quired to transmit a binary symbol over the channel is proportional to the symbol rate. In an uncoded system, the symbol rate is equal to the information bit rate. Thus, a bandwidth expansion is required by a coded system. This is one of the costs of using error correction.

6.3 FUNDAMENTALS OF BLOCK CODING

The promise of information theory is that message error probability can be made to approach zero by increasing waveform complexity at a constant E_b/N_0. This promise is valid as long as the total information transmission rate is below the capacity of the channel. With the information theoretic bounds in hand, communications technologists began searching for specific techniques to reduce message error rate without increasing transmitter power or similarly to achieve the same message error probability with reduced transmitter power. Their goal was to search for waveforms which have the structure necessary to mitigate channel distortion and which can be received using a receiver with reasonable complexity. The field of coding theory is dedicated to the search for efficient methods of structuring the transmitted waveform.

When there is no possibility of feedback from the receiver to the transmitter, *forward error control* techniques are used. In this case the waveform is designed so that a specified end-to-end message (or bit) error probability limit is not exceeded at the minimum expected received E_b/N_0. When feedback from the receiver to the transmitter is possible, the receiver can tell the transmitter when a received message is too distorted to detect reliably and can request a retransmission. Techniques using feedback are called *automatic repeat request* strategies; these strategies will be discussed in Chapter 8.

Forward error control techniques include both *block coding* techniques and *convolutional coding* techniques. Both of these techniques will be discussed separately. In this section it is assumed that the source has been ideally source coded so that its output is a sequence of equally likely independent digits from the binary alphabet {0, 1}. The source output sequence is modified within the encoder to add the necessary structure for correcting transmission errors at the receiver. In most cases, the encoder input and output alphabets will be binary. In order to have the resources necessary to add structure to the transmitted signal, a binary input–output encoder's output symbol rate must be somewhat higher than its input bit rate.

The channel is assumed to be a BSC with transition probabilities $\Pr(y \mid x)$ with y and $x \in \{0,1\}$. When the channel input and output alphabets are identical, the channel is called a *hard decision channel* and the associated decoder is a *hard decision decoder.* When discussing convolutional coding in Chapter 7, the channel will be permitted to output symbols from an alphabet that is larger than the input alphabet. The larger output alphabet permits the channel to give decision reliability information to the decoder. When reliability information is output from the channel, the channel is called a *soft decision channel* and the associated decoder is a *soft decision decoder.*

6.3.1 Basic Concepts

As their name suggests, block coding techniques are those coding techniques that process information in blocks. A binary input–output *block encoder* collects blocks of k binary information symbols and maps them into blocks of n binary output symbols. The *code rate* for the block code is defined as $R = k/n$ and is equal to the rate of information transmission per channel use. That is, each time an encoder output symbol is transmitted, R bits of information are transmitted. A block of n channel uses (one encoder output block) transmits $nR = n(k/n) = k$ information bits. The information transmission rate must be lower than the channel capacity C_N if coding is expected to improve signaling reliability. For a binary input channel the capacity is at most one bit per channel use so that $R \le C_N \le 1.0$, and it follows that $k \le n$.

6.3.1.1 Definition of a Block Code. A block of k information bits will be represented by a k-vector $\mathbf{w_m} = w_{m0}\, w_{m1}\, w_{m2} \ldots w_{m(k-1)}$, where each w_{mi} is either a 0 or 1 and the subscript m identifies the particular message being considered. There are 2^k possible encoder input messages corresponding to the 2^k binary k-vectors $\mathbf{w_m}$. The probability that the source outputs the message m is denoted by $Q_M(m)$; for the binary symmetric source assumed here, all output messages are equally likely so that $Q_M(m) = 2^{-k}$ for all m. The encoder maps each unique $\mathbf{w_m}$ into a unique binary n-vector $\mathbf{x_m} = x_{m0}\, x_{m1}\, x_{m2} \ldots x_{m(n-1)}$. This mapping is one to one, meaning that no two messages are assigned to the same $\mathbf{x}$, that is, $\mathbf{x_m} \neq \mathbf{x_{m'}}$ for $m \neq m'$. An *(n, k) block code* is the set of all $\mathbf{x_m}$ along with their assignments to messages m. Each $\mathbf{x_m}$ is a *codeword* of the code. There are 2^n possible codewords and 2^k possible messages. Since $k < n$, the number of messages 2^k is less than the number of possible codewords 2^n so that not all possible n-vectors are used for codewords. In general, the fraction of all possible codewords used is $2^{k-n} = 2^{k(1-1/R)}$; observe that for a constant rate $R < 1.0$ this fraction decreases as k increases. The error correction capability of the code is due to the fact that not all possible n-vectors are used by the code. Because of this it is possible to generate codes with codewords selected so that a number of transmission errors must occur before one codeword is confused with another in the receiver.

6.3.1.2 Hamming Distance and Hamming Weight. A typical block code is illustrated in Table 6–4. This block code collects groups of four information bits ($k = 4$) and maps them into codewords of length seven ($n = 7$). The code rate is $R = 4/7$.

Since $n = 7$, there are $2^7 = 128$ possible codewords of which only 16 have been selected for the code. Examination of the codewords of Table 6–4 will show that any two codewords differ in at least three positions. This implies that one or two transmission errors cannot transform one codeword into another codeword. In this case, the decoder is able to tell the user that transmission errors have occurred although it may not be able to correct these errors. Three transmission errors, however, can modify the transmitted codeword into a block identical to another valid codeword. In this case the decoder will not be able to detect or correct the errors. For example, if the codeword for message number 2

TABLE 6–4 Typical Block Code

Message Number	Message	Codeword
0	0000	0000000
1	1000	1101000
2	0100	0110100
3	1100	1011100
4	0010	1110010
5	1010	0011010
6	0110	1000110
7	1110	0101110
8	0001	1010001
9	1001	0111001
10	0101	1100101
11	1101	0001101
12	0011	0100011
13	1011	1001011
14	0111	0010111
15	1111	1111111

was transmitted and errors were made in positions (counting from the left) 1, 2, and 4, the received block would be identical to the codeword for message number 3. If more than three errors occur, the decoder will be able to detect errors only if the received vector is not identical to a codeword. The number of differences between a pair of codewords $\mathbf{x}_m$ and $\mathbf{x}_{m'}$ is an extremely important property and is called the *Hamming distance,* denoted d_H or $d_H(\mathbf{x}_m, \mathbf{x}_{m'})$, between the codewords. The minimum Hamming distance between any two codewords in the code is called the *minimum distance* of the code and is denoted $d_{\min}$. The code of Table 6–4 is one of a family of codes called *Hamming codes* after their discoverer R. W. Hamming. The minimum distance of a Hamming code is three. It will be shown later that these codes are able to correct a single transmission error in any transmitted codeword block.

The Hamming distance between a pair of codewords is equal to the number of 1s in the modulo-2 vector sum of the codewords. The modulo-2 vector sum of two codewords is the term-by-term sum of the vector components (without carry) using modulo-2 rules of addition as defined in Table 6–5 which also defines modulo-2 multiplication, which will be needed later in this chapter. Using Table 6–5 it is found that the Hamming distance between codewords 2 and 3 of the code of Table 6–4 is the number of 1s in the sum.

$$\begin{array}{rr} \mathbf{x}_2 & 0\,1\,1\,0\,1\,0\,0 \\ +\,\mathbf{x}_3 & 1\,0\,1\,1\,1\,0\,0 \\ \hline = & 1\,1\,0\,1\,0\,0\,0 \end{array} \tag{6–24}$$

TABLE 6–5 Modulo-2 Addition and Multiplication

+	**0**	**1**	•	**0**	**1**
0	0	1	**0**	0	0
1	1	0	**1**	0	1

The number of 1s in any vector is called the *Hamming weight* of the codeword and is denoted w_H or $w_H(\mathbf{x_m})$.

6.3.1.3 Error Vectors. The codeword $\mathbf{x_m}$ is transmitted over the BSC. It is convenient to model the errors which can occur during this transmission by an error n-vector $\mathbf{e} = e_0\ e_1\ e_2 \ldots e_{n-1}$. The error vector contains a 0 in every position where no transmission error occurs and a 1 in every position where an error does occur. The BSC output vector is found by calculating the modulo-2 sum of the transmitted codeword vector $\mathbf{x_m}$ and the error vector $\mathbf{e}$. For example, the error 7-vector denoting errors in positions 1, 2, and 4 of a 7-bit codeword is $\mathbf{e} = 1101000$. If the codeword corresponding to message 2 ($m = 2$) of the code of Table 6–4 were transmitted and were distorted by this error vector, the received 7-vector would be

$$\begin{array}{rr} \mathbf{x_2} & 0\,1\,1\,0\,1\,0\,0 \\ +\,\mathbf{e} & 1\,1\,0\,1\,0\,0\,0 \\ \hline =\,\mathbf{y} & 1\,0\,1\,1\,1\,0\,0 \end{array} \tag{6–25}$$

The *received* n-vector is denoted by $\mathbf{y} = y_0\ y_1\ y_2 \ldots y_{n-1}$.

The channel is assumed to be memoryless, meaning that the probability of making an error on any particular symbol is independent of what occurs on all other symbols. Further, it is assumed that the BSC transition probabilities $\Pr(y_i \mid x_i)$ are constant for all transmissions. Then

$$\Pr[\mathbf{y} \mid \mathbf{x}] = \prod_{n'=0}^{n-1} \Pr(y_{n'} \mid x_{n'}) \tag{6–26}$$

For the BSC denote $\Pr(1 \mid 0) = \Pr(0 \mid 1) = p$ so that $\Pr(0 \mid 0) = \Pr(1 \mid 1) = 1 - p$. Then, the probability that the BSC causes the error vector $\mathbf{e} = 1101000$ is

$$\begin{aligned} \Pr[\mathbf{e} = 1101000] &= p \times p \times (1-p) \times p \times (1-p) \\ &\quad \times (1-p) \times (1-p) \end{aligned} \tag{6–27}$$

In general, the probability that the BSC causes a sequence of e' channel errors in *specific positions* in a codeword block of length n is

$$\Pr\begin{bmatrix} e' \text{ errors in} \\ \text{specific locations} \\ \text{in codeword} \end{bmatrix} = p^{e'}(1-p)^{n-e'} \tag{6–28}$$

This result is used to find $\Pr[\mathbf{y} \mid \mathbf{x}_m]$, the probability of receiving $\mathbf{y}$ given that $\mathbf{x}_m$ was transmitted. This probability is also given by

$$\Pr(\mathbf{y} \mid \mathbf{x}_m) = p^{d_H}(1-p)^{n-d_H} \tag{6–29}$$

where d_H is the Hamming distance between $\mathbf{x}_m$ and $\mathbf{y}$. The probability that the BSC causes e' errors without regard to the specific locations of the errors in $\mathbf{e}$ equals the number of possible sequences with e' errors times the probability of a specific sequence occurring. The number of specific sequences of length n having e' errors is given by a binomial coefficient

$$\binom{n}{e'} = \frac{n!}{e'!(n-e')!} \tag{6–30}$$

Therefore, the probability that the BSC causes *some* sequence of e' errors in a block of length n is

$$\Pr\begin{bmatrix} e' \text{ errors in} \\ \text{block of } n \end{bmatrix} = \binom{n}{e'} p^{e'}(1-p)^{n-e'} \tag{6–31}$$

6.3.1.4 Optimum Decoding Rule. The decoder inputs are the received vector $\mathbf{y}$, prior knowledge of the entire set of codewords $\mathbf{x}_m$ for $m = 0, \ldots, 2^k - 1$, knowledge of the source output message probabilities $Q_M(m)$, and knowledge of the channel transition probabilities $\Pr(\mathbf{y} \mid \mathbf{x}_m)$. Using this information the decoder must be designed to make the best possible estimate of the transmitted codeword. The "best possible" estimate is the estimate that results in the minimum average number of bit errors delivered to the information user. The bit error probability will be denoted P_b.

First, consider the much simpler problem of choosing a decoding rule which results in the minimum probability of incorrectly estimating the transmitted codeword without regard to the number of information bit errors that a particular codeword error causes. This block decoding error probability is denoted P_B. Denote the decoder estimate of the transmitted codeword by $\hat{\mathbf{x}}$. Given that $\mathbf{y}$ was received and $\hat{\mathbf{x}} = \mathbf{x}_m$, the decoder estimate is correct if $\mathbf{x}_m$ was indeed transmitted. Therefore, P_B is minimized if the decoder chooses as its estimate $\hat{\mathbf{x}}$ the codeword that was most likely to have been transmitted. That is, choose $\hat{\mathbf{x}}$ to be that $\mathbf{x}_m$ with the largest posterior probability $\Pr[\mathbf{x}_m \mid \mathbf{y}]$. Using this rule, a decoding table can be created that identifies the decoder output $\hat{\mathbf{x}}$ for each $\mathbf{y}$. Since there are 2^n possible $\mathbf{y}$ and only 2^k codewords, many $\mathbf{y}$ will be mapped into the same $\mathbf{x}_m$ by this decoding rule. The set of all $\mathbf{y}$ for which the optimum decoder estimates $\hat{\mathbf{x}} = \mathbf{x}_m$ is called the *decision region* for $\mathbf{x}_m$. The space of all 2^n possible $\mathbf{y}$ is partitioned into 2^k non-overlapping decision regions; there is a decision region for each possible message.

In order to calculate the posterior probabilities $\Pr[\mathbf{x}_m \mid \mathbf{y}]$, Bayes' rule is used. Recall from probability theory [see (A-6)] that

$$\Pr[\mathbf{x}_m \mid \mathbf{y}] \Pr[\mathbf{y}] = \Pr[\mathbf{y} \mid \mathbf{x}_m] \Pr[\mathbf{x}_m] \tag{6–32}$$

Therefore,

$$\Pr[\mathbf{x}_m \mid \mathbf{y}] = \frac{\Pr[\mathbf{y} \mid \mathbf{x}_m] \Pr[\mathbf{x}_m]}{\Pr[\mathbf{y}]} \tag{6–33}$$

The denominator on the right side of this equation is

$$\Pr[\mathbf{y}] = \sum_{m'=0}^{2^k-1} Q_M[m'] \Pr[\mathbf{y} \mid \mathbf{x}_{m'}] \tag{6–34}$$

which is positive and independent of the message m.

Therefore, the decoding rule may be restated: Choose $\hat{\mathbf{x}}$ to be that $\mathbf{x}_m$ for which

$$\Pr[\mathbf{y} \mid \mathbf{x}_m] \Pr[\mathbf{x}_m] = \Pr[\mathbf{y} \mid \mathbf{x}_m] Q_M[m] \tag{6–35}$$

is maximum. This equation is evaluated using knowledge of the channel transition probabilities as in (6–26) and the prior probabilities for the messages.

When all message probabilities are equal, the decoding rule simplifies further to the following: Choose $\hat{\mathbf{x}}$ to be that $\mathbf{x}_m$ for which

$$\Pr[\mathbf{y} \mid \mathbf{x}_m] \tag{6–36}$$

is maximum. Observe that the concepts used to derive the optimum decoding rule are the same concepts used to derive the optimum receiver demodulator in Chapter 4. As in Chapter 4, the decoder that makes decisions without regard to the message prior probabilities is called a *maximum likelihood decoder.*

For the binary symmetric channel $\Pr[\mathbf{y} \mid \mathbf{x}_m]$ is given by (6–29). Since the logarithm is a monotone increasing function of an increasing argument, the logarithm of $\Pr[\mathbf{y} \mid \mathbf{x}_m]$ may be used in the maximum likelihood decoding rule without changing any decoder decisions. Taking the logarithm of (6–29) the decoding rule for the BSC can be stated as follows: Choose $\hat{\mathbf{x}}$ to be that $\mathbf{x}_m$ for which

$$\begin{aligned} \log\{\Pr[\mathbf{y} \mid \mathbf{x}_m]\} &= d_H \log(p) + (n - d_H) \log(1 - p) \\ &= d_H \log\left(\frac{p}{1-p}\right) + n \log(1 - p) \end{aligned} \tag{6–37}$$

is maximum. In this equation, d_H is the Hamming distance between the received n-vector and the codeword n-vector being considered.

Since $\log[p/(1-p)] < 0$ for $p < 0.5$, maximizing (6–37) is equivalent to minimizing d_H. Therefore, the decoding rule is a *minimum distance decoding rule.* For the BSC with $p < 0.5$ the decoder will choose as its estimate $\hat{\mathbf{x}}$ the codeword which is closest to the received vector $\mathbf{y}$ in terms of Hamming distance.

EXAMPLE 6–7

Consider the $n = 7$ and $k = 4$ Hamming code of Table 6–4. Suppose that the binary symmetric channel output is the n-tuple $\mathbf{y} = 1\ 0\ 1\ 1\ 0\ 1\ 0$. The decoder calculates the Hamming distances $d_H(\mathbf{y}, \mathbf{x}_m)$ between $\mathbf{y}$ and all possible codewords $\mathbf{x}_m$. The decoder output estimate $\hat{\mathbf{x}}$ is that $\mathbf{x}_m$, which has the minimum distance from $\mathbf{y}$. In this case that minimum distance is 1 and is message $\mathbf{x}_5 = 0011010 = \hat{\mathbf{x}}$.

The decoding rules just described achieve the minimum possible *block* decoding error probability. The minimization of block decoding error probability is assumed to also minimize decoded *bit* error probability. This will be true with proper assignment of messages to codewords. Messages are assigned such that the most probable block decoding errors cause the minimum number of bit errors. "Good" block codes all have this property.

6.3.1.5 Decoding Regions and Error Probability. The minimum distance decoder functions by finding the codeword $\mathbf{x}_m$ that is closest to the received n-vector $\mathbf{y}$. As mentioned before, conceptually this may be accomplished by partitioning the space of all possible received $\mathbf{y}$ into regions. All the $\mathbf{y}$ in any region are chosen to be closer in terms of Hamming distance to a particular $\mathbf{x}_m$ than to any other $\mathbf{x}_{m'}$. Therefore, the decoder may operate by finding the region containing the received $\mathbf{y}$ and estimating $\hat{\mathbf{x}}$ to be the $\mathbf{x}_m$ associated with that region.

Table 6–6 illustrates the partitioning of the set of all possible 7-vectors for the Hamming code of Table 6–4. The first column of Table 6–6 contains the 16 codewords of the code. The remaining columns contain all other 7-vectors. The decoding region for each codeword is the set of 7-vectors in the same row as the codeword and includes the code-

TABLE 6–6 Decoding Table for Code of Table 6–4

0000000	0000001	0000010	0000100	0001000	0010000	0100000	1000000
1101000	1101001	1101010	1101100	1100000	1111000	1001000	0101000
0110100	0110101	0110110	0110000	0111100	0100100	0010100	1110100
1011100	1011101	1011110	1011000	1010100	1001100	1111100	0011100
1110010	1110011	1110000	1110110	1111010	1100010	1010010	0110010
0011010	0011011	0011000	0011110	0010010	0001010	0111010	1011010
1000110	1000111	1000100	1000010	1001110	1010110	1100110	0000110
0101110	0101111	0101100	0101010	0100110	0111110	0001110	1101110
1010001	1010000	1010011	1010101	1011001	1000001	1110001	0010001
0111001	0111000	0111011	0111101	0110001	0101001	0011001	1111001
1100101	1100100	1100111	1100001	1101101	1110101	1000101	0100101
0001101	0001100	0001111	0001001	0000101	0011101	0101101	1001101
0100011	0100010	0100001	0100111	0101011	0110011	0000011	1100011
1001011	1001010	1001001	1001111	1000011	1011011	1101011	0001011
0010111	0010110	0010101	0010011	0011111	0000111	0110111	1010111
1111111	1111110	1111101	1111011	1110111	1101111	1011111	0111111

word as well as the 7-vectors separated from the codeword by the vertical dashed line. All 128 binary 7-vectors are represented in Table 6–6 which has 8 columns and 16 rows. The method used to construct Table 6–6 will be discussed later. It can be verified manually that any 7-vector in the decoding region for a codeword is Hamming distance 0 or 1 from the codeword. It can also be verified manually that *all* 7-vectors that are Hamming distance 0 or 1 from a codeword are in the decoding region for that codeword. The codeword itself is considered to be in the decoding region.

The block decoding error probability can be calculated once a decoding table of the form of Table 6–6 has been generated. A block decoding error will occur if a codeword $\mathbf{x_m}$ is transmitted and errors occur that cause the received n-vector $\mathbf{y}$ to be within the decoding region for a different codeword, say, $\mathbf{x_{m'}}$. Denote the decoding region for message m or codeword $\mathbf{x_m}$ by Λ_m and all n-vectors not in Λ_m by $\bar{\Lambda}_m$. When the received n-vector $\mathbf{y}$ is within Λ_m, the decoder output $\hat{\mathbf{x}} = \mathbf{x_m}$.

A decoding error occurs whenever $\mathbf{x_m}$ is transmitted and $\mathbf{y}$ is within $\bar{\Lambda}_m$. Given that message m was transmitted, the conditional block error probability P_{B_m} is given by

$$\begin{aligned} P_{B_m} &= \Pr(\mathbf{y} \in \bar{\Lambda}_m \mid \mathbf{x_m}) \\ &= \sum_{\mathbf{y} \in \bar{\Lambda}_m} \Pr(\mathbf{y} \mid \mathbf{x_m}) \end{aligned} \tag{6–38}$$

where the sum is over all n-vectors $\mathbf{y}$ that are not in the decoding region for message m. The condition on message m is removed by averaging over all messages, yielding

$$P_B = \sum_{m=0}^{2^k-1} Q_M(m) P_{B_m} \tag{6–39}$$

EXAMPLE 6–8

For the Hamming code of Table 6–4 and the decoding Table 6–6, and for any transmitted message, a decoding error occurs whenever more than a single transmission error occurs. Therefore, P_{B_m} is independent of m and is

$$\begin{aligned} P_{B_m} &= \sum_{e'=2}^{7} \Pr\begin{bmatrix} e' \text{ errors in} \\ \text{block of 7} \end{bmatrix} \\ &= \sum_{e'=2}^{7} \binom{7}{e'} p^{e'} (1-p)^{7-e'} \end{aligned}$$

The prior message probability $Q_M(m) = 2^{-k}$ for any m so that the block error probability P_B is given by

$$P_B = \sum_{m=0}^{15} \frac{1}{16} P_{B_m}$$

This relationship is relatively easy to calculate to obtain a curve of P_B as a function of the BSC crossover probability.

The relationships given relate the *block* error probability to the BSC error probability p. To complete the analysis, the block error probability P_B must be related to the decoder output *bit* error probability P_b. The number of information bit errors that occur when a block error occurs is a function of the specific decoder information output that occurs for that block error. Denote the number of bit errors that occur when transmitted message m is decoded as m' by $B(m, m')$. For example, when codeword $m = 2$ of the code of Table 6–4 is transmitted and the decoder estimates that codeword $m = 3$ was transmitted, a single bit error occurs in the fourth bit position and $B(2,3) = 1$. The probability of decoding a specific message m as another specific message m' is denoted $P_B(m, m')$. The probability $P_B(m, m')$ is the probability of receiving a vector $\mathbf{y}$ within $\Lambda_{m'}$ when $\mathbf{x_m}$ is transmitted. This probability is

$$P_B(m, m') = \sum_{\mathbf{y} \in \Lambda_{m'}} \Pr(\mathbf{y} \mid \mathbf{x_m}) \tag{6–40}$$

so that the exact bit error probability of interest is

$$P_b = \frac{1}{k} \sum_{m=0}^{2^k-1} Q_M(m) \sum_{\substack{m'=0 \\ m' \neq m}}^{2^k-1} B(m, m') \tag{6–41}$$

where the leading $1/k$ factor is due to the fact that k information bits are decoded with each codeword transmission. Without this term, the result would be the average number of bit errors per block. Except for the simplest of codes, the evaluation of this expression is tedious and long. For large k the calculation is not possible due to computation time bounds. For linear codes, which will be defined in the next section, this relationship can be simplified. In addition, reasonable assumptions about the decoder for linear codes will enable the generation of good upper bounds on the bit error probability.

