Chapter 11

ERROR CONTROL CODING

11.1 INTRODUCTION

In this chapter we treat the subject of designing codes for the reliable transmission of digital information over a noisy channel. Codes can either correct or merely detect errors, depending on the amount of redundancy contained in the code. Codes that can detect errors are called *error-detecting* codes, and codes that can correct errors are known as *error-correcting* codes. There are many different error control codes. These are classified into *block codes* and *convolutional codes*. The codes described here are binary codes for which the alphabet consists of only two elements: 0 and 1. The set $\{0, 1\}$ is denoted by K.

11.2 CHANNEL CODING

A. Channel Coding:

A basic block diagram for the channel coding is shown in Fig. 11-1. The binary message sequence at the input of the channel encoder may be the output of a source encoder or the output of a source directly. The channel encoder introduces systematic redundancy into the data stream by adding bits to the message bits in such a way as to facilitate the detection and correction of bit errors in the original binary message sequence at the receiver. The channel decoder in the receiver exploits the redundancy to decide which message bits are actually transmitted. The combined objective of the channel encoder and decoder is to minimize the effect of channel noise.

B. Channel Coding Theorem:

The channel coding theorem for a DMC is stated as follows:

Given a DMS X with entropy $H(X)$ bits/symbol and a DMC with capacity C_s bits/symbol, if $H(X) \leqslant C_s$, there exists a coding scheme for which the source output can be transmitted over the channel with an arbitrarily small probability of error.

Conversely, if $H(X) > C_s$, it is not possible to transmit information over the channel with an arbitrarily small probability of error.

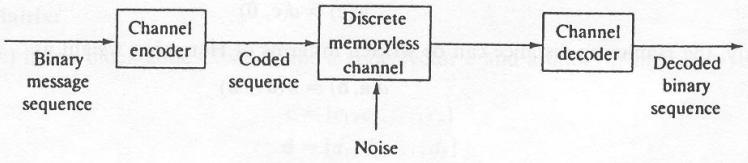

Fig. 11-1 Channel coding

Note that the channel coding theorem only asserts the existence of codes but it does not tell us how to construct these codes.

11.3 BLOCK CODES

A block code is a code having all its words of the same length. In block codes the binary message or data sequence is divided into sequential blocks each k bits long, and each k-bit block is converted into an n-bit block, where $n > k$. The resultant block code is called an (n, k) block code. The k-bit blocks form 2^k distinct message sequences referred to as k-tuples. The n-bit blocks can form as many as 2^n distinct sequences referred to as n-tuples. The set of all n-tuples defined over $K = \{0, 1\}$ is denoted by K^n. The channel encoder performs a mapping

$$T: U \rightarrow V \tag{11.1}$$

where U is a set of binary data words of length k and V is a set of binary code words of length n with $n > k$. Each of the 2^k data words is mapped to a unique code word. The ratio k/n is called the *code rate*.

11.4 LINEAR BLOCK CODES

A. Binary Field:

The set $K = \{0, 1\}$ is a *binary field*. The binary field has two operations, addition and multiplication such that the results of all operations are in K. The rules of addition and multiplication are as follows:

Addition:

$$0 \oplus 0 = 0 \quad 1 \oplus 1 = 0 \quad 0 \oplus 1 = 1 \oplus 0 = 1$$

Multiplication:

$$0 \cdot 0 = 0 \quad 1 \cdot 1 = 1 \quad 0 \cdot 1 = 1 \cdot 0 = 0$$

B. Linear Codes:

Let $\mathbf{a} = (a_1, a_2, \ldots, a_n)$, and $\mathbf{b} = (b_1, b_2, \ldots, b_n)$ be two code words in a code C. The sum of $\mathbf{a}$ and $\mathbf{b}$, denoted by $\mathbf{a} \oplus \mathbf{b}$, is defined by $(a_1 \oplus b_1, a_2 \oplus b_2, \ldots, a_n \oplus b_n)$. A code C is called *linear* if the sum of two code words is also a code word in C. A linear code C must contain the zero code word $\mathbf{0} = (0, 0, \ldots, 0)$, since $\mathbf{a} \oplus \mathbf{a} = \mathbf{0}$.

C. Hamming Weight and Distance:

Let $\mathbf{c}$ be a code word of length n. The *Hamming weight* of $\mathbf{c}$, denoted by $w(\mathbf{c})$, is the number of 1's in $\mathbf{c}$.

Let $\mathbf{a}$ and $\mathbf{b}$ be code words of length n. The *Hamming distance* between $\mathbf{a}$ and $\mathbf{b}$, denoted by $d(\mathbf{a}, \mathbf{b})$, is the number of positions in which $\mathbf{a}$ and $\mathbf{b}$ differ. Thus, the Hamming weight of a code word $\mathbf{c}$ is the Hamming distance between $\mathbf{c}$ and $\mathbf{0}$, that is

$$w(\mathbf{c}) = d(\mathbf{c}, \mathbf{0}) \tag{11.2}$$

Similarly, the Hamming distance can be written in terms of Hamming weight as

$$d(\mathbf{a}, \mathbf{b}) = w(\mathbf{a} \oplus \mathbf{b}) \tag{11.3}$$

D. Minimum Distance:

The minimum distance $d_{\min}$ of a linear code C is defined as the smallest Hamming distance between any pair of code words in C.

