + \cdots + a_{n-r-1}X^{n-r-1})\mathbf{g}(X) \\ &= a_0\mathbf{g}(X) + a_1X\mathbf{g}(X) + \cdots + a_{n-r-1}X^{n-r-1}\mathbf{g}(X).\end{aligned}$$

Because $\mathbb{v}(X)$ is a linear combination of the code polynomials $\mathbb{g}(X)$, $X\mathbb{g}(X)$, $\cdots$, $X^{n-r-1}\mathbb{g}(X)$, it is a code polynomial in $C$. This proves the first part of the theorem—that if a polynomial of degree $n-1$ or less is a multiple of $\mathbb{g}(X)$, it is a code polynomial.

Now, let $\mathbb{v}(X)$ be a code polynomial in $C$. Dividing $\mathbb{v}(X)$ by $\mathbb{g}(X)$, we obtain

$$\mathbb{v}(X) = \mathbb{a}(X)\mathbb{g}(X) + \mathbb{b}(X),$$

where either $\mathbb{b}(X)$ is identical to zero, or the degree of $\mathbb{b}(X)$ is less than the degree of $\mathbb{g}(X)$. Rearranging the preceding equation, we have

$$\mathbb{b}(X) = \mathbb{v}(X) + \mathbb{a}(X)\mathbb{g}(X).$$

It follows from the first part of the theorem that $\mathbb{a}(X)\mathbb{g}(X)$ is a code polynomial. Because both $\mathbb{v}(X)$ and $\mathbb{a}(X)\mathbb{g}(X)$ are code polynomials, $\mathbb{b}(X)$ must also be a code polynomial. If $\mathbb{b}(X) \neq 0$, then $\mathbb{b}(X)$ is a nonzero code polynomial whose degree is less than the degree of $\mathbb{g}(X)$. This contradicts the assumption that $\mathbb{g}(X)$ is the nonzero code polynomial of minimum degree. Thus, $\mathbb{b}(X)$ must be identical to zero. This proves the second part of the theorem—that a code polynomial is a multiple of $\mathbb{g}(X)$. Q.E.D.

The number of binary polynomials of degree $n-1$ or less that are multiples of $\mathbb{g}(X)$ is $2^{n-r}$. It follows from Theorem 5.3 that these polynomials form all the code polynomials of the $(n, k)$ cyclic code $C$. Because there are $2^k$ code polynomials in $C$, then $2^{n-r}$ must be equal to $2^k$. As a result, we have $r = n - k$ [i.e., the degree of $\mathbb{g}(X)$ is $n-k$]. Hence, the nonzero code polynomial of minimum degree in an $(n, k)$ cyclic code is of the following form:

$$\mathbb{g}(X) = 1 + \mathbb{g}_1 X + \mathbb{g}_2 X^2 + \cdots + \mathbb{g}_{n-k-1}X^{n-k-1} + X^{n-k}.$$

Summarizing the preceding results, we have the following theorem:

THEOREM 5.4 In an $(n, k)$ cyclic code, there exists one and only one code polynomial of degree $n-k$,

$$\mathbb{g}(X) = 1 + \mathbb{g}_1 X + \mathbb{g}_2 X^2 + \cdots + \mathbb{g}_{n-k-1}X^{n-k-1} + X^{n-k}. \tag{5.4}$$

Every code polynomial is a multiple of $\mathbb{g}(X)$, and every binary polynomial of degree $n-1$ or less that is a multiple of $\mathbb{g}(X)$ is a code polynomial.

It follows from Theorem 5.4 that every code polynomial $\mathbb{v}(X)$ in an $(n, k)$ cyclic code can be expressed in the following form:

$$\begin{aligned}\mathbb{v}(X) &= \mathbb{u}(X)\mathbb{g}(X) \\ &= (u_0 + u_1 X + \cdots + u_{k-1}X^{k-1})\mathbb{g}(X).\end{aligned}$$

If the coefficients of $\mathbb{u}(X)$, $u_0, u_1, \cdots, u_{k-1}$, are the $k$ information digits to be encoded, $\mathbb{v}(X)$ is the corresponding code polynomial. Hence, the encoding can be achieved by multiplying the message $\mathbb{u}(X)$ by $\mathbb{g}(X)$. Therefore, an $(n, k)$ cyclic code is completely specified by its nonzero code polynomial of minimum degree, $\mathbb{g}(X)$,

given by (5.4). The polynomial $\mathbf{g}(X)$ is called the *generator polynomial* of the code. The degree of $\mathbf{g}(X)$ is equal to the number of parity-check digits of the code. The generator polynomial of the (7, 4) cyclic code given in Table 4.1 is $\mathbf{g}(X) = 1+X+X^3$. We see that each code polynomial is a multiple of $\mathbf{g}(X)$.

The next important property of a cyclic code is given in the following theorem.

**THEOREM 5.5** The generator polynomial $\mathbf{g}(X)$ of an $(n, k)$ cyclic code is a factor of $X^n + 1$.

***Proof.*** Multiplying $\mathbf{g}(X)$ by $X^k$ results in a polynomial $X^k\mathbf{g}(X)$ of degree $n$. Dividing $X^k\mathbf{g}(X)$ by $X^n + 1$, we obtain

$$X^k\mathbf{g}(X) = (X^n + 1) + \mathbf{g}^{(k)}(X), \tag{5.5}$$

where $\mathbf{g}^{(k)}(X)$ is the remainder. It follows from (5.1) that $\mathbf{g}^{(k)}(X)$ is the code polynomial obtained by shifting $\mathbf{g}(X)$ to the right cyclically $k$ times. Hence, $\mathbf{g}^{(k)}(X)$ is a multiple of $\mathbf{g}(X)$, say $\mathbf{g}^{(k)}(X) = \mathbf{a}(X)\mathbf{g}(X)$. From (5.5) we obtain

$$X^n + 1 = \{X^k + \mathbf{a}(X)\}\mathbf{g}(X).$$

Thus, $\mathbf{g}(X)$ is a factor of $X^n + 1$. Q.E.D.

At this point, a natural question is whether, for any $n$ and $k$, there exists an $(n, k)$ cyclic code. This question is answered by the following theorem.

**THEOREM 5.6** If $\mathbf{g}(X)$ is a polynomial of degree $n-k$ and is a factor of $X^n+1$, then $\mathbf{g}(X)$ generates an $(n, k)$ cyclic code.

***Proof.*** Consider the $k$ polynomials $\mathbf{g}(X), X\mathbf{g}(X), \cdots, X^{k-1}\mathbf{g}(X)$, all of degree $n-1$ or less. A linear combination of these $k$ polynomials,

$$\begin{aligned}\mathbf{v}(X) &= a_0\mathbf{g}(X) + a_1X\mathbf{g}(X) + \cdots + a_{k-1}X^{k-1}\mathbf{g}(X)\\ &= (a_0 + a_1X + \cdots + a_{k-1}X^{k-1})\mathbf{g}(X),\end{aligned}$$

is also a polynomial of degree $n-1$ or less and is a multiple of $\mathbf{g}(X)$. There are a total of $2^k$ such polynomials, and they form an $(n, k)$ linear code.

Let $\mathbf{v}(X) = v_0 + v_1X + \cdots + v_{n-1}X^{n-1}$ be a code polynomial in this code. Multiplying $\mathbf{v}(X)$ by $X$, we obtain

$$\begin{aligned}X\mathbf{v}(X) &= v_0X + v_1X^2 + \cdots + v_{n-2}X^{n-1} + v_{n-1}X^n\\ &= v_{n-1}(X^n + 1) + (v_{n-1} + v_0X + \cdots + v_{n-2}X^{n-1})\\ &= v_{n-1}(X^n + 1) + \mathbf{v}^{(1)}(X),\end{aligned}$$

where $\mathbf{v}^{(1)}(X)$ is a cyclic shift of $\mathbf{v}(X)$. Since both $X\mathbf{v}(X)$ and $X^n+1$ are divisible by $\mathbf{g}(X)$, $\mathbf{v}^{(1)}(X)$ must be divisible by $\mathbf{g}(X)$. Thus, $\mathbf{v}^{(1)}(X)$ is a multiple of $\mathbf{g}(X)$ and is a linear combination of $\mathbf{g}(X), X\mathbf{g}(X), \cdots, X^{k-1}\mathbf{g}(X)$. Hence, $\mathbf{v}^{(1)}(X)$ is also a code polynomial. It follows from Definition 5.1 that the linear code generated by $\mathbf{g}(X), X\mathbf{g}(X), \cdots, X^{k-1}\mathbf{g}(X)$ is an $(n, k)$ cyclic code. Q.E.D.

Theorem 5.6 says that any factor of $X^n + 1$ with degree $n - k$ generates an $(n, k)$ cyclic code. For large $n$, $X^n + 1$ may have many factors of degree $n - k$. Some of these polynomials generate good codes, and some generate bad codes. How to select generator polynomials to produce good cyclic codes is a very difficult problem, and coding theorists have expended much effort in searching for good cyclic codes. Several classes of good cyclic codes have been discovered, and they can be practically implemented.

---

EXAMPLE 5.1

The polynomial $X^7 + 1$ can be factored as follows:

$$X^7 + 1 = (1 + X)(1 + X + X^3)(1 + X^2 + X^3).$$

There are two factors of degree 3, and each generates a (7, 4) cyclic code. The (7, 4) cyclic code given by Table 5.1 is generated by $\mathbb{g}(X) = 1 + X + X^3$. This code has a minimum distance of 3 and it is a single-error-correcting code. Notice that the code is not in systematic form. Each code polynomial is the product of a message polynomial of degree 3 or less and the generator polynomial $\mathbb{g}(X) = 1 + X + X^3$. For example, let $\mathbb{u} = (1\,0\,1\,0)$ be the message to be encoded. The corresponding message polynomial is $\mathbb{u}(X) = 1 + X^2$. Multiplying $\mathbb{u}(X)$ by $\mathbb{g}(X)$ gives us the following code polynomial:

$$\begin{aligned}\mathbb{v}(X) &= (1 + X^2)(1 + X + X^3) \\ &= 1 + X + X^2 + X^5,\end{aligned}$$

or the codeword $(1\,1\,1\,0\,0\,1\,0)$.

---

Given the generator polynomial $\mathbb{g}(X)$ of an $(n, k)$ cyclic code, we can put the code into systematic form (i.e., the rightmost $k$ digits of each codeword are the unaltered information digits, and the leftmost $n - k$ digits are parity-check digits). Suppose that the message to be encoded is $\mathbb{u} = (u_0, u_1, \cdots, u_{k-1})$. The corresponding message polynomial is

$$\mathbb{u}(X) = u_0 + u_1 X + \cdots + u_{k-1} X^{k-1}.$$

Multiplying $\mathbb{u}(X)$ by $X^{n-k}$, we obtain a polynomial of degree $n - 1$ or less:

$$X^{n-k}\mathbb{u}(X) = u_0 X^{n-k} + u_1 X^{n-k+1} + \cdots + u_{k-1} X^{n-1}.$$

Dividing $X^{n-k}\mathbb{u}(X)$ by the generator polynomial $\mathbb{g}(X)$, we have

$$X^{n-k}\mathbb{u}(X) = \mathbb{a}(X)\mathbb{g}(X) + \mathbb{b}(X) \tag{5.6}$$

where $\mathbb{a}(X)$ and $\mathbb{b}(X)$ are the quotient and the remainder, respectively. Because the degree of $\mathbb{g}(X)$ is $n - k$, the degree of $\mathbb{b}(X)$ must be $n - k - 1$ or less; that is,

$$\mathbb{b}(X) = b_0 + b_1 X + \cdots + b_{n-k-1} X^{n-k-1}.$$

Rearranging (5.6), we obtain the following polynomial of degree $n-1$ or less:

$$\mathbb{b}(X) + X^{n-k}\mathbb{u}(X) = \mathbb{a}(X)\mathbb{g}(X). \tag{5.7}$$

This polynomial is a multiple of the generator polynomial $\mathbb{g}(X)$ and therefore it is a code polynomial of the cyclic code generated by $\mathbb{g}(X)$. Writing out $\mathbb{b}(X) + X^{n-k}\mathbb{u}(X)$, we have

$$\begin{aligned}\mathbb{b}(X) + X^{n-k}\mathbb{u}(X) = {} & b_0 + b_1X + \cdots + b_{n-k-1}X^{n-k-1} \\ & + u_0X^{n-k} + u_1X^{n-k+1} + \cdots + u_{k-1}X^{n-1},\end{aligned} \tag{5.8}$$

which corresponds to the codeword

$$(b_0, b_1, \cdots, b_{n-k-1}, u_0, u_1, \cdots, u_{k-1}).$$

We see that the codeword consists of $k$ unaltered information digits $(u_0, u_1, \cdots, u_{k-1})$ followed by $n-k$ parity-check digits. The $n-k$ parity-check digits are simply the coefficients of the remainder resulting from dividing the message polynomial $X^{n-k}\mathbb{u}(X)$ by the generator polynomial $\mathbb{g}(X)$. The preceding process yields an $(n, k)$ cyclic code in systematic form. In connection with cyclic codes in systematic form, the following convention is used: the first $n-k$ symbols, the coefficients of $1, X, \cdots, X^{n-k-1}$, are taken as parity-check digits, and the last $k$ symbols, the coefficients of $X^{n-k}, X^{n-k+1}, \cdots, X^{n-1}$, are taken as the information digits. In summary, encoding in systematic form consists of three steps:

**Step 1.** Premultiply the message $\mathbb{u}(X)$ by $X^{n-k}$.
**Step 2.** Obtain the remainder $\mathbb{b}(X)$ (the parity-check digits) from dividing $X^{n-k}\mathbb{u}(X)$ by the generator polynomial $\mathbb{g}(X)$.
**Step 3.** Combine $\mathbb{b}(X)$ and $X^{n-k}\mathbb{u}(X)$ to obtain the code polynomial $\mathbb{b}(X) + X^{n-k}\mathbb{u}(X)$.

---

**EXAMPLE 5.2**

Consider the $(7, 4)$ cyclic code generated by $\mathbb{g}(X) = 1 + X + X^3$. Let $\mathbb{u}(X) = 1 + X^3$ be the message to be encoded. Dividing $X^3\mathbb{u}(X) = X^3 + X^6$ by $\mathbb{g}(X)$,

$$\begin{array}{rl} & X^3+X \quad \text{(quotient)} \\ X^3+X+1 \,|\, & \underline{X^6 \qquad\quad X^3} \\ & \underline{X^6 \quad +X^4+ \; X^3} \\ & \quad X^4 \\ & \quad \underline{X^4 \qquad\quad +X^2+X} \\ & \qquad\qquad X^2+X \quad \text{(remainder)} \end{array}$$

we obtain the remainder $\mathbb{b}(X) = X + X^2$. Thus, the code polynomial is $\mathbb{v}(X) = \mathbb{b}(X) + X^3\mathbb{u}(X) = X + X^2 + X^3 + X^6$, and the corresponding codeword is $\mathbb{v} = (0\,1\,1\,1\,0\,0\,1)$, where the four rightmost digits are the information digits. The 16 codewords in systematic form are listed in Table 5.2.

---

TABLE 5.2: A $(7, 4)$ cyclic code in systematic form generated by $g(X) = 1 + X + X^3$.

| Message | Codeword | |
|---|---|---|
| (0000) | (0000000) | $0 = 0 \cdot g(X)$ |
| (1000) | (1101000) | $1 + X + X^3 = g(X)$ |
| (0100) | (0110100) | $X + X^2 + X^4 = Xg(X)$ |
| (1100) | (1011100) | $1 + X^2 + X^3 + X^4 = (1 + X)g(X)$ |
| (0010) | (1110010) | $1 + X + X^2 + X^5 = (1 + X^2)g(X)$ |
| (1010) | (0011010) | $X^2 + X^3 + X^5 = X^2g(X)$ |
| (0110) | (1000110) | $1 + X^4 + X^5 = (1 + X + X^2)g(X)$ |
| (1110) | (0101110) | $X + X^3 + X^4 + X^5 = (X + X^2)g(X)$ |
| (0001) | (1010001) | $1 + X^2 + X^6 = (1 + X + X^3)g(X)$ |
| (1001) | (0111001) | $X + X^2 + X^3 + X^6 = (X + X^3)g(X)$ |
| (0101) | (1100101) | $1 + X + X^4 + X^6 = (1 + X^3)g(X)$ |
| (1101) | (0001101) | $X^3 + X^4 + X^6 = X^3g(X)$ |
| (0011) | (0100011) | $X + X^5 + X^6 = (X + X^2 + X^3)g(X)$ |
| (1011) | (1001011) | $1 + X^3 + X^5 + X^6 = (1 + X + X^2 + X^3)g(X)$ |
| (0111) | (0010111) | $X^2 + X^4 + X^5 + X^6 = (X^2 + X^3)g(X)$ |
| (1111) | (1111111) | $1 + X + X^2 + X^3 + X^4 + X^5 + X^6$ $= (1 + X^2 + X^5)g(X)$ |

## 5.2 GENERATOR AND PARITY-CHECK MATRICES OF CYCLIC CODES

Consider an $(n, k)$ cyclic code $C$ with generator polynomial $g(X) = g_0 + g_1X + \cdots + g_{n-k}X^{n-k}$. In Section 5.1 we showed that the $k$ code polynomials $g(X)$, $Xg(X)$, $\cdots$, $X^{k-1}g(X)$ span $C$. If the $n$-tuples corresponding to these $k$ code polynomials are used as the rows of a $k \times n$ matrix, we obtain the following generator matrix for $C$:

$$\mathbb{G} = \begin{bmatrix} g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & g_2 & \cdots & g_{n-k} & 0 & \cdots & 0 \\ \cdot & & & & & & & & & \cdot \\ \cdot & & & & & & & & & \cdot \\ \cdot & & & & & & & & & \cdot \\ 0 & 0 & \cdots & 0 & g_0 & g_1 & g_2 & \cdots & \cdots & g_{n-k} \end{bmatrix}. \tag{5.9}$$

(Note that $g_0 = g_{n-k} = 1$.) In general, $\mathbb{G}$ is not in systematic form; however, we can put it into systematic form with row operations. For example, the $(7, 4)$ cyclic code given in Table 5.1 with generator polynomial $g(X) = 1 + X + X^3$ has the following matrix as a generator matrix:

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

Clearly, $\mathbb{G}$ is not in systematic form. If we add the first row to the third row, and if we add the sum of the first two rows to the fourth row, we obtain the following matrix:

$$\mathbb{G}' = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix},$$

which is in systematic form. This matrix generates the same code as $\mathbb{G}$.

Recall that the generator polynomial $\mathbf{g}(X)$ is a factor of $X^n + 1$, say

$$X^n + 1 = \mathbf{g}(X)\mathbb{h}(X), \tag{5.10}$$

where the polynomial $\mathbb{h}(X)$ has degree $k$ and is of the following form:

$$\mathbb{h}(X) = h_0 + h_1 X + \cdots + h_k X^k$$

with $h_0 = h_k = 1$. Next, we want to show that a parity-check matrix of $C$ may be obtained from $\mathbb{h}(X)$. Let $\mathbf{v} = (v_0, v_1, \cdots, v_{n-1})$ be a codeword in $C$. Then, $\mathbf{v}(X) = \mathbf{a}(X)\mathbf{g}(X)$. Multiplying $\mathbf{v}(X)$ by $\mathbb{h}(X)$, we obtain

$$\begin{aligned} \mathbf{v}(X)\mathbb{h}(X) &= \mathbf{a}(X)\mathbf{g}(X)\mathbb{h}(X) \\ &= \mathbf{a}(X)(X^n + 1) \\ &= \mathbf{a}(X) + X^n\mathbf{a}(X). \end{aligned} \tag{5.11}$$

Because the degree of $\mathbf{a}(X)$ is $k - 1$ or less, the powers $X^k, X^{k+1}, \cdots, X^{n-1}$ do not appear in $\mathbf{a}(X) + X^n\mathbf{a}(X)$. If we expand the product $\mathbf{v}(X)\mathbb{h}(X)$ on the left-hand side of (5.11), the coefficients of $X^k, X^{k+1}, \cdots, X^{n-1}$ must be equal to zero. Therefore, we obtain the following $n - k$ equalities:

$$\sum_{i=0}^{k} h_i v_{n-i-j} = 0 \quad \text{for } 1 \le j \le n - k. \tag{5.12}$$

Now, we take the *reciprocal* of $\mathbb{h}(X)$, which is defined as follows:

$$X^k\mathbb{h}(X^{-1}) \triangleq h_k + h_{k-1}X + h_{k-2}X^2 + \cdots + h_0 X^k. \tag{5.13}$$

We can easily see that $X^k\mathbb{h}(X^{-1})$ is also a factor of $X^n + 1$. The polynomial $X^k\mathbb{h}(X^{-1})$ generates an $(n, n-k)$ cyclic code with the following $(n-k) \times n$ matrix as a generator matrix:

$$\mathbb{H} = \begin{bmatrix} h_k & h_{k-1} & h_{k-2} & \cdot & \cdot & \cdot & \cdot & \cdot & h_0 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & h_k & h_{k-1} & h_{k-2} & \cdot & \cdot & \cdot & \cdot & \cdot & h_0 & 0 & \cdot & \cdot & 0 \\ 0 & 0 & h_k & h_{k-1} & h_{k-2} & \cdot & \cdot & \cdot & \cdot & \cdot & h_0 & \cdot & \cdot & 0 \\ \vdots & & & & & & & & & & & & & \vdots \\ 0 & 0 & \cdot & \cdot & \cdot & 0 & h_k & h_{k-1} & h_{k-2} & \cdot & \cdot & \cdot & \cdot & h_0 \end{bmatrix}. \tag{5.14}$$

It follows from the $n - k$ equalities of (5.12) that any codeword $\mathbf{v}$ in $C$ is orthogonal to every row of $\mathbb{H}$. Therefore, $\mathbb{H}$ is a parity-check matrix of the cyclic code $C$, and the row space of $\mathbb{H}$ is the dual code of $C$. Since the parity-check matrix $\mathbb{H}$ is obtained from the polynomial $\mathbb{h}(X)$, we call $\mathbb{h}(X)$ the *parity polynomial* of $C$. Hence, a cyclic code is also uniquely specified by its parity polynomial.

Besides deriving a parity-check matrix for a cyclic code, we have also proved another important property, which is stated in the following theorem.

THEOREM 5.7 Let $C$ be an $(n, k)$ cyclic code with generator polynomial $\mathbb{g}(X)$. The dual code of $C$ is also cyclic and is generated by the polynomial $X^k\mathbb{h}(X^{-1})$, where $\mathbb{h}(X) = (X^n + 1)/\mathbb{g}(X)$.

---

EXAMPLE 5.3

Consider the (7, 4) cyclic code given in Table 5.1 with generator polynomial $\mathbb{g}(X) = 1 + X + X^3$. The parity polynomial is

$$\begin{aligned}\mathbb{h}(X) &= \frac{X^7 + 1}{\mathbb{g}(X)} \\ &= 1 + X + X^2 + X^4.\end{aligned}$$

The reciprocal of $\mathbb{h}(X)$ is

$$\begin{aligned}X^4\mathbb{h}(X^{-1}) &= X^4(1 + X^{-1} + X^{-2} + X^{-4}) \\ &= 1 + X^2 + X^3 + X^4.\end{aligned}$$

This polynomial $X^4\mathbb{h}(X^{-1})$ divides $X^7 + 1 : (X^7 + 1)/X^4\mathbb{h}(X^{-1}) = 1 + X^2 + X^3$. If we construct all the codewords of the (7, 3) code generated by $X^4\mathbb{h}(X^{-1}) = 1 + X^2 + X^3 + X^4$, we will find that it has a minimum distance of 4. Hence, it is capable of correcting any single error and simultaneously detecting any combination of double errors.