6.3.1.6 Coding Gain. The purpose of using forward error correction is to enable improved communications efficiency in terms of the transmitter power necessary to achieve a required bit error probability. The *coding gain* of the system is the difference between the E_b/N_0 required to achieve a specified P_b without coding and the E_b/N_0 required to achieve the same P_b with coding. The coding gain is typically a function of P_b.

Calculation of coding gain is straightforward; however, care must be taken to assure that correct signal-to-noise ratio is used in the calculation of the BSC error probability for the coded case. Recall that the encoder output symbol rate is larger than its input bit rate. Therefore, the transmitted energy per encoder output symbol E_s is less than the energy per bit E_b. These two energies are related by the code rate R, specifically, $E_s = RE_b$.

Consider, for example, a system that uses binary antipodal phase-shift keying (BPSK) for which the bit error rate without FEC is given by (6–22). If the system is using FEC with a code rate R and still using BPSK, the error rate p used in the calculation of decoded bit error probability is calculated using (6–23) with $R_{m'} = R$. The calculated coding gain will be a function of technique used to calculate P_b for the coded system. In most

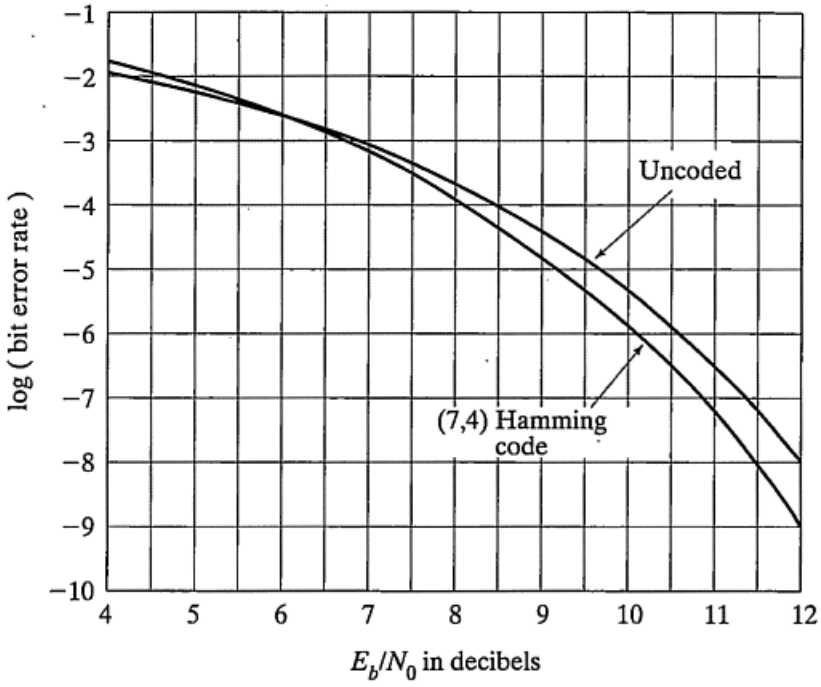


FIGURE 6–13 Bit error rate versus E_b/N_0 for the (7, 4) Hamming code.

cases bounds for P_b will be used so that the calculated coding gain will be a lower bound on the actual coding gain.

EXAMPLE 6–9

Consider a communications system that uses coherent BPSK modulation either with or without the $R = 4/7$ Hamming code of the previous three examples. Without coding the bit error probability is given by (6–22). With coding the bit error probability is calculated using (6–41) with the BSC error probability calculated using (6–23) with $R = 4/7$. The parameters necessary for the calculation of (6–41) were determined by a detailed examination of all of the codeword pairs using a digital computer. Figure 6–13 illustrates the results of this exact calculation. The coding gain is equal to the separation in decibels of the two curves of Figure 6–13. At a bit error probability of 10^{-6}, the coding gain is about 0.5 dB. The system using the $R = 4/7$ Hamming code uses 0.5 dB less transmitter power than the uncoded system to achieve the bit error probability of 10^{-6}. Observe that the separation between the two curves is not constant so that coding gain changes with signal-to-noise ratio. Finally, observe that the curves cross at an E_b/N_0 of approximately 5.75 dB indicating that the communicator is better off not using this particular error correcting code at low signal-to-noise ratios.

6.3.1.7 Summary. An (n, k) binary block code is a one-to-one mapping of encoder input k-bit vectors (messages) to encoder output n-bit vectors (codewords). The encoder output alphabet does not have to be binary. However, in many useful cases the alphabet is binary. When the encoder output is binary, $R \leq 1.0$ and $k < n$. The total number of messages 2^k is less than the number of possible n-vectors 2^n so that a small fraction $(= 2^{k-n})$ of the possible n-vectors will be used for codewords; this fact enables the code to correct transmission

errors. The 2^k codewords are selected from 2^n possibilities so that the number of differences between any codeword pair, the Hamming distance, is maximized. The assignment of messages to codewords is done so that the most likely decoding errors cause few message bit errors. The minimum Hamming distance between any two codewords in a code is the minimum distance of the code. The codewords are assumed to be transmitted over a discrete memoryless channel whose output alphabet is the same as its input alphabet (hard decision channel). The decoder uses prior knowledge of the message probabilities, the codeword set, and the channel transition probabilities along with the received n-vector (includes errors) to estimate the transmitted message. The minimum error probability decoding rule estimates the transmitted codeword to be the codeword that is closest to the received vector in terms of Hamming distance.

6.3.2 Linear Codes

The discussion of the previous section was very general in that codes were merely sets of n-vectors that were *very wisely chosen* and *very wisely assigned to message k-vectors* so that good communications performance resulted. In this section the set of code vectors are constrained to be very particular subsets of the set of all possible n-vectors. These particular subsets will have properties that facilitate both the encoding and decoding operations. In addition the properties make the prediction of the performance of the coded system easier. In order to discuss the codes of interest efficiently, an understanding of the fundamentals of modulo-2 vector arithmetic and linear vector spaces is needed.

6.3.2.1 Modulo-2 Vector Arithmetic. Consider two binary vectors $\mathbf{a} = a_0\ a_1\ a_2 \ldots a_{n-1}$ and $\mathbf{b} = b_0\ b_1\ b_2 \ldots b_{n-1}$. All components of $\mathbf{a}$ and $\mathbf{b}$ are elements of the set $\{0, 1\}$. The *modulo-2 sum*, $\mathbf{c} = \mathbf{a} + \mathbf{b}$, of these two vectors, as mentioned earlier, is the term-by-term modulo-2 sum of the vector components. Modulo-2 addition was defined in Table 6–5. Thus, $c_0 = a_0 + b_0$, $c_1 = a_1 + b_1$, and so on. The *scalar product* $\mathbf{c} = b\mathbf{a}$ of a vector $\mathbf{a}$ and a scalar element of b from the set $\{0, 1\}$ is defined to be the term-by-term modulo-2 product of the vector components and the scalar b. Modulo-2 multiplication is also defined in Table 6–5. Thus, $c_0 = b \cdot a_0$, $c_1 = b \cdot a_1$, and so on. The *dot product* $\mathbf{c} = \mathbf{a} \cdot \mathbf{b}$ is a scalar defined by

$$\mathbf{a} \cdot \mathbf{b} = (a_0 \cdot b_0) + (a_1 \cdot b_1) + (a_2 \cdot b_2) + \cdots + (a_{n-1} \cdot b_{n-1}) \tag{6–42}$$

where all additions and multiplications are performed modulo-2. If the dot product of two vectors is equal to 0, the two vectors are defined to be *orthogonal.*

Using the rules of vector arithmetic linear combinations of vectors are defined. Specifically, given any set of k binary n-vectors $\mathbf{g_0}, \mathbf{g_1}, \ldots, \mathbf{g_{k-1}}$ and any set of k binary scalars $w_0, w_1, w_2, \ldots, w_{k-1}$, the sum

$$(w_0 \cdot \mathbf{g_0}) + (w_1 \cdot \mathbf{g_1}) + (w_2 \cdot \mathbf{g_2}) + \cdots + (w_{k-1} \cdot \mathbf{g_{k-1}}) \tag{6–43}$$

is called a *linear combination* of the vectors $\mathbf{g}$. The set of k binary n-vectors is said to be *linearly independent* if there is no set of scalars $w_0, w_1, w_2, \ldots, w_{k-1}$ that are not all equal

to 0 such that the linear combination is equal to $\mathbf{0} = 0\ 0\ 0\ \ldots\ 0$. Otherwise, the set is defined to be *linearly dependent.*

EXAMPLE 6–10

Consider the set of four 7-vectors

$$\mathbf{g}_0 = 1\ 1\ 0\ 1\ 0\ 0\ 0$$
$$\mathbf{g}_1 = 0\ 1\ 1\ 0\ 1\ 0\ 0$$
$$\mathbf{g}_2 = 1\ 1\ 1\ 0\ 0\ 1\ 0$$
$$\mathbf{g}_3 = 1\ 0\ 1\ 0\ 0\ 0\ 1$$

There are 16 possible linear combinations of these four vectors corresponding to the 16 different binary 4-vectors $\mathbf{w} = w_0\ w_1\ w_2\ w_3$. For example, the linear combination corresponding to $\mathbf{w} = 1\ 0\ 1\ 1$ is

$$\mathbf{x} = (1 \cdot \mathbf{g}_0) + (0 \cdot \mathbf{g}_1) + (1 \cdot \mathbf{g}_2) + (1 \cdot \mathbf{g}_3)$$

$$\begin{array}{rr} = & 1\ 1\ 0\ 1\ 0\ 0\ 0 \\ & +\ 0\ 0\ 0\ 0\ 0\ 0\ 0 \\ & +\ 1\ 1\ 1\ 0\ 0\ 1\ 0 \\ & +\ 1\ 0\ 1\ 0\ 0\ 0\ 1 \\ \hline & 1\ 0\ 0\ 1\ 0\ 1\ 1 \end{array}$$

Although the exercise is tedious, it is straightforward to calculate all 16 linear combinations of the four vectors **g.** These combinations are given in Table 6–7. Observe that there is a sin-

TABLE 6–7 Linear Combinations

w_0	w_1	w_2	w_3	**Linear Combination**
0	0	0	0	0000000
1	0	0	0	1101000
0	1	0	0	0110100
1	1	0	0	1011100
0	0	1	0	1110010
1	0	1	0	0011010
0	1	1	0	1000110
1	1	1	0	0101110
0	0	0	1	1010001
1	0	0	1	0111001
0	1	0	1	1100101
1	1	0	1	0001101
0	0	1	1	0100011
1	0	1	1	1001011
0	1	1	1	0010111
1	1	1	1	1111111

gle linear combination which is equal to the zero vector $\mathbf{0} = 0\ 0\ 0\ 0\ 0\ 0\ 0$ and that for this combination $\mathbf{w} = 0\ 0\ 0\ 0$. Therefore, the set of vectors $\mathbf{g}_0, \ldots, \mathbf{g}_3$ is linearly independent. Comparing Table 6–7 with Table 6–4, which defines the $R = 4/7$ Hamming code, it is seen that the tables are identical. Thus, the 16 possible linear combinations of the vectors $\mathbf{g}_0$ through $\mathbf{g}_3$ generate all 16 possible Hamming codewords. The particular linear combination for a codeword is defined by the binary representation of the message vector. That is, the components of the message vector w_0, w_1, w_2, w_3 define which gs combine to form the codeword. The $R = 4/7$ Hamming code is therefore completely specified by the four vectors $\mathbf{g}_0$ through $\mathbf{g}_3$, which are called the generator vectors of the code.

6.3.2.2 Binary Linear Vector Spaces. A *binary linear vector* space is a set of K binary n-vectors $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{K-1}$ that satisfy certain conditions. The following conditions must be satisfied:

1. The modulo-2 sum of any two vectors in the set is another vector in the set.
2. The modulo-2 scalar product of an element of {0, 1} and any vector in the set is another vector in the set.
3. A distributive law is satisfied. That is, if b_1 and b_2 are scalars from the set {0, 1} and $\mathbf{x}_1$ and $\mathbf{x}_2$ are vectors from the set, then

$$\begin{aligned} b_1 \cdot (\mathbf{x}_1 + \mathbf{x}_2) &= (b_1 \cdot \mathbf{x}_1) + (b_1 \cdot \mathbf{x}_2) \\ (b_1 + b_2) \cdot \mathbf{x}_1 &= (b_1 \cdot \mathbf{x}_1) + (b_2 \cdot \mathbf{x}_1) \end{aligned} \tag{6–44}$$

4. An associative law is satisfied. That is, if b_1 and b_2 are scalars from the set {0, 1} and $\mathbf{x}_1$ is a vector from the set, then

$$(b_1 \cdot b_2) \cdot \mathbf{x}_1 = b_1(b_2 \cdot \mathbf{x}_1) \tag{6–45}$$

The set of all binary n-vectors S is a vector space since all four conditions are satisfied. The first condition is satisfied since the modulo-2 sum of any two binary n-vectors is another n-vector due to the rules of vector addition and since all n-vectors are in the set. The second condition is satisfied since each component of the scalar products are binary elements so that the result is a binary n-vector. Since all binary n-vectors are in the set the scalar product is in the set. The third and fourth conditions can be similarly verified.

In all cases the binary forward error correction codes of interest in the remainder of this chapter are *subsets* of the set of all n-vectors. For example, the $R = 4/7$ Hamming code is a 16-element subset of the set of all 128 binary 7-vectors. A *subspace* of a linear vector space is a subset of the elements of the linear vector space that satisfies the four conditions given. Because the subset consists of elements of a set that is known to satisfy all four conditions, a smaller set of conditions are sufficient to define a subspace. It can be shown [8] that a subset of a linear vector space is a subspace S_{sub} of the original space if the following two conditions are satisfied.

1. The modulo-2 sum of any two vectors in the subset is also a vector in the subset.
2. The scalar product of an element b of {0, 1} and a vector $\mathbf{x}$ in the subset is also a vector in the subset.

Consider the 16 Hamming code vectors of Table 6–4. The vector sum $\mathbf{x}_2 + \mathbf{x}_4 = \mathbf{x}_6$ is another Hamming code vector:

$$\begin{array}{rr} \mathbf{x}_2 & 0\,1\,1\,0\,1\,0\,0 \\ +\,\mathbf{x}_4 & 1\,1\,1\,0\,0\,1\,0 \\ \hline =\,\mathbf{x}_6 & 1\,0\,0\,0\,1\,1\,0 \end{array}$$

Although it would be tedious, all 120 possible sums of pairs of Hamming codewords could be calculated to show that any sum is another Hamming codeword. Therefore, the set of Hamming codewords is a linear subspace of the vector space of all 7-vectors.

Now consider the subset of *any* k vectors $\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k-1}$ of the binary linear space of all possible n-vectors and form a larger set of $K = 2^k$ vectors $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{K-1}$ by forming all possible linear combinations of the k vectors. Note that if the vectors $\mathbf{g}$ are not linearly independent that all $\mathbf{x}_m$ will not be distinct. This set of K vectors $\mathbf{x}_m$ is a linear subspace of the space of all n-vectors. The linear combinations of $\mathbf{g}$ are defined by

$$\mathbf{x}_m = (w_{m0} \cdot \mathbf{g}_0) + (w_{m1} \cdot \mathbf{g}_1) + (w_{m2} \cdot \mathbf{g}_2) + \cdots + (w_{m,k-1} \cdot \mathbf{g}_{k-1}) \qquad (6\text{–}46)$$

where $w_{m0}, w_{m1}, w_{m2}, \ldots, w_{m,k-1}$ are k elements from the set {0, 1}. In order to demonstrate that the subset is a linear subspace consider the sum of any two subset vectors $\mathbf{x}_m$ and $\mathbf{x}_{m'}$. This sum is

$$\begin{array}{lllll} \mathbf{x}_m = & w_{m0} \cdot \mathbf{g}_0 + & w_{m1} \cdot \mathbf{g}_1 + \cdots + & w_{m,k-1} \cdot \mathbf{g}_{k-1} \\ \mathbf{x}_{m'} = & w_{m'0} \cdot \mathbf{g}_0 + & w_{m'1} \cdot \mathbf{g}_1 + \cdots + & w_{m',k-1} \cdot \mathbf{g}_{k-1} \\ \hline = (w_{m0} + w_{m'0}) \cdot \mathbf{g}_0 + & (w_{m1} + w_{m'1}) \cdot \mathbf{g}_1 + \cdots + & (w_{m,k-1} + w_{m',k-1}) \cdot \mathbf{g}_{k-1} \end{array}$$

In the result, the modulo-2 sums $(w_{m1} + w_{m'1})$ are also binary scalars so that the result is also a linear combination of the vectors $\mathbf{g}$. Therefore, since *all* linear combinations of $\mathbf{g}$ are in the set, the sum of any two vectors $\mathbf{x}$ is another vector in the set and the first condition for subspaces is met. Since $\mathbf{w} = \mathbf{0}$ defines a valid linear combination, the zero vector is in the set. The product of the scalar 0 and any vector is $0 \cdot \mathbf{x}_m = (0 \cdot x_{m0})\ (0 \cdot x_{m1}) \cdots (0 \cdot x_{m,k-1}) = 0\,0 \cdots 0$, which was just shown to be in the set so the first half of the second condition is met. Finally, the product of the scalar 1 and any vector is $1 \cdot \mathbf{x}_m = (1 \cdot x_{m0})\ (1 \cdot x_{m1}) \cdots (1 \cdot x_{m,k-1}) = x_{m0}\, x_{m1} \cdots x_{m,k-1} = \mathbf{x}_m$, which is obviously in the set. Therefore, the second half of the second condition is met and the subset is indeed a linear subspace. Because any vector in the subspace is a linear combination of the vectors $\mathbf{g}$, the vectors $\mathbf{g}$ are said to *span* the subspace.

It has been shown that a linear vector subspace S_{sub} can be constructed by forming all linear combinations of k binary and n-vectors of a linear vector space. No conditions are placed on the k vectors except that they be members of a linear vector space. The

converse of this is also true, that is, a set of k vectors $\mathbf{g}$ can always be found that can construct *any* linear subspace S_{sub}. The trivial case where the set $\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k-1}$ is identical to the subspace $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_{K-1}$ demonstrates this. The smallest set of $\mathbf{g}$s that can be found to construct a subspace is always linearly independent. To see this suppose that the set of $\mathbf{g}$s is linearly dependent and $\mathbf{g}_1 = \mathbf{g}_j + \cdots + \mathbf{g}_l$ for some combination of $\mathbf{g}$s on the right side of the equation. Any $\mathbf{x}_m$ constructed using $\mathbf{g}_1$ can also be constructed using $\mathbf{g}_j + \cdots + \mathbf{g}_l$ so that $\mathbf{g}_1$ can be discarded from the set of $\mathbf{g}$s thereby reducing the number of vectors in the set. When all linear dependencies have been eliminated, the set of $\mathbf{g}$s is linearly independent and contains the fewest possible number of elements. The set of k linearly independent vectors $\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k-1}$ which span the vector subspace is called the *basis* of the subspace. The number of vectors in the basis is the *dimension* of the subspace.

Finally, consider the subset S_{null} of all n-vectors in the space S that are orthogonal to all n-vectors in S_{sub}. That is, a vector $\mathbf{y}$ in S_{null} is orthogonal to any n-vector in S_{sub} and any vector in S that is orthogonal to all $\mathbf{x}$ in S_{sub} is in S_{null}. The set S_{null} is called the *null space* of the space S_{sub}. It can be shown that the dimensionality of the null space is $n-k$ where n is the dimensionality of S and k is the dimensionality of S_{sub}. It can also be shown that the null space is a valid subspace of the space S and that S_{sub} is the null space of S_{null}.

6.3.2.3 Linear Block Codes. A *linear block code* is defined as a block code whose codewords form a k-dimensional subspace of the linear vector space of all possible n-vectors. Applying the properties of linear spaces, it is immediately concluded that the sum of any two code vectors of a linear block code is another code vector. Also, since the sum of a code vector and itself is a valid code vector and is equal to the zero vector, the all-zero vector is a code vector in every linear block code.

EXAMPLE 6–11

Consider again the set $\mathbf{g}_0, \mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3$ of the previous example. In that example it was shown that this set is linearly independent since no linear combination except $(0 \cdot \mathbf{g}_0) + (0 \cdot \mathbf{g}_1) + (0 \cdot \mathbf{g}_2) + (0 \cdot \mathbf{g}_3)$ equals the zero vector. It was also shown that the 16 linear combinations of $\mathbf{g}$s are the 16 codewords of the Hamming code. Since all linear combinations of $\mathbf{g}$s are included in the Hamming code, the code is a linear subspace of the space of all 7-vectors. Since the $\mathbf{g}$s are independent, they are the smallest set of vectors that can be used to generate the subspace and the dimension of the subspace is 4, the number of vectors $\mathbf{g}$. Since all vectors in the subspace are generated by linear combinations of $\mathbf{g}$, the $\mathbf{g}$s span the subspace. Therefore, the Hamming code is a linear block code having dimension $k = 4$.

Since a block code forms a k-dimensional subspace of the space of all n-vectors, all code vectors can be generated by linear combinations of the k linearly independent basis vectors of the subspace. The k basis vectors are called the *generator vectors* of the code. It is convenient to use matrix arithmetic to calculate the codewords of a code. Arrange the generator vectors of a code in a column to form the *code generator matrix* $\mathbf{G}$ defined by

$$\mathbf{G} = \begin{bmatrix} \mathbf{g_0} \\ \mathbf{g_1} \\ \vdots \\ \mathbf{g_{k-1}} \end{bmatrix} = \begin{bmatrix} g_{00} & g_{01} & \cdots & g_{0,n-1} \\ g_{10} & g_{11} & \cdots & g_{1,n-1} \\ \vdots & \vdots & & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{bmatrix} \tag{6–47}$$

Let the message vectors be defined as before by k-vectors $\mathbf{w_m} = w_{m0}\ w_{m1}\ w_{m2} \ldots w_{m,k-1}$. Then the codeword row vector $\mathbf{x_m}$ is calculated by the matrix multiplication of the row vector $\mathbf{w_m}$ and $\mathbf{G}$. That is,

$$\begin{aligned} \mathbf{x_m} &= \mathbf{w_m} \times \mathbf{G} \\ &= [w_{m0}\ w_{m1} \cdots w_{m,k-1}] \times \begin{bmatrix} g_{00} & g_{01} & \cdots & g_{0,n-1} \\ g_{10} & g_{11} & \cdots & g_{1,n-1} \\ \vdots & \vdots & & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{bmatrix} \end{aligned} \tag{6–48}$$

From this equation it is apparent that each codeword symbol is a modulo-2 linear combination of the message symbols. The specific message symbols combined for a particular code vector symbol are defined by the generator matrix. Each code vector symbol can also be calculated using

$$x_{m,j} = \sum_{i=0}^{k-1} w_{m,i} \cdot g_{i,j} \tag{6–49}$$

where arithmetic is performed modulo-2. One of the benefits of using linear codes is that codeword generators implementing this equation are easily built.