From the closure property of linear codes—that is, the sum (modulo 2) of two code words is also a code word—we can derive the following theorem (Problem 11.9).

THEOREM 11.1

The minimum distance $d_{\min}$ of a linear code C is the smallest Hamming weight of the nonzero code word in the C.

E. Error Detection and Correction Capabilities:

The minimum distance $d_{\min}$ of a linear code C is an important parameter of C. It determines the error detection and correction capabilities of C. This is stated in the following theorems.

THEOREM 11.2

A linear code C of minimum distance $d_{\min}$ can detect up to t errors if and only if

$$d_{\min} \geqslant t + 1 \tag{11.4}$$

THEOREM 11.3

A linear code C of minimum distance $d_{\min}$ can correct up to t errors if and only if

$$d_{\min} \geqslant 2t + 1 \tag{11.5}$$

Equation (*11.5*) can be illustrated geometrically by Fig. 11-2. In Fig. 11-2 two Hamming spheres, each of radius t, are constructed around the points that represent code words $\mathbf{c}_i$ and $\mathbf{c}_j$. Figure 11-2(*a*) depicts the case where two spheres are disjoint, that is, $d(\mathbf{c}_i, \mathbf{c}_j) \geqslant 2t + 1$. For this case, if the code word $\mathbf{c}_i$ is transmitted, the received word is $\mathbf{r}$, and $d(\mathbf{c}_i, \mathbf{r}) \leqslant t$, then it is clear that the decoder will choose $\mathbf{c}_i$, since it is the code word closest to the received word $\mathbf{r}$. On the other hand, Fig. 11-2(*b*) depicts the case where the two spheres intersect, that is, $d(\mathbf{c}_i, \mathbf{c}_j) < 2t$. In this case, we see that if the code word $\mathbf{c}_i$ is transmitted, there exists a received word $\mathbf{r}$ such that $d(\mathbf{c}_i, \mathbf{r}) \leqslant t$, and yet $\mathbf{r}$ is as close to $\mathbf{c}_j$ as it is to $\mathbf{c}_i$. Thus, the decoder may choose $\mathbf{c}_j$, which is incorrect.

(*a*) $d(\mathbf{c}_i, \mathbf{c}_j) \geq 2t + 1$ (*b*) $d(\mathbf{c}_i, \mathbf{c}_j) \geq 2t$

Fig. 11-2 Hamming distance

F. Generator Matrix:

In an (n, k) linear block code C, we define a code vector $\mathbf{c}$ and a data (or message) vector $\mathbf{d}$ as follows:

$$\mathbf{c} = [c_1, c_2, \ldots, c_n]$$
$$\mathbf{d} = [d_1, d_2, \ldots, d_k]$$

If the data bits appear in specified location of $\mathbf{c}$, then the code C is called *systematic*. Otherwise, it is called nonsystematic. Here we assume that the first k bits of $\mathbf{c}$ are the data bits and the last $(n-k)$ bits are the parity-check bits formed by linear combination of data bits, that is,

$$\begin{aligned} c_1 &= d_1 \\ c_2 &= d_2 \\ &\vdots \\ c_{k+1} &= p_{11}d_1 \oplus p_{12}d_2 \oplus \cdots \oplus p_{1k}d_k \\ c_{k+2} &= p_{21}d_1 \oplus p_{22}d_2 \oplus \cdots \oplus p_{2k}d_k \\ &\vdots \\ c_{k+m} &= p_{m1}d_1 \oplus p_{m2}d_2 \oplus \cdots \oplus p_{mk}d_k \end{aligned} \tag{11.6}$$

where $m = n - k$. Equation (*11.6*) can be written in a matrix form as

$$\mathbf{c} = \mathbf{d}G = [d_1 \quad d_2 \quad \cdots \quad d_k] \begin{bmatrix} 1 & 0 & \cdots & 0 & p_{11} & p_{21} & \cdots & p_{m1} \\ 0 & 1 & \cdots & 0 & p_{12} & p_{22} & \cdots & p_{m2} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 0 & 0 & \cdots & 1 & p_{1k} & p_{2k} & \cdots & p_{mk} \end{bmatrix} \tag{11.7}$$

where

$$G = [I_k \, P^T] \tag{11.8}$$

where I_k is the kth-order identity matrix and P^T is the transpose of the matrix P given by

$$P = \begin{bmatrix} p_{11} & p_{12} & \cdots & p_{1k} \\ p_{21} & p_{22} & \cdots & p_{2k} \\ \cdots & \cdots & \cdots & \cdots \\ p_{m1} & p_{m2} & \cdots & p_{mk} \end{bmatrix} \tag{11.9}$$

The $k \times n$ matrix G is called the *generator matrix*. Note that a generator matrix for C must have k rows and n columns, and it must have rank k; that is, the k rows of G are linearly independent.