---

We also can easily form the generator matrix in systematic form. Dividing $X^{n-k+i}$ by the generator polynomial $\mathbb{g}(X)$ for $i = 0, 1, \cdots, k - 1$, we obtain

$$X^{n-k+i} = \mathbb{a}_i(X)\mathbb{g}(X) + \mathbb{b}_i(X), \tag{5.15}$$

where $\mathbb{b}_i(X)$ is the remainder with the following form:

$$\mathbb{b}_i(X) = b_{i0} + b_{i1}X + \cdots + b_{i,n-k-1}X^{n-k-1}.$$

Because $\mathbb{b}_i(X) + X^{n-k+i}$ for $i = 0, 1, \cdots, k - 1$ are multiples of $\mathbb{g}(X)$, they are code polynomials. Arranging these $k$ code polynomials as rows of a $k \times n$ matrix, we obtain

$$\mathbb{G} = \begin{bmatrix} b_{00} & b_{01} & b_{02} & \cdots & b_{0,n-k-1} & 1 & 0 & 0 & \cdots & 0 \\ b_{10} & b_{11} & b_{12} & \cdots & b_{1,n-k-1} & 0 & 1 & 0 & \cdots & 0 \\ b_{20} & b_{21} & b_{22} & \cdots & b_{2,n-k-1} & 0 & 0 & 1 & \cdots & 0 \\ & \vdots & & & \vdots & & & & & \vdots \\ b_{k-1,0} & b_{k-1,1} & b_{k-1,2} & \cdots & b_{k-1,n-k-1} & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}, \tag{5.16}$$

which is the generator matrix of $C$ in systematic form. The corresponding parity-check matrix for $C$ is

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & b_{00} & b_{10} & b_{20} & \cdots & b_{k-1,0} \\ 0 & 1 & 0 & \cdots & 0 & b_{01} & b_{11} & b_{21} & \cdots & b_{k-1,1} \\ 0 & 0 & 1 & \cdots & 0 & b_{02} & b_{12} & b_{22} & \cdots & b_{k-1,2} \\ & \vdots & & & & & \vdots & & & \vdots \\ 0 & 0 & 0 & \cdots & 1 & b_{0,n-k-1} & b_{1,n-k-1} & b_{2,n-k-1} & \cdots & b_{k-1,n-k-1} \end{bmatrix}. \tag{5.17}$$

**EXAMPLE 5.4**

Again, consider the (7, 4) cyclic code generated by $\mathbf{g}(X) = 1 + X + X^3$. Dividing $X^3$, $X^4$, $X^5$, and $X^6$ by $\mathbf{g}(X)$, we have

$$\begin{aligned} X^3 &= \mathbf{g}(X) + (1 + X), \\ X^4 &= X\mathbf{g}(X) + (X + X^2), \\ X^5 &= (X^2 + 1)\mathbf{g}(X) + (1 + X + X^2), \\ X^6 &= (X^3 + X + 1)\mathbf{g}(X) + (1 + X^2). \end{aligned}$$

Rearranging the preceding equations, we obtain the following four code polynomials:

$$\begin{aligned} \mathbf{v}_0(X) &= 1 + X + X^3, \\ \mathbf{v}_1(X) &= X + X^2 + X^4, \\ \mathbf{v}_2(X) &= 1 + X + X^2 + X^5, \\ \mathbf{v}_3(X) &= 1 + X^2 + X^6. \end{aligned}$$

Taking these four code polynomials as rows of a $4 \times 7$ matrix, we obtain the following generator matrix in systematic form for the (7, 4) cyclic code:

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

which is identical to the matrix $G'$ obtained earlier in this section.

## 5.3 ENCODING OF CYCLIC CODES

As shown in Section 5.1, encoding of an $(n, k)$ cyclic code in systematic form consists of three steps: (1) multiplying the message polynomial $\mathbf{u}(X)$ by $X^{n-k}$; (2) dividing $X^{n-k}\mathbf{u}(X)$ by $\mathbf{g}(X)$ to obtain the remainder $\mathbf{b}(X)$; and (3) forming the codeword $\mathbf{b}(X) + X^{n-k}\mathbf{u}(X)$. All three steps can be accomplished with a division circuit that is a linear $(n - k)$-stage shift register with feedback connections based on the generator polynomial $\mathbf{g}(X) = 1 + g_1X + g_2X^2 + \cdots + g_{n-k-1}X^{n-k-1} + X^{n-k}$. Such a circuit is shown in Figure 5.1. The encoding operation is carried out as follows:

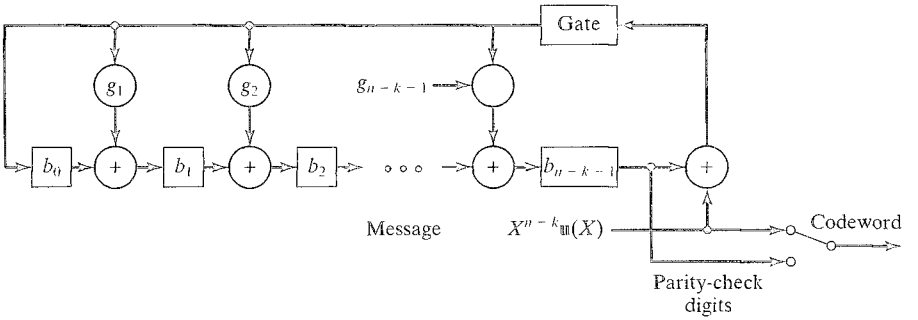


FIGURE 5.1: Encoding circuit for an $(n, k)$ cyclic code with generator polynomial $\mathbb{g}(X) = 1 + g_1X^2 + \cdots + g_{n-k-1}X^{n-k-1} + X^{n-k}$.

Step 1. Turn on the gate. The $k$ information digits $u_0, u_1, \cdots, u_{k-1}$ [or $\mathbb{u}(X) = u_0 + u_1X + \cdots + u_{k-1}X^{k-1}$ in polynomial form] are shifted into the circuit and simultaneously into the communication channel. Shifting the message $\mathbb{u}(X)$ into the circuit from the front end is equivalent to premultiplying $\mathbb{u}(X)$ by $X^{n-k}$. As soon as the complete message has entered the circuit, the $n-k$ digits in the register form the remainder, and thus they are the parity-check digits.

Step 2. Break the feedback connection by turning off the gate.

Step 3. Shift the parity-check digits out and send them into the channel. These $n-k$ parity-check digits $b_0, b_1, \cdots, b_{n-k-1}$, together with the $k$ information digits, form a complete codeword.

---

**EXAMPLE 5.5**

Consider the $(7, 4)$ cyclic code generated by $\mathbb{g}(X) = 1 + X + X^3$. The encoding circuit based on $\mathbb{g}(X)$ is shown in Figure 5.2. Suppose that the message $\mathbb{u} = (1\,0\,1\,1)$ is to be encoded. As the message digits are shifted into the register the contents of the register change as follows:

| Input | Register contents |
|---|---|
| | 0 0 0 (initial state) |
| 1 | 1 1 0 (first shift) |
| 1 | 1 0 1 (second shift) |
| 0 | 1 0 0 (third shift) |
| 1 | 1 0 0 (fourth shift) |

After four shifts, the contents of the register are $(1\,0\,0)$. Thus, the complete codeword is $(1\,0\,0\,1\,0\,1\,1)$, and the code polynomial is $1 + X^3 + X^5 + X^6$.

---

Encoding of a cyclic code can also be accomplished by using its parity polynomial $\mathbb{h}(X) = h_0 + h_1X + \cdots + h_kX^k$. Let $\mathbb{v} = (v_0, v_1, \cdots, v_{n-1})$ be a codeword. We have shown in Section 5.2 that the components of $\mathbb{v}$ satisfy the $n-k$ equalities

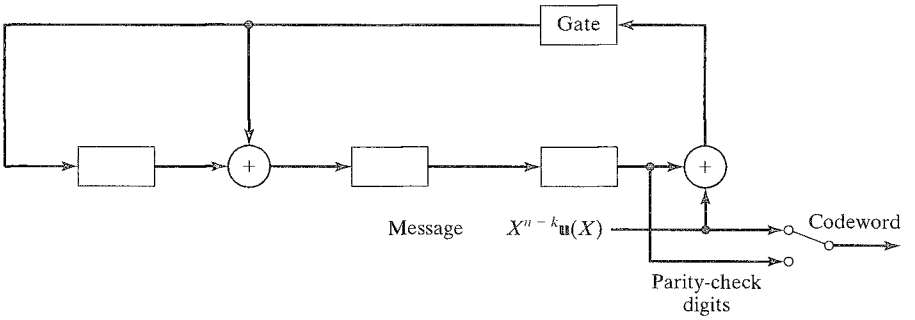


FIGURE 5.2: Encoder for the (7, 4) cyclic code generated by $\mathfrak{g}(X) = 1 + X + X^3$.

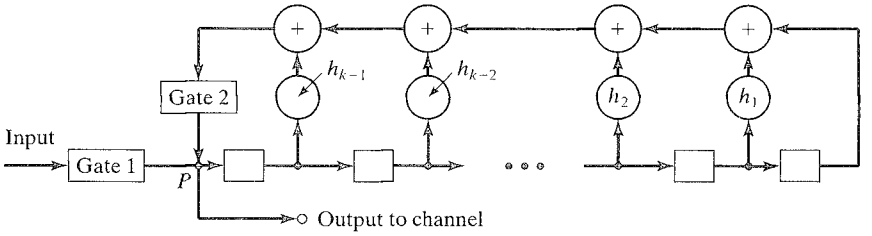


FIGURE 5.3: Encoding circuit for an $(n, k)$ cyclic code based on the parity polynomial $\mathfrak{h}(X) = 1 + h_1 X + \cdots + X^k$.

of (5.12). Since $h_k = 1$, the equalities of (5.12) can be put into the following form:

$$v_{n-k-j} = \sum_{i=0}^{k-1} h_i v_{n-i-j} \quad \text{for } 1 \le j \le n-k \tag{5.18}$$

which is known as a *difference equation*. For a cyclic code in systematic form, the components $v_{n-k}, v_{n-k+1}, \cdots, v_{n-1}$ of each codeword are the information digits. Given these $k$ information digits, (5.18) is a rule for determining the $n-k$ parity-check digits, $v_0, v_1, \cdots, v_{n-k-1}$. An encoding circuit based on (5.18) is shown in Figure 5.3. The feedback connections are based on the coefficients of the parity polynomial $\mathfrak{h}(X)$. (Note that $h_0 = h_k = 1$.) The encoding operation can be described in the following steps:

**Step 1.** Initially, gate 1 is turned on and gate 2 is turned off. The $k$ information digits $\mathfrak{u}(X) = u_0 + u_1 X + \cdots + u_{k-1} X^{k-1}$ are shifted into the register and the communication channel simultaneously.

**Step 2.** As soon as the $k$ information digits have entered the shift register, gate 1 is turned off and gate 2 is turned on. The first parity-check digit,

$$\begin{aligned} v_{n-k-1} &= h_0 v_{n-1} + h_1 v_{n-2} + \cdots + h_{k-1} v_{n-k} \\ &= u_{k-1} + h_1 u_{k-2} + \cdots + h_{k-1} u_0, \end{aligned}$$

is formed and appears at point $P$.

Step 3. The register is shifted once. The first parity-check digit is shifted into the channel and is also shifted into the register. Now, the second parity-check digit,

$$\begin{aligned} v_{n-k-2} &= h_0 v_{n-2} + h_1 v_{n-3} + \cdots + h_{k-1} v_{n-k-1} \\ &= u_{k-2} + h_1 u_{k-3} + \cdots + h_{k-2} u_0 + h_{k-1} v_{n-k-1}, \end{aligned}$$

is formed at $P$.

Step 4. Step 3 is repeated until $n-k$ parity-check digits have been formed and shifted into the channel. Then, gate 1 is turned on and gate 2 is turned off. The next message is now ready to be shifted into the register.

The preceding encoding circuit employs a $k$-stage shift register. Comparing the two encoding circuits presented in this section, we can make the following remark: for codes with more parity-check digits than message digits, the $k$-stage encoding circuit is more economical; otherwise, the $(n-k)$-stage encoding circuit is preferable.

---

EXAMPLE 5.6

The parity polynomial of the (7, 4) cyclic code generated by $g(X) = 1 + X + X^3$ is

$$h(X) = \frac{X^7 + 1}{1 + X + X^3} = 1 + X + X^2 + X^4.$$

The encoding circuit based on $h(X)$ is shown in Figure 5.4. Each codeword is of the form $\mathbf{v} = (v_0, v_1, v_2, v_3, v_4, v_5, v_6)$, where $v_3$, $v_4$, $v_5$, and $v_6$ are message digits, and $v_0$, $v_1$, and $v_2$ are parity-check digits. The difference equation that determines the parity-check digits is

$$\begin{aligned} v_{3-j} &= 1 \cdot v_{7-j} + 1 \cdot v_{6-j} + 1 \cdot v_{5-j} + 0 \cdot v_{4-j} \\ &= v_{7-j} + v_{6-j} + v_{5-j} \qquad \text{for } 1 \le j \le 3. \end{aligned}$$

Suppose that the message to be encoded is (1 0 1 1). Then, $v_3 = 1, v_4 = 0, v_5 = 1, v_6 = 1$. The first parity-check digit is

$$v_2 = v_6 + v_5 + v_4 = 1 + 1 + 0 = 0.$$

The second parity-check digit is

$$v_1 = v_5 + v_4 + v_3 = 1 + 0 + 1 = 0.$$

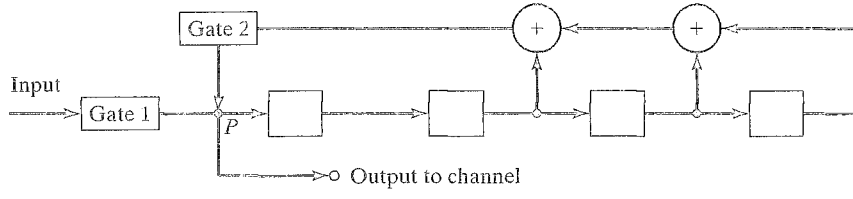


FIGURE 5.4: Encoding circuit for the (7, 4) cyclic code based on its parity polynomial $h(X) = 1 + X + X^2 + X^4$.

The third parity-check digit is

$$v_0 = v_4 + v_3 + v_2 = 0 + 1 + 0 = 1.$$

Thus, the codeword that corresponds to the message (1 0 1 1) is (1 0 0 1 0 1 1).

## 5.4 SYNDROME COMPUTATION AND ERROR DETECTION

Suppose that a codeword is transmitted. Let $\mathbf{r} = (r_0, r_1, \cdots, r_{n-1})$ be the received vector. Because of the channel noise, the received vector may not be the same as the transmitted codeword. In the decoding of a linear code, the first step is to compute the syndrome $\mathbf{s} = \mathbf{r} \cdot \mathbf{H}^T$, where $\mathbf{H}$ is the parity-check matrix. If the syndrome is zero, $\mathbf{r}$ is a codeword, and the decoder accepts $\mathbf{r}$ as the transmitted codeword. If the syndrome is not identical to zero, $\mathbf{r}$ is not a codeword, and the presence of errors has been detected.

We have shown that for a linear systematic code the syndrome is simply the vector sum of the received parity-check digits and the parity-check digits recomputed from the received information digits. For a cyclic code in systematic form, the syndrome can be computed easily. The received vector $\mathbf{r}$ is treated as a polynomial of degree $n - 1$ or less,

$$\mathbf{r}(X) = r_0 + r_1 X + r_2 X^2 + \cdots + r_{n-1} X^{n-1}.$$

Dividing $\mathbf{r}(X)$ by the generator polynomial $\mathbf{g}(X)$, we obtain

$$\mathbf{r}(X) = \mathbf{a}(X)\mathbf{g}(X) + \mathbf{s}(X). \tag{5.19}$$

The remainder $\mathbf{s}(X)$ is a polynomial of degree $n - k - 1$ or less. The $n - k$ coefficients of $\mathbf{s}(X)$ form the syndrome $\mathbf{s}$. It is clear from Theorem 5.4 that $\mathbf{s}(X)$ is identical to zero if and only if the received polynomial $\mathbf{r}(X)$ is a code polynomial. Hereafter, we will simply call $\mathbf{s}(X)$ the syndrome. The syndrome can be computed with a division circuit as shown in Figure 5.5, which is identical to the $(n - k)$-stage encoding circuit, except that the received polynomial $\mathbf{r}(X)$ is shifted into the register from the left end. The received polynomial $\mathbf{r}(X)$ is shifted into the register with all stages initially set to 0. As soon as the entire $\mathbf{r}(X)$ has been shifted into the register, the contents in the register form the syndrome $\mathbf{s}(X)$.

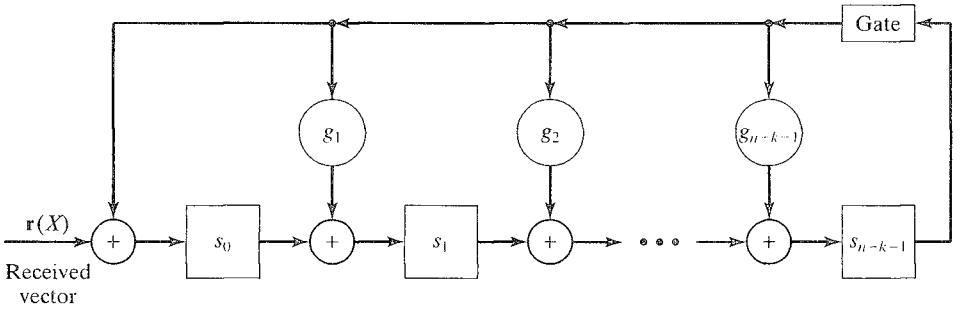


FIGURE 5.5: An $(n - k)$-stage syndrome circuit with input from the left end.

Because of the cyclic structure of the code, the syndrome $\mathbf{s}(X)$ has the following property.

THEOREM 5.8 Let $\mathbf{s}(X)$ be the syndrome of a received polynomial $\mathbf{r}(X) = r_0 + r_1 X + \cdots + r_{n-1}X^{n-1}$. Then, the remainder $\mathbf{s}^{(1)}(X)$ resulting from dividing $X\mathbf{s}(X)$ by the generator polynomial $\mathbf{g}(X)$ is the syndrome of $\mathbf{r}^{(1)}(X)$, which is a cyclic shift of $\mathbf{r}(X)$.

*Proof.* It follows from (5.1) that $\mathbf{r}(X)$ and $\mathbf{r}^{(1)}(X)$ satisfy the following relationship:

$$X\mathbf{r}(X) = r_{n-1}(X^n + 1) + \mathbf{r}^{(1)}(X). \tag{5.20}$$

Rearranging (5.20), we have

$$\mathbf{r}^{(1)}(X) = r_{n-1}(X^n + 1) + X\mathbf{r}(X). \tag{5.21}$$

Dividing both sides of (5.21) by $\mathbf{g}(X)$ and using the fact that $X^n + 1 = \mathbf{g}(X)\mathbf{h}(X)$, we obtain

$$\mathbf{c}(X)\mathbf{g}(X) + \boldsymbol{\rho}(X) = r_{n-1}\mathbf{g}(X)\mathbf{h}(X) + X[\mathbf{a}(X)\mathbf{g}(X) + \mathbf{s}(X)], \tag{5.22}$$

where $\boldsymbol{\rho}(X)$ is the remainder resulting from dividing $\mathbf{r}^{(1)}(X)$ by $\mathbf{g}(X)$. Then, $\boldsymbol{\rho}(X)$ is the syndrome of $\mathbf{r}^{(1)}(X)$.

Rearranging (5.22), we obtain the following relationship between $\boldsymbol{\rho}(X)$ and $X\mathbf{s}(X)$:

$$X\mathbf{s}(X) = [\mathbf{c}(X) + r_{n-1}\mathbf{h}(X) + X\mathbf{a}(X)]\mathbf{g}(X) + \boldsymbol{\rho}(X). \tag{5.23}$$

From (5.23) we see that $\boldsymbol{\rho}(X)$ is also the remainder resulting from dividing $X\mathbf{s}(X)$ by $\mathbf{g}(X)$. Therefore, $\boldsymbol{\rho}(X) = \mathbf{s}^{(1)}(X)$. This completes the proof. Q.E.D.

It follows from Theorem 5.8 that the remainder $\mathbf{s}^{(i)}(X)$ resulting from dividing $X^i\mathbf{s}(X)$ by the generator polynomial $\mathbf{g}(X)$ is the syndrome of $\mathbf{r}^{(i)}(X)$, which is the $i$th cyclic shift of $\mathbf{r}(X)$. This property is useful in decoding cyclic codes. We can obtain the syndrome $\mathbf{s}^{(1)}(X)$ of $\mathbf{r}^{(1)}(X)$ by shifting (or clocking) the syndrome register once with $\mathbf{s}(X)$ as the initial contents and with the input gate disabled. This is because shifting the syndrome register once with $\mathbf{s}(X)$ as the initial contents is equivalent to dividing $X\mathbf{s}(X)$ by $\mathbf{g}(X)$. Thus, after the shift, the register contains $\mathbf{s}^{(1)}(X)$. To obtain the syndrome $\mathbf{s}^{(i)}(X)$ of $\mathbf{r}^{(i)}(X)$, we simply shift the syndrome register $i$ times with $\mathbf{s}(X)$ as the initial contents.

---

EXAMPLE 5.7

A syndrome circuit for the (7, 4) cyclic code generated by $\mathbf{g}(X) = 1 + X + X^3$ is shown in Figure 5.6. Suppose that the received vector is $\mathbf{r} = (0\,0\,1\,0\,1\,1\,0)$. The syndrome of $\mathbf{r}$ is $\mathbf{s} = (1\,0\,1)$. Table 5.3 shows the contents in the register as the received vector is shifted into the circuit. At the end of the seventh shift, the register contains the syndrome $\mathbf{s} = (1\,0\,1)$. If the register is shifted once more with the input gate disabled, the new contents will be $\mathbf{s}^{(1)} = (1\ 0\ 0)$, which is the syndrome of $\mathbf{r}^{(1)}(X) = (0\,0\,0\,1\,0\,1\,1)$, a cyclic shift of $\mathbf{r}$.

---

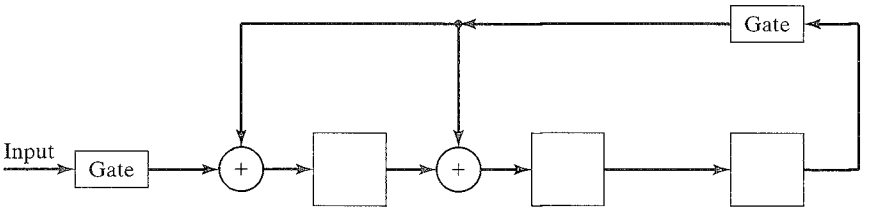


FIGURE 5.6: Syndrome circuit for the (7, 4) cyclic code generated by $\mathbf{g}(X) = 1 + X + X^3$.

TABLE 5.3: Contents of the syndrome register shown in Figure 5.6 with $\mathbf{r} = (0010110)$ as input.

| Shift | Input | Register contents |
|---|---|---|
| | | 000 (initial state) |
| 1 | 0 | 000 |
| 2 | 1 | 100 |
| 3 | 1 | 110 |
| 4 | 0 | 011 |
| 5 | 1 | 011 |
| 6 | 0 | 111 |
| 7 | 0 | 101 (syndrome $\mathbf{s}$) |
| 8 | — | 100 (syndrome $\mathbf{s}^{(1)}$) |
| 9 | — | 010 (syndrome $\mathbf{s}^{(2)}$) |

We may shift the received vector $\mathbf{r}(X)$ into the syndrome register from the right end, as shown in Figure 5.7; however, after the entire $\mathbf{r}(X)$ has been shifted into the register, the contents in the register do not form the syndrome of $\mathbf{r}(X)$; rather, they form the syndrome $\mathbf{s}^{(n-k)}(X)$ of $\mathbf{r}^{(n-k)}(X)$, which is the $(n-k)$th cyclic shift of $\mathbf{r}(X)$. To show this, we notice that shifting $\mathbf{r}(X)$ from the right end is equivalent to premultiplying $\mathbf{r}(X)$ by $X^{n-k}$. When the entire $\mathbf{r}(X)$ has entered the register, the register contains the remainder $\rho(X)$ resulting from dividing $X^{n-k}\mathbf{r}(X)$ by the generator polynomial $\mathbf{g}(X)$. Thus, we have

$$X^{n-k}\mathbf{r}(X) = \mathbf{a}(X)\mathbf{g}(X) + \rho(X). \tag{5.24}$$

It follows from (5.1) that $\mathbf{r}(X)$ and $\mathbf{r}^{(n-k)}(X)$ satisfy the following relation:

$$X^{n-k}\mathbf{r}(X) = \mathbf{b}(X)(X^n + 1) + \mathbf{r}^{(n-k)}(X). \tag{5.25}$$

Combining (5.24) and (5.25) and using the fact that $X^n + 1 = \mathbf{g}(X)\mathbf{h}(X)$, we have

$$\mathbf{r}^{(n-k)}(X) = [\mathbf{b}(X)\mathbf{h}(X) + \mathbf{a}(X)]\mathbf{g}(X) + \rho(X).$$

This equation says that when $\mathbf{r}^{(n-k)}(X)$ is divided by $\mathbf{g}(X)$, $\rho(X)$ is also the remainder. Therefore, $\rho(X)$ is indeed the syndrome of $\mathbf{r}^{(n-k)}(X)$.