EXAMPLE 6–12

The generator matrix for the Hamming code of the previous examples is

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

and the codeword corresponding to the message vector $\mathbf{w} = 1\ 0\ 1\ 1$ is

$$\begin{aligned} \mathbf{x} &= \mathbf{w} \times \mathbf{G} \\ &= [1\ 0\ 1\ 1] \times \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix} \\ &= [1\ 0\ 0\ 1\ 0\ 1\ 1] \end{aligned}$$

A linear code is a linear subspace of the space of all n-vectors. The null space of this subspace is also a subspace and can be represented by generator basis vectors or a generator matrix. Denote the generator matrix of the null space by $\mathbf{H}$ and the generator (basis) vectors by $\mathbf{h}_0, \mathbf{h}_1, \ldots, \mathbf{h}_{n-k-1}$. The null space generator matrix

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_0 \\ \mathbf{h}_1 \\ \vdots \\ \mathbf{h}_{n-k-1} \end{bmatrix} = \begin{bmatrix} h_{00} & h_{01} & \cdots & h_{0,n-1} \\ h_{10} & h_{11} & \cdots & h_{1,n-1} \\ \vdots & \vdots & & \vdots \\ h_{n-k-1,0} & h_{n-k-1,1} & \cdots & h_{n-k-1,n-1} \end{bmatrix} \tag{6–50}$$

This matrix is called the *parity check matrix* of the code defined by **G.** It can be proven [10] that the dimension of the null space is n–k when the dimension of the code space is k. By definition, all the vectors in the null space are orthogonal to all the vectors of the code. Since the vectors $\mathbf{h}_j$ are members of the null space, they are orthogonal to any code vector. Thus, $\mathbf{x}_m \cdot \mathbf{h}_j = 0$ for any $m = 0, \ldots, 2^k - 1$ and $j = 0, \ldots, n - k - 1$. In matrix notation

$$\begin{aligned} \mathbf{0} &= \mathbf{x}_m \times \mathbf{H}^T \\ &= \mathbf{x}_m \times [\mathbf{h}_0^T \mathbf{h}_1^T \cdots \mathbf{h}_{n-k-1}^T] \\ &= [x_{m0}\, x_{m1} \cdots x_{m,n-1}] \times \begin{bmatrix} h_{00} & h_{10} & \cdots & h_{n-k-1,0} \\ h_{01} & h_{11} & \cdots & h_{n-k-1,1} \\ \vdots & \vdots & & \vdots \\ h_{0,n-1} & h_{1,n-1} & \cdots & h_{n-k-1,n-1} \end{bmatrix} \end{aligned} \tag{6–51}$$

where the superscript T denotes the matrix transpose. It is also true that any vector $\mathbf{y}$ that is not a codeword does not satisfy this relationship. Thus, a vector $\mathbf{y}$ is a codeword if and only if

$$\mathbf{0} = \mathbf{y} \times \mathbf{H}^T \tag{6–52}$$

is satisfied. A linear block code can be specified either by its generator matrix $\mathbf{G}$ or by its parity check matrix $\mathbf{H}$. Both are used in practice.

EXAMPLE 6–13

The parity check matrix for the (7, 4) linear block code of the previous examples

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

The matrix product of the n-vector $\mathbf{y} = 1\ 1\ 1\ 1\ 0\ 0\ 1$ and the transpose of $\mathbf{H}$ is

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

which is not zero, indicating that **y** is not a valid codeword.

6.3.2.4 Systematic Linear Block Codes. As specified, any symbol of a code vector is a linear combination of message symbols. Thus, the message symbols themselves do not necessarily appear in the codewords. *Systematic block codes* are codes having generators of a form such that the message vector appears directly in the code vector. Systematic codes are a subset of all linear codes. The message symbols appear directly in a codeword if **G** has the form shown here:

$$\begin{aligned}\mathbf{x_m} &= [x_{m0}\, x_{m1} \cdots x_{m,n-k-1}\, w_{m0}\, w_{m1} \cdots w_{m,k-1}]\\ &= [w_{m0}\, w_{m1} \cdots w_{m,k-1}]\\ &\quad \times \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-k-1} & 1 & 0 & 0 & \cdots & 0\\ g_{1,0} & g_{1,1} & \cdots & g_{1,n-k-1} & 0 & 1 & 0 & \cdots & 0\\ g_{2,0} & g_{2,1} & \cdots & g_{2,n-k-1} & 0 & 0 & 1 & \cdots & 0\\ \vdots & \vdots & & & & & & & \vdots\\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-k-1} & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}\end{aligned} \tag{6–53}$$

Observe that an identity matrix appears in the right side of **G.** For systematic codes the first n–k code vector symbols are parity checks of the message symbols. It can be shown that a systematic code exists whose error correction performance is identical to any given linear code. Therefore, the communications system engineer may limit consideration to systematic codes without fear of overlooking a better code which is not systematic. The Hamming code of Table 6–4 is systematic with the message symbols appearing at the right in each code vector.

For systematic codes the parity check matrix **H** can be determined directly from the generator matrix. Specifically, with the generator matrix as specified, the parity check matrix is

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & g_{0,0} & g_{1,0} & \cdots & g_{k-1,0}\\ 0 & 1 & 0 & \cdots & 0 & g_{0,1} & g_{1,1} & \cdots & g_{k-1,1}\\ 0 & 0 & 1 & \cdots & 0 & g_{0,2} & g_{1,2} & \cdots & g_{k-1,2}\\ \vdots & \vdots & \vdots & & & & & & \\ 0 & 0 & 0 & \cdots & 1 & g_{0,n-k-1} & g_{1,n-k-1} & \cdots & g_{k-1,n-k-1} \end{bmatrix} \tag{6–54}$$

6.3.2.5 Distance Properties of Linear Block Codes. The distance properties of codes are key to their error correction capabilities. In good codes the Hamming distance between any two codewords is as large as possible for the codeword length n and the code rate R. For minimum distance decoders, block decoding errors are made when channel errors cause the received n-vector to be closer in Hamming distance to a codeword different than the transmitted codeword. Knowledge of the Hamming distance between all codeword pairs is necessary in order to calculate either block error or bit error probabilities.

Consider the Hamming distance $d_H(\mathbf{x}_x, \mathbf{x}_{m'})$ between any two code vectors $\mathbf{x}_m$ and $\mathbf{x}_{m'}$ of a linear block code. Suppose that third code vector $\mathbf{x}_a$ is added to both $\mathbf{x}_m$ and to $\mathbf{x}_{m'}$. The addition changes the code vector components of both $\mathbf{x}_m$ and $\mathbf{x}_{m'}$ in the same positions, specifically, the positions where $\mathbf{x}_a$ has 1s. Since both $\mathbf{x}_m$ and $\mathbf{x}_{m'}$ are changed in the same positions, the Hamming distance between the sum vectors is the same as the Hamming distance between the original vectors. That is, $d_H(\mathbf{x}_m + \mathbf{x}_a, \mathbf{x}_{m'} + \mathbf{x}_a) = d_H(\mathbf{x}_m, \mathbf{x}_{m'})$. Since $\mathbf{x}_a$ is an arbitrary code vector, let $\mathbf{x}_a = \mathbf{x}_{m'}$ so that $d_H(\mathbf{x}_m + \mathbf{x}_{m'}, \mathbf{x}_{m'} + \mathbf{x}_{m'}) = d_H(\mathbf{x}_m + \mathbf{x}_{m'}, \mathbf{0})$. Thus, the Hamming distance between any two code vectors of a linear block code is equal to the Hamming distance between the zero code vector and some code vector in the code. In fact, the distances between any particular code vector and all other code vectors is equal to the set of distances from the zero code vector to all nonzero code vectors. The minimum distance $d_{\min}$ of the block code is equal to the smallest Hamming distance between any distinct pair of code vectors. Using the linearity of the code, the minimum distance of the code is equal to the smallest distance between the zero code vector and any nonzero code vector. The minimum distance is thus the Hamming weight of the lowest weight nonzero code vector. The now familiar Hamming code of Table 6–4 has minimum distance three. Examination of Table 6–4 will show that the nonzero code vector with the minimum number of 1s has three 1s, which is consistent with the code's minimum distance.

A minimum distance decoder for a block code with a minimum distance $d_{\min}$ is able to decode correctly any transmission where $\lfloor (d_{\min} - 1)/2 \rfloor \equiv t$ or fewer errors occur. In this expression the notation $\lfloor x \rfloor$ is the largest integer that is smaller than or equal to x. To prove this, consider the distances $d_H(\mathbf{x}_m, \mathbf{y})$ and $d_H(\mathbf{x}_{m'}, \mathbf{y})$, where $\mathbf{x}_m$ is the transmitted code vector, $\mathbf{y}$ is the received vector, and $\mathbf{x}_m$ and $\mathbf{x}_{m'}$, are nearest neighbors; that is, $d_H(\mathbf{x}_m, \mathbf{x}_{m'}) = d_{\min}$. The decoder will decode correctly if it is impossible for $\mathbf{y}$ to be closer to $\mathbf{x}_{m'}$ than to $\mathbf{x}_m$ when t errors occur. If t transmission errors occur, $d_H(\mathbf{x}_m, \mathbf{y}) = t$ and $\mathbf{x}_m$ differs from $\mathbf{y}$ in t positions. Assume that the errors occur in the worst possible locations for decoding $\mathbf{x}_{m'}$ rather than $\mathbf{x}_m$; these positions are positions where $\mathbf{x}_m$ and $\mathbf{x}_{m'}$ differ. Every error that occurs in these positions causes $\mathbf{y}$ to be one unit closer to $\mathbf{x}_{m'}$ and one unit farther from $\mathbf{x}_m$. With t errors in these positions, $d_H(\mathbf{x}_{m'}, \mathbf{y}) = d_{\min} - t$. Since $t \leq (d_{\min} - 1)/2$, $d_{\min} - t \geq (d_{\min} + 1)/2 \geq (d_{\min} - 1)/2$. Therefore, $\mathbf{y}$ is closer to $\mathbf{x}_m$ than to $\mathbf{x}_{m'}$ and correct decoding always occurs. This property will be used to obtain useful relationships for block and bit error probability. Note that, although t transmission errors can always be corrected, a minimum distance decoder can also sometimes correct error patterns having more than t errors.

6.3.2.6 Decoding Using the Standard Array. A convenient method is available for constructing the decoding regions Λ_m for linear block codes. Recall that decoding regions partition the space of all n-vectors into subsets such that all received $\mathbf{y}$ within Λ_m are most likely to have been caused by the transmission of message vector $\mathbf{x}_m$. For the binary symmetric channel with error probability < 0.5, this means that all $\mathbf{y}$ within Λ_m are closer in Hamming distance to $\mathbf{x}_m$ than to any other code vector $\mathbf{x}_{m'}$. Brute force construction of the decoding regions is accomplished by calculating the Hamming distance between each of the 2^n possible received $\mathbf{y}$ and all code vectors $\mathbf{x}_m$ and placing $\mathbf{y}$ in the appropriate region.

The construction of the decoding regions for the binary symmetric channel is greatly facilitated by using the following procedure. First, place all of the code vectors $\mathbf{x}_m$ in a row with the all-zero vector on the left. Next select an error vector $\mathbf{e}_1$ that is not identical to any codeword in the first row and place it below the all zero vector. When this procedure is completed, the error vector just selected will be a correctable error vector, meaning that, if it occurs, the received vector will be correctly decoded. Although the selection of the error vector is arbitrary, it can be shown that the ability to correct the most probable error vectors will result in the best communication system performance. Therefore, since the probability of an error vector increases with decreasing weight, the best error vector to select is the vector having the smallest weight. In all cases this first selection will be a vector with unit Hamming weight. Now add $\mathbf{e}_1$ to every nonzero code vector and place the results in a row to the right of $\mathbf{e}_1$ below the corresponding code vector. Now select another error vector $\mathbf{e}_2$ and place it below the all-zero vector. The vector $\mathbf{e}_2$ should be a vector with the lowest possible Hamming weight that does not appear in either of the first two rows. Form the sum of $\mathbf{e}_2$ and all code vectors and place the sums in a row under the corresponding code vector. Repeat this procedure until all n-vectors have been exhausted. The result is

$$\begin{array}{cccccc} \mathbf{x}_0 & \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{2^k-1} & \\ \mathbf{e}_1 + \mathbf{x}_0 & \mathbf{e}_1 + \mathbf{x}_1 & \mathbf{e}_1 + \mathbf{x}_2 & \cdots & \mathbf{e}_1 + \mathbf{x}_{2^k-1} & \\ \mathbf{e}_2 + \mathbf{x}_0 & \mathbf{e}_2 + \mathbf{x}_1 & \mathbf{e}_2 + \mathbf{x}_2 & \cdots & \mathbf{e}_2 + \mathbf{x}_{2^k-1} & (6\text{–}55) \\ \vdots & \vdots & \vdots & & \vdots & \\ \mathbf{e}_{2^{n-k}-1} + \mathbf{x}_0 & \mathbf{e}_{2^{n-k}-1} + \mathbf{x}_1 & \mathbf{e}_{2^{n-k}-1} + \mathbf{x}_2 & \cdots & \mathbf{e}_{2^{n-k}-1} + \mathbf{x}_{2^k-1} & \end{array}$$

The array just constructed is called the *standard array*. It can be shown [8] that, because of the construction procedure used and because the code is linear, each n-vector in the array is distinct and that all 2^n possible n-vectors appear exactly once in the array. There are always 2^{n-k} rows in the array.

The standard array is used for decoding by defining the decoding region Λ_m to be all of the n-vectors in the column with the code vector $\mathbf{x}_m$ (including $\mathbf{x}_m$). Whenever a received vector $\mathbf{y}$ falls within the same column as $\mathbf{x}_m$, message m is decoded. Since all n-vectors appear in the array once and only once, all $\mathbf{y}$ appear in a single Λ_m (column) and are uniquely decoded. Examination of the standard array will show that, for any code vector $\mathbf{x}_m$, if the actual error pattern is any of the error vectors $\mathbf{e}_j$, the received $\mathbf{y}$ will be in the

same column as $\mathbf{x}_m$ and will be correctly decoded. Thus, the selected error patterns are correctable. The only conditions placed on the selection of $\mathbf{e}_j$ is that a new selection does not appear in any of the rows above it. It can be proven that any error vectors that are *not* in the first column of the array are *not* correctable and, in fact, result in a block decoding error. For any (n, k) linear block code, standard array decoding is able to correct exactly 2^{n-k} error vectors, including the all-zero error vector.

Good communications performance is achieved by selecting correctable error vectors having the lowest weight possible of all vectors not appearing in any row already constructed. It can be proven that the decoding regions constructed from a standard array using this rule are in fact minimum distance decoding regions. Therefore, generating the standard array is a correct procedure for generating the decoding regions Λ_m for a linear block code.

Standard array decoding can be simplified by making use of the parity check matrix. Recall that any codeword of a linear code satisfies

$$\mathbf{0} = \mathbf{x}_m \times \mathbf{H}^T$$

and consider any received vector $\mathbf{y} = \mathbf{x}_m + \mathbf{e}_j$. The matrix product of $\mathbf{y}$ and the transpose of the parity check matrix is called the *syndrome* of $\mathbf{y}$ and is

$$\begin{aligned}(\mathbf{e}_j + \mathbf{x}_m) \times \mathbf{H}^T &= \mathbf{e}_j \times \mathbf{H}^T + \mathbf{x}_m \times \mathbf{H}^T \\ &= \mathbf{e}_j \times \mathbf{H}^T + \mathbf{0}\end{aligned} \tag{6–56}$$

The syndrome is a function only of the error vector $\mathbf{e}_j$. It is easily shown that the syndromes of all the received vectors in a single row of the standard array are equal and further [8] that the syndromes of received vectors appearing in different rows of the array are different. Because of these facts, decoding can be accomplished using the following steps:

1. Calculate the syndrome of the received vector $\mathbf{y}$.
2. Find the row in the standard array which has the same syndrome and estimate the error vector to be the error vector for that row.
3. Add the estimated error vector to the received vector to estimate the transmitted codeword.

If the actual error pattern is one of the patterns used in creating the standard array, correct decoding will occur.

EXAMPLE 6–14

Table 6–8 is the standard array for the (7, 4) Hamming code that has been considered in the previous examples. The first row contains all 16 codewords. The Hamming code can correct all single-error patterns. These error patterns are shown in the first column of the array. Suppose that the received vector is $\mathbf{y} = [1\ 1\ 1\ 0\ 1\ 0\ 1]$, which is shown in the dashed box of the array. What is the decoder's estimate of the transmitted codeword?

TABLE 6–8 Standard Array for the Hamming (7, 4) Code

0000000	1101000	0110100	1011100	1110010	0011010	1000110	0101110	1010001	0111001	1100101	0001101	0100011	1001011	0010111	1111111
0000001	1101001	0110101	1011101	1110011	0011011	1000111	0101111	1010000	0111000	1100100	0001100	0100010	1001010	0010110	1111110
0000010	1101010	0110110	1011110	1110000	0011000	1000100	0101100	1010011	0111011	1100111	0001111	0100001	1001001	0010101	1111101
0000100	1101100	0110000	1011000	1110110	0011110	1000010	0101010	1010101	0111101	1100001	0001001	0100111	1001111	0010011	1111011
0001000	1100000	0111100	1010100	1111010	0010010	1001110	0100110	1011001	0110001	1101101	0000101	0101011	1000011	0011111	1110111
0010000	1111000	0100100	1001100	1100010	0001010	1010110	0111110	1000001	0101001	1110101	0011101	0110011	1011011	0000111	1101111
0100000	1001000	0010100	1111100	1010010	0111010	1100110	0001110	1110001	0011001	1000101	0101101	0000011	1101011	0110111	1011111
1000000	0101000	1110100	0011100	0110010	1011010	0000110	1101110	0010001	1111001	0100101	1001101	1100011	0001011	1010111	0111111

In this simple case where the location of the received vector is easily found in the array, the decoder estimates the transmitted codeword to be the codeword at the head of the column that contains the received vector. In this case the decoder output is $\mathbf{x} = [1\ 1\ 0\ 0\ 1\ 0\ 1]$. In more complicated cases where n is large, the decoder calculates the syndrome of the received vector and compares that syndrome with that of all correctable error patterns. The parity check matrix for the (7, 4) Hamming code is

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

and its transpose is

$$\mathbf{H}^T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

Prior to beginning operation, the decoder will have calculated the syndromes of all the correctable error patterns. For example, the syndrome of error pattern for $\mathbf{e} = [0\ 0\ 0\ 1\ 0\ 0\ 0]$ is

$$[0\ 0\ 0\ 1\ 0\ 0\ 0] \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} = [1\ 1\ 0]$$

The syndromes of the other error patterns are calculated similarly. The results are

$$\begin{array}{ccccccccccc} 0 & 0 & 0 & 0 & 0 & 0 & 1 & \rightarrow & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & \rightarrow & 1 & 1 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & \rightarrow & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & \rightarrow & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & \rightarrow & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & \rightarrow & 0 & 1 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & \rightarrow & 1 & 0 & 0 \end{array}$$

After receiving $\mathbf{y}$, the decoder calculates its syndrome. The result is

$$[1\ 1\ 1\ 0\ 1\ 0\ 1] \times \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} = [0\ 0\ 1]$$

This syndrome $\mathbf{s} = [0\ 0\ 1]$ is the same as the precalculated syndrome for the error pattern $\mathbf{e} = [0\ 0\ 1\ 0\ 0\ 0\ 0]$, which the decoder now estimates as the received error pattern. The transmitted codeword is the modulo-2 sum of received vector and the estimated error vector or $[1\ 1\ 1\ 0\ 1\ 0\ 1] + [0\ 0\ 1\ 0\ 0\ 0\ 0] = [1\ 1\ 0\ 0\ 1\ 0\ 1] = \mathbf{x}$. This is the same result found previously.

Since standard array decoding is able to correct all of the error vectors used in the generation of the array and no others, the block error rate P_B is

$$P_B = 1.0 - \sum_{j=0}^{2^{n-k}-1} p^{w_H(\mathbf{e}_j)}(1-p)^{n-w_H(\mathbf{e}_j)} \tag{6–57}$$

where $w_H(\mathbf{e}_j)$ is the Hamming weight of $\mathbf{e}_j$. This equation is 1 minus the probability of occurrence of any one of the correctable error vectors.

6.3.2.7 Error Probabilities for Linear Codes. Bounding techniques can be used to simplify the calculation of block error probability for linear block codes. First, reconsider the block error probability result discussed previously. The average block error probability for any code (linear or nonlinear) is the average (over all messages m) of the probability that the received vector $\mathbf{y}$ is not in the decoding region Λ_m. This error probability is

$$P_B = \sum_{m=0}^{2^k-1} Q_M(m) P_{B_m} \tag{6–58}$$

where

$$P_{B_m} = \Pr(\mathbf{y} \in \overline{\Lambda}_m \mid \mathbf{x_m}) \tag{6–59}$$

$$= \sum_{y \in \overline{\Lambda}_m} \Pr(\mathbf{y} \mid \mathbf{x_m})$$

For large k and n the brute-force calculation using this relationship is difficult. To facilitate the prediction of performance of coded systems, coding theorists have applied bounding techniques to this problem.

Examine the region $\bar{\Lambda}_m$ in detail. The region $\bar{\Lambda}_m$ can be defined by

$$\begin{aligned}\overline{\Lambda}_m &= \{\mathbf{y} \mid \ln[\Pr(\mathbf{y} \mid \mathbf{x}_{m'})] \geq \ln[\Pr(\mathbf{y} \mid \mathbf{x}_m)] \text{ for some } m' \neq m\} \\ &= \bigcup_{m' \neq m} [\mathbf{y} \mid \ln[\Pr(\mathbf{y} \mid \mathbf{x}_{m'})] \geq \ln[\Pr(\mathbf{y} \mid \mathbf{x}_m)]\} \\ &= \bigcup_{m' \neq m} \Lambda_{mm'}\end{aligned} \tag{6–60}$$

where

$$\Lambda_{mm'} = \left\{ \mathbf{y} \left| \ln\left[\frac{\Pr(\mathbf{y} \mid \mathbf{x}_{m'})}{\Pr(\mathbf{y} \mid \mathbf{x}_m)} \right] \geq 0 \right. \right\} \tag{6–61}$$

For each m and m' the region $\Lambda_{mm'}$ consists of all $\mathbf{y}$ that are more likely to be due to the transmission of $\mathbf{x}_{m'}$ than to the transmission of $\mathbf{x}_m$ without consideration of any other codewords. That is, the space of all $\mathbf{y}$ has been partitioned into two decoding regions, one for $\mathbf{x}_m$ and one for $\mathbf{x}_{m'}$. Since, for a particular m and m', all $\mathbf{y}$ are included within either $\Lambda_{mm'}$ or $\overline{\Lambda}_{mm'}$, the $\Lambda_{mm'}$ for different m and m' are not disjoint. Recall from probability theory that the probability of a union of events is less than or equal to the sum of the probabilities of the component events. Thus the block error probability can be overbounded by

$$\begin{aligned}P_{B_m} &= \Pr(\mathbf{y} \in \overline{\Lambda}_m \mid \mathbf{x}_m) \\ &= \Pr\left(\mathbf{y} \in \bigcup_{m' \neq m} \Lambda_{mm'} \mid \mathbf{x}_m \right) \\ &\leq \sum_{m \neq m'} \Pr(\mathbf{y} \in \Lambda_{mm'} \mid \mathbf{x}_m) \\ &= \sum_{m \neq m'} P_B'(m, m')\end{aligned} \tag{6–62}$$

where $P_B'(m, m')$ is the probability of block error under the assumption that there are only two codewords m and m' in the code.