G. Parity-Check Matrix:

Let H denote an $m \times n$ matrix defined by

$$H = [P \quad I_m] \tag{11.10}$$

where $m = n - k$ and I_m is the mth-order identity matrix. Then

$$H^T = \begin{bmatrix} P^T \\ I_m \end{bmatrix} \tag{11.11}$$

Using Eqs. (*11.8*) and (*11.11*), we have

$$GH^T = [I_k \quad P^T] \begin{bmatrix} P^T \\ I_m \end{bmatrix} = P^T \oplus P^T = O \tag{11.12}$$

where O denotes the $k \times m$ zero matrix. Now by Eqs. (*11.7*) and (*11.12*), we have

$$\mathbf{c}H^T = \mathbf{d}GH^T = \mathbf{0} \tag{11.13}$$

where $\mathbf{0}$ denotes the $1 \times m$ zero vector.

The matrix H is called the *parity-check matrix* of C. Note that the rank of H is $m = n - k$ and the rows of H are linearly independent. The minimum distance $d_{\min}$ of a linear block code C is closely related to the structure of the parity-check matrix H of C. This is stated in the following theorem (Prob. 11.22).

THEOREM 11.4

The minimum distance $d_{\min}$ of a linear block code C is equal to the minimum number of rows of H^T that sum to $\mathbf{0}$, where H^T is the transpose of the parity-check matrix H of C.

H. Syndrome Decoding:

Let $\mathbf{r}$ denote the received word of length n when code word $\mathbf{c}$ of length n was sent over a noisy channel. Then $\mathbf{r} = \mathbf{c} \oplus \mathbf{e}$, where $\mathbf{e}$ is called the error pattern. Note that $\mathbf{e} = \mathbf{r} + \mathbf{c}$.

Consider first the case of a single error in the ith position. Then we can represent $\mathbf{e}$ by

$$\mathbf{e} = [0 \cdots 010 \cdots 0] \qquad (11.14)$$
$$\uparrow$$
$$i\text{th position}$$

Next, we evaluate $\mathbf{r}H^T$ and obtain

$$\mathbf{r}H^T = (\mathbf{c} \oplus \mathbf{e})H^T = \mathbf{c}H^T \oplus \mathbf{e}H^T = \mathbf{e}H^T = \mathbf{s} \qquad (11.15)$$

in view of Eq. (*11.13*) and $\mathbf{s}$ is called the *syndrome* of $\mathbf{r}$.

Thus, using $\mathbf{s}$ and noting that $\mathbf{e}H^T$ is the ith row of H^T, we can identify the error position by comparing $\mathbf{s}$ to the rows of H^T. Decoding by this simple comparison method is called *syndrome decoding*. Note that not all error patterns can be correctly decoded by syndrome decoding. The zero syndrome indicates that $\mathbf{r}$ is a code word and is presumably correct.

With syndrome decoding, an (n, k) linear block code can correct up to t errors per codeword if n and k satisfy the following *Hamming bound:*

$$2^{n-k} \geqslant \sum_{i=0}^{t} \binom{n}{i} \qquad (11.16)$$

where $$\binom{n}{i} = \frac{n!}{(n-1)!i!}$$

A block code for which the equality holds for Eq. (*11.16*) is known as the *perfect code*. Single error-correcting perfect codes are called *Hamming codes*.

Note that the Hamming bound is necessary but not sufficient for the construction of a t-error correcting linear block code.

11.5. CYCLIC CODES

A. Definition:

Let $\mathbf{c} = (c_0, c_1, \ldots, c_{n-1})$ be a codeword in code C. The *cyclic shift* $\sigma(\mathbf{c})$ of $\mathbf{c}$ is defined by

$$\sigma(\mathbf{c}) = \mathbf{c}^{(1)} = (c_{n-1}, c_0, c_1, \ldots, c_{n-2}) \qquad (11.17a)$$

and a second cyclic shift produces

$$\sigma^2(\mathbf{c}) = \sigma\{\sigma(\mathbf{c})\} = \mathbf{c}^{(2)} = (c_{n-2}, c_{n-1}, c_0, \ldots, c_{n-3}) \qquad (11.17b)$$

Note that $\sigma^n(\mathbf{c}) = \mathbf{c}$.

A code C is said to be a *cyclic code* if the cyclic shift of each code word is also a code word. Cyclic codes are a subclass of linear block code.

B. Code Polynomials:

We define a code polynomial $c(x)$ corresponding to $\mathbf{c}$ as

$$c(x) = c_0 + c_1 x + \cdots + c_{n-1}x^{n-1} \tag{11.18}$$

Note that $c(x)$ is a polynomial of degree $(n-1)$ over $K = \{0, 1\}$, where coefficients $c_0, c_1, \ldots, c_{n-1}$ are elements of K. Thus, a code C of length n can be represented as a set of polynomials over K of degree at most $n-1$. The polynomials over K are added and multiplied in the usual fashion, except that since $1 \oplus 1 = 0$, we have $x^k \oplus x^k = 0$. The set of all polynomials over K is denoted by $K[x]$.

Let $f(x)$ and $h(x)$ be in $K[x]$ with $h(x) \neq 0$. Then there exists unique polynomials $q(x)$ and $r(x)$ in $K[x]$ such that

$$f(x) = q(x)h(x) + r(x) \tag{11.19}$$

with $r(x) = 0$ or degree$[r(x)] <$ degree $[h(x)]$. The polynomial $q(x)$ is called the *quotient*, and $r(x)$ is called the *remainder*. The procedure for finding $q(x)$ and $r(x)$ is the familiar long-division process with modulo 2 arithmetic among the coefficients.