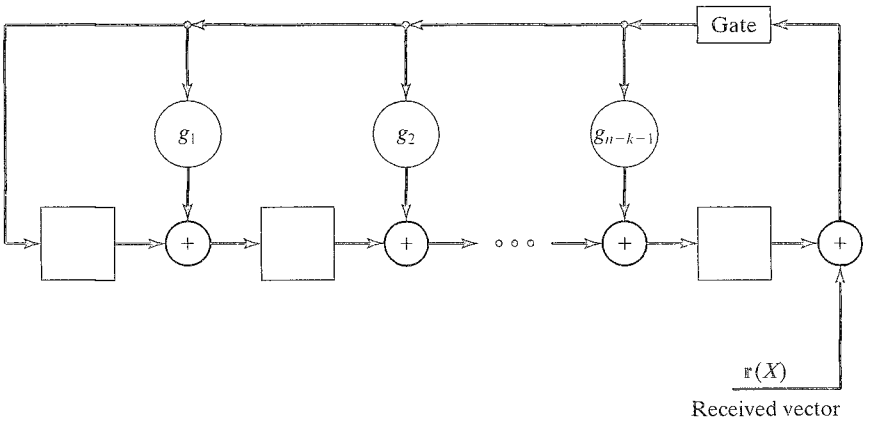


FIGURE 5.7: An $(n-k)$-stage syndrome circuit with input from the right end.

Let $\mathbb{v}(X)$ be the transmitted codeword, and let $\mathbb{e}(X) = e_0 + e_1 X + \cdots + e_{n-1}X^{n-1}$ be the error pattern. Then, the received polynomial is

$$\mathbb{r}(X) = \mathbb{v}(X) + \mathbb{e}(X). \tag{5.26}$$

Because $\mathbb{v}(X)$ is a multiple of the generator polynomial $\mathbb{g}(X)$, combining (5.19) and (5.26), we have the following relationship between the error pattern and the syndrome:

$$\mathbb{e}(X) = [\mathbb{a}(X) + \mathbb{b}(X)]\mathbb{g}(X) + \mathbb{s}(X), \tag{5.27}$$

where $\mathbb{b}(X)\mathbb{g}(X) = \mathbb{v}(X)$. This shows that the syndrome is equal to the remainder resulting from dividing the error pattern by the generator polynomial. The syndrome can be computed from the received vector; however, the error pattern $\mathbb{e}(X)$ is unknown to the decoder. Therefore, the decoder has to estimate $\mathbb{e}(X)$ based on the syndrome $\mathbb{s}(X)$. If $\mathbb{e}(X)$ is a coset leader in the standard array and if table-lookup decoding is used, $\mathbb{e}(X)$ can correctly be determined from the syndrome.

From (5.27) we see that $\mathbb{s}(X)$ is identical to zero if and only if either the error pattern $\mathbb{e}(X) = 0$ or $\mathbb{e}(X)$ is identical to a codeword. If $\mathbb{e}(X)$ is identical to a code polynomial, $\mathbb{e}(X)$ is an undetectable error pattern. Cyclic codes are very effective for detecting errors, random or burst. The error-detection circuit is simply a syndrome circuit with an OR gate whose inputs are the syndrome digits. If the syndrome is not zero, the output of the OR gate is 1, and the presence of errors has been detected.

Now, we investigate the error-detecting capability of an $(n, k)$ cyclic code. Suppose that the error pattern $\mathbb{e}(X)$ is a burst of length $n-k$ or less (i.e., errors are confined to $n-k$ or fewer consecutive positions). Then, we can express $\mathbb{e}(X)$ in the following form:

$$\mathbb{e}(X) = X^j \mathbb{B}(X),$$

where $0 \leq j \leq n-1$, and $\mathbb{B}(X)$ is a polynomial of degree $n-k-1$ or less. Because the degree of $\mathbb{B}(X)$ is less than the degree of the generator polynomial $\mathbb{g}(X)$, $\mathbb{B}(X)$ is not divisible by $\mathbb{g}(X)$. Since $\mathbb{g}(X)$ is a factor of $X^n + 1$, and $X$ is not a factor of $\mathbb{g}(X)$, $\mathbb{g}(X)$ and $X^j$ must be relatively prime. Therefore, $\mathbb{e}(X) = X^j\mathbb{B}(X)$ is not

divisible by $\mathbf{g}(X)$. As a result, the syndrome caused by $\mathbf{e}(X)$ is not equal to zero. This implies that an $(n, k)$ cyclic code is capable of detecting any error burst of length $n - k$ or less. For a cyclic code, an error pattern with errors confined to $i$ high-order positions and $l - i$ low-order positions is also regarded as a burst of length $l$ or less. Such a burst is called an *end-around* burst. For example,

$$\mathbf{e} = (\underset{\longleftarrow}{1}\ \underset{\longleftarrow}{0}\ \underset{\hookleftarrow}{1}\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ \underset{\hookrightarrow}{1}\ \underset{\longrightarrow}{1}\ \underset{\longrightarrow}{0}\ \underset{\longrightarrow}{1})$$

is an end-around burst of length 7. An $(n, k)$ cyclic code is also capable of detecting all the end-around error bursts of length $n - k$ or less (the proof of this is left as a problem). Summarizing the preceding results, we have the following property:

**Theorem 5.9** An $(n, k)$ cyclic code is capable of detecting any error burst of length $n - k$ or less, including the end-around bursts.

In fact, a large percentage of error bursts of length $n - k + 1$ or longer can be detected. Consider the bursts of length $n - k + 1$ starting from the $i$th digit position and ending at the $(i + n - k)$th digit position (i.e., errors are confined to digits $e_i, e_{i+1}, \cdots, e_{i+n-k}$, with $e_i = e_{i+n-k} = 1$). There are $2^{n-k-1}$ such bursts. Among these bursts, the only one that cannot be detected is

$$\mathbf{e}(X) = X^i \mathbf{g}(X).$$

Therefore, the fraction of undetectable bursts of length $n - k + 1$ starting from the $i$th digit position is $2^{-(n-k-1)}$. This fraction applies to bursts of length $n - k + 1$ starting from any digit position (including the end-around case). Therefore, we have the following result:

**Theorem 5.10** The fraction of undetectable bursts of length $n - k + 1$ is $2^{-(n-k-1)}$.

For $l > n - k + 1$, there are $2^{l-2}$ bursts of length $l$ starting from the $i$th digit position and ending at the $(i + l - 1)$th digit position. Among these bursts, the undetectable ones must be of the following form:

$$\mathbf{e}(X) = X^i \mathbf{a}(X)\mathbf{g}(X),$$

where $\mathbf{a}(X) = a_0 + a_1 X + \cdots + a_{l-(n-k)-1} X^{l-(n-k)-1}$, with $a_0 = a_{l-(n-k)-1} = 1$. The number of such bursts is $2^{l-(n-k)-2}$. Therefore, the fraction of undetectable bursts of length $l$ starting from the $i$th digit position is $2^{-(n-k)}$. Again, this fraction applies to bursts of length $l$ starting from any digit position (including the end-around case), which leads to the following conclusion:

**Theorem 5.11** For $l > n - k + 1$, the fraction of undetectable error bursts of length $l$ is $2^{-(n-k)}$.

The preceding analysis shows that cyclic codes are very effective for burst-error detection.

EXAMPLE 5.8

The (7, 4) cyclic code generated by $g(X) = 1 + X + X^3$ has a minimum distance of 3. It is capable of detecting any combination of two or fewer random errors or any burst of length 3 or less. It also detects many bursts of length greater than 3.

## 5.5 DECODING OF CYCLIC CODES

Decoding of cyclic codes consists of the same three steps as for decoding linear codes: syndrome computation, association of the syndrome with an error pattern, and error correction. It was shown in Section 5.4 that syndromes for cyclic codes can be computed with a division circuit whose complexity is linearly proportional to the number of parity-check digits (i.e., $n - k$). The error-correction step is simply adding (modulo-2) the error pattern to the received vector. This addition can be performed with a single EXCLUSIVE-OR gate if correction is carried out serially (i.e., one digit at a time); $n$ EXCLUSIVE-OR gates are required if correction is carried out in parallel, as shown in Figure 3.8. The association of the syndrome with an error pattern can be completely specified by a decoding table. A straightforward approach to the design of a decoding circuit is via a combinational logic circuit that implements the table-lookup procedure; however, the limit to this approach is that the complexity of the decoding circuit tends to grow exponentially with the code length and with the number of errors that are going to be corrected. Cyclic codes have considerable algebraic and geometric properties. If these properties are properly used, decoding circuits can be simplified.

The cyclic structure of a cyclic code allows us to decode a received vector $\mathbb{r}(X) = r_0 + r_1X + r_2X^2 + \cdots + r_{n-1}X^{n-1}$ serially. The received digits are decoded one at a time, and each digit is decoded with the same circuitry. As soon as the syndrome has been computed the decoding circuit checks whether the syndrome $\mathbb{s}(X)$ corresponds to a correctable error pattern $\mathbb{e}(X) = e_0 + e_1X + \cdots + e_{n-1}X^{n-1}$ with an error at the highest-order position $X^{n-1}$ (i.e., $e_{n-1} = 1$). If $\mathbb{s}(X)$ does not correspond to an error pattern with $e_{n-1} = 1$, the received polynomial (stored in a buffer register) and the syndrome register are cyclically shifted once simultaneously. Thus, we obtain $\mathbb{r}^{(1)}(X) = r_{n-1} + r_0X + \cdots + r_{n-2}X^{n-1}$, and the new contents in the syndrome register form the syndrome $\mathbb{s}^{(1)}(X)$ of $\mathbb{r}^{(1)}(X)$. Now, the second digit $r_{n-2}$ of $\mathbb{r}(X)$ becomes the first digit of $\mathbb{r}^{(1)}(X)$. The same decoding circuit will check whether $\mathbb{s}^{(1)}(X)$ corresponds to an error pattern with an error at location $X^{n-1}$.

If the syndrome $\mathbb{s}(X)$ of $\mathbb{r}(X)$ does correspond to an error pattern with an error at location $X^{n-1}$ (i.e., $e_{n-1} = 1$), the first received digit $r_{n-1}$ is an erroneous digit, and it must be corrected. The correction is carried out by taking the sum $r_{n-1} \oplus e_{n-1}$. This correction results in a modified received polynomial, denoted by $\mathbb{r}_1(X) = r_0 + r_1X + \cdots + r_{n-2}X^{n-2} + (r_{n-1} \oplus e_{n-1})X^{n-1}$. The effect of the error digit $e_{n-1}$ on the syndrome is then removed from the syndrome $\mathbb{s}(X)$, by adding the syndrome of $\mathbb{e}'(X) = X^{n-1}$ to $\mathbb{s}(X)$. This sum is the syndrome of the modified received polynomial $\mathbb{r}_1(X)$. Now, $\mathbb{r}_1(X)$ and the syndrome register are cyclically shifted once simultaneously. This shift results in a received polynomial $\mathbb{r}_1^{(1)}(X) = (r_{n-1} \oplus e_{n-1}) + r_0X + \cdots + r_{n-2}X^{n-1}$. The syndrome $\mathbb{s}_1^{(1)}(X)$ of $\mathbb{r}_1^{(1)}(X)$ is the remainder resulting from dividing $X[\mathbb{s}(X) + X^{n-1}]$ by the generator polynomial

$\mathbf{g}(X)$. Because the remainders resulting from dividing $X\mathbf{s}(X)$ and $X^n$ by $\mathbf{g}(X)$ are $\mathbf{s}^{(1)}(X)$ and 1, respectively, we have

$$\mathbf{s}_1^{(1)}(X) = \mathbf{s}^{(1)}(X) + 1.$$

Therefore, if 1 is added to the left end of the syndrome register while it is shifted, we obtain $\mathbf{s}_1^{(1)}(X)$. The decoding circuitry proceeds to decode the received digit $r_{n-2}$. The decoding of $r_{n-2}$ and the other received digits is identical to the decoding of $r_{n-1}$. Whenever an error is detected and corrected, its effect on the syndrome is removed. The decoding stops after a total of $n$ shifts. If $\mathbf{e}(X)$ is a correctable error pattern, the contents of the syndrome register should be zero at the end of the decoding operation, and the received vector $\mathbf{r}(X)$ has been correctly decoded. If the syndrome register does not contain all 0's at the end of the decoding process, an uncorrectable error pattern has been detected.

A general decoder for an $(n, k)$ cyclic code is shown in Figure 5.8. It consists of three major parts: (1) a syndrome register, (2) an error-pattern detector, and (3) a buffer register to hold the received vector. The received polynomial is shifted into the syndrome register from the left end. To remove the effect of an error digit on the syndrome, we simply feed the error digit into the shift register from the left end through an EXCLUSIVE-OR gate. The decoding operation is as follows:

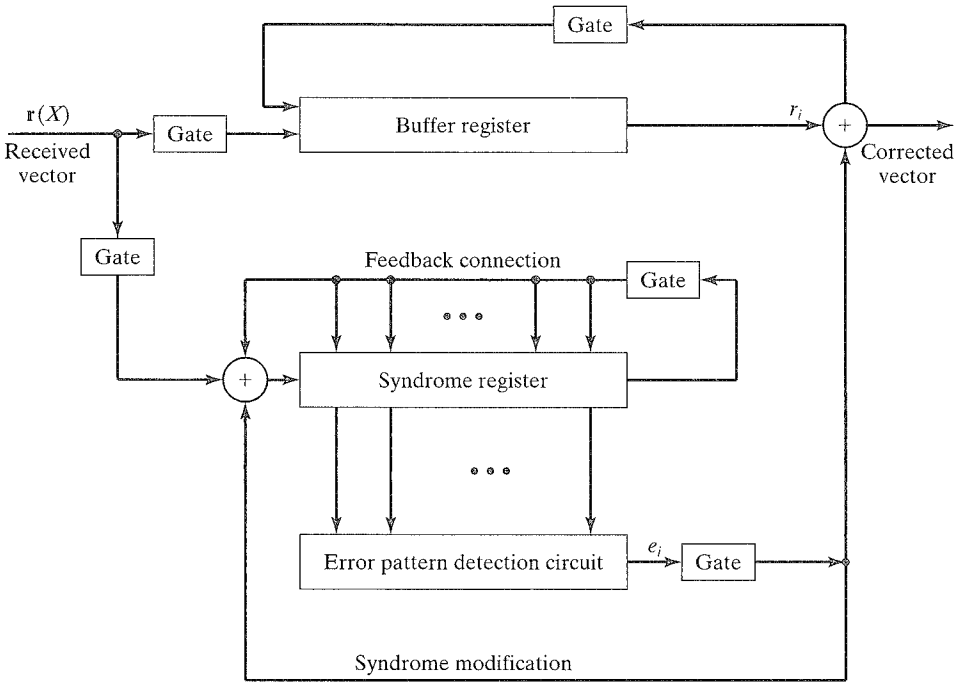


FIGURE 5.8: General cyclic code decoder with received polynomial $\mathbf{r}(X)$ shifted into the syndrome register from the left end.

Step 1. The syndrome is formed by shifting the entire received vector into the syndrome register. The received vector is simultaneously stored in the buffer register.

Step 2. The syndrome is read into the detector and is tested for the corresponding error pattern. The detector is a combinational logic circuit that is designed in such a way that its output is 1 if and only if the syndrome in the syndrome register corresponds to a correctable error pattern with an error at the highest-order position $X^{n-1}$. That is, if a 1 appears at the output of the detector, the received symbol in the rightmost stage of the buffer register is assumed to be erroneous and must be corrected; if a 0 appears at the output of the detector, the received symbol at the rightmost stage of the buffer register is assumed to be error-free, and no correction is necessary. Thus, the output of the detector is the estimated error value for the symbol to come out of the buffer.

Step 3. The first received symbol is read out of the buffer. At the same time, the syndrome register is shifted once. If the first received symbol is detected to be an erroneous symbol, it is then corrected by the output of the detector. The output of the detector is also fed back to the syndrome register to modify the syndrome (i.e., to remove the error effect from the syndrome). This operation results in a new syndrome, which corresponds to the altered received vector shifted one place to the right.

Step 4. The new syndrome formed in step 3 is used to detect whether the second received symbol (now at the rightmost stage of the buffer register) is an erroneous symbol. The decoder repeats steps 2 and 3. The second received symbol is corrected in exactly the same manner as the first received symbol was corrected.

Step 5. The decoder decodes the received vector symbol by symbol in the manner outlined until the entire received vector is read out of the buffer register.

The preceding decoder is known as a Meggitt decoder [11], which applies in principle to any cyclic code. But whether it is practical depends entirely on its error-pattern detection circuit. In some cases the error-pattern detection circuits are simple. Several of these cases are discussed in subsequent chapters.

---

EXAMPLE 5.9

Consider the decoding of the (7, 4) cyclic code generated by $g(X) = 1 + X + X^3$. This code has a minimum distance of 3 and is capable of correcting any single error over a block of seven digits. There are seven single-error patterns. These seven error patterns and the all-zero vector form all the coset leaders of the decoding table. Thus, they form all the correctable error patterns. Suppose that the received polynomial $r(X) = r_0 + r_1X + r_2X^2 + r_3X^3 + r_4X^4 + r_5X^5 + r_6X^6$ is shifted into the syndrome register from the left end. The seven single-error patterns and their corresponding syndrome are listed in Table 5.4.

TABLE 5.4: Error patterns and their syndromes with the received polynomial $\mathbf{r}(X)$ shifted into the syndrome register from the left end.

| Error pattern $\mathbf{e}(X)$ | Syndrome $\mathbf{s}(X)$ | Syndrome vector $(s_0, s_1, s_2)$ |
|---|---|---|
| $\mathbf{e}_6(X) = X^6$ | $\mathbf{s}(X) = 1 + X^2$ | (1 0 1) |
| $\mathbf{e}_5(X) = X^5$ | $\mathbf{s}(X) = 1 + X + X^2$ | (1 1 1) |
| $\mathbf{e}_4(X) = X^4$ | $\mathbf{s}(X) = X + X^2$ | (0 1 1) |
| $\mathbf{e}_3(X) = X^3$ | $\mathbf{s}(X) = 1 + X$ | (1 1 0) |
| $\mathbf{e}_2(X) = X^2$ | $\mathbf{s}(X) = X^2$ | (0 0 1) |
| $\mathbf{e}_1(X) = X^1$ | $\mathbf{s}(X) = X$ | (0 1 0) |
| $\mathbf{e}_0(X) = X^0$ | $\mathbf{s}(X) = 1$ | (1 0 0) |

We see that $\mathbf{e}_6(X) = X^6$ is the only error pattern with an error at location $X^6$. When this error pattern occurs, the syndrome in the syndrome register will be (1 0 1) after the entire received polynomial $\mathbf{r}(X)$ has entered the syndrome register. The detection of this syndrome indicates that $r_6$ is an erroneous digit and must be corrected. Suppose that the single error occurs at location $X^i$ [i.e., $\mathbf{e}_i(X) = X^i$] for $0 \leq i < 6$. After the entire received polynomial has been shifted into the syndrome register, the syndrome in the register will not be (1 0 1); however, after another $6 - i$ shifts, the contents in the syndrome register will be (1 0 1), and the next received digit to come out of the buffer register will be the erroneous digit. Therefore, only the syndrome (1 0 1) needs to be detected, and this can be accomplished with a single three-input AND gate. The complete decoding circuit is shown in Figure 5.9. Figure 5.10 illustrates the decoding process. Suppose that the codeword $\mathbf{v} = (1\,0\,0\,1\,0\,1\,1)$ [or $\mathbf{v}(X) = 1 + X^3 + X^5 + X^6$] is transmitted and $\mathbf{r} = (1\,0\,1\,1\,0\,1\,1)$ [or $\mathbf{r}(X) = 1 + X^2 + X^3 + X^5 + X^6$] is received. A single error occurs at location $X^2$. When the entire received polynomial has been shifted into the syndrome and buffer registers, the syndrome register contains (0 0 1). In Figure 5.10, the contents in the syndrome register and the contents in the buffer register are recorded after each shift. Also, there is a pointer to indicate the error location after each shift. We see that after four more shifts the contents in the syndrome register are (1 0 1), and the erroneous digit $r_2$ is the next digit to come out from the buffer register.

---

The (7, 4) cyclic code considered in Example 5.9 is the same code considered in Example 3.9. Comparing the decoding circuit shown in Figure 3.9 with the decoding circuit shown in Figure 5.9, we see that the circuit shown in Figure 5.9 is simpler than the circuit shown in Figure 3.9. Thus, the cyclic structure does simplify the decoding circuit; however, the circuit shown in Figure 5.9 takes a longer time to decode a received vector because the decoding is carried out serially. In general, there is a trade-off between speed and simplicity, as they cannot be achieved at the same time.

The Meggitt decoder described decodes a received polynomial $\mathbf{r}(X) = r_0 + r_1 X + \cdots + r_{n-1} X^{n-1}$ from the highest-order received digit $r_{n-1}$ to the lowest-order received digit $r_0$. After decoding the received digit $r_i$, both the buffer and syndrome

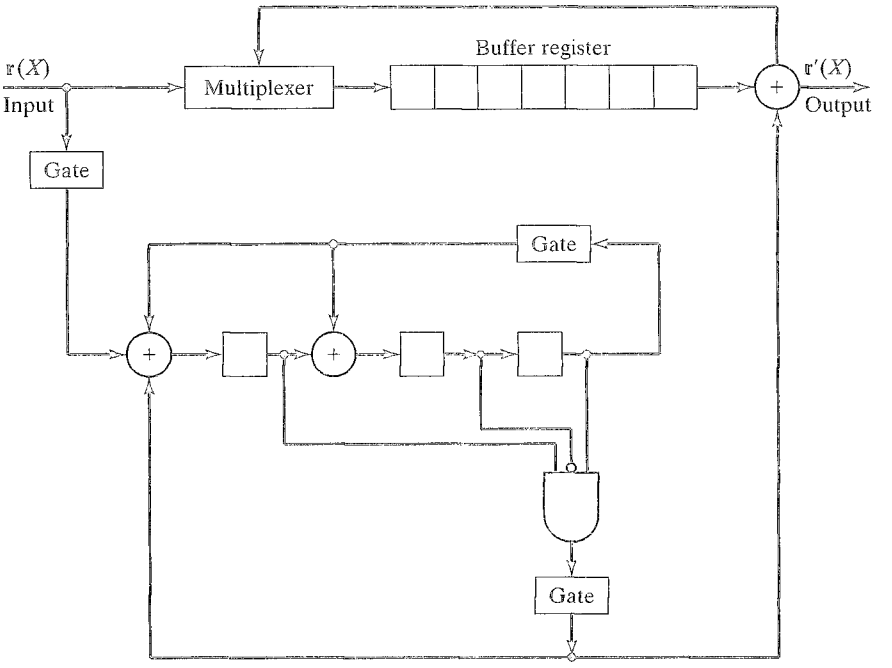


FIGURE 5.9: Decoding circuit for the $(7, 4)$ cyclic code generated by $\mathbb{g}(X) = 1+X+X^3$.

registers are shifted once to the right. The next received digit to be decoded is $r_{i-1}$. It is possible to implement a Meggitt decoder to decode a received polynomial in the reverse order (i.e., to decode a received polynomial from the lowest-order received digit $r_0$ to the highest-order received digit $r_{n-1}$). After decoding the received digit $r_i$, both the buffer and syndrome registers are shifted once to the left. The next received digit to be decoded is $r_{i+1}$. The details of this decoding of a received polynomial in reverse order are left as an exercise.