The evaluation of $P_B'(m, m')$ is done using methods already described. Specifically, consider a code consisting of two codewords, $\mathbf{x}_m$ and $\mathbf{x}_{m'}$, separated by Hamming distance $d_H(\mathbf{x}_m, \mathbf{x}_{m'})$, and determine the probability of incorrect decoding with a minimum distance decoder. Suppose that $\mathbf{x}_m$ is transmitted. A decoding error occurs whenever transmission errors are made that cause $\mathbf{y}$ to be closer in Hamming distance to $\mathbf{x}_{m'}$ than to $\mathbf{x}_m$. Transmission errors in positions where the two codewords are identical increase the distance between $\mathbf{x}_m$ and $\mathbf{y}$ and the distance between $\mathbf{x}_{m'}$ and $\mathbf{y}$ equally and therefore have no effect on the decoding result. Thus, for d_H even, a decoding error is made whenever $(d_H/2) + 1$ or more transmission errors occur in the positions where $\mathbf{x}_m$ and $\mathbf{x}_{m'}$ differ. When only $(d_H/2)$ errors occur, $\mathbf{y}$ is equidistant from $\mathbf{x}_m$ and $\mathbf{x}_{m'}$, and a decoding error is assumed to be made one-half of the time. Thus the two-codeword error probability for d_H even is

$$P'_B(m,m') = \sum_{e=(d_H/2+1)}^{d_H} \binom{d_H}{e} p^e(1-p)^{d_H-e}, \quad d_H \text{ even}$$
$$+ \frac{1}{2}\binom{d_H}{\frac{d_H}{2}} p^{d_H/2}(1-p)^{d_H/2} \tag{6–63}$$

For d_H odd, a decoding error is made whenever $(d_H + 1)/2$ or more transmission errors occur in the positions where $\mathbf{x_m}$ and $\mathbf{x_{m'}}$ differ, and the two-codeword error probability for d_H odd is

$$P'_B(m, m') = \sum_{e=(d_H+1)/2}^{d_H} \binom{d_H}{e} p^e(1-p)^{d_H-e}, \quad d_H \text{ odd} \tag{6–64}$$

Using these relationships, the total block decoding error probability is bounded by

$$P_B \leq \sum_{m=0}^{2^k-1} Q_M(m) \sum_{\substack{m'=0 \\ m' \neq m}}^{2^k-1} P'_B(m,m') \tag{6–65}$$

The calculation of the average block error probability can be somewhat simplified through the use of upper bounds replacing (6–63) and (6–64). It can be shown [8, 12] that

$$P'_B(m, m') \leq [\sqrt{4p(1-p)}]^{d_H(m,m')} \tag{6–66}$$

where $d_H(m, m')$ is the Hamming distance between $\mathbf{x_m}$ and $\mathbf{x_{m'}}$.

Further simplification of this result is dependent on the properties of linear codes.

The specific property of importance is the fact that the set of Hamming distances $d_H(\mathbf{x_m}, \mathbf{x_{m'}})$ between a codeword $\mathbf{x_m}$ and all other codewords $\mathbf{x_{m'}}$ for $m' \neq m$, $m' = 0, \ldots, 2^k - 1$ is the same for all codewords $\mathbf{x_m}$. Because of this property, the second summation of the bound of (6–65) is the same for any choice of m. Therefore, the error probability is unchanged if $m = 0$ is chosen for all calculations of the second sum. Thus,

$$\begin{aligned} P_B &\leq \sum_{m=0}^{2^k-1} Q_M(m) \sum_{m'=1}^{2^k-1} P'_B(0, m') \\ &= \sum_{m'=1}^{2^k-1} P'_B(0, m') \sum_{m=0}^{2^k-1} Q_M(m) \\ &= \sum_{m'=1}^{2^k-1} P'_B(0, m') \end{aligned} \tag{6–67}$$

where $P'_B(0, m')$ is calculated using (6–63) and (6–64). For linear codes, therefore, the total average block error probability may be calculated by calculating only the error probability assuming that message $m = 0$ was transmitted.

In principle, the codewords for any specific code can be enumerated to determine the number of codewords that are Hamming distance d from the all-zeros codeword. The function $P_B'(0, m')$ is completely determined if the Hamming distance between $\mathbf{0}$ and $\mathbf{x}_{\mathbf{m}'}$ is known. Therefore, the block error probability can also be expressed as a sum over values of Hamming distance

$$\begin{aligned} P_B &\leq \sum_{d=d_{\min}}^{n} a_d P_B'(d) \\ &\leq \sum_{d=d_{\min}}^{n} a_d \left[\sqrt{4p(1-p)}\right]^d \end{aligned} \tag{6–68}$$

where a_d is the number of codewords that are Hamming distance d from the all-zero codeword of $P_B'(d)$ is the probability of error considering only two codewords separated by Hamming distance d. Note that $P_B'(d)$ denotes the value of $P_B'(m, m')$ for any m and m' such that $d_H(\mathbf{x}_{\mathbf{m}}, \mathbf{x}_{\mathbf{m}'}) = d$. The set of values a_d for $d = d_{\text{inin}}, \ldots, n$ is called the *weight structure* of the code.

For codes with moderate k it is possible to find a_d with the aid of a digital computer. For a small set of codes a_d has been found analytically. Unfortunately, the communications systems engineer is often required to calculate P_B for codes with large k where even a fast digital computer would not be able to enumerate all codewords in a reasonable time. In these cases results can still be obtained by modifying the decoding rules somewhat and limiting consideration to systematic codes. Specifically, assume that the decoder will correct up to a maximum of M channel errors and no more. Decoders having this property are called *bounded distance decoders.* Note that a maximum likelihood decoder is guaranteed to be able to correct t channel errors but it can, in many cases, correct more than t errors. A bounded distance decoder is guaranteed to make a block decoding error whenever more than M channel errors occur. Therefore, the block error probability is simply the probability of $M + 1$ or more channel errors occurring, which is

$$P_B = \sum_{i=M+1}^{n} \binom{n}{i} p^i (1-p)^{n-1} \tag{6–69}$$

This result is an upper bound on P_B for a maximum likelihood decoder that is guaranteed to correct at least $M = t$ errors.

The results just discussed permit the communication system designer to predict the *block error probability* performance for any linear code. Determining the *bit error probability* requires knowledge of the number of bit errors associated with every possible block decoding error. An exact expression for bit error probability was given previously. One approach to estimating bit error probability is to use bounds that are a function of block error probability. When a block decoding error is made, a minimum of one bit error will always occur. Since k information bits are associated with each block, a single bit error corresponds to an average bit error probability of P_B/k. Thus a lower bound on bit error probability is

$$\frac{P_B}{k} \leq P_b \tag{6–70}$$

Similarly, not more than k-bit errors will occur with each block decoding error. Since k bits are decoded per block, an upper bound on bit error probability is

$$P_b \leq P_B \tag{6–71}$$

These bounds are often adequate for system design purposes.

For linear systematic block codes, improved bit error probability estimates can be made. Recall that, for a linear code, the number of codewords that are Hamming distance d from any particular codeword is the same for all codewords. Using this fact, the block error probability calculation was simplified to the calculation of the block error probability assuming that codeword **0** was transmitted. The brute-force calculation of bit error probability weights each possible block error event by the associated number of bit errors as in (6–41). Using arguments similar to those used to simplify the block error probability calculation, it can be shown that for linear systematic block codes the average bit error probability calculation simplifies to the calculation of bit error probability assuming the **0** codeword was transmitted. This simplification is possible because (1) the number of codewords Hamming distance d from any particular codeword is the same for all codewords, and (2) the number of bit errors caused by any Hamming distance d block decoding error is the same for all codewords. Consider, for example, the (7, 4) Hamming code. If message 7 was transmitted and message 13 was decoded, two bit errors would occur. The Hamming distance between the transmitted and decoded codewords is 4. There is a corresponding error event associated with the transmission of the **0** codeword. Specifically, if message 0 was transmitted and message 9 was decoded, two bit errors would occur. The Hamming distance between the transmitted and decoded codewords is 4.

The probability of information bit error P_b is calculated by assuming that message **0** is transmitted and weighting each block error event by the associated number of information bit errors. Let $B(m')$ denote the number of bit errors that occur when $m = 0$ is transmitted and m' is decoded. The average bit error probability is then bounded by extending the bound of (6–67)

$$P_b \leq \sum_{m'=1}^{2^k-1} \frac{1}{k} B(m') P'_B(0, m') \tag{6–72}$$

where $P'_B(0, m')$ is the probability of incorrectly decoding m'. Observe that $P'_B(0, m')$ is a function of the Hamming distance between **0** and $\mathbf{x}_{m'}$ and is the same for all codewords with the same Hamming distance. As before let $P'_B(d)$ denote the value of $P'_B(m, m')$ for all m and m' for which $d_H(\mathbf{x}_m, \mathbf{x}_{m'}) = d$. Let B_d denote the total number of bit errors that occur in all block error events involving codewords that are distance d from the all-zeros codeword. Equation (6–72) can be rewritten as

$$P_b < \frac{1}{k} \sum_{d=d_{\min}}^{n} B_d P'_B(d) \tag{6–73}$$

where $d_{\min}$ is the minimum distance of the code. This equation provides a good bound on bit error probability for any particular code. The values of B_d may be given or may be calculated using a computer for any particular code. For low error rates, the bit error probability is often bounded using only the first term of this equation.

The usefulness of the techniques just described are limited by the availability of values for B_d for any specific code. For moderate k, B_d can be found with a digital computer. However, for large k other techniques are used. Specifically, once again limit consideration to systematic codes and the bounded distance decoder. With a bounded distance decoder, the received vector will be changed ("corrected") in at most M positions by the decoder. Following Odenwalder [23], in the best case these changes will correct information bit errors in the information portion of the systematic codeword. In the worst case, all changes will still be in the information portion of the codeword but will change already correct information bits making them incorrect. Thus, when $i \leq M$ channel errors occur, correct decoding occurs and no bit errors occur. When $i > M$ channel errors occur, a block decoding error occurs and the maximum number of information bit errors is $\min[k, i + M]$. Clearly no more than all of the information bits can be incorrect and $i + M$ information bit errors can occur if all channel errors occur in the information portion of the codeword and M correct bits are changed by the decoder. Therefore, the bit error probability is overbounded by

$$P_b \leq \frac{1}{k} \sum_{i=M+1}^{n} \min[k, i + M] \binom{n}{i} p^i (1 - p)^{n-i} \tag{6–74}$$

where the $1/k$ factor is due to the fact that k information bits are decoded per block. The binomial coefficient is the usual number of ways in which i errors can occur within a block of n symbols.

Many formulas for block and bit error probability have been given in this discussion. These results are useful when used prudently by the system designer. When specific codes are discussed later, results will be calculated using a number of these methods so that the reader can begin to appreciate the conditions under which various formulas are appropriate.

6.3.3 Cyclic Codes

Recall from Section 6.3.1 that an (n, k) block code is simply a smart mapping of k information bits to n symbol codewords. The codewords and the mapping must be carefully chosen for the code to be useful with respect to improving the performance of the communication system. With no further constraints on code structure the encoding of an (n, k) block code is accomplished using table lookup. In general decoding can be accomplished using the minimum distance decoding rule also using table lookup. For large n and k, table lookup cannot be used due to the size of the required tables. In Section 6.3.2 consideration was limited to linear block codes. The linearity requirement enabled significant simplification of the encoding and decoding processes. Encoding of linear codes can be accomplished by a matrix multiplication of the information k-vector by a generator matrix $\mathbf{G}$. Decoding of linear codes

was greatly simplified using the standard array and the concept of the syndrome. In this section the encoding and decoding functions are simplified further by requiring the linear code to have additional structure. This additional structure will enable the encoder to take the form of a simple feedforward shift register. Similarly, the decoder will be a feedback shift register with some additional logic used to calculate the error vector. The subject of this section is cyclic linear codes which are a subset of all the linear codes.[5]

6.3.3.1 Definition of Cyclic Codes. A *cyclic code* is a linear code that has, in addition to the normal properties of a linear code, the property that any cyclic shift of a codeword is also a valid codeword. Thus if $\mathbf{x} = x_0\, x_1\, x_2 \cdots x_{n-2}\, x_{n-1}$ is a codeword in a cyclic code, $\mathbf{x}' = x_{n-1}\, x_0\, x_1 x_2 \cdots x_{n-2}$ is also a codeword of the code. Applying this property repeatedly, the qth cyclic shift, denoted $\mathbf{x}^{(q)}$, of a codeword $\mathbf{x}$ is also a codeword where $\mathbf{x}^{(q)} = x_{n-q}\, x_{n-q+1} \cdots x_0 x_1 \cdots x_{n-q-1}$. Cyclic codes can be described using a generator matrix $\mathbf{G}$ or parity matrix $\mathbf{H}$; however, they are usually described using the concept of a generator polynomial. Before defining the generator polynomial, the arithmetic of polynomial multiplication and division will be reviewed.

6.3.3.2 Polynomial Arithmetic. Let $\mathbf{w}(D) = w_0 + w_1 D + w_2 D^2 + \cdots + w_{k-1} D^{k-1}$ be a polynomial in D with each w_j from the set $\{0, 1\}$. The degree of this polynomial is the largest power of D that has a nonzero coefficient. For example, the degree of $\mathbf{w}(D) = 1 + D + D^4$ is 4. It is convenient to not show polynomial terms with zero coefficients and also to not show the multiplication by 1 explicitly. That is, write $\mathbf{w}(D) = (1 \cdot D^0) + (1 \cdot D^1) + (0 \cdot D^2) + (0 \cdot D^3) + (1 \cdot D^4)$ as $\mathbf{w}(D) = 1 + D + D^4$. This polynomial is yet another means of representing the k-bit information vector $\mathbf{w} = w_0\, w_1\, w_2 \cdots w_{k-1}$. Let $\mathbf{g}(D) = g_0 + g_1 D + g_2 D^2 + \cdots + g_{n-k} D^{n-k}$ be another polynomial. Define the polynomials $\mathbf{y}(D)$, $\mathbf{x}(D)$, and $\mathbf{e}(D)$ similarly. The modulo-2 *sum of two polynomials* $\mathbf{y}(D) = \mathbf{x}(D) + \mathbf{e}(D)$ is a polynomial $\mathbf{y}(D) = y_0 + y_1 D + y_2 D^2 + \cdots + y_t D^t$ whose coefficients are the modulo-2 sums of the coefficients of like powers of D in $\mathbf{x}(D)$ and $\mathbf{e}(D)$. That is, $y_j = x_j + e_j$. The *product of two polynomials* is a polynomial $\mathbf{x}(D) = x_0 + x_1 D + x_2 D^2 + \cdots + x_{n-1} D^{n-1}$ with coefficients given by

$$
\begin{aligned}
\mathbf{g}(D)\mathbf{w}(D) &= (g_{n-k} \cdot w_{k-1})\, D^{n-1} \\
&\quad + (g_{n-k} \cdot w_{k-2} + g_{n-k-1} \cdot w_{k-1})\, D^{n-2} \\
&\quad + (g_{n-k} \cdot w_{k-3} + g_{n-k-1} \cdot w_{k-2} + g_{n-k-2} \cdot w_{k-1})\, D^{n-3} \\
&\quad \vdots \\
&\quad + (g_0 \cdot w_0)\, D^0 \\
&= x_{n-1}\, D^{n-1} + x_{n-2}\, D^{n-2} + \cdots + x_0\, D^0
\end{aligned}
\tag{6–75}
$$

$$
x_j = \sum_{i=0}^{j} w_i \cdot g_{j-i}
$$

[5]The description of cyclic codes in this section follows the description in Lin and Costello [8]. Significant additional detail can be found in this reference.

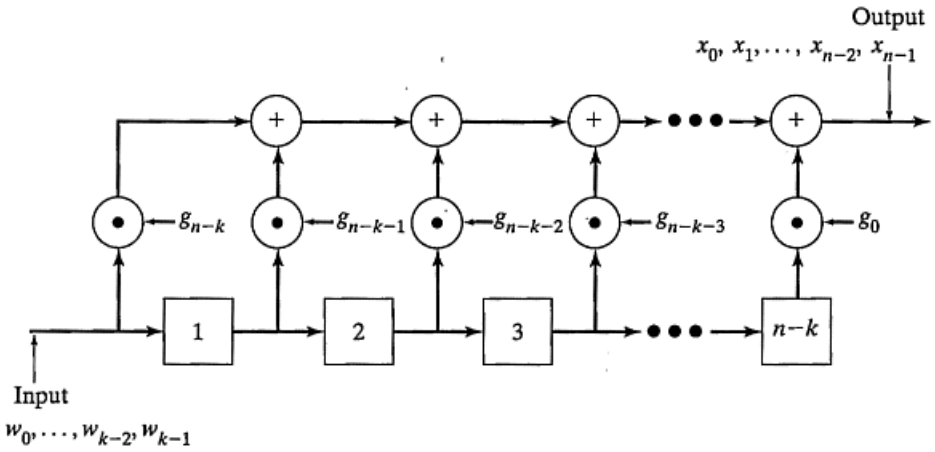
where all arithmetic is modulo-2 and coefficients outside the valid range of a polynomial are assumed equal to zero. The polynomial $\mathbf{x}(D)$ is another means of representing a code vector $\mathbf{x} = x_0\, x_1\, x_2 \cdots x_{n-1}$.

Equation (6–75) suggests a simple circuit that can perform polynomial multiplication. Figure 6–14 is a conceptual illustration of a circuit for calculating $\mathbf{x}(D) = \mathbf{w}(D) \cdot \mathbf{g}(D)$. Initially the shift registers of Figure 6–14 contain all 0s. The coefficients of $\mathbf{w}(D)$ are input one at a time starting with w_{k-1}. The first output on the right of the figure is $x_{n-1} = w_{k-1} \cdot g_{n-k}$. The shift register is now clocked once so that w_{k-1} is stored in register 1. The output is $x_{n-2} = (w_{k-2} \cdot g_{n-k}) + (w_{k-1} \cdot g_{n-k-1})$. The shift register is clocked a total of $n-1$ times to produce all n coefficients of $\mathbf{x}(D)$. After the kth input coefficient, 0s are input to the shift register. It will be seen later that circuits such as this greatly simplify the encoding for cyclic codes.

Next consider the quotient of the two polynomials $\mathbf{x}(D) \div \mathbf{g}(D)$. This division is performed using the normal rules of polynomial division however modulo-2 arithmetic is used everywhere. The division ends with a remainder $\rho(D)$ if $\mathbf{x}(D)$ is not divisible by $\mathbf{g}(D)$. Denoting the quotient by $\mathbf{q}(D)$

$$\mathbf{x}(D) = \mathbf{q}(D) \cdot \mathbf{g}(D) + \rho(D) \tag{6–76}$$

The remainder $\rho(D)$ is very important in the study of cyclic codes. Modulo-2 polynomial division is most easily explained by example. Therefore consider the division of $\mathbf{x}(D) = 1 + D + D^2 + D^4 + D^5$ by $\mathbf{g}(D) = 1 + D + D^3$. This long division is



1) Input high order coefficients first

2) First output is highest order coefficient $x_{n-1} = w_{k-1} g_{n-k}$ and is present at output before first shift register clock pulse

3) Clock shift register $(n-1)$ times using zero inputs following w_0

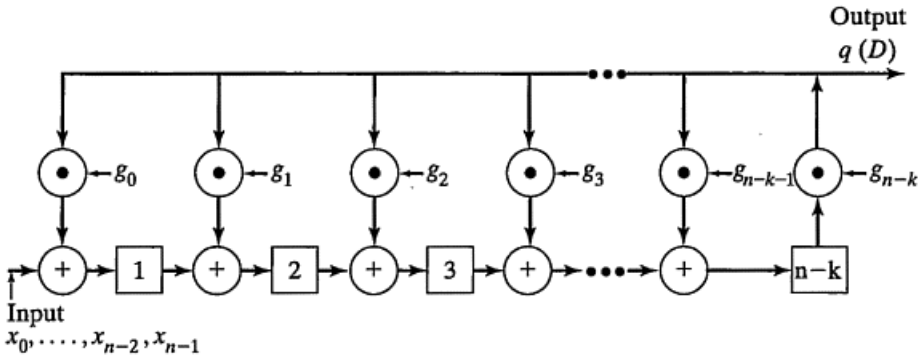
4) $x_0 = w_0 g_0$ is present at output following $(n-1)^{th}$ clock pulse

FIGURE 6–14 Circuit for multiplying polynomials $w(D)g(D) = x(D)$. (Reproduced from Reference [8] with permission.)

$$
\begin{array}{r}
\qquad\qquad\qquad\qquad\qquad\qquad D^2+D+1 \leftarrow \text{Quotient} \\
D^3+D+1 \;\overline{\big|\; D^5+D^4 \qquad\quad +D^2+D+1} \\
\underline{D^5+\qquad D^3+D^2\qquad\qquad\qquad} \\
\qquad\quad D^4+D^3+\qquad D+1 \\
\qquad\quad \underline{D^4+\qquad D^2+D\qquad\quad} \\
\qquad\qquad\quad D^3+D^2+\qquad 1 \\
\qquad\qquad\quad \underline{D^3+\qquad D+1} \\
\qquad\qquad\qquad\quad D^2+D \qquad \leftarrow \text{Remainder}
\end{array}
$$

Observe that the order of the terms is reversed from the order used when writing the polynomials themselves. Again, in nearly all cases the result of interest for cyclic codes is the remainder. The remainder of this division is denoted by $R_{\mathbf{g}(D)}[\mathbf{x}(D)] = \rho(D)$.

Although it is not obvious from this discussion, a simple circuit exists for performing polynomial division. Figure 6–15 is a conceptual illustration of a division circuit for dividing $\mathbf{x}(D)$ by $\mathbf{g}(D)$. As in polynomial multiplication, the high-order coefficients are input and output first as explained in the notes of Figure 6–15. With $\mathbf{x}$ of degree $n-1$ and $\mathbf{g}$ of degree $n-k$ the shift register must be clocked a total $n-1$ times to produce all coefficients of the quotient $\mathbf{q}$. A last shift is required to produce the remainder, which appears as the contents of the shift register with lowest-order coefficient on the left. Figure 6–16



1) Input high order coefficients first

2) First output is coefficient of D^{n-1} of quotient (always equal to zero but mentioned here to associate outputs with correct power of D in quotient) and is present before first shift register clock pulse

3) First non- zero output occurs after $(n-k)^{\text{th}}$ clock pulse and is coefficient of D^{n-k} in quotient

4) Last term of quotient appears at output after $(n-1)^{\text{th}}$ clock pulse and is coefficient of D^0 in quotient

5) Shift register contains coefficients of remainder $r(D) = r_0 + r_1 D + \ldots r_{n-k-1} D^{n-k-1}$ from left to right after n^{th} clock pulse

FIGURE 6–15 Circuit for dividing $x(D)$ by $g(D)$.