We say that $f(x)$ modulo $h(x)$ is $r(x)$ if $r(x)$ is the remainder when $f(x)$ is divided by $h(x)$, and we write

$$r(x) = f(x) \mod h(x) \tag{11.20}$$

If $c(x) \bmod h(x) = r(x) = b(x) \bmod h(x)$, then we say that $c(x)$ and $b(x)$ are equivalent modulo $h(x)$ denoted by

$$c(x) = b(x) \mod h(x) \tag{11.21}$$

Note that

$$x^n = 1 \mod (1 + x^n) \tag{11.22}$$

C. Generator Polynomial:

The code polynomial $c^{(1)}(x)$ corresponding to $\sigma(\mathbf{c})$ [Eq. (*11.17a*)] is

$$c^{(1)}(x) = c_{n-1} + c_0 x + c_1 x^2 + \cdots + c_{n-2}x^{n-1} \tag{11.23}$$

Now from Eq. (*11.18*)

$$xc(x) = c_0 x + c_1 x^2 + \cdots + c_{n-1}x^n$$

Using Eq. (*11.22*), we have

$$c^{(1)}(x) = xc(x) \mod (1 + x^n) \tag{11.24}$$

In a similar fashion, we get

$$c^{(i)}(x) = x^i c(x) \mod (1 + x^n) \tag{11.25}$$

THEOREM 11.5

If $g(x)$ is a polynomial of degree $(n-k)$ that is a factor of $1 + x^n$, then $g(x)$ generates an (n, k) cyclic code C in which the code polynomial $c(x)$ for a data word $\mathbf{d} = (d_0, d_1, \ldots, d_{k-1})$ is generated by

$$c(x) = d(x)g(x) \tag{11.26}$$

where $d(x) = d_0 + d_1 x + \ldots + d_{k-1}x^{k-1}$ is the data polynomial corresponding to the data word.

The polynomial $g(x)$ is called the *generator polynomial* (Prob. 11.27). Note that the resultant cyclic code C is, in general, not systematic. To construct a systematic cyclic code, see Prob. 11.35.

D. Parity-Check Polynomial:

Let $h(x)$ be a polynomial of degree k and also a factor of $1 + x^n$. Then $h(x)$ is the parity-check polynomial of an (n, k) cyclic code C. The parity-check polynomial $h(x)$ and the generating polynomial $g(x)$ of C are related by

$$g(x)h(x) = 0 \mod (1 + x^n) \tag{11.27a}$$

or

$$g(x)h(x) = 1 + x^n \tag{11.27b}$$

E. Syndrome Polynomial:

Let $r(x)$ be the received word polynomial when the code word polynomial $c(x)$ was sent.

Then

$$r(x) = c(x) + e(x) \tag{11.28}$$

where $e(x)$ is the most likely error polynomial. The *syndrome polynomial* $s(x)$ is defined by

$$s(x) = r(x) \mod g(x) \tag{11.29}$$

Assuming $g(x)$ has degree $n - k$, then $s(x)$ will have degree less than $n - k$ and will correspond to a binary word $\mathbf{s}$ of length $n - k$. Since $c(x) = d(x)g(x)$, we have

$$s(x) = e(x) \mod g(x) \tag{11.30}$$

Thus, the syndrome polynomial is dependent only on the error. Note that if degree $[e(x)] <$ degree $[g(x)]$, then $e(x) = e(x) \bmod g(x)$ and thus $s(x) = e(x)$. The following theorem specifies the condition for the existence of a unique syndrome polynomial (Prob. 11.32).

THEOREM 11.6

Let C be a cyclic code with minimum distance $d_{\min}$. Every error polynomial of weight less than $\frac{1}{2}d_{\min}$ has a unique syndrome polynomial.

From the preceding, we state that the error-correction problem then is to find the unique $e(x)$ with the least weight satisfying Eq. (*11.30*). (Note that the weight of a polynomial is the number of nonzero coefficients in the polynomial.) This can be done as follows: For each correctable $e(x)$, compute and tabulate $s(x)$ as shown in Table 11.1. The table is called the *syndrome evaluator* table. A decoder finds the error polynomial $e(x)$ by computing $s(x)$ from $r(x)$ and then finding $s(x)$ in the syndrome evaluator table, thereby finding the corresponding $e(x)$ (see Prob. 11.34).

Table 11-1 Syndrome Evaluator Table

$e(x)$	$s(x)$
1	1 mod $g(x)$
x	x mod $g(x)$
x^2	x^2 mod $g(x)$
⋮	⋮
$1+x$	$1+x$ mod $g(x)$
$1+x^2$	$1+x^2$ mod $g(x)$
⋮	⋮

F. Implementation:

One of the advantages of cyclic codes is their easier implementation. The cyclic encoders can be implemented by shift registers. These devices consist of m registers (or delay elements) and a "clock" that controls the shifting of the data contained in the registers.