To decode a cyclic code, the received polynomial $\mathbb{r}(X)$ may be shifted into the syndrome register from the right end for computing the syndrome. When $\mathbb{r}(X)$ has been shifted into the syndrome register, the register contains $\mathbb{s}^{(n-k)}(X)$, which is the syndrome of $\mathbb{r}^{(n-k)}(X)$, the $(n-k)$th cyclic shift of $\mathbb{r}(X)$. If $\mathbb{s}^{(n-k)}(X)$ corresponds to an error pattern $\mathbb{e}(X)$ with $e_{n-1} = 1$, the highest-order digit $r_{n-1}$ of $\mathbb{r}(X)$ is erroneous and must be corrected. In $\mathbb{r}^{(n-k)}(X)$, the digit $r_{n-1}$ is at the location $X^{n-k-1}$. When $r_{n-1}$ is corrected, the error effect must be removed from $\mathbb{s}^{(n-k)}(X)$. The new syndrome, denoted by $\mathbb{s}_1^{(n-k)}(X)$, is the sum of $\mathbb{s}^{(n-k)}(X)$ and the remainder $\rho(X)$ resulting from dividing $X^{n-k-1}$ by the generator polynomial $\mathbb{g}(X)$. Because the degree of $X^{n-k-1}$ is less than the degree of $\mathbb{g}(X)$,

$$\rho(X) = X^{n-k-1}.$$

Therefore,

$$\mathbb{s}_1^{(n-k)}(X) = \mathbb{s}^{(n-k)}(X) + X^{n-k-1},$$

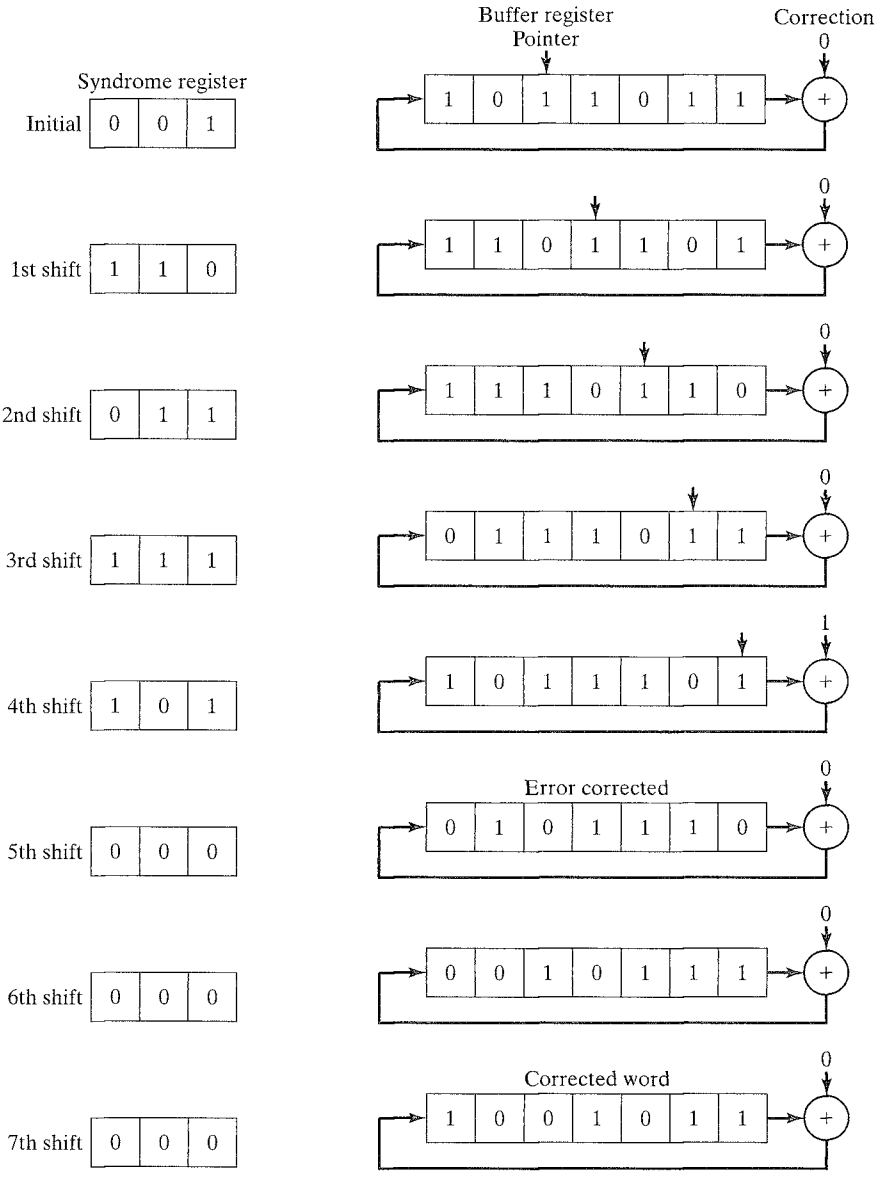


FIGURE 5.10: Error-correction process of the circuit shown in Figure 5.9.

which indicates that the effect of an error at the location $X^{n-1}$ on the syndrome can be removed by feeding the error digit into the syndrome register from the right end through an EXCLUSIVE-OR gate, as shown in Figure 5.11. The decoding process of the decoder shown in Figure 5.11 is identical to the decoding process of the decoder shown in Figure 5.8.

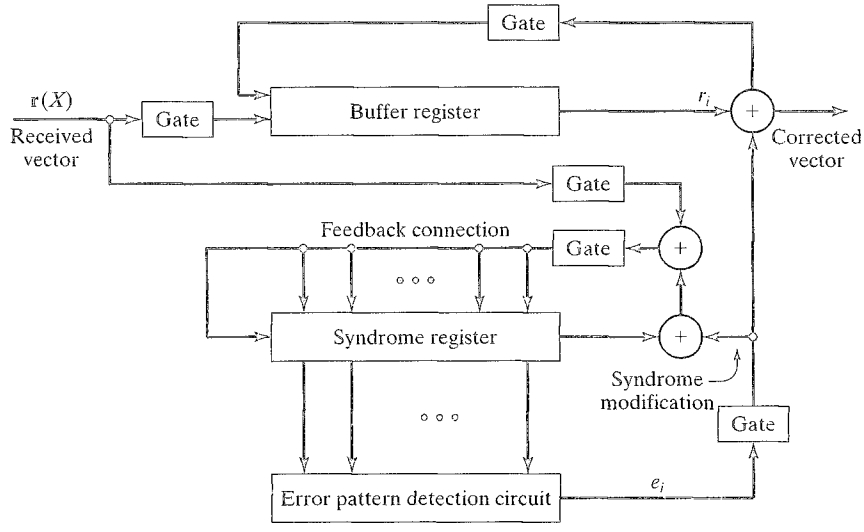


FIGURE 5.11: General cyclic code decoder with received polynomial $\mathrm{r}(X)$ shifted into the syndrome register from the right end.

EXAMPLE 5.10

Again, we consider the decoding of the (7, 4) cyclic code generated by $\mathrm{g}(X) = 1 + X + X^3$. Suppose that the received polynomial $\mathrm{r}(X)$ is shifted into the syndrome register from the right end. The seven single-error patterns and their corresponding syndromes are listed in Table 5.5.

We see that only when $\mathrm{e}(X) = X^6$ occurs, the syndrome is (0 0 1) after the entire polynomial $\mathrm{r}(X)$ has been shifted into the syndrome register. If the single error occurs at the location $X^i$ with $i \neq 6$, the syndrome in the register will not be

TABLE 5.5: Error patterns and their syndromes with the received polynomial $\mathrm{r}(X)$ shifted into the syndrome register from the right end.

| Error pattern $\mathrm{e}(X)$ | Syndrome $\mathrm{s}^{(3)}(X)$ | Syndrome vector $(s_0, s_1, s_2)$ |
|---|---|---|
| $\mathrm{e}(X) = X^6$ | $\mathrm{s}^{(3)}(X) = X^2$ | (0 0 1) |
| $\mathrm{e}(X) = X^5$ | $\mathrm{s}^{(3)}(X) = X$ | (0 1 0) |
| $\mathrm{e}(X) = X^4$ | $\mathrm{s}^{(3)}(X) = 1$ | (1 0 0) |
| $\mathrm{e}(X) = X^3$ | $\mathrm{s}^{(3)}(X) = 1 + X^2$ | (1 0 1) |
| $\mathrm{e}(X) = X^2$ | $\mathrm{s}^{(3)}(X) = 1 + X + X^2$ | (1 1 1) |
| $\mathrm{e}(X) = X$ | $\mathrm{s}^{(3)}(X) = X + X^2$ | (0 1 1) |
| $\mathrm{e}(X) = X^0$ | $\mathrm{s}^{(3)}(X) = 1 + X$ | (1 1 0) |

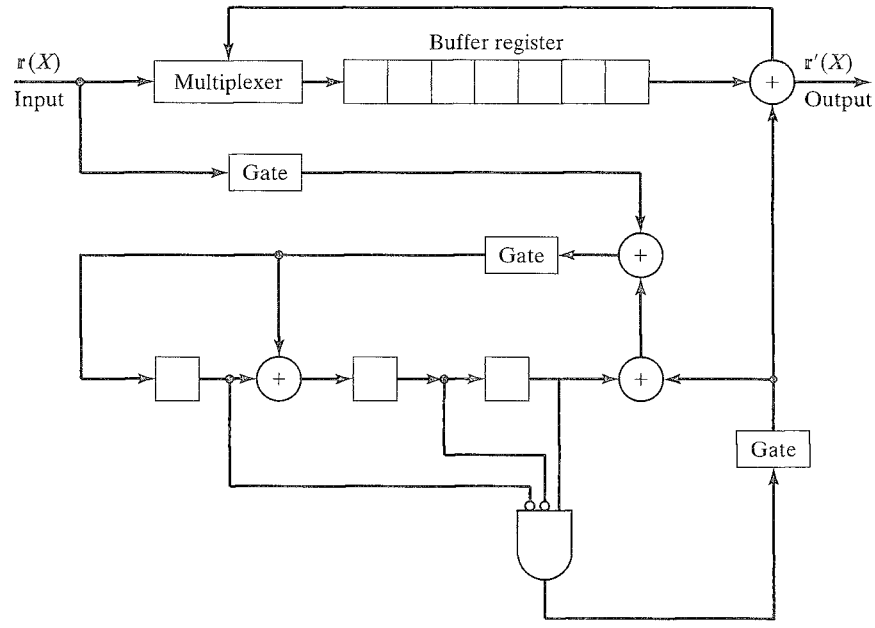


FIGURE 5.12: Decoding circuit for the (7, 4) cyclic code generated by $\mathbb{g}(X) = 1 + X + X^3$.

(0 0 1) after the entire received polynomial $\mathbb{r}(X)$ has been shifted into the syndrome register; however, after another $6 - i$ shifts, the syndrome register will contain (0 0 1). Based on this result, we obtain another decoding circuit for the (7, 4) cyclic code generated by $\mathbb{g}(X) = 1 + X + X^3$, as shown in Figure 5.12. We see that the circuit shown in Figure 5.9 and the circuit shown in Figure 5.12 have the same complexity.

## 5.6 CYCLIC HAMMING CODES

The Hamming codes presented in Section 4.1 can be put in cyclic form. A cyclic Hamming code of length $2^m - 1$ with $m \geq 3$ is generated by a primitive polynomial $\mathbb{p}(X)$ of degree $m$.

In the following we show that the cyclic code defined previously is indeed a Hamming code, by examining its parity-check matrix in systematic form. The method presented in Section 5.2 is used to form the parity-check matrix. Dividing $X^{m+i}$ by the generator polynomial $\mathbb{p}(X)$ for $0 \leq i < 2^m - m - 1$, we obtain

$$X^{m+i} = \mathbb{a}_i(X)\mathbb{p}(X) + \mathbb{b}_i(X), \tag{5.28}$$

where the remainder $\mathbb{b}_i(X)$ is of the form

$$\mathbb{b}_i(X) = b_{i0} + b_{i1}X + \cdots + b_{i,m-1}X^{m-1}.$$

Because $X$ is not a factor of the primitive polynomial $\mathbb{p}(X)$, $X^{m+i}$ and $\mathbb{p}(X)$ must be relatively prime. As a result, $\mathbb{b}_i(X) \neq 0$. Moreover, $\mathbb{b}_i(X)$ consists of at least two

terms. Suppose that $\mathbb{b}_i(X)$ has only one term, say $X^j$ with $0 \leq j < m$. It follows from (5.28) that

$$X^{m+i} = \mathbb{a}_i(X)\mathbb{p}(X) + X^j.$$

Rearranging the preceding equation, we have

$$X^j(X^{m+i-j} + 1) = \mathbb{a}_i(X)\mathbb{p}(X).$$

Because $X^j$ and $\mathbb{p}(X)$ are relatively prime, the foregoing equation implies that $\mathbb{p}(X)$ divides $X^{m+i-j} + 1$; however, this is impossible since $m + i - j < 2^m - 1$, and $\mathbb{p}(X)$ is a primitive polynomial of degree $m$. [Recall that the smallest positive integer $n$ such that $\mathbb{p}(X)$ divides $X^n + 1$ is $2^m - 1$.] Therefore, for $0 \leq i < 2^m - m - 1$, the remainder $\mathbb{b}_i(X)$ contains at least two terms. Next we show that for $i \neq j$, $\mathbb{b}_i(X) \neq \mathbb{b}_j(X)$. From (5.28) we obtain

$$\mathbb{b}_i(X) + X^{m+i} = \mathbb{a}_i(X)\mathbb{p}(X),$$
$$\mathbb{b}_j(X) + X^{m+j} = \mathbb{a}_j(X)\mathbb{p}(X).$$

Suppose that $\mathbb{b}_i(X) = \mathbb{b}_j(X)$. Assuming that $i < j$ and combining the preceding two equations above we obtain the following relation:

$$X^{m+i}(X^{j-i} + 1) = [\mathbb{a}_i(X) + \mathbb{a}_j(X)]\mathbb{p}(X).$$

This equation implies that $\mathbb{p}(X)$ divides $X^{j-i} + 1$, but this is impossible, since $i \neq j$ and $j - i < 2^m - 1$. Therefore, $\mathbb{b}_i(X) \neq \mathbb{b}_j(X)$.

Let $\mathbb{H} = [\mathbb{I}_m \ \mathbb{Q}]$ be the parity-check matrix (in systematic form) of the cyclic code generated by $\mathbb{p}(X)$, where $\mathbb{I}_m$ is an $m \times m$ identity matrix and $\mathbb{Q}$ is an $m \times (2^m - m - 1)$ matrix. Let $\mathbb{b}_i = (b_{i0}, b_{i1}, \cdots, b_{i,m-1})$ be the $m$-tuple corresponding to $\mathbb{b}_i(X)$. It follows from (5.17) that the matrix $\mathbb{Q}$ has the $2^m - m - 1$ $\mathbb{b}_i$'s with $0 \leq i < 2^m - m - 1$ as all its columns. It follows from the preceding analysis that no two columns of $\mathbb{Q}$ are alike, and each column has at least two 1's. Therefore, the matrix $\mathbb{H}$ is indeed a parity-check matrix of a Hamming code, and $\mathbb{p}(X)$ generates this code.

The polynomial $\mathbb{p}(X) = 1 + X + X^3$ is a primitive polynomial. Therefore, the (7, 4) cyclic code generated by $\mathbb{p}(X) = 1 + X + X^3$ is a Hamming code. A list of primitive polynomials with degree $\geq 3$ is given in Table 2.7.

Cyclic Hamming codes can easily be decoded. To devise the decoding circuit, all we need to know is how to decode the first received digit. All the other received digits will be decoded in the same manner and with the same circuitry. Suppose that a single error has occurred at the highest-order position, $X^{2^m-2}$, of the received vector $\mathbb{r}(X)$ [i.e., the error polynomial is $\mathbb{e}(X) = X^{2^m-2}$]. Suppose that $\mathbb{r}(X)$ is shifted into the syndrome register from the right end. After the entire $\mathbb{r}(X)$ has entered the register, the syndrome in the register is equal to the remainder resulting from dividing $X^m \cdot X^{2^m-2}$ (the error polynomial preshifted $m$ times) by the generator polynomial $\mathbb{p}(X)$. Because $\mathbb{p}(X)$ divides $X^{2^m-1} + 1$, the syndrome is of the following form:

$$\mathbb{s}(X) = X^{m-1}.$$

Therefore, if a single error occurs at the highest-order location of $\mathbb{r}(X)$, the resultant syndrome is $(0, 0, \cdots, 0, 1)$. If a single error occurs at any other location of

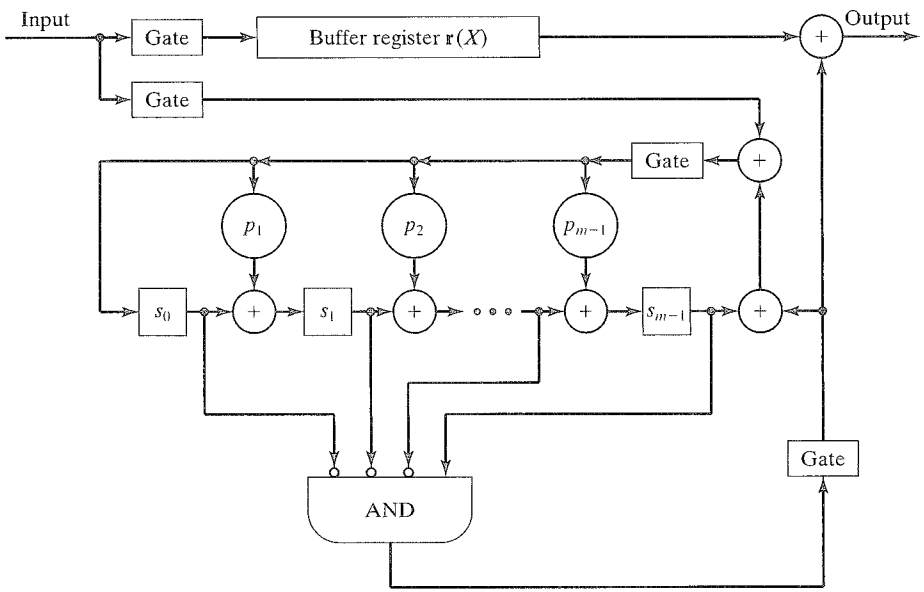


FIGURE 5.13: Decoder for a cyclic Hamming code.

$\mathbf{r}(X)$, the resultant syndrome will be different from $(0, 0, \cdots, 0, 1)$. Based on this result only a single $m$-input AND gate is needed to detect the syndrome pattern $(0, 0, \cdots, 0, 1)$. The inputs to this AND gate are $s_0'$, $s_1'$, $\cdots$, $s_{m-2}'$ and $s_{m-1}$, where $s_i$ is a syndrome digit and $s_i'$ denotes its complement. A complete decoding circuit for a cyclic Hamming code is shown in Figure 5.13. The decoding operation is described in the following steps:

**Step 1.** The syndrome is obtained by shifting the entire received vector into the syndrome register. At the same time, the received vector is stored in the buffer register. If the syndrome is zero, the decoder assumes that no error has occurred, and no correction is necessary. If the syndrome is not zero, the decoder assumes that a single error has occurred.

**Step 2.** The received word is read out of the buffer register digit by digit. As each digit is read out of the buffer register the syndrome register is shifted cyclically once. As soon as the syndrome in the register is $(0, 0, 0, \cdots, 0, 1)$ the next digit to come out of the buffer is the erroneous digit, and the output of the $m$-input AND gate is 1.

**Step 3.** The erroneous digit is read out of the buffer register and is corrected by the output of the $m$-input AND gate. The correction is accomplished by an EXCLUSIVE-OR gate.

**Step 4.** The syndrome register is reset to zero after the entire received vector is read out of the buffer.

The cyclic Hamming code presented here can be modified to correct any single error and simultaneously to detect any combination of double errors. Let

$\mathbb{g}(X) = (X + 1)\mathbb{p}(X)$, where $\mathbb{p}(X)$ is a primitive polynomial of degree $m$. Because both $X + 1$ and $\mathbb{p}(X)$ divide $X^{2^m-1} + 1$ and since they are relatively prime, $\mathbb{g}(X)$ must also divide $X^{2^m-1} + 1$. A single-error-correcting and double-error-detecting cyclic Hamming code of length $2^m - 1$ is generated by $\mathbb{g}(X) = (X + 1)\mathbb{p}(X)$. The code has $m + 1$ parity-check digits. We show next that the minimum distance of this code is 4.

For convenience, we denote the single-error-correcting cyclic Hamming code by $C_1$ and denote the cyclic code generated by $\mathbb{g}(X) = (X + 1)\mathbb{p}(X)$ by $C_2$. In fact, $C_2$ consists of the even-weight codewords of $C_1$ as all its codewords, because any odd-weight code polynomial in $C_1$ does not have $X + 1$ as a factor. Therefore, an odd-weight code polynomial of $C_1$ is not divisible by $\mathbb{g}(X) = (X + 1)\mathbb{p}(X)$, and it is not a code polynomial of $C_2$; however, an even-weight code polynomial of $C_1$ has $X + 1$ as a factor. Therefore, it is divisible by $\mathbb{g}(X) = (X + 1)\mathbb{p}(X)$, and it is also a code polynomial in $C_2$. As a result, the minimum weight of $C_2$ is at least 4.

Next, we show that the minimum weight of $C_2$ is exactly 4. Let $i$, $j$, and $k$ be three distinct nonnegative integers less than $2^m - 1$ such that $X^i + X^j + X^k$ is not divisible by $\mathbb{p}(X)$. Such integers do exist. For example, we first choose $i$ and $j$. Dividing $X^i + X^j$ by $\mathbb{p}(X)$, we obtain

$$X^i + X^j = \mathbb{a}(X)\mathbb{p}(X) + \mathbb{b}(X),$$

where $\mathbb{b}(X)$ is the remainder with degree $m - 1$ or less. Because $X^i + X^j$ is not divisible by $\mathbb{p}(X)$, $\mathbb{b}(X) \neq 0$. Now, we choose an integer $k$ such that when $X^k$ is divided by $\mathbb{p}(X)$, the remainder is not equal to $\mathbb{b}(X)$. Therefore, $X^i + X^j + X^k$ is not divisible by $\mathbb{p}(X)$. Dividing this polynomial by $\mathbb{p}(X)$, we have

$$X^i + X^j + X^k = \mathbb{c}(X)\mathbb{p}(X) + \mathbb{d}(X). \tag{5.29}$$

Next, we choose a nonnegative integer $l$ less than $2^m - 1$ such that when $X^l$ is divided by $\mathbb{p}(X)$, the remainder is $\mathbb{d}(X)$; that is,

$$X^l = \mathbb{f}(X)\mathbb{p}(X) + \mathbb{d}(X). \tag{5.30}$$

The integer $l$ cannot be equal to any of the three integers $i$, $j$, or $k$. Suppose that $l = i$. From (5.29) and (5.30) we would obtain

$$X^j + X^k = [\mathbb{c}(X) + \mathbb{f}(X)]\mathbb{p}(X).$$

This result implies that $\mathbb{p}(X)$ divides $X^{k-j} + 1$ (assuming that $j < k$), which is impossible, since $k - j < 2^m - 1$, and $\mathbb{p}(X)$ is a primitive polynomial. Therefore, $l \neq i$. Similarly, we can show that $l \neq j$ and $l \neq k$. Using this fact and combining (5.29) and (5.30), we obtain

$$X^i + X^j + X^k + X^l = [\mathbb{c}(X) + \mathbb{f}(X)]\mathbb{p}(X).$$

Because $X+1$ is a factor of $X^i+X^j+X^k+X^l$ and it is not a factor of $\mathbb{p}(X)$, $\mathbb{c}(X)+\mathbb{f}(X)$ must be divisible by $X + 1$. As a result, $X^i + X^j + X^k + X^l$ is divisible by $\mathbb{g}(X) = (X + 1)\mathbb{p}(X)$. Therefore, it is a code polynomial in the code generated by $\mathbb{g}(X)$. It has weight 4. This proves that the cyclic code $C_2$ generated by $\mathbb{g}(X) = (X + 1)\mathbb{p}(X)$ has a minimum weight (or distance) of 4. Hence, it is capable of correcting any single error and simultaneously detecting any combination of double errors.