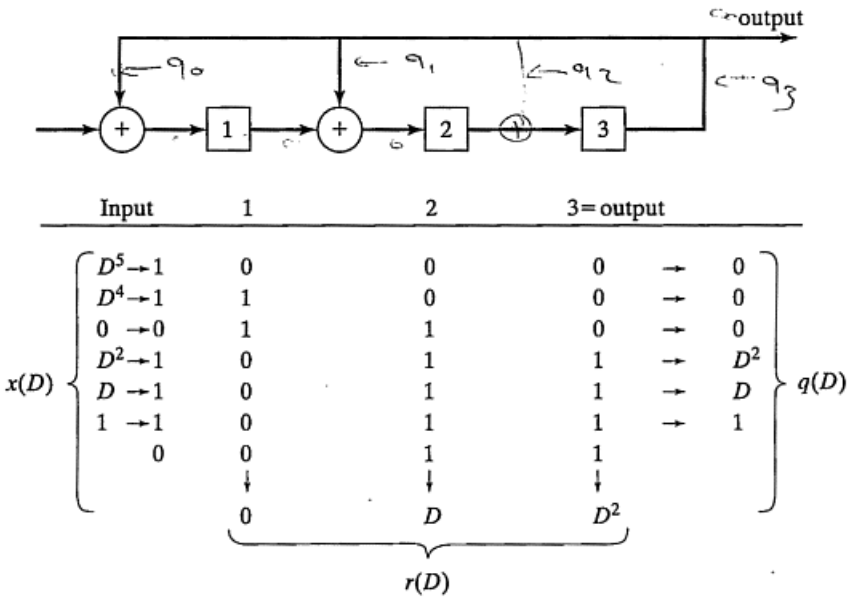


FIGURE 6.16 Circuit for dividing $x(D) = D^5 + D^4 + D^2 + D + 1$ by $g(D) = D^3 + D + 1$.

illustrates the operation of the shift register for the polynomial long division just performed.

6.3.3.3 Properties of Cyclic Codes. Return now to the discussion of cyclic codes. For any (n, k) cyclic code all of the code vectors (code polynomials) can be represented as the product of a generator polynomial $\mathbf{g}(D)$ having degree $r = n - k$ and a message polynomial $\mathbf{w}(D)$ having degree $k - 1$. The product $\mathbf{x}(D) = \mathbf{w}(D) \cdot \mathbf{g}(D)$ has degree $(k - 1) + (n - k) = n - 1$ as expected. There are 2^k distinct polynomials $\mathbf{w}(D) = w_0 + w_1 D + w_2 D^2 + \cdots + w_{k-1} D^{k-1}$ corresponding to all possible binary choices for the coefficients $\mathbf{w}_j$. Therefore, there are 2^k code polynomials. The codewords of a binary cyclic code are easily generated using a circuit as shown in Figure 6–14. Finding the generator polynomials for cyclic codes is beyond the scope of this discussion; however, a few useful properties of the generator polynomials are the following:

1. The generator polynomial for any cyclic code is a factor of $D^n + 1$. Thus, there will be no remainder if the long division of $D^n + 1$ by $\mathbf{g}(D)$ is carried out. Further, any polynomial of degree $n - k$ that is a factor of $D^n + 1$ is the generator for an (n, k) cyclic code.
2. The degree of $\mathbf{g}(D)$ is always $n - k$.
3. The qth cyclic shift of a codeword is the remainder of the division of $D^q \mathbf{x}(D)$ by $D^n + 1$.
4. Given the generator polynomial $\mathbf{g}(D) = g_0 + g_1 D + g_2 D^2 + \cdots + g_{n-k} D^{n-k}$ for a cyclic code, the $k \times n$ generator matrix $\mathbf{G}$ is

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & \cdot & \cdot & g_{n-k-1} & g_{n-k} & 0 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & g_0 & g_1 & \cdot & \cdot & g_{n-k-1} & g_{n-k} & 0 & 0 & \cdot & \cdot & 0 \\ 0 & 0 & g_0 & g_1 & \cdot & \cdot & g_{n-k-1} & g_{n-k} & 0 & 0 & \cdot & 0 \\ \vdots & \vdots & & & & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & 0 & g_0 & g_1 & \cdot & \cdot & g_{n-k-1} & g_{n-k} \end{bmatrix} \quad (6\text{–}77)$$

5. The $(n-k) \times n$ parity check matrix for a cyclic code is

$$\begin{bmatrix} h_k & h_{k-1} & \cdot & \cdot & h_1 & h_0 & 0 & 0 & \cdot & \cdot & \cdot & 0 \\ 0 & h_k & h_{k-1} & \cdot & \cdot & h_1 & h_0 & 0 & 0 & \cdot & \cdot & 0 \\ 0 & 0 & h_k & h_{k-1} & \cdot & \cdot & h_1 & h_0 & 0 & 0 & \cdot & 0 \\ \vdots & \vdots & & & & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & 0 & h_k & h_{k-1} & \cdot & \cdot & h_1 & h_0 \end{bmatrix} \quad (6\text{–}78)$$

where $\mathbf{h}(D) = h_0 + h_1 D + h_2 D^2 + \cdots + h_k D^k$ is the result of the division of $D^n + 1$ by $\mathbf{g}(D)$. Note that there is no remainder due to property 1. The polynomial $\mathbf{h}(D)$ is called the *parity polynomial.*

6.3.3.4 Encoding of Cyclic Codes. Performing the multiplication of the generator polynomial by the message polynomial as described will generate all valid code polynomials; however, these polynomials will not be in systematic form. A simple procedure is described by Lin [8] that generates the code polynomials in systematic form using the polynomial division circuitry of Figure 6–15 or via manual polynomial long division. The procedure [8] is given here without proof:

Step 1. Premultiply the message $\mathbf{w}(D)$ by D^{n-k}.
Step 2. Obtain the remainder $\rho(D)$ of the division of $D^{n-k}\mathbf{w}(D)$ by the generator $\mathbf{g}(D)$.
Step 3. The code polynomial is $\mathbf{x}(D) = \rho(D) + D^{n-k}\mathbf{w}(D)$.

6.3.3.5 Decoding of Cyclic Codes. The structure of cyclic codes enables significant simplification of the decoding process relative to noncyclic linear codes. The justification for the decoding procedure which follows can be found in many excellent references on algebraic coding [8, 10, 11]; due to the complexity of the subject, no attempt at justification will be made here.

Assume that a code polynomial $\mathbf{x}(D)$ was transmitted and that errors were added in the channel. The errors are represented by an error polynomial $\mathbf{e}(D) = e_0 + e_1 D + \cdots + e_{n-1} D^{n-1}$, where $e_j = 1$ indicates that an error has occurred in the jth position. The received vector is represented by a polynomial $\mathbf{y}(D) = y_0 + y_1 D + \cdots + y_{n-1} D^{n-1} = \mathbf{x}(D) + \mathbf{e}(D)$. Just as was done for decoding with the standard array earlier, decoding of cyclic codes begins with the calculation of a syndrome polynomial $\mathbf{s}(D)$. The syndrome polynomial depends only on the error polynomial; that is, the syndrome of $\mathbf{x}_m(D) + \mathbf{e}(D)$ is the same for all messages m. The syndrome is the remainder when the received polynomial

$\mathbf{y}(D)$ is divided by $\mathbf{g}(D)$; that is $\mathbf{s}(D) = R_{\mathbf{g}(D)}[\mathbf{y}(D)]$. The syndrome is calculated using the circuitry of Figure 6–15. The syndrome is zero if and only if the received polynomial is identical to some code polynomial, which is the case if no errors were made or if the error polynomial is identical to some code polynomial.

The next step in the decoding process is the association of the syndrome with the most probable error polynomial. This procedure is dependent on the specific code being considered and therefore details cannot be presented here. Assume that a table is available that associates syndromes with error patterns via a table lookup. With the error polynomial determined from the syndrome, the decoding is completed by adding the estimated error polynomial to the received polynomial to determine the estimated code polynomial. In this process advantage is taken of the cyclic structure of the code. The cyclic structure enables the error estimates $e_0, e_1, \cdots, e_{n-1}$ to be made one at a time starting with e_{n-1}. The decoder operates by first estimating e_{n-1} from the initial syndrome. Then the decoder circuitry is clocked to generate another syndrome to estimate of e_{n-2} and so on through e_0. The simplification of the decoder is due to the fact that the same circuitry is used sequentially to estimate $e_{n-1}, e_{n-2}, \cdots, e_0$. As errors are detected by the decoder they are corrected in the received polynomial and at the same time the effect of the corrected error is removed from syndrome calculation circuitry. Figure 6–17 is conceptual block diagram of a decoder for a cyclic code. The received polynomial is input (highest-order term first) into a buffer register (top) and a long division circuit. Gates a, b, and d in Figure 6–17 are closed and gates c and e are open while the received sequence is being read into the de-

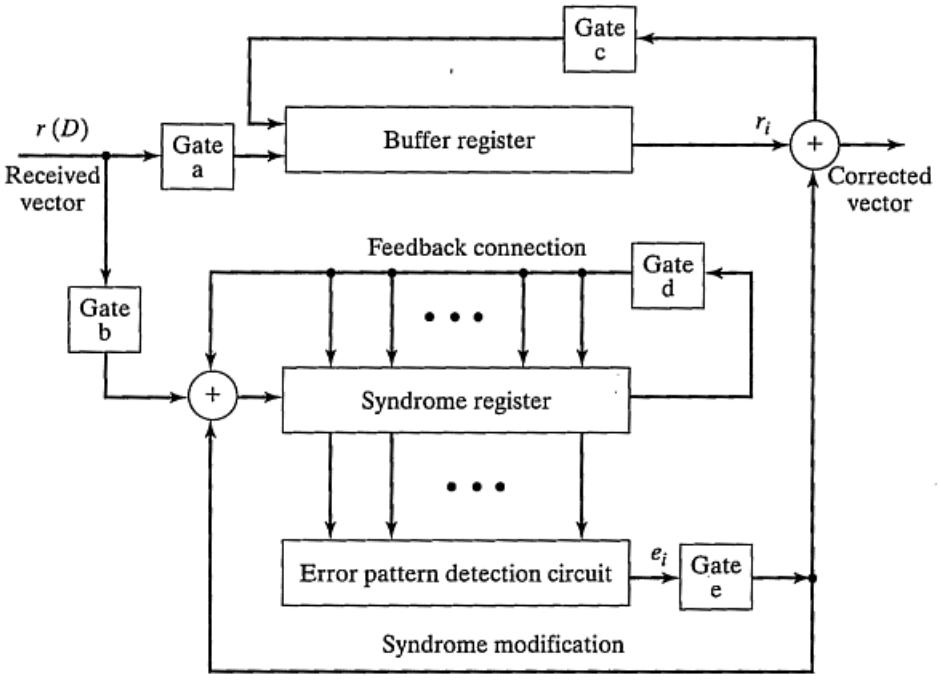


FIGURE 6.17 General cyclic code decoder with received polynomial $r(D)$ shifted into the syndrome register from the left end. (Reproduced from Reference [8] with permission.)

coder. After the received sequence is input, gates a, b and d are opened and gates c and e are closed. The long division performs the division of the received polynomial by the generator polynomial $\mathbf{g}(D)$to find the remainder $\rho(D)$, which is the first syndrome $\mathbf{s}(D)$. The syndrome is input to the "error pattern detection circuit," which will be code specific and can be quite simple. The error detection circuit outputs a 1 if it estimates that an error $e_{n-1} = 1$ has occurred in the highest-order position r_{n-1} of the received polynomial. The buffer register and the division register are simultaneously clocked to output the first decoded symbol which is corrected (if required) by the exclusive-or circuit in the upper right corner. The error estimate is simultaneously input to the division circuit to eliminate its effect on the remaining syndrome. This procedure is repeated until all n received symbols have been corrected. The complexity of the decoder for cyclic codes is proportional to the length n of the codewords. This is in contrast to the complexity of a table lookup scheme for an arbitrary block code, which must have 2^n entries (one for each possible received vector) or the complexity of a standard array decoder, which must have a table with an entry for each correctable error pattern.

6.3.4 Hamming Codes

6.3.4.1 Definition of Hamming Codes. Hamming codes are a family of codes discovered by R. W. Hamming [4] in about 1950. The Hamming codes all have minimum distance 3 and are able to correct a single error ($t = 1$) in a block of n and to detect all double errors. Hamming codes are linear cyclic systematic codes. Hamming codes with $n = 2^j - 1$ and $k = n - j$ exist for any integer $j \geq 3$. The rate of these codes is $R = k/n = (2^j - 1 - j)/(2^j - 1)$ which approaches 1.0 as j increases. Table 6–9 is a short list of n, k, and R for the first eight Hamming codes.

The Hamming codes are always defined by their parity check matrix, which has $n - k = j$ rows and n columns. The columns consist of all possible nonzero j-component vectors. For example, the parity check matrix for the (7, 4) Hamming code which has been the subject of many of the previous examples is

TABLE 6–9 Codes Rates for Hamming Codes

j	k	n	$R = k/n$
3	4	7	0.57
4	11	15	0.73
5	26	31	0.84
6	57	63	0.90
7	120	127	0.94
8	247	255	0.97
9	502	511	0.98
10	1013	1023	0.99

TABLE 6–10 List of Code Generators for Hamming Codes

j		j	
3	$1+D+D^3$	14	$1+D+D^6+D^{10}+D^{14}$
4	$1+D+D^4$	15	$1+D+D^{15}$
5	$1+D^2+D^5$	16	$1+D+D^3+D^{12}+D^{16}$
6	$1+D+D^6$	17	$1+D^3+D^{17}$
7	$1+D^3+D^7$	18	$1+D^7+D^{18}$
8	$1+D^2+D^3+D^4+D^8$	19	$1+D+D^2+D^5+D^{19}$
9	$1+D^4+D^9$	20	$1+D^3+D^{20}$
10	$1+D^3+D^{10}$	21	$1+D^2+D^{21}$
11	$1+D^2+D^{11}$	22	$1+D+D^{22}$
12	$1+D+D^4+D^6+D^{12}$	23	$1+D^5+D^{23}$
13	$1+D+D^3+D^4+D^{13}$	24	$1+D+D^2+D^7+D^{24}$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 1 \end{bmatrix} \tag{6–79}$$

Observe that all nonzero 3-tuples are used as the columns of this matrix. The generator matrix for the Hamming codes is found via the relationships of (6–53) and (6–54) for linear systematic codes. The generator matrix for this code was given in a previous example. It can be shown that this definition implies that the generator polynomial $\mathbf{g}(D)$ is a member of a special class of polynomials called *primitive polynomials.* Detailed treatments of these special polynomials can be found in advanced texts [6, 8, 10, 13]. Sets of primitive polynomials[6] can be found in these references. Table 6–10 is a short list of primitive polynomials (from [8]) that can be used as generators for Hamming codes up to block length $n = 2^{24} - 1$.

6.3.4.2 Encoding of Hamming Codes. The encoder used for the Hamming codes can take a variety of forms. If encoding speed is not critical, the encoder can be implemented in software by calculating the generator matrix multiplication directly. If speed is critical, however, other implementations are usually used. Figure 6–18 illustrates a straightforward implementation for a Hamming encoder that can be built using high-speed logic. The $(k - 1)$-bit shift register at the top of Figure 6–18 receives the information bits from the data source. After the input register is filled, the output of the modulo-2 adders are the correct codeword symbols. Each symbol is a modulo-2 sum of certain information bits defined by the generator matrix. The generator matrix can be found from the generator polynomial as was done in Section 6.3.3.3, specifically

[6]Also see Table 9–2.

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & \cdots & g_{n-k-1} & g_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_{n-k-1} & g_{n-k} & 0 & 0\cdots & 0 \\ 0 & 0 & g_0 & g_1 & \cdots & g_{n-k-1} & g_{n-k} & 0\,0\cdots & 0 \\ \vdots & \vdots & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & g_0 & g_1 & \cdots g_{n-k-1} & g_{n-k} \end{bmatrix} \quad (6\text{–}80)$$

The codeword symbols are strobed into the lower shift register by closing the switches below the modulo-2 adders. The codeword is then clocked out of the encoder to a modulator for transmission. The clock rates for the upper and lower shift registers are different with the lower register clock $1/R$ times the upper register clock rate. New information bits are read into the upper register at the same time that the codeword is being read out of the lower register. The encoder of Figure 6–18 can be used for any cyclic code with proper selection of the generator matrix coefficients. Observe that the complexity of this encoder is proportional to $n \times (n - k)$, which quickly becomes impractical for large block length.

Since Hamming codes are cyclic, a much simpler implementation of the encoder is possible. Recall that codewords for cyclic codes in systematic form are generated by (1) multiplying the message $\mathbf{w}(D)$ polynomial by D^{n-k}, (2) calculating the remainder $\rho(D) = R_{g(D)}[\mathbf{w}(D) \cdot D^{n-k}]$, and (3) $\mathbf{x}(D) = \rho(D) + \mathbf{w}(D) \cdot D^{n-k}$. The encoding circuit of Figure 6–19 performs exactly this calculation. This circuit is a small variation of the circuit of Figure 6–15 for dividing two polynomials. The input has been moved $n - k$ positions to the right, which implements the premultiplication of $\mathbf{w}(D)$ by D^{n-k}. Operation of the encoder begins with the switches in the positions shown. The k message bits are simultaneously input to the division circuitry and output to the user. After the last message bit is input, the shift register contains the remainder of the long division, and the positions of

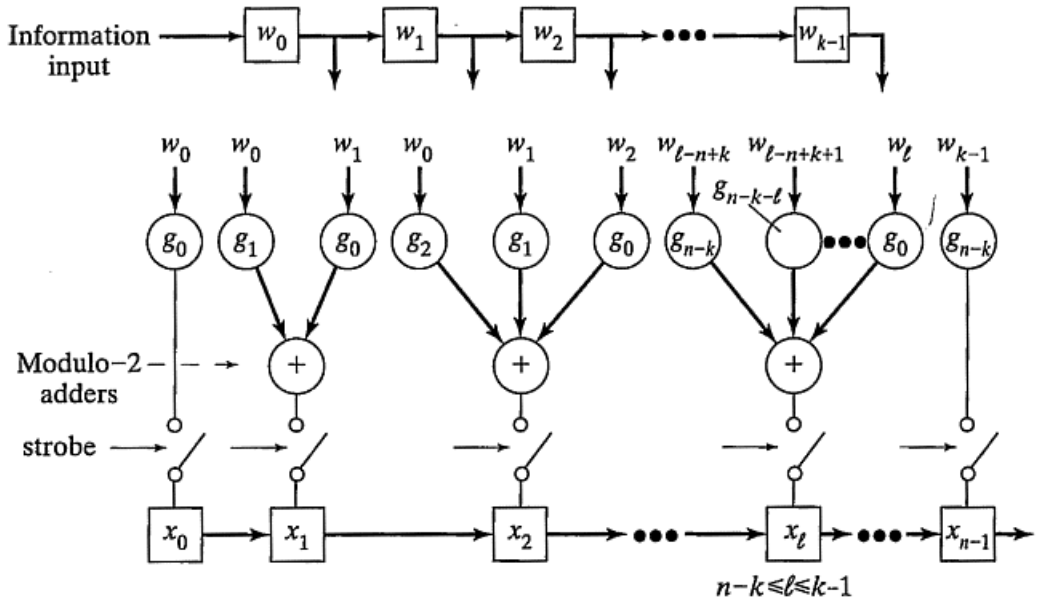


FIGURE 6–18 Hamming encoder.

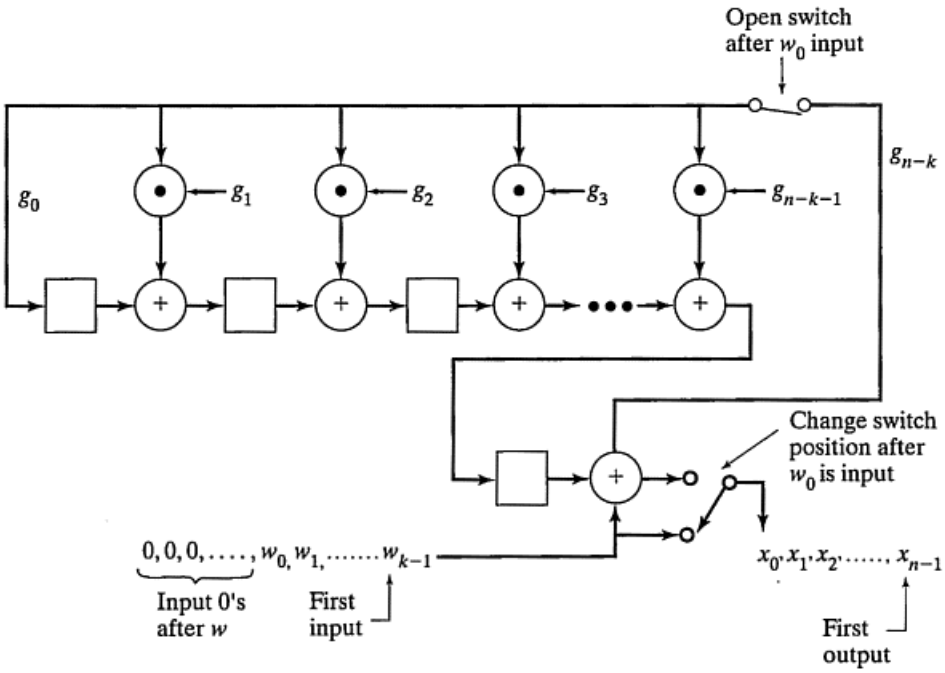


FIGURE 6–19 Feedback shift register implementation of Hamming encoder. (Reproduced from Reference [8] with permission.)

both switches are changed. The upper switch is opened and the output switch is changed to output the contents of the shift register. The shift register is clocked just enough times to output all of the parity bits to complete the encoding process. The complexity of this encoder is proportional to $n - k$, making it the clear choice of encoders for large n.

6.3.4.3 Decoding of Hamming Codes. Decoding of Hamming codes can be accomplished using any of the techniques described previously, including table lookup, the standard array, and generalized cyclic decoder of Figure 6–17. Because of its simplicity, only the decoder of Figure 6–17 will be discussed here. It can be shown that the error pattern detection circuit of Figure 6–17 takes a particularly simple form of an $(n - k)$-input AND gate for the single error correcting Hamming codes. The decoding circuit for Hamming codes is illustrated in Figure 6–20. An error is detected (and corrected) only when the syndrome contained in the shift register is $\mathbf{s}(D) = (0 \cdot D^0) + (0 \cdot D^1) + \cdots + (0 \cdot D^{n-k-2}) + (1 \cdot D^{n-k-1})$. The operation of the circuit is exactly as described earlier for Figure 6–17. Validation that this is the proper decoder can be found in [8]. The simplicity of this decoder is due to the algebraic structure of the code.