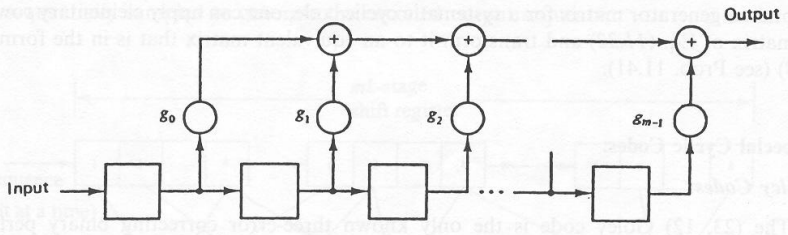

Fig. 11-3 m-stage shift register

An m-stage shift register, shown in Fig. 11-3, is a shift register with m registers. The output of an m-stage shift register is a linear combination of the contents of the registers and can be described using coefficients $g_0, g_1, \ldots, g_{m-1}$ with $g_i \in K = \{0, 1\}$. Let c_t be the output at time t, then

$$c_t = g_0 X_0(t) + g_1 X_1(t) + \cdots + g_{m-1} X_{m-1}(t) \qquad (11.31)$$

where $X_i(t)$ is the value of the contents of register X_i at time t.

Given a fixed generator polynomial $g(x)$ of degree $n - k$ for an (n, k) linear cyclic code C, we can build an $n - k + 1$ stage shift register with generator $g(x)$ to implement polynomial encoding of data polynomial $d(x)$.

Polynomial division (and thus polynomial decoding for cyclic codes) can be implemented by devices known as *feedback shift registers*. A feedback shift register is a shift register with the output fed back into the shift register as shown in Fig. 11-4.

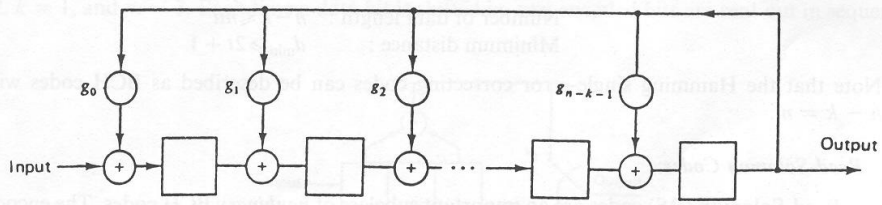

Fig. 11-4 Feedback shift register

Feeding $r(x) = r_0 + r_1 x + \cdots + r_{n-1} x^{n-1}$ into a feedback shift register with $g(x)$ with high-order coefficients fed in first (i.e. $r_{n-1}, r_{n-2}, \ldots, r_0$) is equivalent to dividing $r(x)$ by $g(x)$. The output after n clock tick will be the quotient (high-order coefficients first) and the contents of registers will be the remainder (high-order digits to the right).

G. Generator Matrix:

There exist many generator matrices G for linear cyclic codes. The simplest is the matrix in which the rows are the code words corresponding to the generator polynomial $g(x)$ and its first $k - 1$ cyclic shifts:

$$G = \begin{bmatrix} g(x) \\ xg(x) \\ \vdots \\ x^{k-1}g(x) \end{bmatrix} \tag{11.32}$$

Note that the resulting generator matrix G is, in general, not in the form shown in Eq. (*11.8*). In order to obtain a generator matrix for a systematic cyclic code, one can apply elementary row operations to the matrix of Eq. (*11.32*) and transform it to an equivalent matrix that is in the form shown in Eq. (*11.8*) (see Prob. 11.41).

H. Special Cyclic Codes:

1. Goley Codes:

The (23, 12) Goley code is the only known three-error correcting binary perfect code. The minimum distance of the code is 7. The (23, 12) Goley code is generated either by the polynomial

$$g_1(x) = 1 + x + x^5 + x^6 + x^7 + x^9 + x^{11} \tag{11.33}$$

or by the polynomial

$$g_2(x) = 1 + x^2 + x^4 + x^5 + x^6 + x^{10} + x^{11} \tag{11.34}$$

Note that

$$1 + x^{23} = (1 + x)g_1(x)g_2(x) \tag{11.35}$$

2. Bose-Chaudhuri-Hocqueqhem (BCH) Codes:

The BCH codes are the most efficient error-correcting cyclic codes known. The BCH codes offer flexibility in the choice of code parameters, that is, block length and code rate. The most common BCH codes are characterized as follows: For any positive integers m ($\geqslant 3$) and t, there exists a binary BCH code with the following parameters:

Block length : $n = 2^m - 1$
Number of data length : $n - k \leqslant mt$
Minimum distance : $d_{\min} \geqslant 2t + 1$

Note that the Hamming single-error correcting codes can be described as BCH codes with $t = 1$, $n - k = m$.

3. Reed-Solomon Codes:

Reed-Solomon (RS) codes are an important subclass of *nonbinary* BCH codes. The encoder for an RS code operates on multiple bits rather than individual bits. Reed-Solomon codes have particularly good distance properties and are useful in situations where errors tend to happen in "bursts" rather than randomly.

11.6. CONVOLUTIONAL CODES

Two major types of codes are in common use: block codes and convolutional codes. In an (n, k) block code the data (or information) sequence is grouped into k-bit block and coded into n-bit block codeword after adding $(n - k)$ parity check bits. A characteristic of a linear block code is that each n-bit encoded block is uniquely determined by the input data k-bit block. In contrast with this, the encoder of an (n, k, m) convolutional code also accepts k-bit blocks of the input sequence and produces n-bit blocks of the encoded sequence. However, each encoded n-bit block depends not only on the corresponding k-bit input block at the same time unit, but also on the previous $(m - 1)$ k-bit

input blocks. Thus, an important characteristic of convolutional codes, different from block codes, is that the encoder has memory. The integer m is a parameter known as the *constraint length* of the convolutional code. In practice, n and k are small integers and m is varied to control the redundancy.