Because the distance-4 Hamming code $C_2$ of length $2^m - 1$ consists of the even-weight codewords of the distance-3 Hamming code $C_1$ of length $2^m - 1$ as its codewords, the weight enumerator $A_2(z)$ for $C_2$ can be determined from the weight enumerator $A_1(z)$ for $C_1$. $A_2(z)$ consists of only the even power terms of $A_1(z)$. Therefore,

$$A_2(z) = \tfrac{1}{2}[A_1(z) + A_1(-z)] \tag{5.31}$$

(see Problem 5.8). Since $A_1(Z)$ is known and is given by (4.1), $A_2(z)$ can be determined from (4.1) and (5.31):

$$A_2(z) = \frac{1}{2(n+1)}\{(1+z)^n + (1-z)^n + 2n(1-z^2)^{(n-1)/2}\}, \tag{5.32}$$

where $n = 2^m - 1$. The dual of a distance-4 cyclic Hamming code is a $(2^m - 1, m + 1)$ cyclic code that has the following weight distribution:

$$B_0 = 1, \quad B_{2^{m-1}-1} = 2^m - 1, \quad B_{2^{m-1}} = 2^m - 1, \quad B_{2^m-1} = 1.$$

Therefore, the weight enumerator for the dual of a distance-4 cyclic Hamming code is

$$B_2(z) = 1 + (2^m - 1)z^{2^{m-1}-1} + (2^m - 1)z^{2^{m-1}} + z^{2^m-1}. \tag{5.33}$$

If a distance-4 cyclic Hamming code is used for error detection on a BSC, its probability of an undetected error, $P_u(E)$, can be computed from (3.33) and (5.32) or from (3.36) and (5.33). Computing $P_u(E)$ from (3.36) and (5.33), we obtain the following expression:

$$P_u(E) = 2^{-(m+1)}\{1 + 2(2^m - 1)(1-p)(1-2p)^{2^{m-1}-1} + (1-2p)^{2^{m-1}-1}\} - (1-p)^{2^m-1}. \tag{5.34}$$

Again, from (5.34), we can show that the probability of an undetected error for the distance-4 cyclic Hamming codes satisfies the upper bound $2^{-(n-k)} = 2^{-(m+1)}$ (see Problem 5.21).

Distance-3 and distance-4 cyclic Hamming codes are often used in communication systems for error detection.

## 5.7 ERROR-TRAPPING DECODING

In principle, the general decoding method of Meggitt's applies to any cyclic code, but refinements are necessary for practical implementation. If we put some restrictions on the error patterns that we intend to correct, the Meggitt decoder can be practically implemented. Consider an $(n, k)$ cyclic code with generator polynomial $\mathbf{g}(X)$. Suppose that a code polynomial $\mathbf{v}(X)$ is transmitted and is corrupted by an error pattern $\mathbf{e}(X)$. Then, the received polynomial is $\mathbf{r}(X) = \mathbf{v}(X) + \mathbf{e}(X)$. We showed in Section 5.4 that the syndrome $\mathbf{s}(X)$ computed from $\mathbf{r}(X)$ is equal to the remainder resulting from dividing the error pattern $\mathbf{e}(X)$ by the generator $\mathbf{g}(X)$ [i.e., $\mathbf{e}(X) = \mathbf{a}(X)\mathbf{g}(X) + \mathbf{s}(X)$]. Suppose that errors are confined to the $n - k$ high-order positions, $X^k$, $X^{k+1}$, $\cdots$, $X^{n-1}$ of $\mathbf{r}(X)$ [i.e., $\mathbf{e}(X) = e_k X^k + e_{k+1}X^{k+1} + \cdots + e_{n-1}X^{n-1}$]. If $\mathbf{r}(X)$ is cyclically shifted $n - k$ times, the errors will be confined to $n - k$ low-order parity positions, $X^0$, $X^1$, $\cdots$, $X^{n-k-1}$ of $\mathbf{r}^{(n-k)}(X)$. The corresponding error pattern is then

$$\mathbf{e}^{(n-k)}(X) = e_k + e_{k+1}X + \cdots + e_{n-1}X^{n-k-1}.$$

Because the syndrome $\mathbb{s}^{(n-k)}(X)$ of $\mathbb{r}^{(n-k)}(X)$ is equal to the remainder resulting from dividing $\mathbb{e}^{(n-k)}(X)$ by $\mathbb{g}(X)$ and since the degree of $\mathbb{e}^{(n-k)}(X)$ is less than $n-k$, we obtain the following equality:

$$\mathbb{s}^{(n-k)}(X) = \mathbb{e}^{(n-k)}(X) = e_k + e_{k+1}X + \cdots + e_{n-1}X^{n-k-1}.$$

Multiplying $\mathbb{s}^{(n-k)}(X)$ by $X^k$, we have

$$\begin{aligned} X^k\mathbb{s}^{(n-k)}(X) &= \mathbb{e}(X) \\ &= e_kX^k + e_{k+1}X^{k+1} + \cdots + e_{n-1}X^{n-1}. \end{aligned}$$

This result says that if errors are confined to the $n-k$ high-order positions of the received polynomial $\mathbb{r}(X)$, the error pattern $\mathbb{e}(X)$ is identical to $X^k\mathbb{s}^{(n-k)}(X)$, where $\mathbb{s}^{(n-k)}(X)$ is the syndrome of $\mathbb{r}^{(n-k)}(X)$, the $(n-k)$th cyclic shift of $\mathbb{r}(X)$. When this event occurs, we simply compute $\mathbb{s}^{(n-k)}(X)$ and add $X^k\mathbb{s}^{(n-k)}(X)$ to $\mathbb{r}(X)$. The resultant polynomial is the transmitted code polynomial.

Suppose that errors are not confined to the $n-k$ high-order positions but are confined to $n-k$ consecutive positions, say $X^i, X^{i+1}, \cdots, X^{(n-k)+i-1}$ of $\mathbb{r}(X)$ (including the end-around case). If $\mathbb{r}(X)$ is cyclically shifted $n-i$ times to the right, errors will be confined to the $n-k$ low-order position of $\mathbb{r}^{(n-i)}(X)$, and the error pattern will be identical to $X^i\mathbb{s}^{(n-i)}(X)$, where $\mathbb{s}^{(n-i)}(X)$ is the syndrome of $\mathbb{r}^{(n-i)}(X)$.

Now, suppose that we shift the received polynomial $\mathbb{r}(X)$ into the syndrome register from the right end. Shifting $\mathbb{r}(X)$ into the syndrome register from the right end is equivalent to multiplying $\mathbb{r}(X)$ by $X^{n-k}$. After the entire $\mathbb{r}(X)$ has been shifted into the syndrome register, the contents of the syndrome register form the syndrome $\mathbb{s}^{(n-k)}(X)$ of $\mathbb{r}^{(n-k)}(X)$. If the errors are confined to $n-k$ high-order positions, $X^k, X^{k+1}, \cdots, X^{n-1}$ of $\mathbb{r}(X)$, they are identical to $\mathbb{s}^{(n-k)}(X)$; however, if the errors are confined to $n-k$ consecutive positions (including end-around) other than the $n-k$ high-order positions of $\mathbb{r}(X)$, after the entire $\mathbb{r}(X)$ has been shifted into the syndrome register, the syndrome register must be shifted a certain number of times before its contents are identical to the error digits. This shifting of the syndrome register until its contents are identical to the error digits is called *error trapping* [14]. If errors are confined to $n-k$ consecutive positions of $\mathbb{r}(X)$ and if we can detect when the errors are trapped in the syndrome register, errors can be corrected simply by adding the contents of the syndrome register to the received digits at the $n-k$ proper positions.

Suppose that a $t$-error-correcting cyclic code is used. To detect the event that the errors are trapped in the syndrome register, we may simply test the *weight* of the syndrome after each shift of the syndrome register. As soon as the weight of the syndrome becomes $t$ or less, we assume that errors are trapped in the syndrome register. If the number of errors in $\mathbb{r}(X)$ is $t$ or less and if they are confined to $n-k$ consecutive positions, the errors are trapped in the syndrome register only when the weight of the syndrome in the register becomes $t$ or less. This result can be shown as follows. An error pattern $\mathbb{e}(X)$ with $t$ or fewer errors that are confined to $n-k$ consecutive positions must be of the form $\mathbb{e}(X) = X^j\mathbb{B}(X)$, where $\mathbb{B}(X)$ has $t$ or fewer terms and has degree $n-k-1$ or less. (For the end-around case, the same form would be obtained after a certain number of cyclic shifts of $\mathbb{e}(X)$.) Dividing $\mathbb{e}(X)$ by the generator polynomial $\mathbb{g}(X)$, we have

$$X^j\mathbb{B}(X) = \mathbb{a}(X)\mathbb{g}(X) + \mathbb{s}(X),$$

where $\mathbf{s}(X)$ is the syndrome of $X^j\mathbb{B}(X)$. Because $\mathbf{s}(X) + X^j\mathbb{B}(X)$ is a multiple of $\mathbf{g}(X)$, it is a code polynomial. The syndrome $\mathbf{s}(X)$ cannot have weight $t$ or less unless $\mathbf{s}(X) = X^j\mathbb{B}(X)$. Suppose that the weight of $\mathbf{s}(X)$ is $t$ or less, and $\mathbf{s}(X) \neq X^j\mathbb{B}(X)$. Then, $\mathbf{s}(X) + X^j\mathbb{B}(X)$ is a nonzero code polynomial with weight less than $2t + 1$. This is impossible, since a $t$-error-correcting code must have a minimum weight of at least $2t + 1$. Therefore, we conclude that the errors are trapped in the syndrome register only when the weight of the syndrome becomes $t$ or less.

Based on the error-trapping concept and the test just described, an error-trapping decoder can be implemented as shown in Figure 5.14. The decoding operation can be described in the following steps:

**Step 1.** The received polynomial $\mathbf{r}(X)$ is shifted into the buffer and syndrome registers simultaneously with gates 1 and 3 turned on and all the other gates turned off. Because we are interested only in the recovery of the $k$ information digits, the buffer register has to store only the $k$ received information digits.

**Step 2.** As soon as the entire $\mathbf{r}(X)$ has been shifted into the syndrome register, the weight of the syndrome in the register is tested by an $(n - k)$-input *threshold* gate whose output is 1 when $t$ or fewer of its inputs are 1; otherwise, it is zero.

**a.** If the weight of the syndrome is $t$ or less, the syndrome digits in the syndrome register are identical to the error digits at the $n - k$ high-order positions $X^k, X^{k+1}, \cdots, X^{n-1}$ of $\mathbf{r}(X)$. Now, gates 2 and 4 are turned on and the other gates are turned off. The received vector is read out of the buffer register one digit at a time and is corrected by the error digits shifted out from the syndrome register.

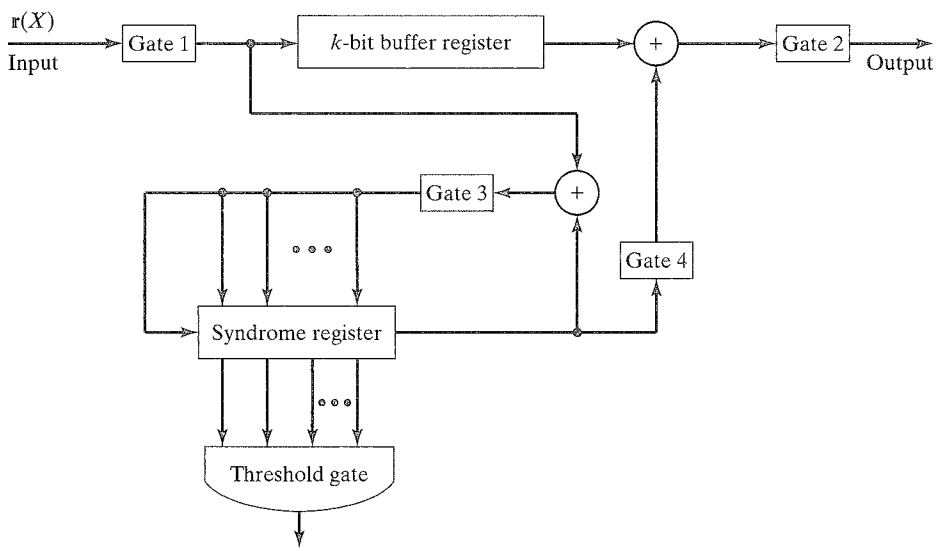


FIGURE 5.14: Error-trapping decoder.

b. If the weight of the syndrome is greater than $t$, the errors are *not* confined to the $n-k$ high-order positions of $\mathbf{r}(X)$, and they have not been trapped in the syndrome register. Go to step 3.

Step 3. The syndrome register is cyclically shifted once with gate 3 turned on and other gates turned off. The weight of the new syndrome is tested. (a) If it is $t$ or less, the errors are confined to the locations $X^{k-1}, X^k, \cdots, X^{n-2}$ of $\mathbf{r}(X)$, and the contents in the syndrome register are identical to the errors at these locations. Because the first received digit $r_{n-1}$ is *error-free*, it is read out of the buffer register with gate 2 turned on. As soon as $r_{n-1}$ has been read out, gate 4 is turned on and gate 3 is turned off. The contents in the syndrome register are shifted out and are used to correct the next $n-k$ received digits to come out from the buffer register. (b) If the weight of the syndrome is greater than $t$, the syndrome register is shifted once more with gate 3 turned on.

Step 4. The syndrome register is continuously shifted until the weight of its contents drops to $t$ or less. If the weight goes down to $t$ or less at the end of the $i$th shift, for $1 \leq i \leq k$, the first $i$ received digits, $r_{n-i}, r_{n-i+1}, \cdots, r_{n-1}$, in the buffer are *error-free*, and the contents in the syndrome register are identical to the errors at the locations $X^{k-i}, X^{k-i+1}, \cdots, X^{n-i-1}$. As soon as the $i$ error-free received digits have been read out of the buffer register, the contents in the syndrome register are shifted out and are used to correct the next $n-k$ received digit to come out from the buffer register. When the $k$ received information digits have been read out of the buffer register and have been corrected, gate 2 is turned off. Any nonzero digits left in the syndrome register are errors in the parity part of $\mathbf{r}(X)$, and they will be ignored.

Step 5. If the weight of the syndrome never goes down to $t$ or less by the time the syndrome register has been shifted $k$ times, either an error pattern with errors confined to $n-k$ consecutive end-around locations has occurred or an uncorrectable error pattern has occurred. The syndrome register keeps being shifted. Suppose that the weight of its contents becomes $t$ or less at the end of $k+l$ shifts with $1 \leq l \leq n-k$. Then, errors are confined to the $n-k$ consecutive end-around locations, $X^{n-l}, X^{n-l+1}, \cdots, X^{n-1}, X^0, X^1, \cdots, X^{n-k-l-1}$ of $\mathbf{r}(X)$. The $l$ digits in the $l$ leftmost stages of the syndrome register match the errors at the $l$ high-order locations $X^{n-l}, X^{n-l+1}, \cdots, X^{n-1}$ of $\mathbf{r}(X)$. Because the errors at the $n-k-l$ parity locations are not needed, the syndrome register is shifted $n-k-l$ times with all the gates turned off. Now, the $l$ errors at the locations $X^{n-l}, X^{n-l+1}, \cdots, X^{n-1}$ of $\mathbf{r}(X)$ are contained in the $l$ rightmost stages of the syndrome register. With gates 2 and 4 turned on and other gates turned off, the received digits in the buffer register are read out and are corrected by the corresponding error digits shifted out from the syndrome register. This completes the decoding operation.

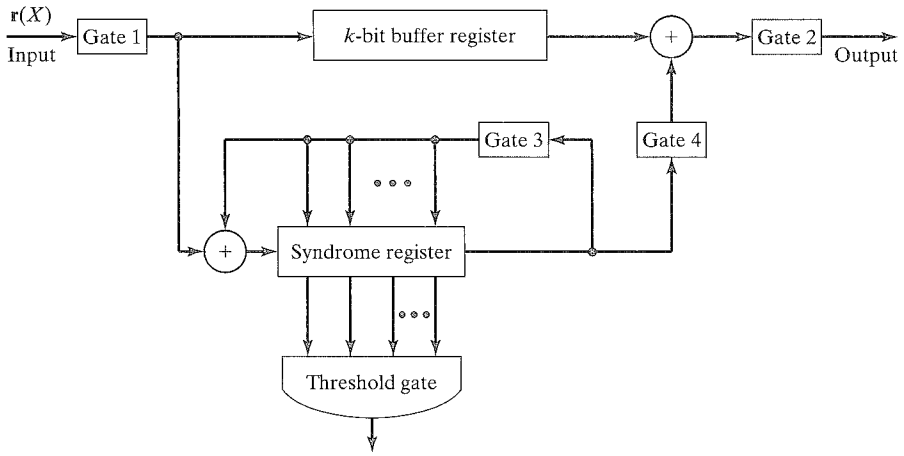


FIGURE 5.15: Another error-trapping decoder.

If the weight of the syndrome never drops to $t$ or less by the time the syndrome register has been shifted a total of $n$ times, either an uncorrectable error pattern has occurred or errors are not confined to $n-k$ consecutive positions. In either case, errors are detected. Except when errors are confined to the $n-k$ consecutive end-around positions of $\mathbf{r}(X)$, the received information digits can be read out of the buffer register, corrected, and delivered to the data sink after at most $k$ cyclic shifts of the syndrome register. When an error pattern with errors confined to $n-k$ consecutive end-around locations of $\mathbf{r}(X)$ occurs, a total of $n$ cyclic shifts of the syndrome register is required before the received message can be read out of the buffer register for corrections. For large $n$ and $n-k$, the number of correctable end-around error patterns becomes big, which causes an undesirably long decoding delay.

It is possible to implement the error-trapping decoding in a different manner so that the error patterns with errors confined to $n-k$ consecutive end-around locations can be corrected as fast as possible. This can be done by shifting the received vector $\mathbf{r}(X)$ into the syndrome register from the *left* end, as shown in Figure 5.15. This variation is based on the following facts. If the errors are confined to $n-k$ low-order parity positions $X^0, X^1, \cdots, X^{n-k-1}$ of $\mathbf{r}(X)$, then after the entire $\mathbf{r}(X)$ has entered the syndrome register, the contents in the register are identical to the error digits at the locations $X^0, X^1, \cdots, X^{n-k-1}$ of $\mathbf{r}(X)$. Suppose that the errors are not confined to the $n-k$ low-order positions of $\mathbf{r}(X)$ but are confined to $n-k$ consecutive locations (including the end-around case), say $X^i, X^{i+1}, \cdots, X^{(n-k)+i-1}$. After $n-i$ cyclic shifts of $\mathbf{r}(X)$, the errors will be shifted to the $n-k$ low-order positions of $\mathbf{r}^{(n-i)}(X)$, and the syndrome of $\mathbf{r}^{(n-i)}(X)$ will be identical to the errors confined to positions $X^i, X^{i+1}, \cdots, X^{(n-k)+i-1}$ of $\mathbf{r}(X)$. The operation of the decoder shown in Figure 5.15 is as follows:

**Step 1.** Gates 1 and 3 are turned on and the other gates are turned off. The received vector $\mathbf{r}(X)$ is shifted into the syndrome register and simultaneously into the buffer register (only the $k$ received information digits

are stored in the buffer register). As soon as the entire $\mathbb{r}(X)$ has been shifted into the syndrome register, the contents of the register form the syndrome $\mathbb{s}(X)$ of $\mathbb{r}(X)$.

Step 2. The weight of the syndrome is tested. (a) If the weight is $t$ or less, the errors are confined to the $(n-k)$ low-order parity positions $X^0, X^1, \cdots, X^{n-k-1}$ of $\mathbb{r}(X)$. Thus, the $k$ received information digits in the buffer register are *error-free*. Gate 2 is then turned on, and the error-free information digits are read out of the buffer with gate 4 turned off. (b) If the weight of the syndrome is greater than $t$, the syndrome register is then shifted once with gate 3 turned on and the other gates turned off. Go to step 3.

Step 3. The weight of the new contents in the syndrome register is tested. (a) If the weight is $t$ or less, the errors are confined to the position $X^{n-1}, X^0, X^1, \cdots, X^{n-k-2}$ of $\mathbb{r}(X)$ (end-around case). The leftmost digit in the syndrome register is identical to the error at the position $X^{n-1}$ of $\mathbb{r}(X)$; the other $n-k-1$ digits in the syndrome register match the errors at parity positions $X^0, X^1, \cdots, X^{n-k-2}$ of $\mathbb{r}(X)$. The output of the threshold gate turns gate 3 off and sets a clock to count from 2. The syndrome register is then shifted (in step with the clock) with gate 3 turned off. As soon as the clock has counted to $n-k$, the contents of the syndrome register will be $(0\,0\cdots 0\,1)$. The rightmost digit matches the error at position $X^{n-1}$ of $\mathbb{r}(X)$. The $k$ received information digits are then read out of the buffer, and the first received information digit is corrected by the 1 coming out from the syndrome register. The decoding is thus completed. (b) If the weight of the contents in the syndrome register is greater than $t$, the syndrome register is shifted once again with gate 3 turned on and other gates turned off. Go to step 4.

Step 4. Step 3(b) repeats until the weight of the contents of the syndrome register drop to $t$ or less. If the weight drops to $t$ or less after the $i$th shift, for $1 \le i \le n-k$, the clock starts to count from $i+1$. At the same time the syndrome register is shifted with gate 3 turned off. As soon as the clock has counted to $n-k$, the rightmost $i$ digits in the syndrome register match the errors in the first $i$ received information digits in the buffer register. The other information digits are error-free. Gates 2 and 4 are then turned on. The received information digits are read out of the buffer for correction.

Step 5. If the weight of the contents of the syndrome register never drops to $t$ or less by the time that the syndrome register has been shifted $n-k$ times (with gate 3 turned on), gate 2 is then turned on, and the received information digits are read out of the buffer one at a time. At the same time the syndrome register is shifted with gate 3 turned on. As soon as the weight of the contents of the syndrome register drops to $t$ or less, the contents match the errors in the next $n-k$ digits to come out of the buffer. Gate 4 is then turned on, and the erroneous information digits are corrected by the digits coming out from the syndrome register with gate 3 turned off. Gate 2 is turned off as soon as $k$ information digits have been read out of the buffer.

With the implementation of the error-trapping decoding just described, the received information digits can be read out of the buffer register after at most $n - k$ shifts of the syndrome register. For large $n - k$, this implementation provides faster decoding than the decoder shown in Figure 5.14; however, when $n - k$ is much smaller than $k$, the first implementation of error trapping is more advantageous in decoding speed than the one shown in Figure 5.15.

The decoding of cyclic Hamming codes presented in Section 5.6 is actually an error-trapping decoding. The syndrome register is cyclically shifted until the single error is trapped in the rightmost stage of the register. Error-trapping decoding is most effective for decoding single-error-correcting block codes and burst-error-correcting codes (decoding of burst-error-correcting codes is discussed in Chapter 20). It is also effective for decoding some short double-error-correcting codes. When error-trapping decoding is applied to long and high-rate codes (small $n - k$) with large error-correcting capability, it becomes very ineffective, and much of the error-correcting capability is sacrificed. Several refinements of this simple decoding technique [12]–[19] have been devised to extend its application to multiple-error-correcting codes. One of the refinements is presented in the next section.