6.3.4.4 Performance of Hamming Codes

Block Error Probability. The block error probability P_B of the single error correcting codes is the probability that two or more errors occur during transmission. This probability is equal to 1.0 minus the probability that 0 or 1 errors occur, which is given by (6–57):

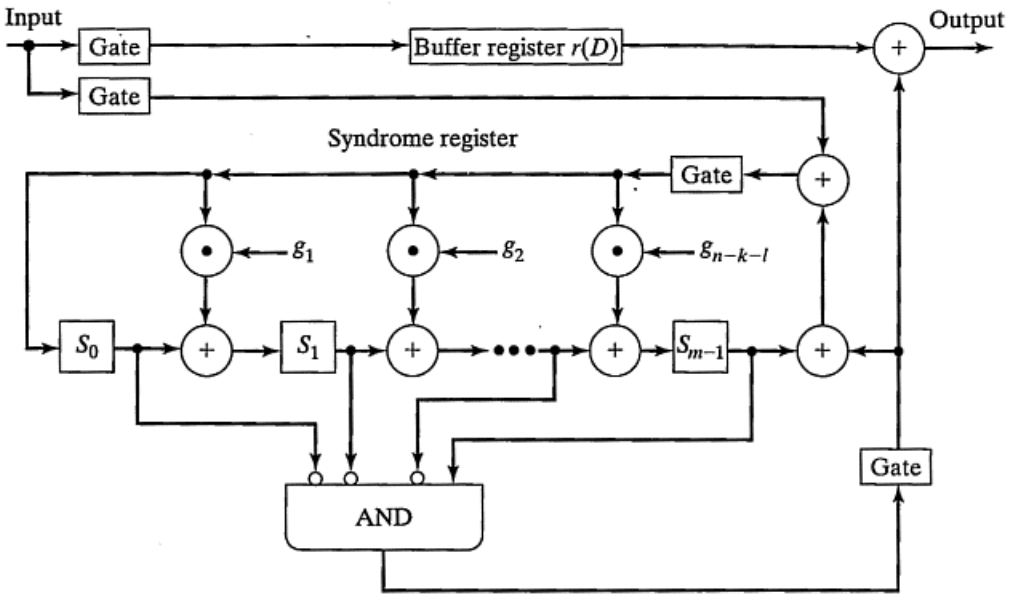


FIGURE 6–20 Decoder for cyclic Hamming code. (Reproduced from Reference [8] with permission.)

$$P_B = 1.0 - (1-p)^n - \sum_{l=1}^{n} p(1-p)^{n-1} \tag{6–81}$$

There is a single "error pattern" with no errors, and it occurs with probability $(1-p)^n$, and there are $n = 2^j - 1$ error patterns with a single error that occur with probability $p(1-p)^{n-1}$. Figure 6–21 illustrates P_B as a function of BSC crossover probability p for some of the Hamming codes of Table 6–10.

Because the Hamming decoder is able to correct all single errors and is not able to correct any other error patterns, the calculation of block error probability (6–81) was very simple. For other codes it may not be possible to derive such a simple result and the bounding techniques described earlier may have to be used. It is instructive to compare the results of the error probability bounds with the result just derived for particular codes. Consider the (7, 4) and (15, 11) Hamming codes that have $d_{\min} = 3$ and can correct a single error per block of 7 or 15 symbols, respectively. An upper bound on block error probability for *any* code is simply the probability that more than $\lfloor (d_{\min} - 1)/2 \rfloor = t$ errors occur. For the Hamming codes $t = 1$ and the probability that more than t errors occur is exactly the result just derived. The result is exact only for the Hamming codes and a few other codes known as "perfect codes."

Another bound on block error probability was given in (6–68):

$$P_B \leq \sum_{d=d_{\min}}^{n} a_d P_B'(d) \tag{6–82a}$$

$$\leq \sum_{d=d_{\min}}^{n} a_d [\sqrt{4p(1-p)}]^d \tag{6–82b}$$

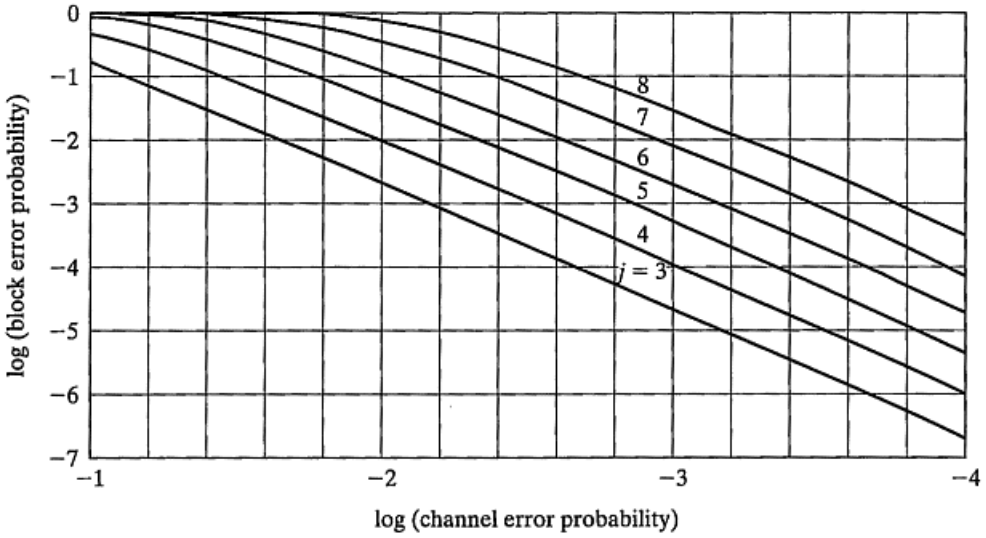


FIGURE 6–21 Block error probability for Hamming codes with $j = 3$ through 8 ($n = 2^j - 1, j \geq 3$).

where $P'_B(d)$ is the probability of decoding the **0** codeword as another codeword that is Hamming distance d from **0**. $P'_B(d)$ is evaluated using (6–63) or (6–64). The successful use of this result requires knowledge of the weight structure of the code. Fortunately, for the Hamming codes a reasonably simple means of determining the weight structure is available. For any Hamming code the number of codewords with Hamming weight d is the coefficient of z^d in the following polynomial in z:

$$\begin{aligned} A(z) &= \sum_{d=0}^{n} a_d z^d \\ &= \frac{1}{n+1}\left[(1+z)^n + n(1+z)^{(n-1)/2}(1-z)^{(n+1)/2}\right] \end{aligned} \tag{6–83}$$

For the (7, 4) Hamming code the result of the expansion is

$$A(z) = 1 + 7z^3 + 7z^4 + z^7$$

showing that there are 7 nonzero codewords with Hamming weight = 3, 7 with Hamming weight 4, and 1 with Hamming weight 7. This can be verified by examination of Table 6–4. For the (15, 11) code, the expansion is

$$\begin{aligned} A(z) = 1 &+ 35z^3 + 105z^4 + 168z^5 + 280z^6 \\ &+ 435z^7 + 435z^8 + 280z^9 + 168z^{10} + 105z^{11} + 35z^{12} + z^{15} \end{aligned}$$

Thus, for the (15, 11) code, $a_3 = 35$, $a_4 = 105$, $a_5 = 168$, and so on. These weight structures were used in (6–82) to calculate P_B as a function of the channel error probability; the results are shown in Figure 6–22. Two different bounds were calculated using the two different formulas of (6–82). Observe that the first bound is quite accurate, whereas the second bound, while easy to calculate, is highly conservative. The differences between the bounds illustrated in Figure 6–22 are a clear indicator that care should be used when selecting a bound for communications system design. A third bound could be calculated using (6–69), which presumes a bounded distance decoder. Because the code is a "perfect code," the performance of a bounded distance decoder and the maximum likelihood decoder are the same. Thus, (6–69) yields the same result for P_B as the exact result.

Bit Error Probability. Bit error probability can be calculated using any of the techniques described previously. Again the Hamming (7, 4) and (15, 11), codes will be considered. Bit error probability will be calculated using (6–70), (6–71), (6–73), and (6–74). Equation (6–73) is the union bound result, while (6–70) and (6–71) are upper and lower bounds which are slightly easier to calculate. Equation (6–74) is the upper bound derived assuming a bounded distance decoder. To use (6–73)

$$P_b < \frac{1}{k} \sum_{d=d_{\min}}^{n} B_d P_d \tag{6–84}$$

the coefficients B_d must be found. For the (7, 4) code these can be found by inspection of the codewords in Table 6–4. There are 7 nonzero codewords with weight 3, and the total

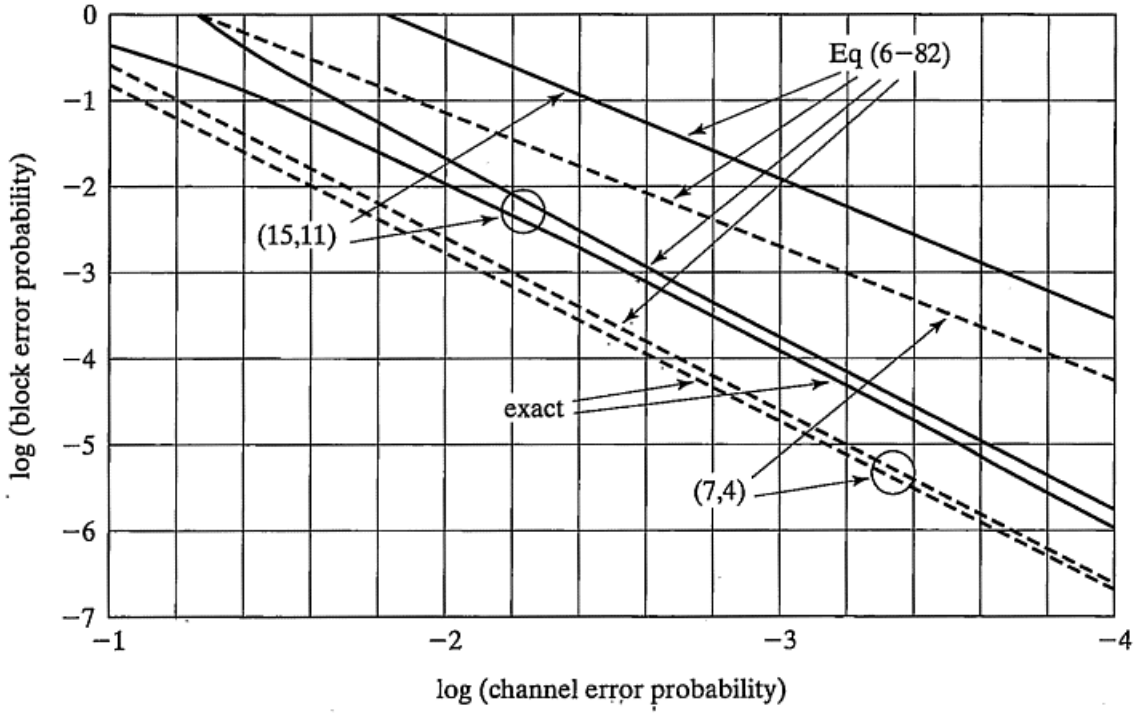


FIGURE 6–22 Comparison of block error probability bounds for (7, 4) and (15, 11) Hamming codes.

number of bit errors associated with these codewords is $B_3 = 1 + 1 + 2 + 2 + 1 + 3 + 2 = 12$. Similarly, there are 7 nonzero codewords with weight 4 and $B_4 = 2 + 1 + 3 + 2 + 2 + 3 + 3 = 16$ and $B_7 = 4$. All other $B_d = 0$. Thus, the bit error probability is bounded by

$$P_b < \frac{1}{4} \sum_{d=3}^{7} B_d P_d$$

$$= \frac{1}{4}(12P_3 + 16P_4 + 4P_7)$$

where P_d denotes $P_B'(d)$, which is calculated from (6–63) and (6–64). The values of B_d for the (15, 11) code were found by writing a short computer program that calculated each codeword in systematic form and then found the codeword weight and the number of attendant bit errors. The resulting equation for bit error probability is

$$P_b < \frac{1}{11} \sum_{d=3}^{15} B_d P_d$$

$$= \frac{1}{11}(77P_3 + 308P_4 + 616P_5 + 1232P_6$$

$$+ 2233P_7 + 2552P_8 + 1848P_9 + 1232P_{10}$$

$$+ 847P_{11} + 308P_{12} + 11P_{15})$$

Results for P_b as a function of BSC error probability p are shown in Figure 6–23 for the (7, 4) code and Figure 6–24 for the (15, 11) code. Bit error probability was also evaluated using the bounded distance decoder result (6–74). Thus, the four curves of Figures 6–23 and 6–24 are the upper bound of (6–71), the union bound of (6–73), the bounded distance decoder bound of (6–74), and the lower bound of (6–70). Examination of Figures 6–23 and 6–24 leads to the conjecture that the bounded distance result is reasonable in terms of its accuracy. This is expected since the Hamming codes are "perfect codes."

Bit error probability as a function of E_b/N_0 is plotted in Figure 6–25 for the Hamming codes with $j = 3$ to 7. BPSK signaling was assumed for Figure 6–25, and the error probability was calculated using (6–74). Comparison of exact result for the (7, 4) Hamming code presented in Figure 6–13 with the results of Figure 6–25 shows that the bound of (6–74) is quite accurate.

6.3.5 BCH Codes

6.3.5.1 Definition and Encoding for BCH Codes. Bose–Chaudhuri–Hocquenghem (BCH) codes were discovered independently by Hocquenghem [25] in 1959 and Bose and Chaudhuri [26] in 1960. These codes are among the most important block codes available since they exist for a wide range of rates, since they can achieve significant coding gain, and since the complexity of their decoders is such that they are

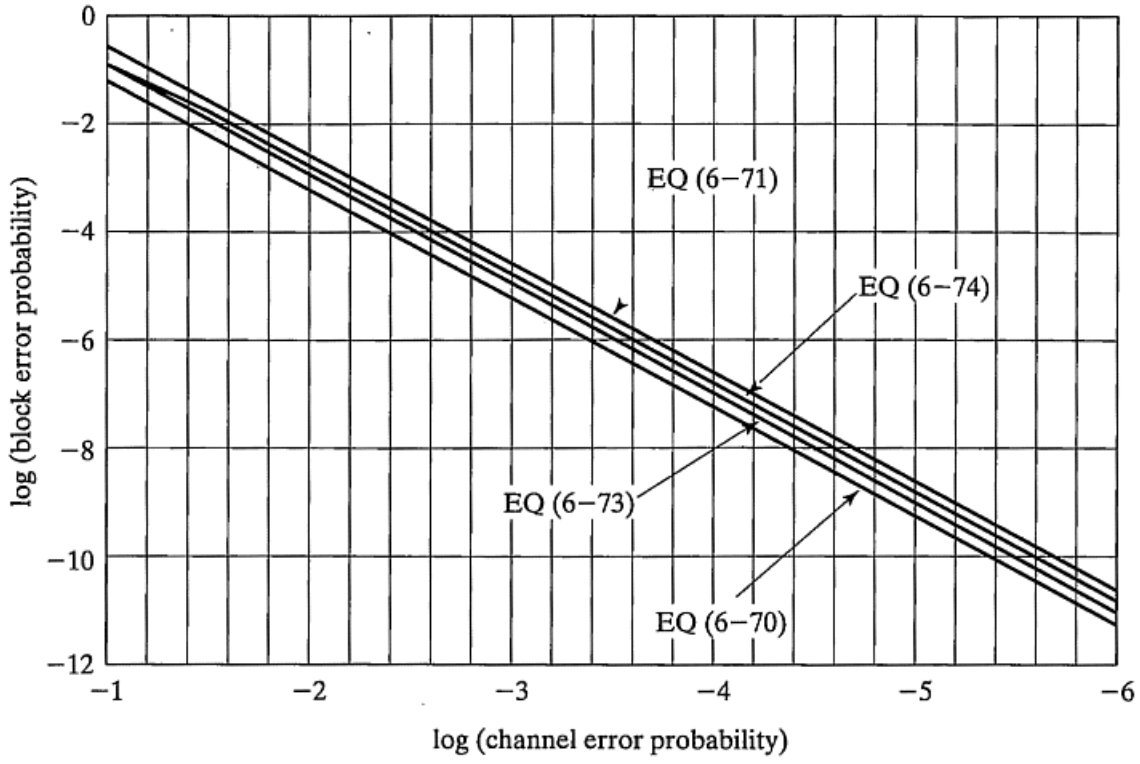


FIGURE 6–23 Bit error rate bounds for (7, 4) Hamming code.

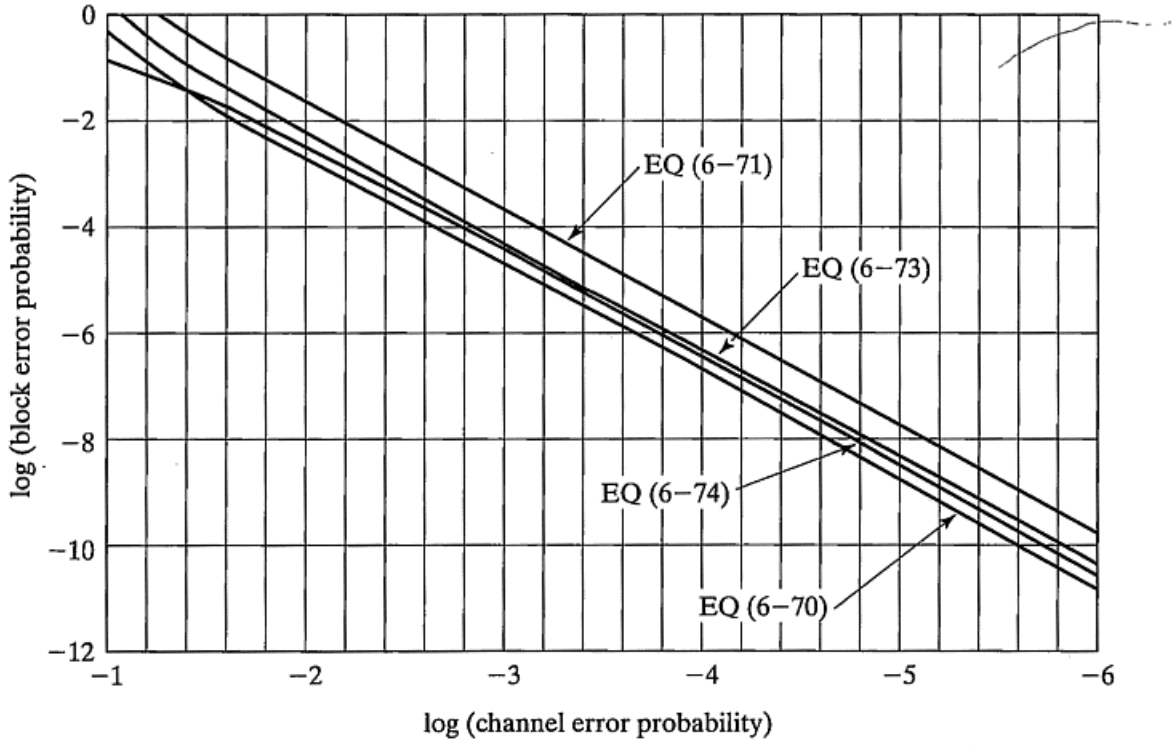


FIGURE 6–24 Bit error rate bounds for (15, 11) Hamming code.

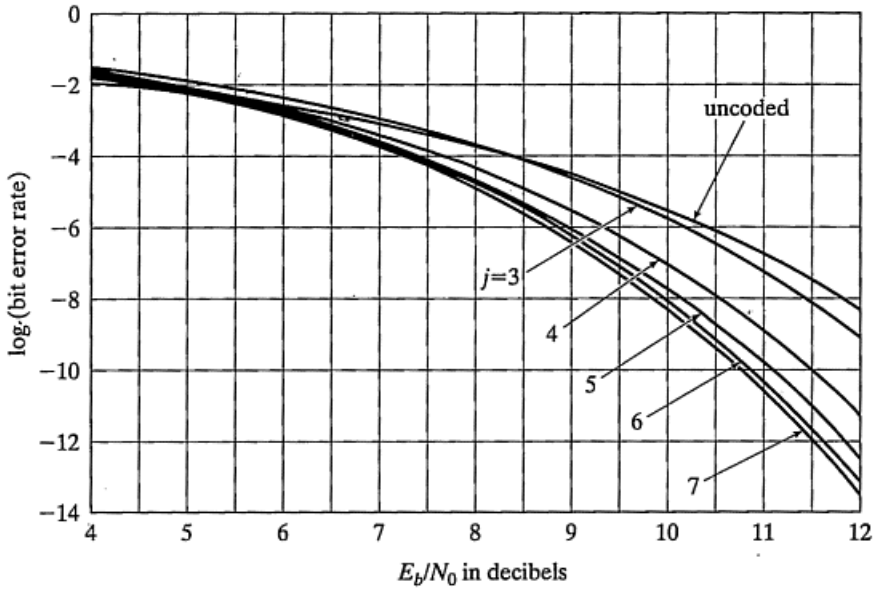


FIGURE 6–25 Bit error rate versus E_b/N_0 for Hamming codes with $j = 3$ through 7 ($n = 2^j - 1, j \geq 3$).

implementable even at high speeds. These codes have an elegant algebraic structure that is extremely interesting to study but is beyond the scope of this chapter. The reader is referred to advanced texts [6, 8, 9, 10, 11, 13] for all the theory behind the results to be presented. The BCH codes are linear cyclic codes that are always defined by their code generator polynomial. Although nonbinary BCH codes exist, discussion will be limited to the binary codes.

The block length n for BCH codes is always $n = 2^m - 1$ for $m \geq 3$, and the number of errors t that can be corrected is bounded by $t < (2^m - 1)/2$. It is also always true that $n - k \leq mt$. Specific values for t and k can be found using algebraic techniques for determining code polynomials; not all values satisfying the inequalities above are possible. Table 6–11 [8] gives all of the known values for n, k, and t for BCH codes with block lengths up to $n = 1023$. Observe that the code rate $R = k/n$ varies over a wide range and that the number of errors that can be corrected increases as the code rate decreases. The BCH code having the highest rate for any block length is a Hamming code. That is, the Hamming codes are a subset of the BCH codes. The code polynomials which generate the codes of Table 6–11 can be found in Lin and Costello [8] in their Appendix C. Another complete list of codes can be found in Peterson and Weldon [13], Table 9–1. Table 6–12 [8] is a short list of the BCH code generator polynomials for block length 63. BCH codes are applied in a variety of modern communications systems. Perhaps the best known application is in first generation cellular mobile telephony where a shortened BCH code is used for the messages that tell the mobile unit, among other things, what power and channel to use.