A general encoder for an (n, k, m) convolutional code is shown in Fig. 11-5. It consists of an mk-stage shift register and n modulo-2 adders. At each unit of time, k bits are shifted into the first k-stage of the register, all bits in the register are shifted k-stage to the right, and the outputs of the n modulo-2 adders are sequentially sampled to generate the encoded output sequence.

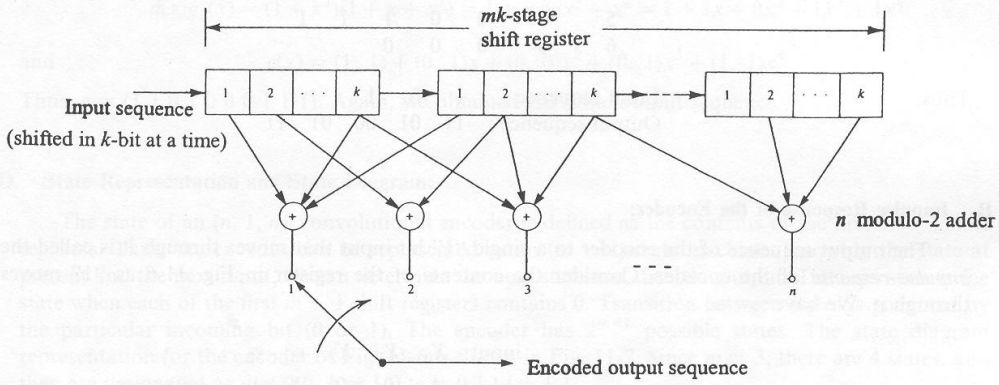

Fig. 11-5 An (n,k,m) convolutional encoder

In the important special case when $k = 1$, the data sequence is not divided into blocks and can be processed continuously. In the following, only this special case of $k = 1$ will be treated.

Convolutional codes are very practical codes. Several methods are used for representing a convolutional encoder: the connection diagram, connection polynomials, the state diagram, the tree diagram, and the trellis diagram. Figure 11-6 shows a simple (2, 1, 3) convolutional encoder with $n = 2$, $k = 1$, and $m = 3$. Each time a data bit is shifted in, two encoded bits are sent out in sequence.

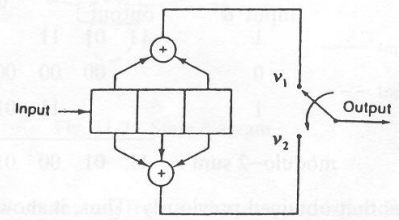

Fig. 11-6 A (2,1,3) convolutional encoder

In the following we shall use this encoder to describe various representations of convolutional codes.

A. Connection Diagram:

Letting the content in each of the three stages of Fig. 11-6 be X_0, X_1, X_2, with the output coded bits v_1, v_2, we have

$$v_1 = X_0 \oplus X_2 \qquad v_2 = X_0 \oplus X_1 \oplus X_2$$

This coding is carried out continuously, on all data bits as they are shifted through the register. Consider the input sequence $\mathbf{d} = (1\ 0\ 1)$. Then from the connection diagram of Fig. 11-6, we have the following results:

time	input	X_0	X_1	X_2	v_1	v_2
1	1	1	0	0	1	1
2	0	0	1	0	0	1
3	1	1	0	1	0	0
4	0	0	1	0	0	1
5	0	0	0	1	1	1
6	0	0	0	0		

Thus,

Input sequence :	1	0	1		
Output sequence :	11	01	00	01	11

B. Impulse Response of the Encoder:

The output sequence of the encoder to a single "1" bit input that moves through it is called the *impulse response* of the encoder. Consider the contents of the register in Fig. 11-6 as "1" moves through it. We have

input	X_0	X_1	X_2	v_1	v_2
1	1	0	0	1	1
0	0	1	0	0	1
0	0	0	1	1	1

Input sequence:	1	0	0
Impulse response:	11	01	11

Note that the impulse response of v_1 is (1 0 1) and the impulse response of v_2 is (1 1 1). For the input sequence $\mathbf{d} = (1\ 0\ 1)$, the output may be found by the linear addition of the time-shifted impulse responses as follows:

input **d**	output				
1	11	01	11		
0		00	00	00	
1			11	01	11
modulo−2 sum :	11	01	00	01	11

This is the same output as that obtained previously. Thus, it shows that the convolutional codes are linear. Note that the output sequence of v_1 (1 0 0 0 1) is the discrete convolution sum (mod 2) of (1 0 1) and its impulse response (1 0 1). Similarly, the output sequence of v_2 (1 1 0 1 1) is the discrete convolution sum (mod 2) of (1 0 1) and its impulse response (1 1 1). This is the reason for the name "convolutional."