---

**EXAMPLE 5.11**

Consider the (15, 7) cyclic code generated by $g(X) = 1 + X^4 + X^6 + X^7 + X^8$. This code has a minimum distance of $d_{min} = 5$, which will be proved in Chapter 6. Hence, the code is capable of correcting any combination of two or fewer errors over a block of 15 digits. Suppose that we decode this code with an error-trapping decoder.

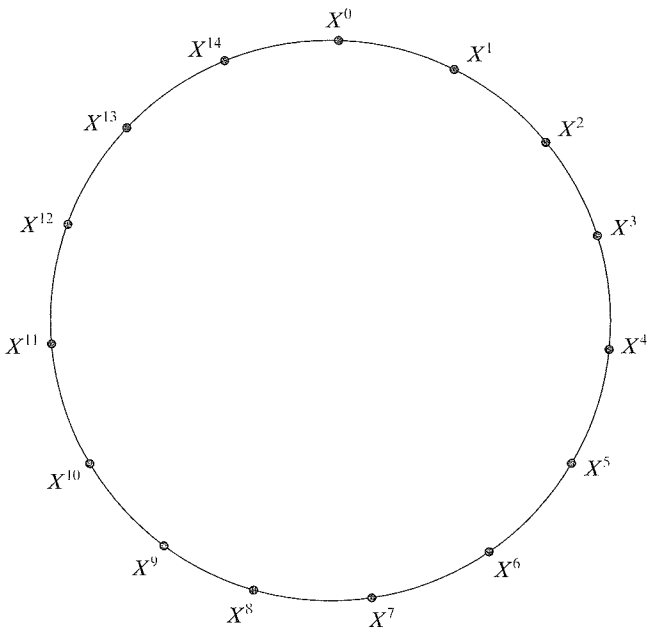


FIGURE 5.16: Ring arrangement of code digit positions.

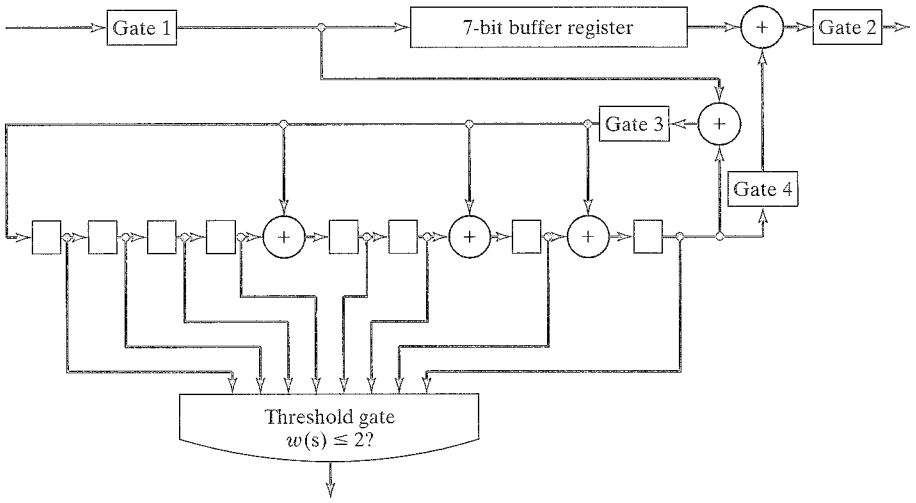


FIGURE 5.17: Error-trapping decoder for the (15, 7) cyclic code generated by $g(X) = 1 + X^4 + X^6 + X^7 + X^8$.

Clearly, any single error is confined to $n - k = 8$ consecutive positions. Therefore, any single error can be trapped and corrected. Now, consider any double errors over a span of 15 digits. If we arrange the 15 digit positions $X^0$ to $X^{14}$ as a ring, as shown in Figure 5.16, we see that any double errors are confined to eight consecutive positions. Hence, any double errors can be trapped and corrected. An error-trapping decoder for the (15, 7) cyclic code generated by $g(X) = 1 + X^4 + X^6 + X^7 + X^8$ is shown in Figure 5.17.

## 5.8 IMPROVED ERROR-TRAPPING DECODING

The error-trapping decoding discussed in Section 5.7 can be improved to correct error patterns such that, for each error pattern, most errors are confined to $n - k$ consecutive positions, and fewer errors are outside the $(n - k)$-digit span. This improvement needs additional circuitry. The complexity of the additional circuitry depends on how many errors outside an $(n - k)$-digit span are to be corrected. An improvement proposed by Kasami [17] is discussed here.

The error pattern $e(X) = e_0 + e_1X + e_2X^2 + \cdots + e_{n-1}X^{n-1}$ that corrupted the transmitted codeword can be divided into two parts:

$$e_p(X) = e_0 + e_1X + \cdots + e_{n-k-1}X^{n-k-1}$$

$$e_I(X) = e_{n-k}X^{n-k} + \cdots + e_{n-1}X^{n-1},$$

where $e_I(X)$ contains the errors in the message section of the received vector, and $e_p(X)$ contains the errors in the parity section of the received vector. Dividing $e_I(X)$ by the code generator polynomial $g(X)$, we obtain

$$e_I(X) = q(X)g(X) + \rho(X), \tag{5.35}$$

where $\rho(X)$ is the remainder with degree $n-k-1$ or less. Adding $\mathfrak{e}_p(X)$ to both sides of (5.35), we obtain

$$\mathfrak{e}(X) = \mathfrak{e}_p(X) + \mathfrak{e}_I(X) = \mathfrak{q}(X)\mathfrak{g}(X) + \rho(X) + \mathfrak{e}_p(X). \tag{5.36}$$

Because $\mathfrak{e}_p(X)$ has degree $n-k-1$ or less, $\rho(X) + \mathfrak{e}_p(X)$ must be the remainder resulting from dividing the error pattern $\mathfrak{e}(X)$ by the generator polynomial. Thus, $\rho(X) + \mathfrak{e}_p(X)$ is equal to the syndrome of the received vector $\mathfrak{r}(X)$,

$$\mathfrak{s}(X) = \rho(X) + \mathfrak{e}_p(X). \tag{5.37}$$

Rearranging (5.37), we have

$$\mathfrak{e}_p(X) = \mathfrak{s}(X) + \rho(X). \tag{5.38}$$

That is, if the error pattern $\mathfrak{e}_I(X)$ in the message positions is known, the error pattern $\mathfrak{e}_p(X)$ in the parity positions can be found.

Kasami's error-trapping decoding requires finding a set of polynomials $[\phi_j(X)]_{j=1}^{N}$ of degree $k-1$ or less such that, for any correctable error pattern $\mathfrak{e}(X)$, there is one polynomial $\phi_j(X)$ such that $X^{n-k}\phi_j(X)$ matches the message section of $\mathfrak{e}(X)$ or the message section of a cyclic shift of $\mathfrak{e}(X)$. The polynomials $\phi_j(X)$'s are called the *covering polynomials*. Let $\rho_j(X)$ be the remainder resulting from dividing $X^{n-k}\phi_j(X)$ by the generator polynomial $\mathfrak{g}(X)$ of the code.

The decoding procedure is as follows:

**Step 1.** Calculate the syndrome $\mathfrak{s}(X)$ by entering the entire received vector into the syndrome register.

**Step 2.** Calculate the weight of the sum $\mathfrak{s}(X)+\rho_j(X)$ for each $j = 1, 2, \cdots, N$ (i.e., $w[\mathfrak{s}(X) + \rho_j(X)]$ for $j = 1, 2, \cdots, N$).

**Step 3.** If, for some $l$,

$$w[\mathfrak{s}(X) + \rho_l(X)] \le t - w[\phi_l(X)],$$

then $X^{n-k}\phi_l(X)$ matches the error pattern in the message section of $\mathfrak{e}(X)$, and $\mathfrak{s}(X) + \rho_l(X)$ matches the error pattern in the parity section of $\mathfrak{e}(X)$. Thus,

$$\mathfrak{e}(X) = \mathfrak{s}(X) + \rho_l(X) + X^{n-k}\phi_l(X).$$

The correction is then performed by taking the modulo-2 sum $\mathfrak{r}(X) + \mathfrak{e}(X)$. This step requires $N$ $(n-k)$-input threshold gates to test the weights of $\mathfrak{s}(X) + \rho_j(X)$ for $j = 1, 2, \cdots, N$.

**Step 4.** If $w[\mathfrak{s}(X) + \rho_j(X)] > t - w[\phi_j(X)]$ for all $j = 1, 2, \cdots, N$, both syndrome and buffer registers are shifted cyclically once. Then, the new contents of the syndrome register, $\mathfrak{s}^{(1)}(X)$, is the syndrome corresponding to $\mathfrak{e}^{(1)}(X)$, which is obtained by shifting the error pattern $\mathfrak{e}(X)$ cyclically one place to the right.

**Step 5.** The weight of $\mathfrak{s}^{(1)}(X) + \rho_j(X)$ is computed for $j = 1, 2, \cdots, N$. If, for some $l$,

$$w[\mathfrak{s}^{(1)}(X) + \rho_l(X)] \le t - w[\phi_l(X)],$$

then $X^{n-k}\phi_l(X)$ matches the errors in the message section of $\mathfrak{e}^{(1)}(X)$, and $\mathfrak{s}^{(1)}(X) + \rho_l(X)$ matches the errors in the parity section of $\mathfrak{e}^{(1)}(X)$. Thus,

$$\mathfrak{e}^{(1)}(X) = s^{(1)}(X) + \rho_l(X) + X^{n-k}\phi_l(X).$$

The correction is then performed by taking the modulo-2 sum $\mathfrak{r}^{(1)}(X) + \mathfrak{e}^{(1)}(X)$. If

$$w[\mathfrak{s}^{(1)}(X) + \rho_j(X)] > t - w[\phi_j(X)]$$

for all $j = 1, 2, \cdots, N$, both syndrome and buffer registers are shifted cyclically once again.

Step 6. The syndrome and buffer registers are continuously shifted until $\mathfrak{s}^{(i)}(X)$ (the syndrome after the $i$th shift) is found such that, for some $l$,

$$w[\mathfrak{s}^{(i)}(X) + \rho_l(X)] \leq t - w[\phi_l(X)].$$

Then,

$$\mathfrak{e}^{(i)}(X) = \mathfrak{s}^{(i)}(X) + \rho_l(X) + X^{n-k}\phi_l(X),$$

where $\mathfrak{e}^{(i)}(X)$ is the $i$th cyclic shift of $\mathfrak{e}(X)$. If the weight $w[\mathfrak{s}^{(i)}(X) + \rho_j(X)]$ never drops to $t - w[\phi_j(X)]$ or less for all $j$ by the time that the syndrome and buffer registers have been cyclically shifted $n - 1$ times, an uncorrectable error pattern is detected.

The complexity of a decoder that employs the decoding method just described depends on $N$, the number of covering polynomials in $\{\phi_j(X)\}_{j=1}^{N}$. The combinational logical circuitry consists of $N$ $(n-k)$-input threshold gates. To find the set of covering polynomials $\{\phi_j(X)\}_{j=1}^{N}$ for a specific code is not an easy problem. Several methods for finding this set can be found in [17], [20], and [21].

This improved error-trapping method is applicable to many double- and triple-error-correcting codes, however, it is still applicable only to relatively short and low-rate codes. When the code length $n$ and error-correcting capability $t$ become large, the number of threshold gates required in the error-detecting logical circuitry becomes very large and impractical.

Other variations of error-trapping decoding can be found in [15], [16], and [18].

## 5.9 THE (23, 12) GOLAY CODE

As pointed out in Section 4.6, the (23, 12) Golay code [22] with a minimum distance of 7 is the only known multiple-error-correcting binary perfect code. This code can be put in cyclic form, and hence it can be encoded and decoded based on its cyclic structure. It is generated either by

$$\mathfrak{g}_1(X) = 1 + X^2 + X^4 + X^5 + X^6 + X^{10} + X^{11}$$

or by

$$\mathfrak{g}_2(X) = 1 + X + X^5 + X^6 + X^7 + X^9 + X^{11}.$$

Both $\mathfrak{g}_1(X)$ and $\mathfrak{g}_2(X)$ are factors of $X^{23} + 1$ and $X^{23} + 1 = (1 + X)\mathfrak{g}_1(X)\mathfrak{g}_2(X)$. The encoding can be accomplished by an 11-stage shift register with feedback

connections according to either $g_1(X)$ or $g_2(X)$. If the simple error-trapping scheme described in Section 5.7 is used for decoding this code, some of the double-error patterns and many of the triple-error patterns cannot be trapped. For example, consider the double-error patterns $e(X) = X^{11} + X^{22}$. The two errors are never confined to $n - k = 11$ consecutive positions, no matter how many times $e(X)$ is cyclically shifted. Therefore, they can never be trapped in the syndrome register and cannot be corrected. We can also readily see that the triple-error pattern $e(X) = X^5 + X^{11} + X^{22}$ cannot be trapped. Therefore, if the simple error-trapping scheme for decoding the Golay code is used, some of its error-correcting capability will be lost; however, the decoding circuitry is simple.

There are several practical ways to decode the (23, 12) Golay code up to its error-correcting capability $t = 3$. Two of the best are discussed in this section. Both are refined error-trapping schemes.

### 5.9.1 Kasami Decoder [17]

The Golay code can easily be decoded by Kasami's error-trapping technique. The set of polynomials $\{\phi_j(X)\}_{j=1}^{N}$ is chosen as follows:

$$\phi_1(X) = 0, \quad \phi_2(X) = X^5, \quad \phi_3(X) = X^6.$$

Let $g_1(X) = 1 + X^2 + X^4 + X^5 + X^6 + X^{10} + X^{11}$ be the generator polynomial. Dividing $X^{11}\phi_j(X)$ by $g_1(X)$ for $j = 1, 2, 3$, we obtain the following remainders:

$$\rho_1(X) = 0,$$
$$\rho_2(X) = X + X^2 + X^5 + X^6 + X^8 + X^9,$$
$$\rho_3(X) = X\rho_2(X) = X^2 + X^3 + X^6 + X^7 + X^9 + X^{10}.$$

A decoder based on Kasami's error-trapping scheme is shown in Figure 5.18. The received vector is shifted into the syndrome register from the rightmost stage; this is equivalent to preshifting the received vector 11 times cyclically. After the entire received vector has entered the syndrome register, the syndrome in the register corresponds to $r^{(11)}(X)$, which is the eleventh cyclic shift of $r(X)$. In this case, if the errors are confined to the first 11 high-order positions $X^{12}, X^{13}, \cdots, X^{22}$ of $r(X)$, the syndrome matches the errors in those positions. The error-correction procedure of this decoder is as follows:

**Step 1.** Gates 1, 3, and 5 are turned on; gates 2 and 4 are turned off. The received vector $r(X)$ is read into the syndrome register and simultaneously into the buffer register. The syndrome $s(X) = s_0 + s_1X + \cdots + s_{10}X^{10}$ is formed and is read into three threshold gates.

**Step 2.** Gates 1, 4, and 5 are turned off; gates 2 and 3 are turned on. The syndrome is tested for correctable error patterns as follows:

**a.** If the weight $w[s(X)] \leq 3$, all the errors are confined to the 11 high-order positions of $r(X)$, and $s(X)$ matches the errors. Thus, the erroneous symbols are the next 11 digits to come out of the buffer register. The output of the threshold gate $T_0$ turns gate 4 on and

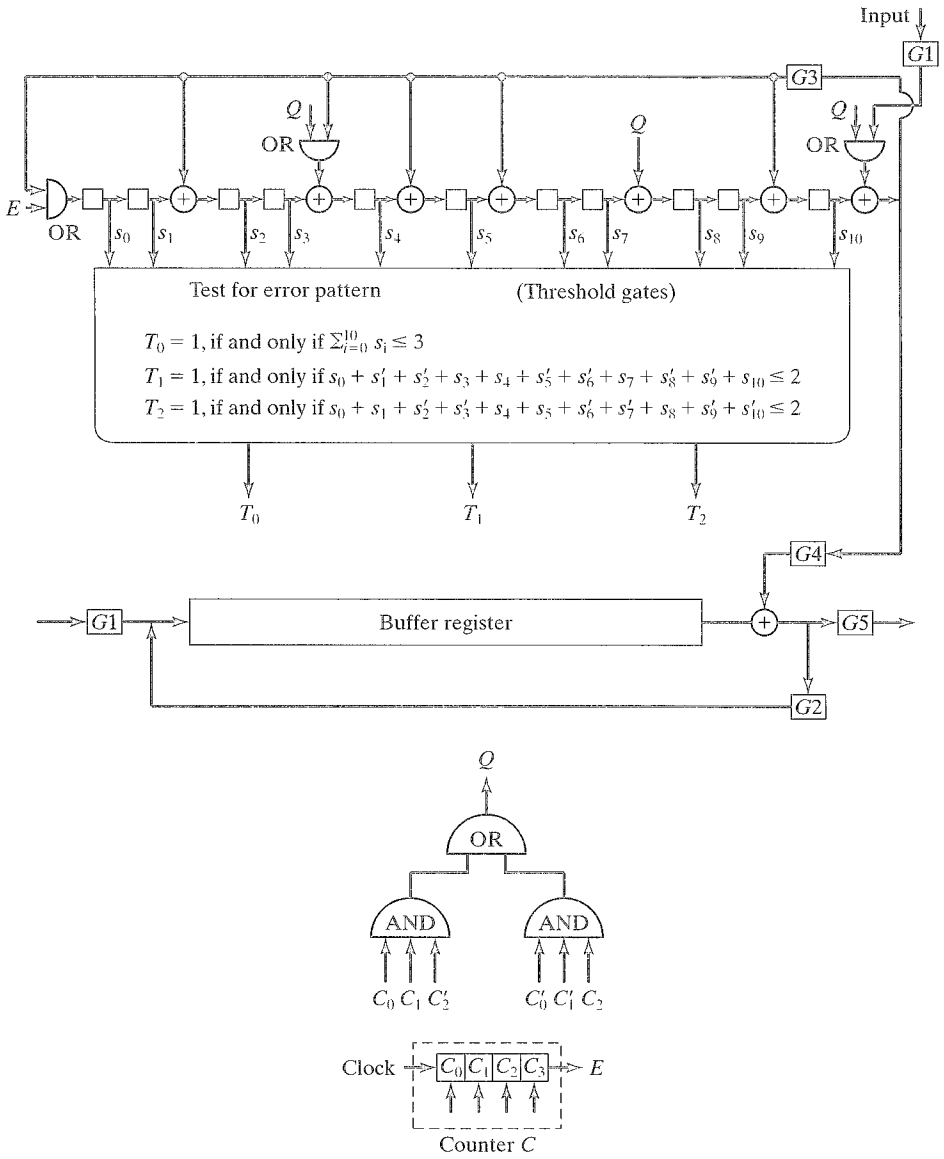


FIGURE 5.18: An error-trapping decoder for the (23, 12) Golay code.

gate 3 off. Digits are read out one at a time from the buffer register. The digit coming out of the syndrome register is added (modulo-2) to the digit coming out of the buffer. This corrects the errors.

b. If $w[\mathbf{s}(X)] > 3$, the weight of $\mathbf{s}(X) + \rho_2(X)$ is tested. If $w[\mathbf{s}(X) + \rho_2(X)] \leq 2$, then $\mathbf{s}(X) + \rho_2(X) = s_0 + s_1'X + s_2'X^2 + s_3X^3 + s_4X^4 + s_5'X^5 + s_6'X^6 + s_7X^7 + s_8'X^8 + s_9'X^9 + s_{10}X^{10}$ is identical to the error

pattern in the 11 high-order positions of the received word, and a single error occurs at location $X^5$, where $s_i'$ is the complement of $s_i$. Gate 4 is turned on, and gate 3 is turned off. The counter $C$ starts to count from 2. At the same time, the syndrome register is shifted without feedback. The output Q, which is 1 when and only when $C$ counts 3 and 4, is fed into the syndrome register to form the error pattern $\mathbf{s}(X) + \rho_2(X)$. When the counter $C$ counts 8, its output $E$ is 1, and the leftmost stage of the syndrome register is set to 1. This 1 is used for correcting the error at location $X^5$ in the received vector $\mathbf{r}(X)$. The digits coming out of the buffer are then corrected by the digits coming out of the syndrome register.

c. If $w[\mathbf{s}(X)] > 3$ and $w[\mathbf{s}(X)+\rho_2(X)] > 2$, the weight of $\mathbf{s}(X)+\rho_3(X)$ is tested. If $w[\mathbf{s}(X) + \rho_3(X)] \leq 2$, then $\mathbf{s}(X) + \rho_3(X) = s_0 + s_1X + s_2'X^2 + s_3'X^3 + s_4X^4 + s_5X^5 + s_6'X^6 + s_7'X^7 + s_8X^8 + s_9'X^9 + s_{10}'X^{10}$ is identical to the error pattern in the 11 high-order positions of the received word and a single error occurs at positions $X^6$. The correction is the same as step (b), except that counter $C$ starts to count from 3. If $w[\mathbf{s}(X)] > 3$, $w[\mathbf{s}(X) + \rho_2(X)] > 2$, and $w[\mathbf{s}(X) + \rho_3(X)] > 2$, then the decoder moves to step 3.

**Step 3.** Both the syndrome and buffer registers are cyclically shifted once with gates 1, 4, and 5 turned off and gates 2 and 3 turned on. The new contents of the syndrome register are $\mathbf{s}^{(1)}(X)$. Step 2 is then repeated.

**Step 4.** The decoding operation is completed as soon as the buffer register has been cyclically shifted 46 times. Gate 5 is then turned on and the vector in the buffer is shifted out to the data sink.

If there are three or fewer errors in the received vector, the vector in the buffer at the end of decoding will be the transmitted codeword. If there are more than three errors in the received vector, the vector in the buffer at the end of decoding will not be the transmitted codeword.

### 5.9.2 Systematic Search Decoder [23]

This decoding method is based on the fact that every pattern of three or fewer errors in a block of 23 digits can be cyclically shifted so that at most one of the errors lies outside a specified 11-digit section of the word. The decoding procedure is as follows:

**Step 1.** Compute the syndrome from the received vector.

**Step 2.** Shift the syndrome and the received vector 23 times, checking whether the weight of the syndrome ever falls to 3 or less. If it does, the syndrome with weight 3 or less matches the error pattern and correction can be made.

**Step 3.** If it does not, the first received information digit is inverted and step 2 is repeated, checking for a syndrome of weight of 2 or less. If one is found, the first received information digit was incorrect and the other two errors are specified by the syndrome. This completes the decoding.

Step 4. If no syndrome of weight 2 or less is found in step 3, the first information digit was originally correct. In this case, this bit must be reinverted.

Step 5. Repeat step 3 by inverting the second, third, $\cdots$, and twelfth information digits. Because not all the errors are in the parity-check section, an error must be corrected in this manner.

In every pattern of three or fewer errors, there is at least one error that if corrected, will leave the remaining error or errors within 11 successive positions. When the digit corresponding to this error is inverted, the remaining errors are corrected as in ordinary error trapping.

Compared with the Kasami decoder, the systematic search decoder has the advantage that only one weight-sensing (threshold) gate is required; however, it has the disadvantage that the clock and timing circuitry is more complex than that of the Kasami decoder, since 12 different digits must be inverted sequentially. Also, the Kasami decoder operates faster than the systematic search decoder.

This systematic search technique can be generalized for decoding other multiple-error-correcting cyclic codes.

The weight enumerator for the (23, 12) Golay code is

$$A(Z) = 1 + 253z^7 + 506z^8 + 1288z^{11} + 1288z^{12} + 506z^{15} + 253z^{16} + z^{23}.$$

If this code is used for error detection on a BSC, its probability of an undetected error $P_u(E)$ can be computed from (3.19). Moreover, $P_u(E)$ satisfies the upper bound $2^{-11}$ [i.e., $P_u(E) \leq 2^{-11}$] [24]. Therefore, the (23, 12) Golay code is a good error-detecting code.