The BCH codes are cyclic codes so that encoding is simply accomplished using the general technique for cyclic codes discussed previously. Given the code generator $\mathbf{g}(D)$, the codewords in systematic form are found using the following steps: (1) multiply the

TABLE 6–11 List of Possible BCH Code Parameters

n	k	t	n	k	t	n	k	t
7	4	1	255	163	12	511	268	29
15	11	1		155	13		259	30
	7	2		147	14		250	31
	5	3		139	15		241	36
31	26	1		131	18		238	37
	21	2		123	19		229	38
	16	3		115	21		220	39
	11	5		107	22		211	41
	6	7		99	23		202	42
63	57	1		91	25		193	43
	51	2		87	26		184	45
	45	3		79	27		175	46
	39	4		71	29		166	47
	36	5		63	30		157	51
	30	6		55	31		148	53
	24	7		47	42		139	54
	18	10		45	43		130	55
	16	11		37	45		121	58
	10	13		29	47		112	59
	7	15		21	55		103	61
127	120	1		13	59		94	62
	113	2		9	63		85	63
	106	3	511	502	1		76	85
	99	4		493	2		67	87
	92	5		484	3		58	91
	85	6		475	4		49	93
	78	7		466	5		40	95
	71	9		457	6		31	109
	64	10		448	7		28	111
	57	11		439	8		19	119
	50	13		430	9		10	121
	43	14		421	10	1023	1013	1
	36	15		412	11		1003	2
	29	21		403	12		993	3
	22	23		394	13		983	4
	15	27		385	14		973	5
	8	31		376	15		963	6
255	247	1		367	16		953	7
	239	2		358	18		943	8
	231	3		349	19		933	9
	223	4		340	20		923	10
	215	5		331	21		913	11
	207	6		322	22		903	12

(*continued*)

TABLE 6–11 *Continued*

n	k	t	n	k	t	n	k	t
	199	7		313	23		893	13
	191	8		304	25		883	14
	187	9		295	26		873	15
	179	10		286	27		863	16
	171	112		277	28		858	17
1023	848	18	1023	553	52	1023	268	103
	838	19		543	53		258	106
	828	20		533	54		248	107
	818	21		523	55		238	109
	808	22		513	57		228	110
	798	23		503	58		218	111
	788	24		493	59		208	115
	778	25		483	60		203	117
	768	26		473	61		193	118
	758	27		463	62		183	119
	748	28		453	63		173	122
	738	29		443	73		163	123
	728	30		433	74		153	125
	718	31		423	75		143	126
	708	34		413	77		133	127
	698	35		403	78		123	170
	688	36		393	79		121	171
	678	37		383	82		111	173
	668	38		378	83		101	175
	658	39		368	85		91	181
	648	41		358	86		86	183
	638	42		348	87		76	187
	628	43		338	89		66	189
	618	44		328	90		56	191
	608	45		318	91		46	219
	598	46		308	93		36	223
	588	47		298	94		26	239
	578	49		288	95		16	147
	573	50		278	102		11	255
	563	51						

TABLE 6–12 Generator Polynomials of All the BCH Codes of Length 63

n	k	t	$g(D)$
63	57	1	$g_1(D) = 1 + D + D^6$
	51	2	$g_2(D) = (1 + D + D^6)(1 + D + D^2 + D^4 + D^6)$
	45	3	$g_3(D) = (1 + D + D^2 + D^5 + D^6)\, g_2(D)$
	39	4	$g_4(D) = (1 + D^3 + D^6)\, g_3(D)$
	36	5	$g_5(D) = (1 + D^2 + D^3)\, g_4(D)$
	30	6	$g_6(D) = (1 + D^2 + D^3 + D^5 + D^6)\, g_5(D)$
	24	7	$g_7(D) = (1 + D + D^3 + D^4 + D^6)\, g_6(D)$
	18	10	$g_{10}(D) = (1 + D^2 + D^4 + D^5 + D^6)\, g_7(D)$
	16	11	$g_{11}(D) = (1 + D + D^2)\, g_{10}(D)$
	10	13	$g_{13}(D) = (1 + D + D^4 + D^5 + D^6)\, g_{11}(D)$
	7	15	$g_{15}(D) = (1 + D + D^3)\, g_{13}(D)$

Source: Reproduced from Reference [8] with permission.

message polynomial $\mathbf{w}(D)$ by D^{n-k}, (2) calculate the remainder $\rho(D) = R_{\mathbf{g}(D)}[D^{n-k}\mathbf{w}(D)]$, and (3) generate the code polynomial $\mathbf{x}(D) = \rho(D) + D^{n-k}\mathbf{w}(D)$. A simple circuit for generating the systematic codewords for Hamming codes was given in Figure 6–19. The encoder for BCH codes is the same.

6.3.5.2 Decoding of BCH Codes [27]. The foundation for all BCH decoding algorithms is the algebraic structure of the codes. The decoders are straightforward once the algebraic fundamentals are established. Decoding consists of the same three steps given previously for any cyclic code. The function of the decoder is to produce an estimate of the error polynomial that was most likely to have occurred given the received polynomial. The decoding procedure is

1. Calculate a syndrome from the received code polynomial. There are $2t$ components of the syndrome $S_1, S_2, \ldots, S_{2t}$. The syndrome of an undistorted code polynomial is zero so that it is a function only of the transmission errors and the code structure. Calculation of each component of the syndrome is done by finding the remainder of division of the received sequence by a polynomial $\phi_i(D)$, which was specified at the time of the definition of the code. The polynomial long division is accomplished using the division circuitry of Figure 6–15. The same division circuitry may sometimes be used to calculate several syndrome components. Figure 6–26 is a simple example of a BCH syndrome calculation circuit from [8]. In this figure four syndromes are calculated for a (15, 7) BCH code using two long division circuits. There are at most t different division circuits required for a t error correcting code.
2. Determine the positions of the errors in the received polynomial using a two-step procedure:

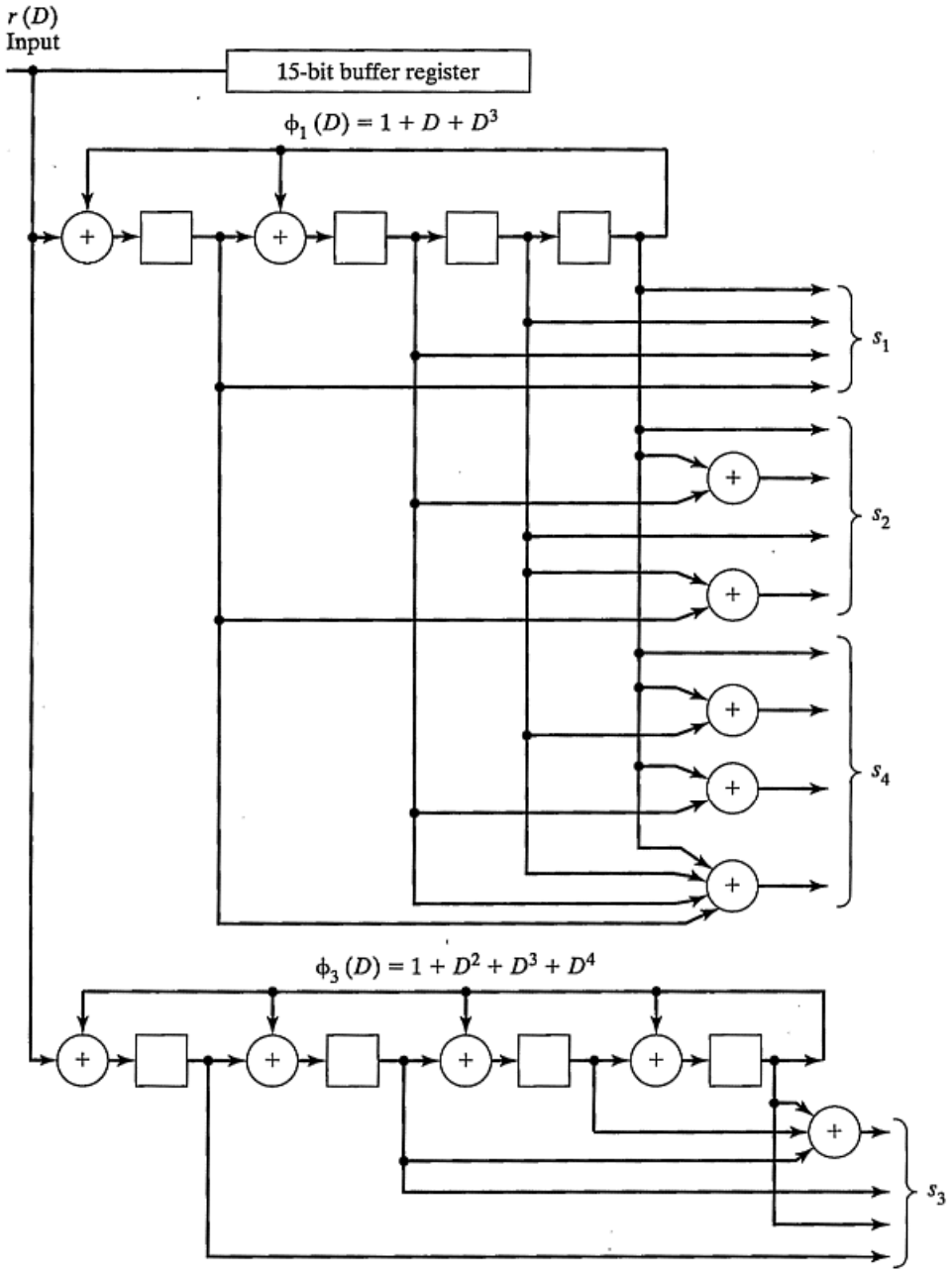


FIGURE 6–26 Syndrome computation circuit for the double error correcting (15, 7) BCH Code. (Reproduced from Reference [8] with permission.)

a. Determine an *error location polynomial* from the syndrome components found in step 1. An interactive algorithm is available [10] for finding this polynomial. This algorithm is called the Berlekamp algorithm, after its inventor. The complexity of this algorithm is proportional to $2t^2$.
b. Find the roots of the polynomial found in step 2a. These roots directly determine the locations of the errors in the received polynomial. A circuit for find-

ing the roots and correcting errors for the (15, 7) BCH code is illustrated in Figure 6–27, which is taken from [8]. The initial shift register loads for Figure 6–27 are the coefficients of the polynomial found in step 2a. The root finding procedure that Figure 6–27 executes is called the *Chien search algorithm* [6, 28].

3. Correct the errors in the received polynomial to find the transmitted codeword and thus the transmitted information.

This brief description of a BCH decoding algorithm is intended only to communicate that, although the decoding is based on concepts beyond this text, the decoding algorithm is well defined. BCH decoders are commercially available and are widely used.

6.3.5.3 Performance of BCH Codes. The coding gain achievable using BCH codes varies as the code parameters n and k vary. Bit error probability for BCH codes is calculated using (6–74), which is a small modification of results from [23, 29]. Equation (6–74) is an overbound on decoding bit error probability assuming a bounded distance decoder. Note that most of the known decoding algorithms for BCH codes are, in fact,

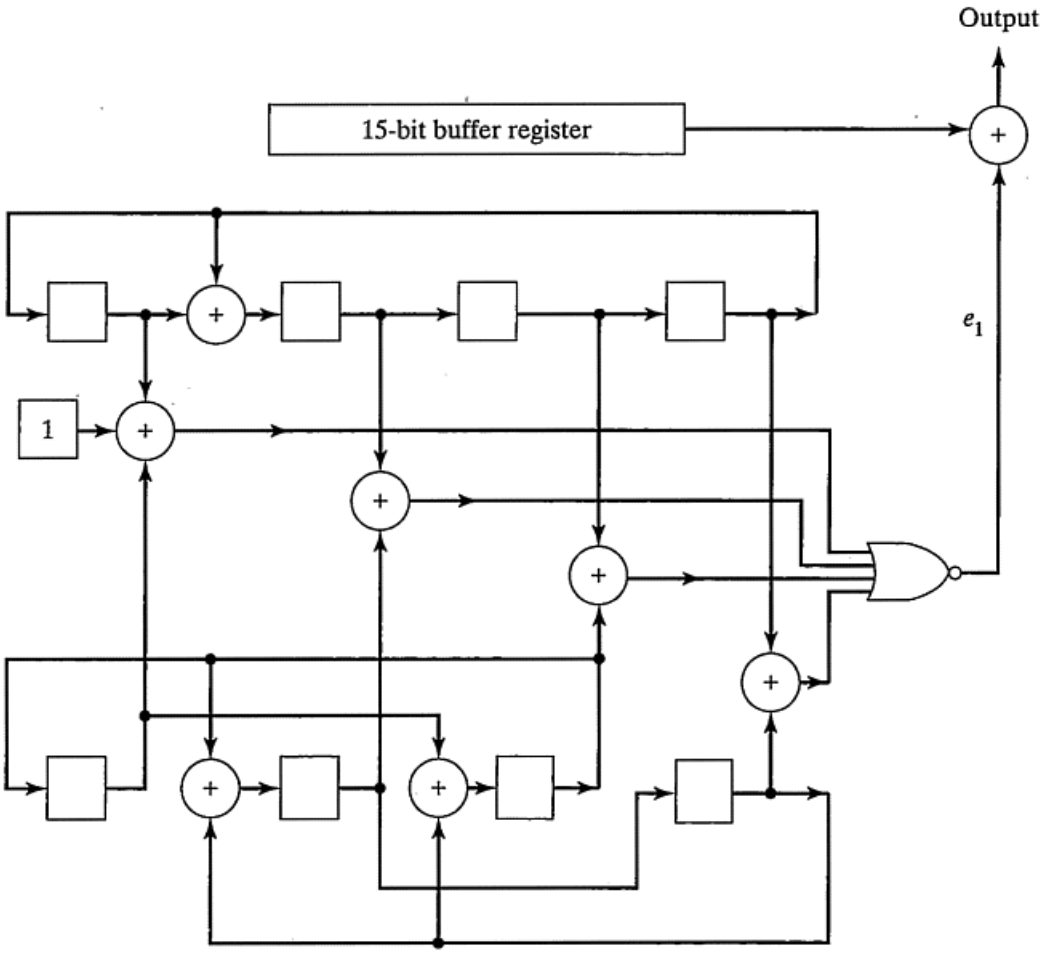


FIGURE 6–27 Chien search circuit for the double-error-correcting (15, 7) BCH code. (Reproduced from Reference [8] with permission.)

bounded distance decoders. Figures 6–28 through 6–30 illustrate bit error probability as a function of received E_b/N_0 for binary phase shift keying modulation. Figure 6–28, 6–29, and 6–30 illustrate P_b for BCH codes having code rates of approximately 1/4, 1/2, and 3/4, respectively. Observe that the best performance (lowest E_b/N_0 for a given P_b) is achieved for the rate 1/2 codes. It can be shown that very high rate ($R > 3/4$) and very low rate ($R < 1/3$) codes have less coding gain than those with rates between 1/3 and 3/4. For the rate 1/2 codes Figure 6–29 shows that coding gain is greater than 4 dB at $P_b = 10^{-5}$ for the $n = 1023$ BCH code. Observe also that in all cases the performance improves as the block length n increases. Coding gain is a function of P_b and the selected modulation scheme. Clark and Cain [9] plot P_b versus E_b/N_0 for a number of BCH codes with orthogonal signaling showing that coding gain is reduced relative to the gain for coherent BPSK illustrated here. Block error probability for BCH codes can be calculated using the bound of (6–68).

6.3.6 Reed–Solomon Codes

6.3.6.1 Definition of the Reed–Solomon Codes. The Reed–Solomon codes are the only nonbinary codes that will be considered in this chapter. They are discussed because of their importance for use in communication systems where errors appear in bursts (rather than independent random errors), because of their importance in concatenated coding systems which will be discussed in Section 7.5.3, and because of their use in compact disc audio technology. The Reed–Solomon (RS) codes were discovered in 1960 by Reed and Solomon [30]. They are nonbinary BCH block codes

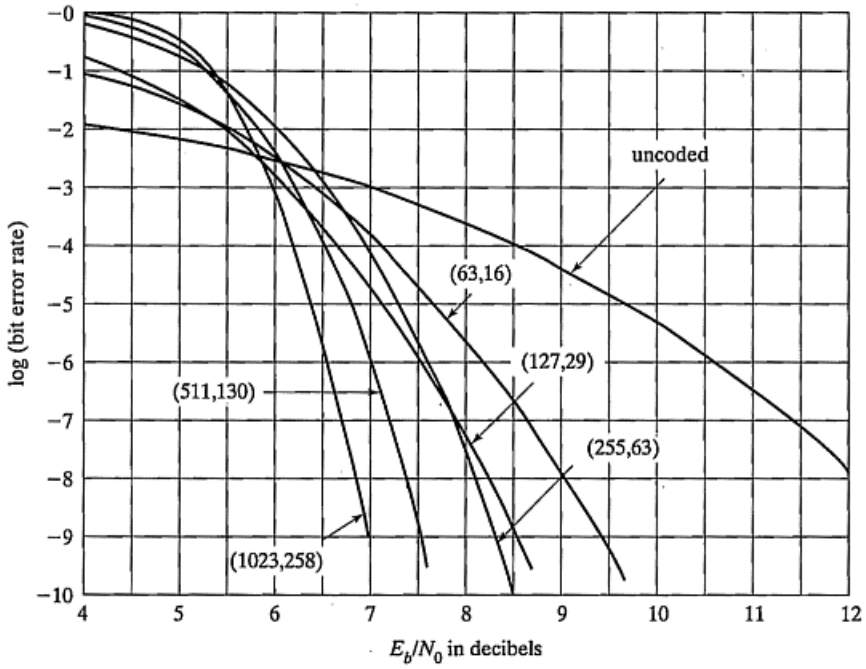


FIGURE 6–28 Bit error probability for BCH codes with $R \cong 1/4$.

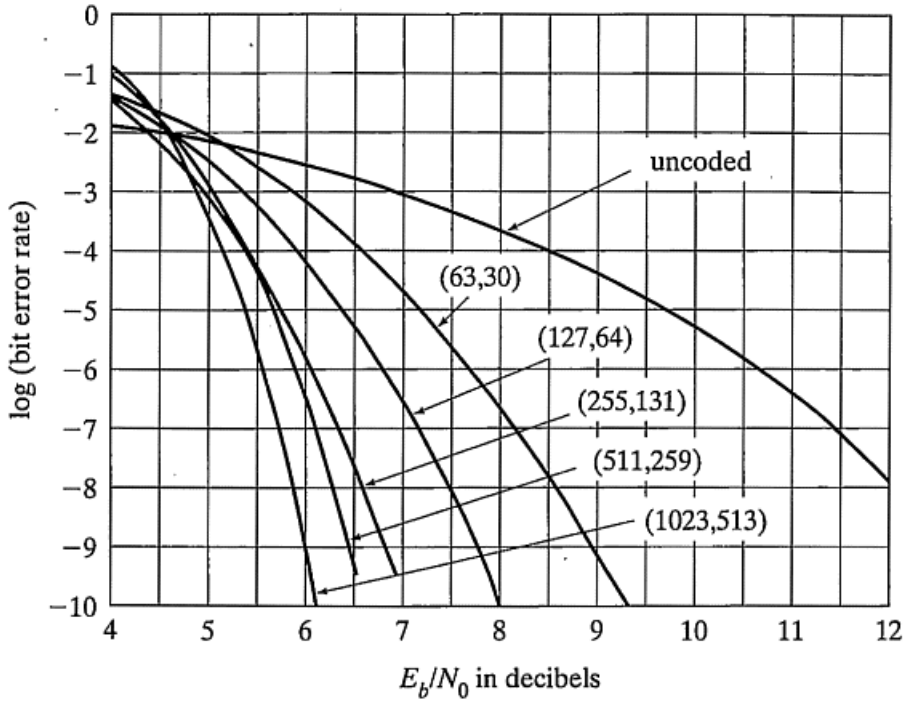


FIGURE 6–29 Bit error probability for BCH codes with $R \cong 1/2$.

that use input and output alphabets having 2^m symbols, $\{0, 1, 2, \ldots, 2^m - 1\}$. The block length of the RS codes is $n = 2^m - 1$; this block length can be extended to $n = 2^m$ or $n = 2^m + 1$ if desired. The codes are designed to correct e_0 errors within a block of n symbols. The number of parity symbols that must be used to correct e_0 errors is $n - k = n - 2e_0 = 2^m - 1 - 2e_0$. The maximum number of errors that can be corrected is a function of the minimum distance $d_{\min}$ of the code, specifically, $d_{\min} = 2e_0 + 1$. The RS codes achieve the largest possible $d_{\min}$ of any linear codes with the same encoder input and output lengths. The minimum distance, and thus the error correction capability of the code, increases with block length. The Hamming distance between nonbinary codewords is defined to be the number of positions in which the codewords differ.

Reed–Solomon codes are often used on channels that are naturally nonbinary.[7] For example, an 8-FSK transmission system can be used with an $m = 3$ RS coding system. If the source and channel are naturally binary, the nonbinary alphabet can be constructed by grouping binary symbols into m-bit blocks. In this case the RS codes can be thought of as accepting $k' = k \cdot m$ information bits and mapping these k' bits into codeword blocks having length $n' = n \cdot m$ binary channel symbols. Thus the rate of the RS codes is $R = k'/n' = k/n = (2^m - 1 - 2e_0)/(2^m - 1)$. RS codes with many different block lengths and rates are known.

The codes are defined by generator polynomials or generator matrices just as the binary BCH codes are defined. For RS codes, however, the code generator polynomial has coefficients from the nonbinary alphabet $\{0, 1, 2, \ldots, 2^m - 1\}$ rather than the binary al-

[7]See T. A. Gulliver, "Matching Q-ary Reed–Solomon Codes with M-ary Modulation,: *IEEE Trans. Commun.*, Vol. 45, pp. 1349–1353, Nov. 1997 for exact relationships between decoder output probability of bit error and modulation symbol probability of error for several choices of M and Q.

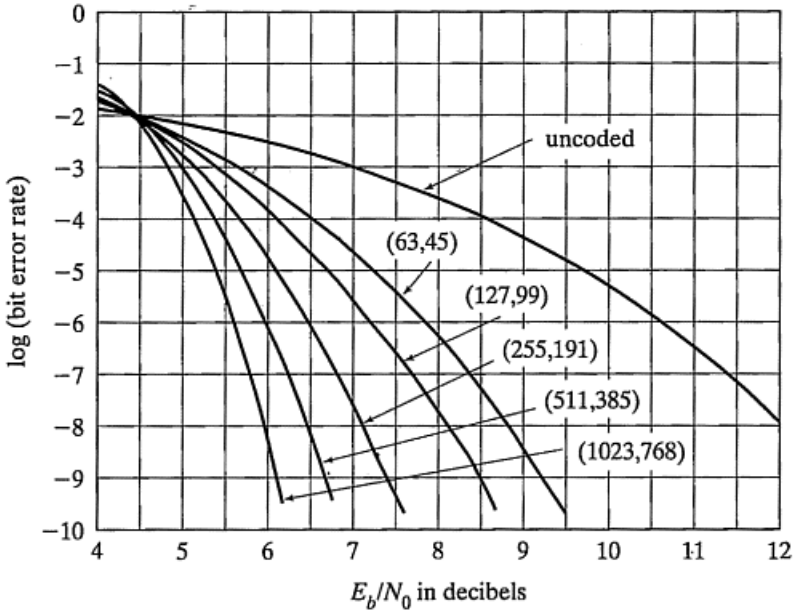


FIGURE 6–30 Bit error probability for BCH codes with $R \cong 3/4$.

phabet {0, 1}. When performing polynomial multiplication and addition, modulo-2^m arithmetic is used. A RS encoder can be implemented using a feedback shift register similar to that of Figure 6–19 but with all arithmetic performed modulo-2^m. The reader is referred to Lin and Costello [8] for the details of the modulo-2^m polynomial arithmetic necessary to generate the Reed–Solomon codes. Reference [8] also defines the method used to find the generator polynomials and provides additional references for the interested reader.