C. Polynomial Representation:

Comparing the structure of Fig. 11-6 with that of Fig. 11-3, we see that we can write the generator polynomial $g_1(x)$ for the upper connection and $g_2(x)$ for the lower connection as

$$g_1(x) = 1 + x^2$$
$$g_2(x) = 1 + x + x^2$$

The output sequence is then

$$c(x) = d(x)g_1(x) \text{ interlaced with } d(x)g_2(x).$$

For $\mathbf{d} = (1\ 0\ 1)$, we have $d(x) = 1 + x^2$. Now

$$d(x)g_1(x) = (1 + x^2)(1 + x^2) = 1 + x^4 = 1 + 0x + 0x^2 + 0x^3 + 1x^4$$
$$d(x)g_2(x) = (1 + x^2)(1 + x + x^2) = 1 + x + x^3 + x^4 = 1 + 1x + 0x^2 + 1x^3 + 1x^4$$

and
$$c(x) = (1,\ 1) + (0,\ 1)x + (0,\ 0)x^2 + (0,\ 1)x^3 + (1,\ 1)x^4$$

Thus, $\mathbf{c} = (1\ 1\ 0\ 1\ 0\ 0\ 0\ 1\ 1\ 1)$. Again, we obtained the same output sequence.

D. State Representation and State Diagram:

The state of an $(n, 1, m)$ convolutional encoder is defined as the contents of the first $m - 1$ shift registers. Thus, the encoder can be represented as an $(m - 1)$-state machine. Knowing the state at present and the next input, we can determine the next state and then the output. The zero state is the state when each of the first $m - 1$ shift registers contains 0. Transition between states is governed by the particular incoming bit (0 or 1). The encoder has 2^{m-1} possible states. The state diagram representation for the encoder of Fig. 11-6 is shown in Fig. 11-7. Since $m = 3$, there are 4 states, and they are designated as $a = 0\ 0$, $b = 1\ 0$, $c = 0\ 1$, $d = 1\ 1$.

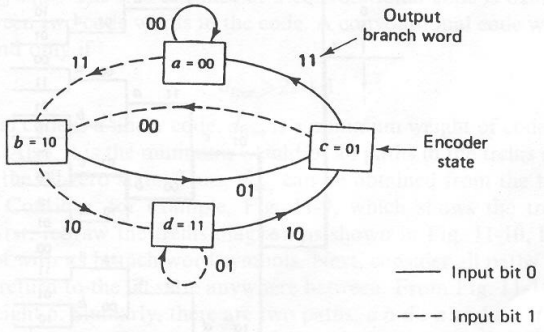

Fig. 11-7 State diagram

There are only two transitions emanating from each state, corresponding to the two possible input bits (0 or 1). Arrows indicate the direction of transition. A transition from one state to another in response to input 0 is represented by a solid line, whereas a transition in response to input 1 is represented as a dashed line. Bits appearing next to each path between states are the output bits associated with the state transition. Assuming that the encoder is initially in state a (the all-zero state), the codeword corresponding to any given input sequence can be obtained by following the path through the state diagram determined by the input sequence and noting the corresponding outputs on the branch labels. Following the last nonzero input sequence, the encoder is returned to state a by a sequence of $m - 1$ all-zero blocks appended to the input data sequence. For example, in Fig. 11-7, if input sequence is data word (1 0 1) followed by (0 0), the path is $a\ b\ c\ b\ c\ a$ and then the output codeword is (1 1 0 1 0 0 0 1 1 1).

E. Tree Diagram:

The tree diagram adds the dimension of time to the state diagram. The tree diagram for the convolutional encoder of Fig. 11-6 is illustrated in Fig. 11-8. A vertical line is called a *node* and a horizontal line is called a *branch*, and the output codeword for each input bit is shown on branches. The rule for the tree structure is as follows: Upward transitions correspond to a 0 data bit at the input; downward transitions correspond to a 1. Following this rule, we see that the input sequence 1 0 1 0 0 traces the heavy line drawn in the tree diagram in Fig. 11-8. This path corresponds to the following output code word sequence: 1 1 0 1 0 0 0 1 1 1. It is further seen that the tree structure is repetitive beyond the third node level; this is because the encoder output depends only on the two previous inputs and the current input. Note that the number of possible sequences (paths) of the transition bits increases exponentially with time.

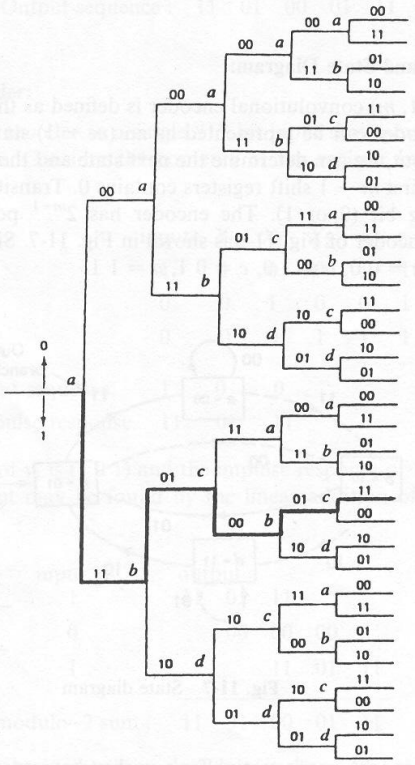

Fig. 11-8 Tree diagram

F. The Trellis Diagram:

The repetitive structure of the tree diagram leads to the redrawing of the tree diagram as a trellis diagram, shown in Fig. 11-9. The four states of this encoder appear along the vertical axis; transition between the states is represented in time along the horizontal axis. In drawing the trellis diagram, we use the same convention for the state diagram; that is, a solid line denotes the output generated by input bit 0, and a dashed line denotes the output generated by input bit 1. The trellis in this example demonstrates the repetitive nature of the structure after trellis depth $m(=3)$(at time t_4).