## 5.10 SHORTENED CYCLIC CODES

In system design, if a code of suitable natural length or suitable number of information digits cannot be found, it may be desirable to shorten a code to meet the requirements. A technique for shortening a cyclic code is presented in this section. This technique leads to simple implementation of the encoding and decoding for the shortened code.

Given an $(n, k)$ cyclic code $C$ consider the set of codewords for which the $l$ leading high-order information digits are identical to zero. There are $2^{k-l}$ such codewords, and they form a linear subcode of $C$. If the $l$ zero information digits are deleted from each of these codewords, we obtain a set of $2^{k-l}$ vectors of length $n - l$. These $2^{k-l}$ shortened vectors form an $(n - l, k - l)$ linear code. This code is called a *shortened cyclic code* (or *polynomial code*), and it is not cyclic. A shortened cyclic code has at least the same error-correcting capability as the code from which it is derived.

The encoding and decoding for a shortened cyclic code can be accomplished by the same circuits as those employed by the original cyclic code. This is so because the deleted $l$ leading-zero information digits do not affect the parity-check and syndrome computations; however, in decoding the shortened cyclic code after the entire received vector has been shifted into the syndrome register, the syndrome register must be cyclically shifted $l$ times to generate the proper syndrome for decoding the first received digit $r_{n-l-1}$. For large $l$, these extra $l$ shifts of the syndrome register

cause undesirable decoding delay; they can be eliminated by modifying either the connections of the syndrome register or the error-pattern detection circuit.

Let $\mathbf{r}(X) = r_0 + r_1 X + \cdots + r_{n-l-1} X^{n-l-1}$ be the received polynomial. Suppose that $\mathbf{r}(X)$ is shifted into the syndrome register from the right end. If the decoding circuit for the original cyclic code is used for decoding the shortened code, the proper syndrome for decoding the received digit $r_{n-l-1}$ is equal to the remainder resulting from dividing $X^{n-k+l}\mathbf{r}(X)$ by the generator polynomial $\mathbf{g}(X)$. Because shifting $\mathbf{r}(X)$ into the syndrome register from the right end is equivalent to premultiplying $\mathbf{r}(X)$ by $X^{n-k}$, the syndrome register must be cyclically shifted another $l$ times after the entire $\mathbf{r}(X)$ has been shifted into the register. Now, we want to show how these extra $l$ shifts can be eliminated by modifying the connections of the syndrome register. Dividing $X^{n-k+l}\mathbf{r}(X)$ by $\mathbf{g}(X)$, we obtain

$$X^{n-k+l}\mathbf{r}(X) = \mathbf{a}_1(X)\mathbf{g}(X) + \mathbf{s}^{(n-k+l)}(X), \tag{5.39}$$

where $\mathbf{s}^{(n-k+l)}(X)$ is the remainder and the syndrome for decoding the received digit $r_{n-l-1}$. Next, we divide $X^{n-k+l}$ by $\mathbf{g}(X)$. Let $\boldsymbol{\rho}(X) = \rho_0 + \rho_1 X + \cdots + \rho_{n-k-1} X^{n-k-1}$ be the remainder resulting from this division. Then, we have the following relation:

$$\boldsymbol{\rho}(X) = X^{n-k+l} + \mathbf{a}_2(X)\mathbf{g}(X). \tag{5.40}$$

Multiplying both sides of (5.40) by $\mathbf{r}(X)$ and using the equality of (5.39), we obtain the following relation between $\boldsymbol{\rho}(X)\mathbf{r}(X)$ and $\mathbf{s}^{(n-k+l)}(X)$:

$$\boldsymbol{\rho}(X)\mathbf{r}(X) = [\mathbf{a}_1(X) + \mathbf{a}_2(X)\mathbf{r}(X)]\mathbf{g}(X) + \mathbf{s}^{(s-k+l)}(X). \tag{5.41}$$

The foregoing equality suggests that we can obtain the syndrome $\mathbf{s}^{(n-k+l)}(X)$ by multiplying $\mathbf{r}(X)$ by $\boldsymbol{\rho}(X)$ and dividing the product $\boldsymbol{\rho}(X)\mathbf{r}(X)$ by $\mathbf{g}(X)$. Computing $\mathbf{s}^{(n-k+l)}(X)$ in this way, we can avoid the extra $l$ shifts of the syndrome register. Simultaneously multiplying $\mathbf{r}(X)$ by $\boldsymbol{\rho}(X)$ and dividing $\boldsymbol{\rho}(X)\mathbf{r}(X)$ by $\mathbf{g}(X)$ can be accomplished by the circuit shown in Figure 5.19. As soon as the received polynomial $\mathbf{r}(X)$ has been shifted into the register, the contents in the register form the syndrome $\mathbf{s}^{(n-k+l)}(X)$, and the first received digit is ready to be decoded.

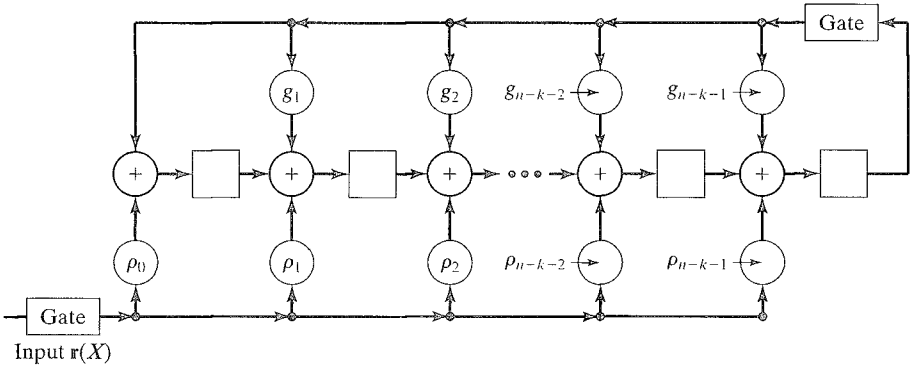


FIGURE 5.19: Circuit for multiplying $\mathbf{r}(X)$ by $\boldsymbol{\rho}(X) = \rho_0 + \rho_1 X + \cdots + \rho_{n-k-1} X^{n-k-1}$ and dividing $\boldsymbol{\rho}(X)\mathbf{r}(X)$ by $\mathbf{g}(X) = 1 + g_1 X + \cdots + X^{n-k}$.

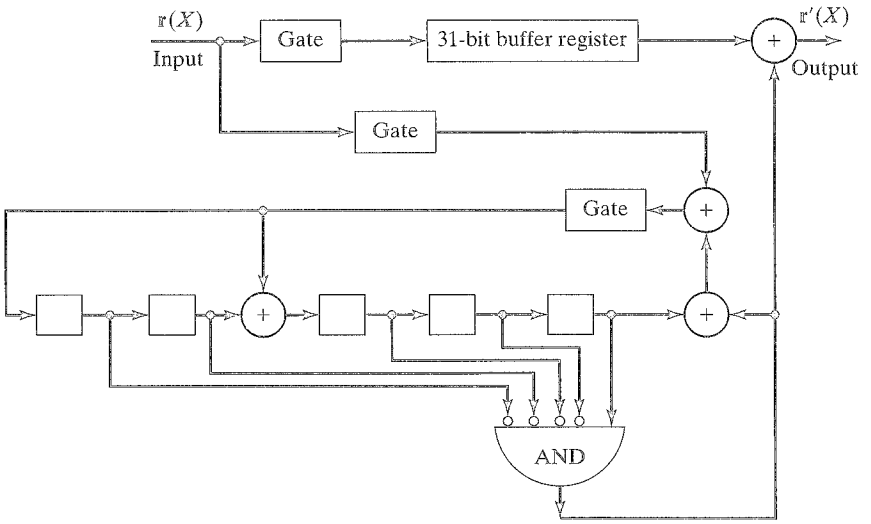


FIGURE 5.20: Decoding circuit for the (31, 26) cyclic Hamming code generated by $g(X) = 1 + X^2 + X^5$.

**EXAMPLE 5.12**

For $m = 5$, there exists a (31, 26) cyclic Hamming code generated by $g(X) = 1 + X^2 + X^5$. Suppose that it is shortened by three digits. The resultant shortened code is a (28, 23) linear code. The decoding circuit for the (31, 26) cyclic code is shown in Figure 5.20.

This circuit can be used to decode the (28, 23) shortened code. To eliminate the extra shifts of the syndrome register, we need to modify the connections of the syndrome register. First, we need to determine the polynomial $\rho(X)$. Dividing $X^{n-k+3}$ by $g(X) = 1 + X^2 + X^5$, we have

$$\begin{array}{r|l} & X^3 + 1 \\ \hline X^5 + X^2 + 1 & X^8 \\ & \underline{X^8 + X^5 + X^3} \\ & \quad X^5 + X^3 \\ & \quad \underline{X^5 \qquad + X^2 + 1} \\ & \qquad\quad X^3 + X^2 + 1 \end{array}$$

and $\rho(X) = 1 + X^2 + X^3$. The modified decoding circuit for the (28, 23) shortened code is shown in Figure 5.21.

The extra $l$ shifts of the syndrome register for decoding the shortened cyclic code can also be avoided by modifying the error-pattern detection circuit of the decoder for the original cyclic code.

The error-pattern detection circuit is redesigned to check whether the syndrome in the syndrome register corresponds to a correctable error pattern $e(X)$

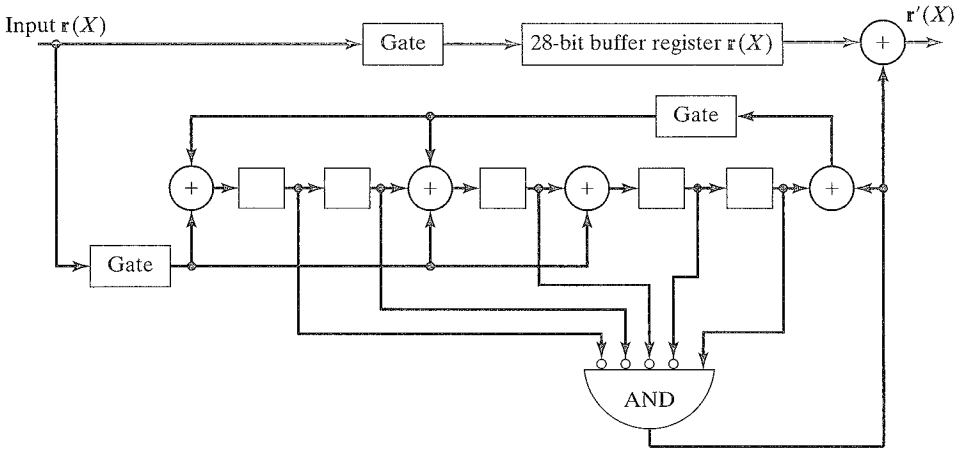


FIGURE 5.21: Decoding circuit for the (28, 23) shortened cyclic code generated by $\mathfrak{g}(X) = 1 + X^2 + X^5$.

with an error at position $X^{n-l-1}$ (i.e., $e_{n-l-1} = 1$). When the received digit $r_{n-l-1}$ is corrected, the effect of the error digit $e_{n-l-1}$ on the syndrome should be removed. Suppose that the received vector is shifted into the syndrome register from the right end. Let $\rho(X) = \rho_0 + \rho_1 X + \cdots + \rho_{n-k-1}X^{n-k-1}$ be the remainder resulting from dividing $X^{n-l-1} \cdot X^{n-k} = X^{2n-k-l-1}$ by the generator polynomial $\mathfrak{g}(X)$. Then, the effect of the error digit $e_{n-l-1}$ on the syndrome is removed by adding $\rho(X)$ to the syndrome in the syndrome register.

**EXAMPLE 5.13**

Consider the (28, 23) shortened cyclic code obtained by deleting three digits from the (31, 26) cyclic Hamming code generated by $\mathfrak{g}(X) = 1 + X^2 + X^5$. Suppose that in decoding this code the received vector is shifted into the syndrome register from the right end. If a single error occurs at the position $X^{27}$ [or $\mathfrak{e}(X) = X^{27}$], the syndrome corresponding to this error pattern is the remainder resulting from dividing $X^5\mathfrak{e}(X) = X^{32}$ by $\mathfrak{g}(X) = 1 + X^2 + X^5$. The resultant syndrome is (0 1 0 0 0). Thus, in decoding the (28, 23) shortened Hamming code, the error-pattern detection circuit may be designed to check whether the syndrome in the syndrome register is (0 1 0 0 0). Doing so avoids the extra three shifts of the syndrome register. The resultant decoding circuit with syndrome resetting is shown in Figure 5.22.

Shortened cyclic codes for error detection in conjunction with ARQ protocols are widely used for error control, particularly in computer communications. In these applications they are often called *cyclic redundancy check* (CRC) codes. A CRC code is, in general, generated by either a primitive polynomial $\mathfrak{p}(X)$ or a polynomial $\mathfrak{g}(X) = (X + 1)\mathfrak{p}(X)$. A number of CRC codes have become international standards for error detection in various contexts. A few standard CRC codes follow:

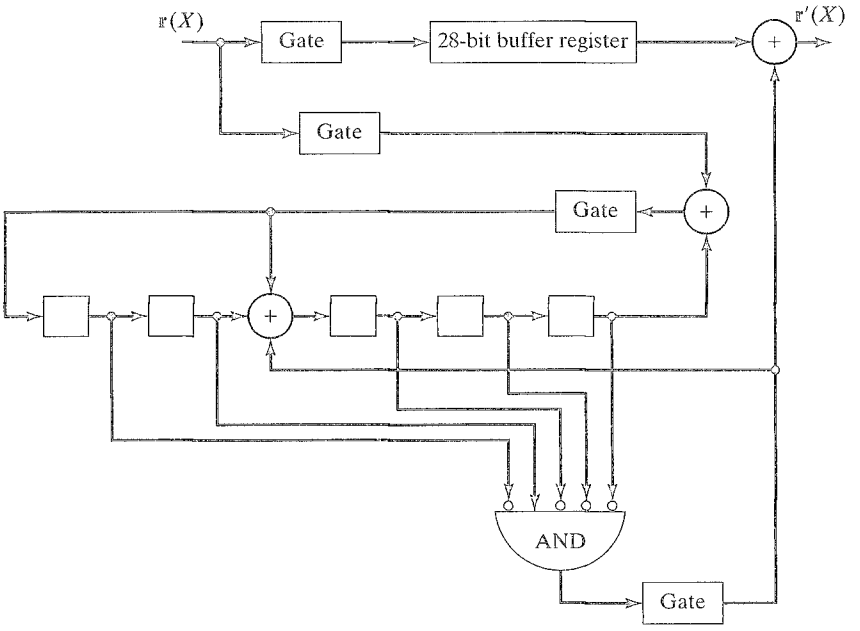


FIGURE 5.22: Another decoding circuit for the (28, 23) shortened Hamming code generated by $g(X) = 1 + X^2 + X^5$.

*CCITT X-25* (Consultative Committee for International Telegraphy and Telephony, Recommendation X-25)

$$\begin{aligned} g(X) &= (X+1)(X^{15} + X^{14} + X^{13} + X^{12} + X^4 + X^3 + X^2 + X + 1) \\ &= X^{16} + X^{12} + X^5 + 1. \end{aligned}$$

*ANSI* (American National Standards Institute)

$$g(X) = (X+1)(X^{15} + X + 1) = X^{16} + X^{15} + X^2 + 1,$$

*IBM-SDLC* (IBM Synchronous Data Link Control)

$$\begin{aligned} g(X) &= (X+1)^2(X^{14} + X^{13} + X^{12} + X^{10} + X^8 + X^6 + X^5 + X^4 + X^3 + X + 1) \\ &= X^{16} + X^{15} + X^{13} + X^7 + X^4 + X^2 + X + 1, \end{aligned}$$

*IEC TC57*

$$\begin{aligned} g(X) &= (X+1)^2(X^{14} + X^{10} + X^9 + X^8 + X^5 + X^3 + X^2 + X + 1) \\ &= X^{16} + X^{14} + X^{11} + X^8 + X^6 + X^5 + X^4 + 1, \end{aligned}$$

*IEEE Standard 802.3*

$$\begin{aligned} g(X) = {} & X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} \\ & + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1. \end{aligned}$$

## 5.11 CYCLIC PRODUCT CODES

The obvious way to implement a product code is to set up the code array and operate on rows and then on columns (or columns and then rows) in encoding and decoding, but there is an alternative that can be extremely attractive. In many cases the product code of cyclic codes is cyclic, and cyclic code implementation is much simpler.

If the component codes $C_1$ and $C_2$ are cyclic, and if their lengths, $n_1$ and $n_2$, are relatively prime, the product code $C_1 \times C_2$ is cyclic if the code digits of a code array are transmitted in a proper order [3, 25, 26]. We start with the upper right corner and move down and to *the left* on a 45° diagonal, as shown in Figure 5.23. When we reach the end of a column, we move to the top of the next column. When we reach the end of a row, we move to the rightmost digit of the next row.

Because $n_1$ and $n_2$ are relatively prime, there exists a pair of integers $a$ and $b$ such that

$$an_1 + bn_2 = 1.$$

Let $\mathbb{g}_1(X)$ and $\mathbb{h}_1(X)$ be the generator and parity polynomials of the $(n_1, k_1)$ cyclic code $C_1$, and let $\mathbb{g}_2(X)$ and $\mathbb{h}_2(X)$ be the generator and parity polynomials of the $(n_2, k_2)$ cyclic code $C_2$. Then, it is possible to show [25, 26] that the generator polynomial $\mathbb{g}(X)$ of the cyclic product code of $C_1$ and $C_2$ is the greatest common divisor (GCD) of $X^{n_1 n_2} - 1$ and $\mathbb{g}_1(X^{bn_2})\mathbb{g}_2(X^{an_1})$; that is,

$$\mathbb{g}(X) = \text{GCD}[X^{n_1 n_2} - 1, \mathbb{g}_1(X^{bn_2})\mathbb{g}_2(X^{an_1})], \tag{5.42}$$

and the parity polynomial $\mathbb{h}(X)$ of the cyclic product code is the greatest common divisor of $\mathbb{h}_1(X^{bn_2})$ and $\mathbb{h}_2(X^{an_1})$; that is,

$$\mathbb{h}(X) = \text{GCD}[\mathbb{h}_1(X^{bn_2}), \mathbb{h}_2(X^{an_1})]. \tag{5.43}$$

The complexity of the decoder for cyclic product codes is comparable to the complexity of the decoders for both the $(n_1, k_1)$ code and the $(n_2, k_2)$ code. At the receiving end of the channel, the received vector may again be rearranged as a rectangular array. Thus, the decoder can decode each of the row (or column) codewords separately and then decode each of the column (or row) codewords.

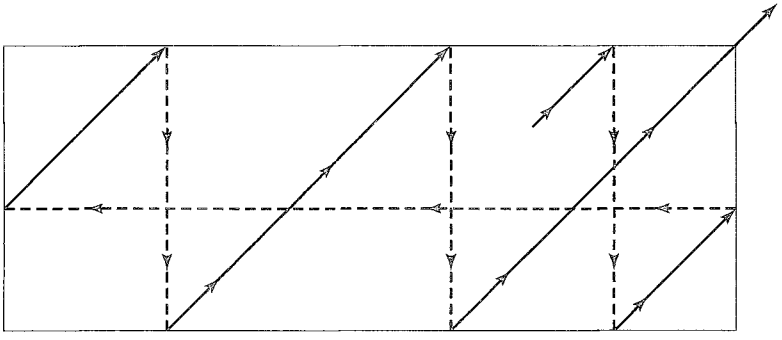

FIGURE 5.23: Transmission of a cyclic product code.

Alternatively, in the transmitted codeword, the set of $n_1$ digits formed by selecting every $(n_2)$th digit are the $n_1$ digits of a codeword of $C_1$ permuted in a fixed way. They can be permuted back to their original form and corrected by a Meggitt-type decoder. The digits in the permuted form are a codeword in a related code and can be decoded directly in this form by a Meggitt-type decoder. Similarly, the column code $C_2$ can be corrected by selecting every $(n_1)$th digit from the large codeword. Thus, the total equipment required is roughly that required to decode the two individual codes.

## 5.12 QUASI-CYCLIC CODES

Cyclic codes possess full cyclic symmetry; that is, cyclically shifting a codeword any number of symbol positions, either to the right or to the left, results in another codeword. This cyclic symmetry structure makes it possible to implement the encoding and decoding of cyclic codes with simple shift registers and logic circuits. There are other linear block codes that do not possess full cyclic symmetry but do have partial cyclic structure, namely, quasi-cyclic codes.

A *quasi-cyclic code* is a linear code for which cyclically shifting a codeword a fixed number $n_0 \neq 1$ ( or a multiple of $n_0$) of symbol positions either to the right or to the left results in another codeword. It is clear that for $n_0 = 1$, a quasi-cyclic code is a cyclic code. The integer $n_0$ is called the *shifting constraint*. It is clear that the dual of a quasi-cyclic code is also quasi-cyclic.

As an example, consider the (9, 3) code generated by the following generator matrix:

$$\mathbb{G} = \begin{bmatrix} 111 & 100 & 110 \\ 110 & 111 & 100 \\ 100 & 110 & 111 \end{bmatrix}.$$

The eight codewords of this code are listed in Table 5.6. Suppose we cyclically shift the fifth codeword in Table 5.6, (001011010) three symbol positions to the right. This shift results in the seventh codeword (010001011) in Table 5.6. If we cyclically shift the fifth codeword one and two positions to the right, we obtain two vectors, (000101101) and (100010110), respectively, which are not codewords given in Table 5.6. Therefore, the (9, 3) code given in Table 5.6 is a quasi-cyclic code with shifting constraint $n_0 = 3$. This code also can be encoded with a shift register as shown in Figure 5.24. Let $(c_0, c_1, c_2)$ be the message to be encoded. As soon as the

TABLE 5.6: The codewords of the (9, 3) quasi-cyclic code.

| | | |
|---|---|---|
| 000 | 000 | 000 |
| 111 | 100 | 110 |
| 110 | 111 | 100 |
| 100 | 110 | 111 |
| 001 | 011 | 010 |
| 011 | 010 | 001 |
| 010 | 001 | 011 |
| 101 | 101 | 101 |

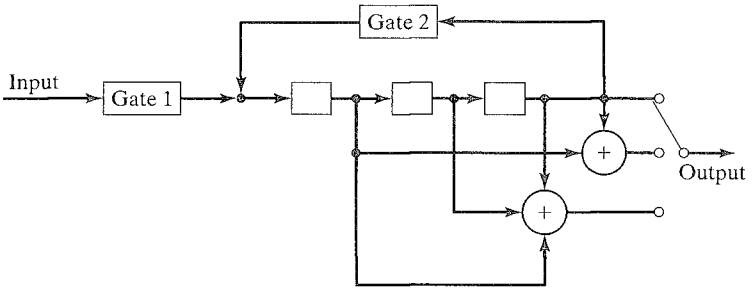


FIGURE 5.24: An encoding circuit for a (9, 3) quasi-cyclic code.

three information symbols have been shifted into the register, gate 1 is deactivated and gate 2 is activated. The information symbol $c_2$ and two parity-check symbols $p_2^{(1)}$ and $p_2^{(2)}$ appear at the output terminals and then are shifted into the channel. The two parity-check symbols are given by

$$p_2^{(1)} = c_0 + c_2,$$
$$p_2^{(2)} = c_0 + c_1 + c_2.$$

Next, the register is shifted once. The content of the register is now $(c_2, c_0, c_1)$. The information symbol $c_1$ and two parity-check symbols $p_1^{(1)}$ and $p_1^{(2)}$ appear at the output terminals, and they are shifted into the channel. The parity-check symbols $p_1^{(1)}$ and $p_1^{(2)}$ are given by

$$p_1^{(1)} = c_1 + c_2,$$
$$p_1^{(2)} = c_0 + c_1 + c_2.$$

At this point, the register is shifted once again. The content of the register is now $(c_1, c_2, c_0)$, and the information symbol $c_0$ and two parity-check symbols $p_0^{(1)}$ and $p_0^{(2)}$ appear at the output terminals.