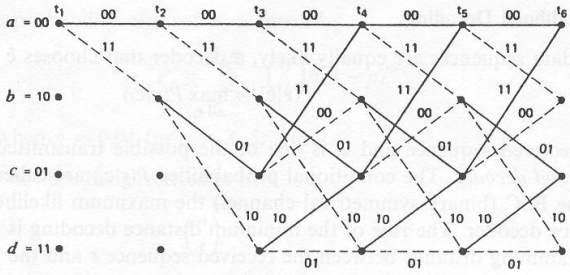

Fig. 11-9 Trellis diagram

The completely repetitive structure of the trellis diagram suggests a further reduction in the representation of the encoder to the state diagram of Fig. 11-7. Thus, the tree, the trellis, and the state diagram all represent the input-output relation of the convolutional encoder and are used for decoding coded sequence by tracing the most likely path that has been traversed while the code was generated. Specially, the trellis diagram is the most useful for probabilistic decoding.

11.7. DECODING OF CONVOLUTIONAL CODES

A. Distance Properties of Convolutional Codes:

As in block codes, the distance property of convolutional codes is also important in evaluating their error-correcting capabilities. The most important distance measure for convolutional codes is the free distance, denoted by d_{free}. The free distance of a convolutional code is defined as the minimum Hamming distance between two code words in the code. A convolutional code with free distance d_{free} can correct t errors if and only if

$$d_{\text{free}} \geq 2t + 1 \tag{11.36}$$

Since a convolutional code is a linear code, d_{free} is a minimum weight of code words generated by nonzero data sequences. Also, it is the minimum weight of all paths in the trellis diagram that diverge from and remerge with the all-zero state. Thus, d_{free} can be obtained from the trellis diagram of the convolutional encoder. Consider, for example, Fig. 11-9, which shows the trellis diagram of the encoder of Fig. 11-6. First, redraw the trellis diagram as shown in Fig. 11-10, labeling each branch with its weight instead of with its branch word symbols. Next, consider all paths that start and end in the 00 state and do not return to the 00 state anywhere between. From Fig. 11-10 we see that there is one path $a\ b\ c\ a$ with weight 5. Similarly, there are two paths, $a\ b\ d\ c\ a$ and $a\ b\ d\ d\ c\ a$, with weight 6, and so on. Thus, the free distance of the convolutional code is seen to be 5 in this example.

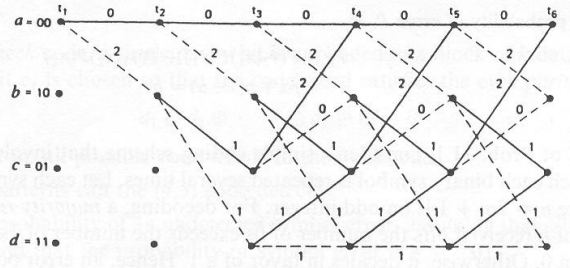

Fig. 11-10

B. Maximum Likelihood Decoding:

If all input data sequences are equally likely, a decoder that chooses $\hat{\mathbf{c}}$ if

$$P(\mathbf{r}|\hat{\mathbf{c}}) = \max_{\text{all } \mathbf{c}_i} P(\mathbf{r}|\mathbf{c}_i) \tag{11.37}$$

where $\mathbf{r}$ is the received sequence and $\mathbf{c}_i$ is one of the possible transmitted sequences, is called the *maximum likelihood decoder*. The conditional probabilities $P(\mathbf{r}|\mathbf{c}_i)$ are called the *likelihood functions*. Note that for the BSC (binary symmetrical channel) the maximum likelihood decoder reduces to a minimum distance decoder. The rule of the minimum distance decoding is as follows: Choose $\hat{\mathbf{c}}$ that minimizes the Hamming distance between the received sequence $\mathbf{r}$ and the transmitted sequence $\mathbf{c}$.

C. The Viterbi Decoding Algorithm:

Maximum likelihood decoding involves searching the entire code space and generally is impractical because of the large associated computational problem. However, a decoding algorithm due to Viterbi provides a maximum likelihood decoding procedure that is practical to use with short constraint length convolutional codes. We may decode a convolutional code by choosing a path in the trellis diagram such that the code sequence corresponding to the chosen path is at minimum distance from the received sequence. The Viterbi decoding algorithm essentially performs minimum Hamming distance decoding; however, it reduces the computational load by taking advantage of the special structure in the code trellis. From the trellis diagram of Fig. 11-9, we note that each of the four states (*a, b, c,* and *d*) can be reached through two states only. Thus, only the path that agrees most with the received sequence $\mathbf{r}$ (the minimum distance path) need be retained for each state. The retained path is called the *survivor* at that state. A Viterbi decoder assigns to each branch of each surviving path a metric that equals its Hamming distance from the corresponding branch of $\mathbf{r}$. Summing the branch metric yields the path metric, and $\mathbf{r}$ is finally decoded as the surviving path with the smallest metric (see Prob. 11.50).