These three symbols are then shifted into the channel. The two parity-check symbols are given by

$$p_0^{(1)} = c_0 + c_1,$$
$$p_0^{(2)} = c_0 + c_1 + c_2.$$

This completes the encoding. The codeword has the form

$$\mathbf{v} = (p_0^{(2)}, p_0^{(1)}, c_0, p_1^{(2)}, p_1^{(1)}, c_1, p_2^{(2)}, p_2^{(1)}, c_2),$$

which consists of three blocks, each of which consists of one unaltered information symbol and two parity-check symbols. This form may also be regarded as a systematic form.

For an $(mn_0, mk_0)$ quasi-cyclic code with shifting constraint $n_0$, the generator matrix in systematic form is

$$\mathbb{G} = \begin{bmatrix} \mathbb{P}_0\mathbb{I} & \mathbb{P}_1\mathbb{0} & \cdots & \mathbb{P}_{m-1}\mathbb{0} \\ \mathbb{P}_{m-1}\mathbb{0} & \mathbb{P}_0\mathbb{I} & \cdots & \mathbb{P}_{m-2}\mathbb{0} \\ \vdots & \vdots & & \vdots \\ \mathbb{P}_1\mathbb{0} & \mathbb{P}_2\mathbb{0} & \cdots & \mathbb{P}_0\mathbb{I} \end{bmatrix}, \tag{5.44}$$

where $\mathbb{I}$ and $\mathbb{0}$ represent the $k_0 \times k_0$ identity and zero matrices, respectively, and $\mathbb{P}_i$ is an arbitrary $k_0 \times (n_0 - k_0)$ matrix. Each row (as a sequence of $m$ $k_0 \times n_0$ matrices) is the cyclic shift (to the right) of the row immediately above it, and the row at the top is the cyclic shift of the bottom row. Each column of $\mathbb{G}$ is the downward cyclic shift of the column on its left (or the upward cyclic shift of the column on its right). A message for the code consists of $m$ $k_0$-bit blocks, and a codeword consists of $m$ $n_0$-bit blocks. Each of these $m$ $n_0$-bit blocks consists of $k_0$ unaltered information symbols and $n_0 - k_0$ parity-check symbols. The parity-check matrix corresponding to the generator matrix given by (5.44) is

$$\mathbb{H} = \begin{bmatrix} \mathbb{I}\mathbb{P}_0^T & \mathbb{0}\mathbb{P}_{m-1}^T & \cdots & \mathbb{0}\mathbb{P}_1^T \\ \mathbb{0}\mathbb{P}_1^T & \mathbb{I}\mathbb{P}_0^T & \cdots & \mathbb{0}\mathbb{P}_2^T \\ \vdots & \vdots & & \vdots \\ \mathbb{0}\mathbb{P}_{m-1}^T & \mathbb{0}\mathbb{P}_{m-2}^T & \cdots & \mathbb{I}\mathbb{P}_0^T \end{bmatrix}, \tag{5.45}$$

where $\mathbb{I}$ and $\mathbb{0}$ represent the $(n_0 - k_0) \times (n_0 - k_0)$ identity and zero matrices, respectively, and $\mathbb{P}_i^T$ is the transpose of $\mathbb{P}_i$. Consider the (9, 3) quasi-cyclic code given previously for which $k_0 = 1$ and $n_0 = 3$. The parity-check matrix in systematic form is

$$\mathbb{H} = \begin{bmatrix} 101 & 001 & 001 \\ 011 & 001 & 000 \\ 001 & 101 & 001 \\ 000 & 011 & 001 \\ 001 & 001 & 101 \\ 001 & 000 & 011 \end{bmatrix}.$$

A more general form for the generator matrix of an $(mn_0, mk_0)$ quasi-cyclic code is

$$\mathbb{G} = \begin{bmatrix} \mathbb{G}_0 & \mathbb{G}_1 & \cdots & \mathbb{G}_{m-1} \\ \mathbb{G}_{m-1} & \mathbb{G}_0 & \cdots & \mathbb{G}_{m-2} \\ \vdots & \vdots & & \vdots \\ \mathbb{G}_2 & \mathbb{G}_3 & \cdots & \mathbb{G}_1 \\ \mathbb{G}_1 & \mathbb{G}_2 & \cdots & \mathbb{G}_0 \end{bmatrix}, \tag{5.46}$$

where each $\mathbb{G}_i$ is a $k_0 \times n_0$ submatrix. We see that $\mathbb{G}$ given by (5.46) displays the cyclic structure among the rows and columns in terms of the submatrices $\mathbb{G}_i$'s. For $0 \leq j < m$, let $\mathbb{M}_j \triangleq [\mathbb{G}_j, \mathbb{G}_{j-1}, \cdots, \mathbb{G}_{j+1}]^T$ denote the $j$th column of $\mathbb{G}$ (with $\mathbb{G}_m = \mathbb{G}_0$). $\mathbb{M}_j$ is an $mk_0 \times n_0$ submatrix. Then, we can put $\mathbb{G}$ in the following form:

$$\mathbb{G} = [\mathbb{M}_0, \mathbb{M}_1, \cdots, \mathbb{M}_{m-1}].$$

For $0 \leq l < n_0$, let $\mathbb{Q}_l$ be the $mk_0 \times m$ submatrix that is formed by taking the $l$th columns from $\mathbb{M}_0, \mathbb{M}_1, \cdots, \mathbb{M}_{m-1}$. Then, we can put $\mathbb{G}$ in the following form:

$$\mathbb{G}_c = [\mathbb{Q}_0, \mathbb{Q}_1, \cdots, \mathbb{Q}_{n_0-1}].$$

Each column of $\mathbb{Q}_l$ consists of $mk_0$ bits that are regarded as $m$ $k_0$-bit bytes (a *byte* is a group of $k_0$ binary digits). In terms of bytes, $\mathbb{Q}_l$ is regarded as an $m \times m$ matrix that has the following cyclic structure: (1) each row is the cyclic shift (to the right) of the row immediately above it, and the top row is the cyclic shift of the bottom row; (2) each column is the downward cyclic shift of the column on its left, and the leftmost column is the downward cyclic shift of the rightmost column. The matrix $\mathbb{Q}_l$ is called a *circulant*. Therefore, $\mathbb{G}_c$ consists of $n_0$ circulants. Most often, quasi-cyclic codes are studied in circulant form.

---

**EXAMPLE 5.14**

Consider the (15, 5) quasi-cyclic code with parameters $m = 5$, $n_0 = 3$, and $k_0 = 1$ that is generated by the following generator matrix:

$$\mathbb{G} = \begin{bmatrix} 001 & 100 & 010 & 110 & 110 \\ 110 & 001 & 100 & 010 & 110 \\ 110 & 110 & 001 & 100 & 010 \\ 010 & 110 & 110 & 001 & 100 \\ 100 & 010 & 110 & 110 & 001 \end{bmatrix}.$$
$$\mathbb{M}_0 \quad \mathbb{M}_1 \quad \mathbb{M}_2 \quad \mathbb{M}_3 \quad \mathbb{M}_4$$

This quasi-cyclic code has a minimum distance of 7. In circulant form, the generator matrix takes the following form:

$$\mathbb{G} = \begin{bmatrix} 01011 & 00111 & 10000 \\ 10101 & 10011 & 01000 \\ 11010 & 11001 & 00100 \\ 01101 & 11100 & 00010 \\ 10110 & 01110 & 00001 \end{bmatrix}.$$
$$\mathbb{Q}_0 \quad \mathbb{Q}_1 \quad \mathbb{Q}_2$$

---

## PROBLEMS

**5.1** Consider the (15, 11) cyclic Hamming code generated by $\mathbf{g}(X) = 1 + X + X^4$.
- **a.** Determine the parity polynomial $\mathbf{h}(X)$ of this code.
- **b.** Determine the generator polynomial of its dual code.
- **c.** Find the generator and parity matrices in systematic form for this code.

**5.2** Devise an encoder and a decoder for the (15, 11) cyclic Hamming code generated by $\mathbf{g}(X) = 1 + X + X^4$.

5.3 Show that $\mathfrak{g}(X) = 1 + X^2 + X^4 + X^6 + X^7 + X^{10}$ generates a (21, 11) cyclic code. Devise a syndrome computation circuit for this code. Let $\mathfrak{r}(X) = 1 + X^5 + X^{17}$ be a received polynomial. Compute the syndrome of $\mathfrak{r}(X)$. Display the contents of the syndrome register after each digit of $\mathfrak{r}$ has been shifted into the syndrome computation circuit.

5.4 Shorten this (15, 11) cyclic Hamming by deleting the seven leading high-order message digits. The resultant code is an (8, 4) shortened cyclic code. Design a decoder for this code that eliminates the extra shifts of the syndrome register.

5.5 Shorten the (31, 26) cyclic Hamming code by deleting the 11 leading high-order message digits. The resultant code is a (20, 15) shortened cyclic code. Devise a decoding circuit for this code that requires no extra shifts of the syndrome register.

5.6 Let $\mathfrak{g}(X)$ be the generator polynomial of a binary cyclic code of length $n$.

a. Show that if $\mathfrak{g}(X)$ has $X + 1$ as a factor, the code contains no codewords of odd weight.

b. If $n$ is odd and $X + 1$ is not a factor of $\mathfrak{g}(X)$, show that the code contains a codeword consisting of all 1's.

c. Show that the code has a minimum weight of at least 3 if $n$ is the smallest integer such that $\mathfrak{g}(X)$ divides $X^n + 1$.

5.7 Consider a binary $(n, k)$ cyclic code $C$ generated by $\mathfrak{g}(X)$. Let

$$\mathfrak{g}^*(X) = X^{n-k}\mathfrak{g}(X^{-1})$$

be the reciprocal polynomial of $\mathfrak{g}(X)$.

a. Show that $\mathfrak{g}^*(X)$ also generates an $(n, k)$ cyclic code.

b. Let $C^*$ denote the cyclic code generated by $\mathfrak{g}^*(X)$. Show that $C$ and $C^*$ have the same weight distribution.

(*Hint:* Show that

$$\mathfrak{v}(X) = v_0 + v_1X + \cdots + v_{n-2}X^{n-2} + v_{n-1}X^{n-1}$$

is a code polynomial in $C$ if and only if

$$X^{n-1}\mathfrak{v}(X^{-1}) = v_{n-1} + v_{n-2}X + \cdots + v_1X^{n-2} + v_0X^{n-1}$$

is a code polynomial in $C^*$.)

5.8 Consider a cyclic code $C$ of length $n$ that consists of both odd-weight and even-weight codewords. Let $\mathfrak{g}(X)$ and $A(z)$ be the generator polynomial and weight enumerator for this code. Show that the cyclic code generated by $(X + 1)\mathfrak{g}(X)$ has weight enumerator

$$A_1(z) = \tfrac{1}{2}[A(z) + A(-z)].$$

5.9 Suppose that the (15, 10) cyclic Hamming code of minimum distance 4 is used for error detection over a BSC with transition probability $p = 10^{-2}$. Compute the probability of an undetected error, $P_u(E)$, for this code.

5.10 Consider the $(2^m - 1, 2^m - m - 2)$ cyclic Hamming code $C$ generated by $\mathfrak{g}(X) = (X + 1)\mathfrak{p}(X)$, where $\mathfrak{p}(X)$ is a primitive polynomial of degree $m$. An error pattern of the form

$$\mathfrak{e}(X) = X^i + X^{i+1}$$

is called a *double-adjacent-error pattern*. Show that no two double-adjacent-error patterns can be in the same coset of a standard array for $C$. Therefore, the code is capable of correcting all the single-error patterns and all the double-adjacent-error patterns.

**5.11** Devise a decoding circuit for the (7, 3) Hamming code generated by $\mathbf{g}(X) = (X+1)(X^3+X+1)$. The decoding circuit corrects all the single-error patterns and all the double-adjacent-error patterns (see Problem 5.10).

**5.12** For a cyclic code, if an error pattern $\mathbf{e}(X)$ is detectable, show that its $i$th cyclic shift $e^{(i)}(X)$ is also detectable.

**5.13** In the decoding of an $(n, k)$ cyclic code, suppose that the received polynomial $\mathbf{r}(X)$ is shifted into the syndrome register from the right end, as shown in Figure 5.11. Show that when a received digit $r_i$ is detected in error and is corrected, the effect of error digit $e_i$ on the syndrome can be removed by feeding $e_i$ into the syndrome register from the right end, as shown in Figure 5.11.

**5.14** Let $\mathbf{v}(X)$ be a code polynomial in a cyclic code of length $n$. Let $l$ be the smallest integer such that

$$\mathbf{v}^{(l)}(X) = \mathbf{v}(X).$$

Show that if $l \neq 0$, $l$ is a factor of $n$.

**5.15** Let $\mathbf{g}(X)$ be the generator polynomial of an $(n, k)$ cyclic code $C$. Suppose $C$ is interleaved to a depth of $\lambda$. Prove that the interleaved code $C^{\lambda}$ is also cyclic and its generator polynomial is $\mathbf{g}(X^{\lambda})$.

**5.16** Construct all the binary cyclic codes of length 15. (*Hint:* Using the fact that $X^{15}+1$ has all the nonzero elements of $GF(2^4)$ as roots and using Table 2.9, factor $X^{15}+1$ as a product of irreducible polynomials.)

**5.17** Let $\beta$ be a nonzero element in the Galois field $GF(2^m)$, and $\beta \neq 1$. Let $\boldsymbol{\phi}(X)$ be the minimum polynomial of $\beta$. Is there a cyclic code with $\boldsymbol{\phi}(X)$ as the generator polynomial? If your answer is yes, find the shortest cyclic code with $\boldsymbol{\phi}(X)$ as the generator polynomial.

**5.18** Let $\beta_1$ and $\beta_2$ be two distinct nonzero elements in $GF(2^m)$. Let $\boldsymbol{\phi}_1(X)$ and $\boldsymbol{\phi}_2(X)$ be the minimal polynomials of $\beta_1$ and $\beta_2$, respectively. Is there a cyclic code with $\mathbf{g}(X) = \boldsymbol{\phi}_1(X) \cdot \boldsymbol{\phi}_2(X)$ as the generator polynomial? If your answer is yes, find the shortest cyclic code with $\mathbf{g}(X) = \boldsymbol{\phi}_1(X) \cdot \boldsymbol{\phi}_2(X)$ as the generator polynomial.

**5.19** Consider the Galois field $GF(2^m)$, which is constructed based on the primitive polynomial $\mathbf{p}(X)$ of degree $m$. Let $\alpha$ be a primitive element of $GF(2^m)$ whose minimal polynomial is $\mathbf{p}(X)$. Show that every code polynomial in the Hamming code generated by $\mathbf{p}(X)$ has $\alpha$ and its conjugates as roots. Show that any binary polynomial of degree $2^m - 2$ or less that has $\alpha$ as a root is a code polynomial in the Hamming code generated by $\mathbf{p}(X)$.

**5.20** Let $C_1$ and $C_2$ be two cyclic codes of length $n$ that are generated by $\mathbf{g}_1(X)$ and $\mathbf{g}_2(X)$, respectively. Show that the code polynomials common to both $C_1$ and $C_2$ also form a cyclic code $C_3$. Determine the generator polynomial of $C_3$. If $d_1$ and $d_2$ are the minimum distances of $C_1$ and $C_2$, respectively, what can you say about the minimum distance of $C_3$?

**5.21** Show that the probability of an undetected error for the distance-4 cyclic Hamming codes is upper bounded by $2^{-(m+1)}$.

**5.22** Let $C$ be a $(2^m-1, 2^m-m-1)$ Hamming code generated by a primitive polynomial $\mathbf{p}(X)$ of degree $m$. Let $C_d$ be the dual code of $C$. Then, $C_d$ is a $(2^m-1, m)$ cyclic code generated by

$$\mathbf{h}^*(X) = X^{2^m-m-1}\mathbf{h}(X^{-1}).$$

where

$$\mathbb{h}(X) = \frac{X^{2^m-1}+1}{\mathbb{p}(X)}.$$

a. Let $\mathbb{v}(X)$ be a codeword in $C_d$ and let $\mathbb{v}^{(i)}(X)$ be the $i$th cyclic shift of $\mathbb{v}(X)$. Show that for $1 \leq i \leq 2^m - 2$, $\mathbb{v}^{(i)}(X) \neq \mathbb{v}(X)$.

b. Show that $C_d$ contains the all-zero codeword and $2^m - 1$ codewords of weight $2^{m-1}$.

(*Hint:* For part (a), use (5.1) and the fact that the smallest integer $n$ such that $X^n + 1$ is divisible by $\mathbb{p}(X)$ is $2^m - 1$. For part (b), use the result of Problem 3.6(b).)

5.23 For an $(n, k)$ cyclic code, show that the syndrome of an end-around burst of length $n - k$ cannot be zero.

5.24 Design a Meggitt decoder that decodes a received polynomial $\mathbb{r}(X) = r_0 + r_1 X + \cdots + r_{n-1}X^{n-1}$ from the lowest-order received digit $r_0$ to the highest-order received digit $r_{n-1}$. Describe the decoding operation and the syndrome modification after each correction.

5.25 Consider the (15, 5) cyclic code generated by the following polynomial:

$$\mathbb{g}(X) = 1 + X + X^2 + X^4 + X^5 + X^8 + X^{10}.$$

This code has been proved to be capable of correcting any combination of three or fewer errors. Suppose that this code is to be decoded by the simple error-trapping decoding scheme.

a. Show that all the double errors can be trapped.

b. Can all the error patterns of three errors be trapped? If not, how many error patterns of three errors cannot be trapped?

c. Devise a simple error-trapping decoder for this code.

5.26 a. Devise a simple error-trapping decoder for the (23, 12) Golay code.

b. How many error patterns of double errors cannot be trapped?

c. How many error patterns of three errors cannot be trapped?

5.27 Suppose that the (23, 12) Golay code is used only for error correction on a BSC with transition probability $p$. If Kasami's decoder of Figure 5.18 is used for decoding this code, what is the probability of a decoding error? (*Hint:* Use the fact that the (23, 12) Golay code is a perfect code.)

5.28 Use the decoder of Figure 5.18 to decode the following received polynomials:

a. $\mathbb{r}(X) = X^5 + X^{19}$

b. $\mathbb{r}(X) = X^4 + X^{11} + X^{21}$

At each step in the decoding process, write down the contents of the syndrome register.

5.29 Consider the following binary polynomial:

$$\mathbb{g}(X) = (X^3 + 1)\mathbb{p}(X),$$

where $(X^3+1)$ and $\mathbb{p}(X)$ are relatively prime, and $\mathbb{p}(X)$ is an irreducible polynomial of degree $m$ with $m \geq 3$. Let $n$ be the smallest integer such that $\mathbb{g}(X)$ divides $X^n + 1$. Thus, $\mathbb{g}(X)$ generates a cyclic code of length $n$.

**a.** Show that this code is capable of correcting all the single-error, double-adjacent-error, and triple-adjacent-error patterns. (*Hint:* Show that these error patterns can be used as coset leaders of a standard array for the code.)

**b.** Devise an error-trapping decoder for this code. The decoder must be capable of correcting all the single-error, double-adjacent-error, and triple-adjacent-error patterns. Design a combinational logic circuit whose output is 1 when the errors are trapped in the appropriate stages of the syndrome register.

**c.** Suppose that $\mathbf{p}(X) = 1 + X + X^4$, which is a primitive polynomial of degree 4. Determine the smallest integer $n$ such that $\mathbf{g}(X) = (X^3 + 1)\mathbf{p}(X)$ divides $X^n + 1$.

**5.30** Let $C_1$ be the (3, 1) cyclic code generated by $\mathbf{g}_1(X) = 1 + X + X^2$, and let $C_2$ be the (7, 3) maximum-length code generated by $\mathbf{g}_2(X) = 1 + X + X^2 + X^4$. Find the generator and parity polynomials of the cyclic product of $C_1$ and $C_2$. What is the minimum distance of this product code? Discuss its error-correcting capability.

**5.31** Devise an encoding circuit for the (15, 5) quasi-cyclic code given in Example 5.14.

## BIBLIOGRAPHY


1. E. Prange, "Cyclic Error-Correcting Codes in Two Symbols," *AFCRC-TN-57, 103*, Air Force Cambridge Research Center, Cambridge, Mass., September 1957.

2. E. R. Berlekamp, *Algebraic Coding Theory*, McGraw-Hill, New York, 1968. (Rev. ed., Aegean Park Press, Laguna Hills, Calif., 1984.)

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

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

5. R. E. Blahut, *Theory and Practice of Error Control Codes,* Addison-Wesley, Reading, Mass., 1984.

6. R. J. McEliece, *The Theory of Information and Coding*, Addison-Wesley, Reading, Mass., 1977.

7. G. Clark and J. Cain, *Error-Correction Codes for Digital Communications,* Plenum, New York, 1981.

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

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

10. W. W. Peterson, "Encoding and Error-Correction Procedures for the Bose–Chaudhuri Codes," *IRE Trans. Inform. Theory*, IT-6, 459–70, September 1960.

11. J. E. Meggitt, "Error Correcting Codes and Their Implementation," *IRE Trans. Inform. Theory*, IT-7: 232–44, October 1961.

12. T. Kasami, "A Decoding Method for Multiple-Error-Correcting Cyclic Codes by Using Threshold Logics," *Conf. Rec. Inform. Process. Soc. Jap.* (in Japanese), Tokyo, November 1961.

13. M. E. Mitchell et al., "Coding and Decoding Operation Research," *G. E. Advanced Electronics Final Report on Contract AF 19 (604)-6183*, Air Force Cambridge Research Labs., Cambridge, Mass., 1961.

14. M. E. Mitchell, "Error-Trap Decoding of Cyclic Codes," *G. E. Report No. 62MCD3,* General Electric Military Communications Dept., Oklahoma City, December 1962.

15. E. Prange, "The Use of Information Sets in Decoding Cyclic Codes," *IEEE Trans. Inform. Theory*, IT-8: 85–80, September 1962.

16. L. Rudolph, "Easily Implemented Error-Correction Encoding-Decoding," *G. E. Report No. 62MCD2*, General Electric Corporation, Oklahoma City, December 1962.

17. T. Kasami, "A Decoding Procedure For Multiple-Error-Correction Cyclic Codes," *IEEE Trans. Inform. Theory*, IT-10: 134–39, April 1964.

18. F. J. MacWilliams, "Permutation Decoding of Systematic Codes," *Bell Syst. Tech. J.*, 43 (p. 1): 485–505, January 1964.

19. L. Rudolph and M. E. Mitchell, "Implementation of Decoders for Cyclic Codes," *IEEE Trans. Inform. Theory*, IT-10: 259–60, July 1964.

20. D. C. Foata, "On a Program for Ray-Chaudhuri's Algorithm for a Minimum Cover of an Abstract Complex," *Commun. ACM*, 4: 504–6, November 1961.

21. I. B. Pyne and E. J. McCluskey, "The Reduction of Redundancy in Solving Prime Implicant Tables," *IRE Trans. Electron. Comput.*, EC-11: 473–82, August 1962.

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

23. E. J. Weldon, Jr., "A Comparison of an Interleaved Golay Code and a Three-Dimensional Product Code," *Final Report, USNELC Contract N0095368M5345*, San Diego, CA, August 1968.

24. S. K. Leung-Yan-Cheong, E. R. Barnes, and D. U. Friedman, "On Some Properties of the Undetected Error Probability of Linear Codes," *IEEE Trans. Inform. Theory*, IT-25 (1): 110–12, January 1979.

25. H. O. Burton and E. J. Weldon, Jr.,"Cyclic Product Codes," *IEEE Trans. Inform. Theory*, IT-11: 433–40, July 1965.

26. S. Lin and E. J. Weldon, Jr., "Further Results on Cyclic Product Codes," *IEEE Trans. Inform. Theory*, IT-16: 452–59, July 1970.

27. W. C. Gore, "Further Results on Product Codes," *IEEE Trans. Inform. Theory*, IT-16: 446–51, July 1970.

28. N. M. Abramson, "Cascade Decoding of Cyclic Product Codes," *IEEE Trans. Commun. Technol.*, COM-16: 398–402, 1968.