6.3.6.2 Decoding the Reed–Solomon Codes. Reed–Solomon codes are decoded using the same concepts used to decode binary BCH codes. The decoding begins by calculating a syndrome from the received block and the known structure of the code. Next, the syndromes are used to determine an error locator polynomial which is then solved to determine the specific error estimates. The errors are then corrected in the received sequence and the information block is output. The reader is referred to Lin and Costello [8], Blahut [11], or Clark and Cain [9] for advanced discussions of RS decoding.

6.3.6.3 Performance of the Reed–Solomon Codes. Consider an RS-coded communication system using a binary source and a naturally 2^m-ary physical link such as MFSK with $M = 2^m$. The RS code uses 2^m-ary symbols that are constructed from the binary source by grouping information bits into blocks of m. The encoder output is a sequence of 2^m-ary symbols that are transmitted to the receiver using MFSK. The performance measure of interest is the probability of decoded bit error P_b as a function of the MFSK symbol error probability p_s. It can be shown [23, 31, 32] that P_b is overbounded by

$$P_b \leq \sum_{i=e_0+1}^{2^m-1} \frac{i}{2(2^m-2)} \binom{2^m-1}{i} p_s^i (1-p_s)^{2^m-1-i} \tag{6–85}$$

For MFSK, the probability of symbol error is calculated using[8]

$$p_s = \frac{1}{M} \exp\left(-\frac{E_s}{N_0}\right) \sum_{q=2}^{M} \binom{M}{q} (-1)^q \exp\left(\frac{E_s}{qN_0}\right) \tag{6–86}$$

where $E_s/N_0 = (E_b/N_0)Rm$, since m binary bits are associated with one 2^m-ary symbol and the code rate is R.

Next, consider an RS-coded communication system using a binary source and a naturally binary physical link such as BPSK. The performance measure of interest is the probability of decoded bit error P_b as a function of the binary channel symbol error probability p. As before, (6–85) is used to calculate P_b. The probability p_s is calculated from the BPSK binary channel symbol error probability p. Specifically, p_s is equal to 1 minus the probability of transmitting an entire m-binary-symbol block without error. Thus

$$p_s = 1 - (1-p)^m \tag{6–87}$$

and

$$p = Q\left(\sqrt{\frac{2E_bR}{N_0}}\right) \tag{6–88}$$

Figure 6–31 illustrates the decoded bit error probability as a function of E_b/N_0 for RS codes with rate $\approx 1/2$ and with $m = 4$, 5, 6, 7, and 8. Observe that the coding gain for RS codes at $P_b = 10^{-5}$ varies from about 1.5 dB for $m = 4$ to 3.4 dB for $m = 8$. The "equivalent" binary block lengths of these codes is $(2^m - 1)m$.

The Reed–Solomon codes have been used in many modern communications systems. A number of manufacturers make large-scale integrated circuits for decoding RS codes. Probably the most common use of RS codes is found in the audio recording industry where they are used in compact discs. In this application a (32, 28) shortened RS code is concatenated with a (28, 24) code [33] resulting in a $R = 3/4$ code. In addition, Reed–Solomon coding is part of a recommendation by the Consultative Committee for Space Data Systems (CCSDS) [34], which is a standards organization having NASA and the European Space Agency (and others) as members. The CCSDS recommends that an RS code be concatenated with a convolutional code. The CCSDS recommendation will be followed in the NASA space station.

6.3.7 The Golay Code

6.3.7.1 Definition of the Golay Code. The Golay code was discovered in 1949 by M. J. E. Golay [5]. The code has $n = 23$, $k = 12$, $d_{\min} = 7$. It is capable of correcting three errors in a block of 23 symbols. The code's importance stems from its significant error correcting capability as well as the fact that it is one of the few known "perfect"

[8] Equation (6–86) can be reduced to (4–124).

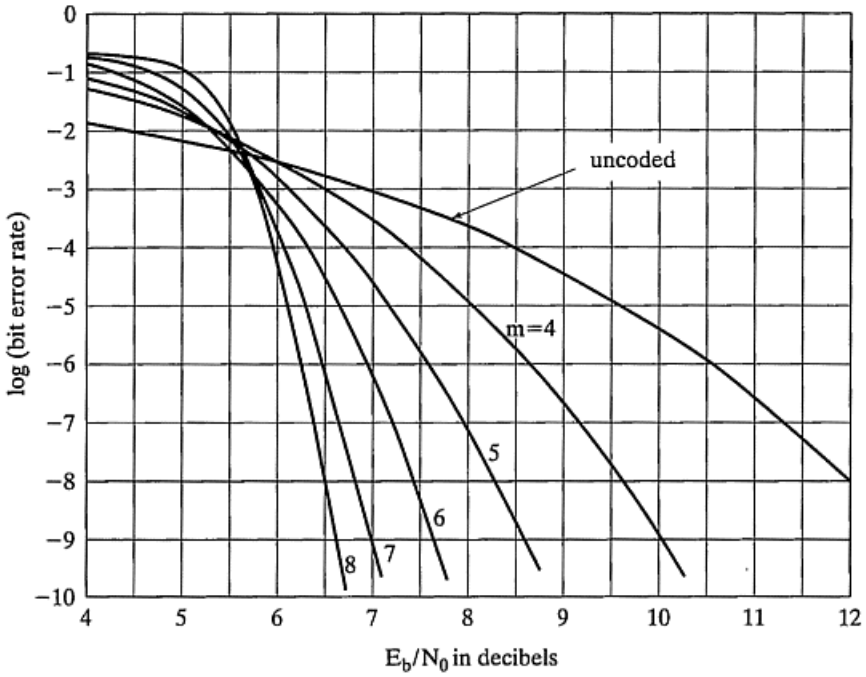


FIGURE 6–31 Bit error probability for Reed–Solomon codes with $R \cong 1/2$.

codes. A perfect code has the interesting property that all error patterns with Hamming weight t or less and no error patterns with weight greater than t are correctable using a minimum-distance maximum-likelihood decoder. The Hamming codes are also perfect codes. The Golay code is decodable using a variety of techniques. The code is a binary cyclic code whose generator polynomial is given by either

$$\mathbf{g}_1(D) = 1 + D^2 + D^4 + D^5 + D^6 + D^{10} + D^{11} \tag{6–89a}$$

or the generator polynomial

$$\mathbf{g}_2(D) = 1 + D + D^5 + D^6 + D^7 + D^9 + D^{11} \tag{6–89b}$$

The same codewords, although with different message associations, are generated by either generator. The encoding circuit for the Golay code is identical in form to the circuit for cyclic Hamming codes of Figure 6–19.

The (23, 12) Golay code is often modified by adding a parity bit making the extended (24, 12) Golay code. The extended code has $d_{\min} = 8$; however, it is not a perfect code. The performance of the extended code is slightly better than the nonextended code and is easier to use since its rate is exactly 1/2. For these reasons, the extended code is often used in place of the perfect (23, 12) code.

6.3.7.2 Decoding the Golay Code. Decoding of the Golay code can be accomplished using several strategies. Of course, table lookup and standard array decoding are always conceptually possible. Since the code is cyclic, the generalized

decoder of Figure 6–17 can be used. A specific implementation of the decoder which is able to correct three errors in a block of 23 bits has been found by Kasami [35] and is described in some detail in [8]. The Kasami decoder implements the error pattern detection circuit of Figure 6–17 by comparing certain sums of syndromes to fixed thresholds. Three syndrome sums are calculated to estimate error patterns.

6.3.7.3 Performance of the Golay Code. The bit error probability for the (23, 12) Golay code is overbounded using (6–74) with $M = 3$, $n = 23$, and $k = 12$

$$P_b \leq \frac{1}{12} \sum_{i=4}^{23} \min[12, i + 3] \binom{23}{i} p^i (1 - p)^{23-i} \tag{6–90}$$

Figure 6–32 illustrates P_b as a function of E_b/N_0 assuming binary phase-shift keying modulation. Observe that the coding gain at $P_b = 10^{-5}$ is approximately 1.9 dB.

The weight structure for the Golay code is known and is given in [8].

$$A(z) = 1 + 253z^7 + 506z^8 + 1288z^{11} + 1288z^{12} + 506z^{15} + 253z^{16} + z^{23} \tag{6–91}$$

Thus, the code has a single Hamming weight 0 codeword, 253 codewords with Hamming weight 7, 506 codewords with Hamming weight 8, and so on. Since the weight structure is known, (6–68) can be used to calculate an upper bound on block decoding error probability. The weight structure of the extended (24, 12) Golay code is also known and is [9]

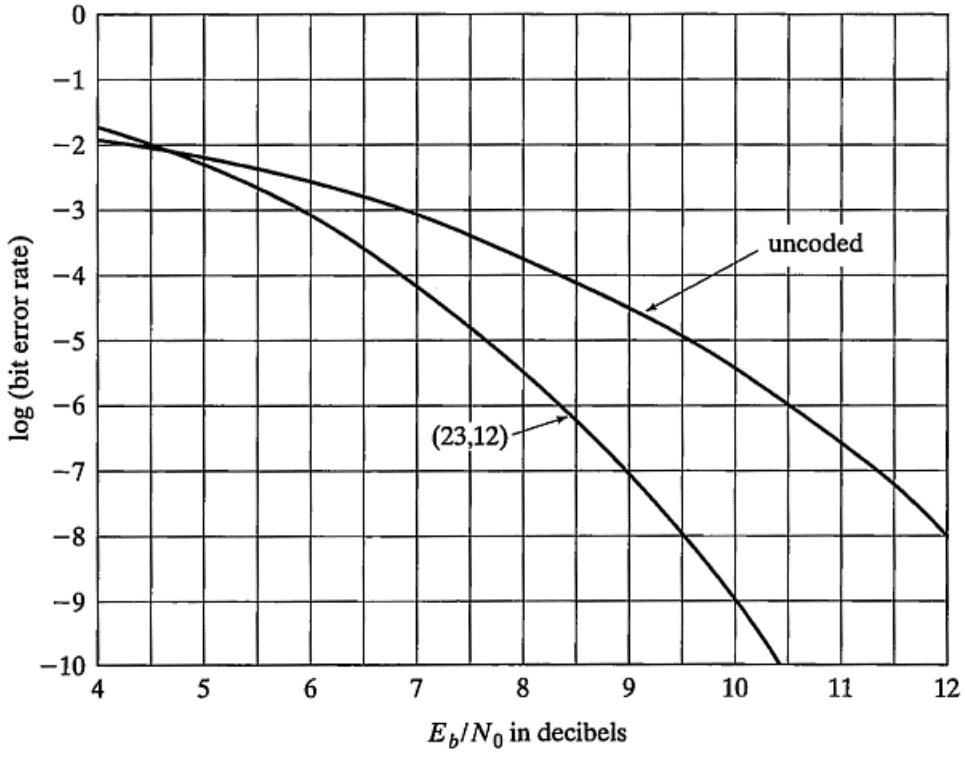


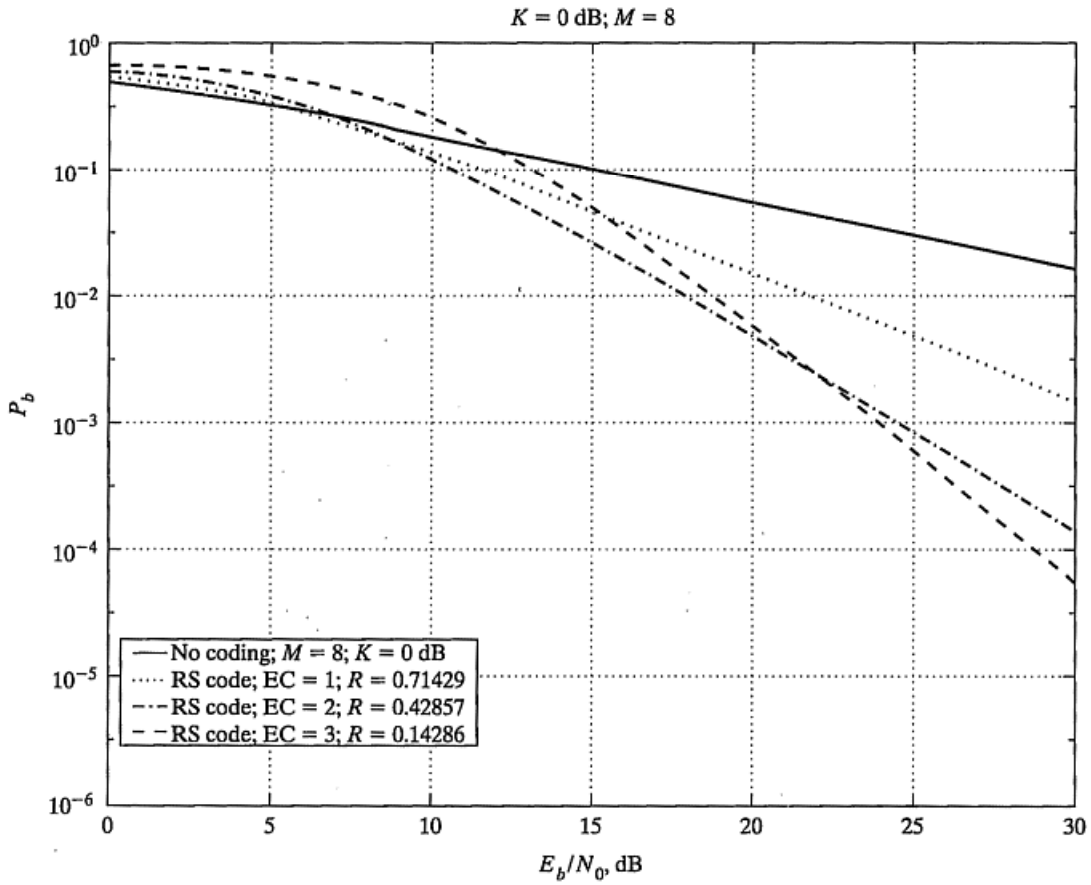
FIGURE 6–32 Bit error probability for Golay (23, 12) code.

$$A(z) = 1 + 759z^8 + 2576z^{12} + 759z^{16} + z^{24} \tag{6–92}$$

The Golay codes have been studied extensively and have been used in modern communication systems, including recent deep space missions [36]. The formula for bit error probability can be improved upon slightly for the extended Golay code. This improvement is made possible by exact calculation of the average number of bit errors caused by a particular number of channel errors. This exact number replaces the min[12, $i + 3$] function in the formula for P_b. The reader is referred to Odenwalder [23] for the more precise expression. Results using improved formula differ from the formula used to generate Figure 6–32 only at low E_b/N_0.

6.4 CODING PERFORMANCE IN SLOW FADING CHANNELS

The purpose of error correction coding is to improve performance in channels where, for one reason or other, uncoded performance is not acceptable. As discussed in previous chapters, slow fading channels are one such example situation. To show the potential of



(a)

FIGURE 6–33 P_b versus E_b/N_0 of Reed–Solomon-coded, noncoherent FSK in slow Ricean fading with $K = 0$ dB. (EC = Errors corrected).

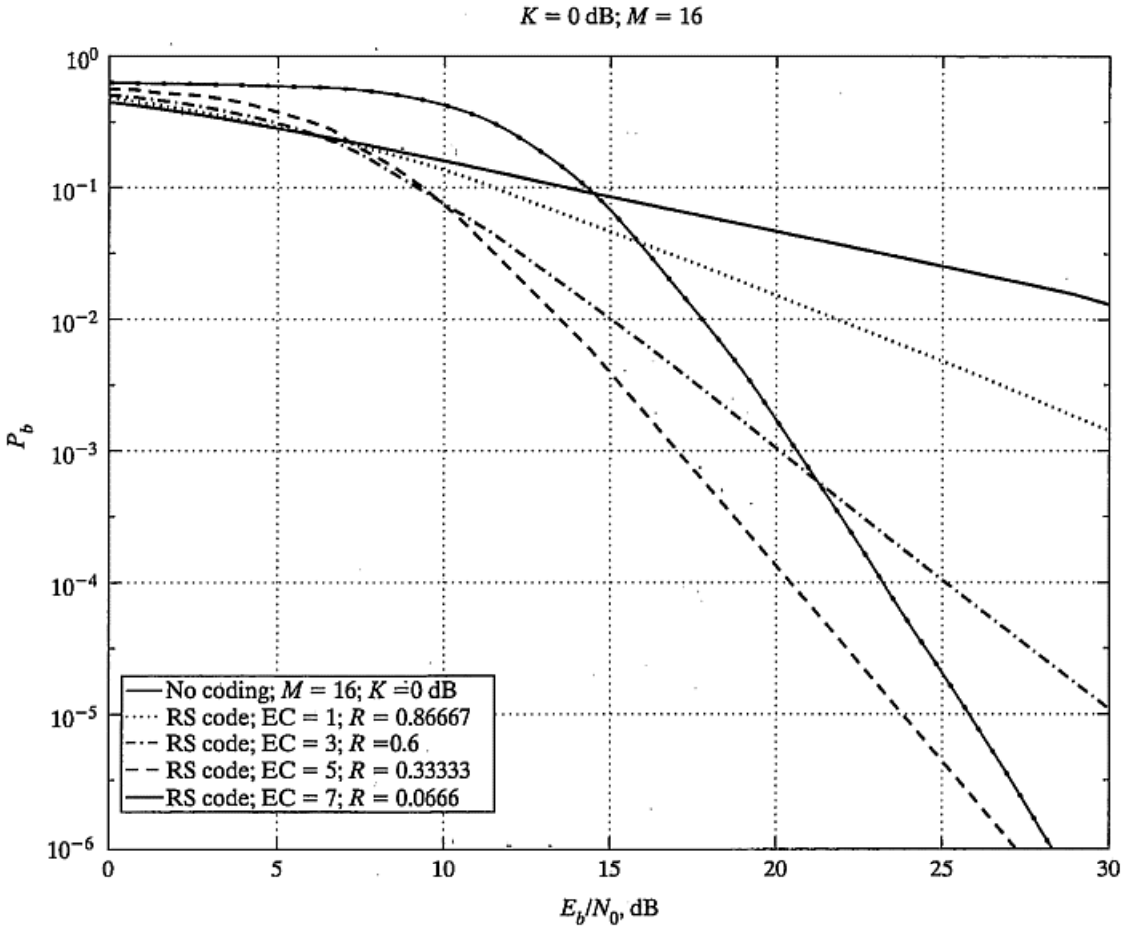


FIGURE 6–33 *Continued*

coding in such situations for improving performance, we consider Reed–Solomon code performance used in conjunction with M-ary noncoherent FSK in Ricean slow fading channels. This is a fairly easy task by using the exact expression for the symbol error probability derived in Chapter 4 for noncoherent FSK along with the bound for bit error probability performance for Reed–Solomon coding given in Section 6.3.6 as (6–85). The results to be presented in this section assume that interleaving at the transmitter and deinterleaving at the receiver are used to randomize the errors introduced in the channel. A further discussion of this technique is given in Chapter 7 (see Figure 7–25 and the accompanying discussion).

We consider two different channel conditions—a specular-to-diffuse power ratio of 0 dB, which is very nearly a Rayleigh fading channel, and a specular-to-diffuse power ratio of 10 dB, where the diffuse component is a factor of 10 below the direct received power. Figure 6–33 illustrates the former case for $M = 8$, 16, and 32, and Figure 6–34 shows the latter case for the same values of M. The code is matched to the modulation in each case in terms of code symbols and modulation symbols being equal. The parameter EC shown in the legends is the number of errors corrected, and the parameter R is the code rate. Thus, more errors corrected result in lower code rates and larger bandwidth expansion factors. Even for high code rates and few errors corrected, it is seen that

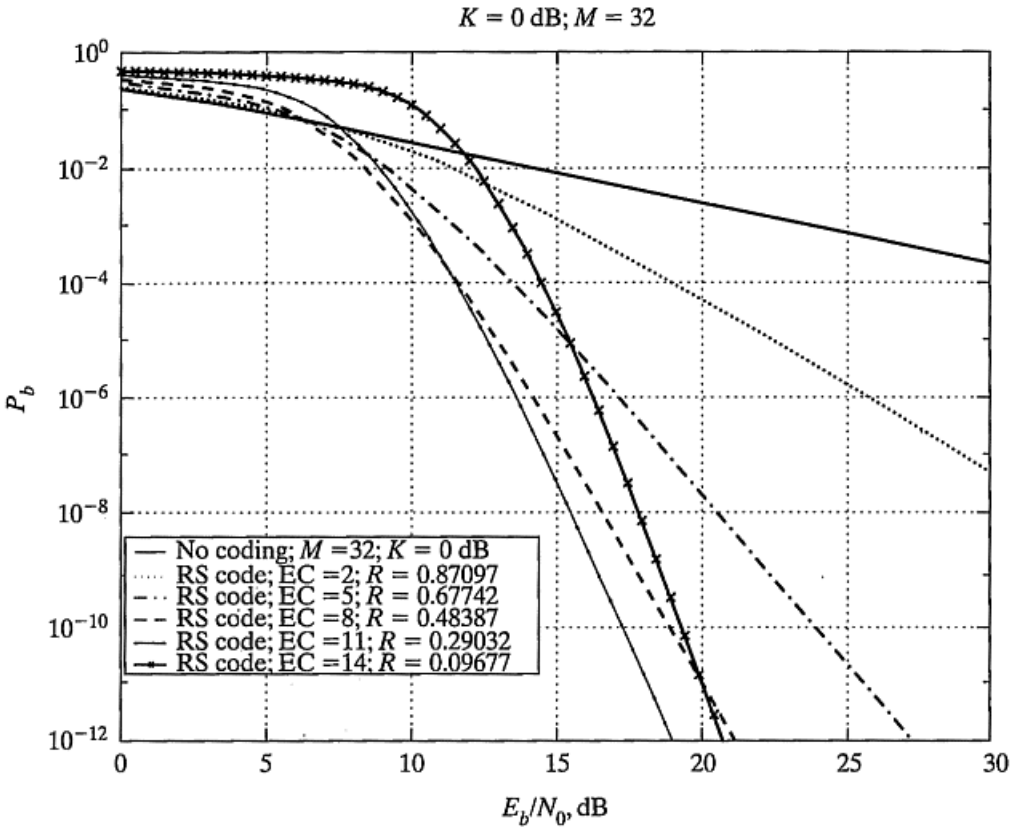


FIGURE 6–33 *Continued*

Reed–Solomon codes are very powerful vehicles for improving performance in slow fading channels.

6.5 SUMMARY

1. The function of the communications systems engineer is to design systems making efficient use of the resources of power, bandwidth, and complexity to reliably communicate information from a source to a destination. An accepted measure of reliability is end-to-end bit error probability P_b.
2. Source coding removes signal redundancy to minimize the total number of bits per second that must be communicated.
3. Channel coding is used to correct as many channel errors as possible thereby improving reliability.
4. A discrete memoryless source represents information using a sequence of independent symbols $w_0, w_1, w_2, \ldots$ from the discrete alphabet $W = \{0, 1, 2, \ldots, q_w - 1\}$. The probability that the DMS outputs symbol j during interval i is $Q_w(j)$.
5. The information content of a single DMS output symbol j is $I(j) = -\log_2 Q_w(j)$. Highly likely DMS output symbols have less information content than do less likely symbols. The entropy of the source is

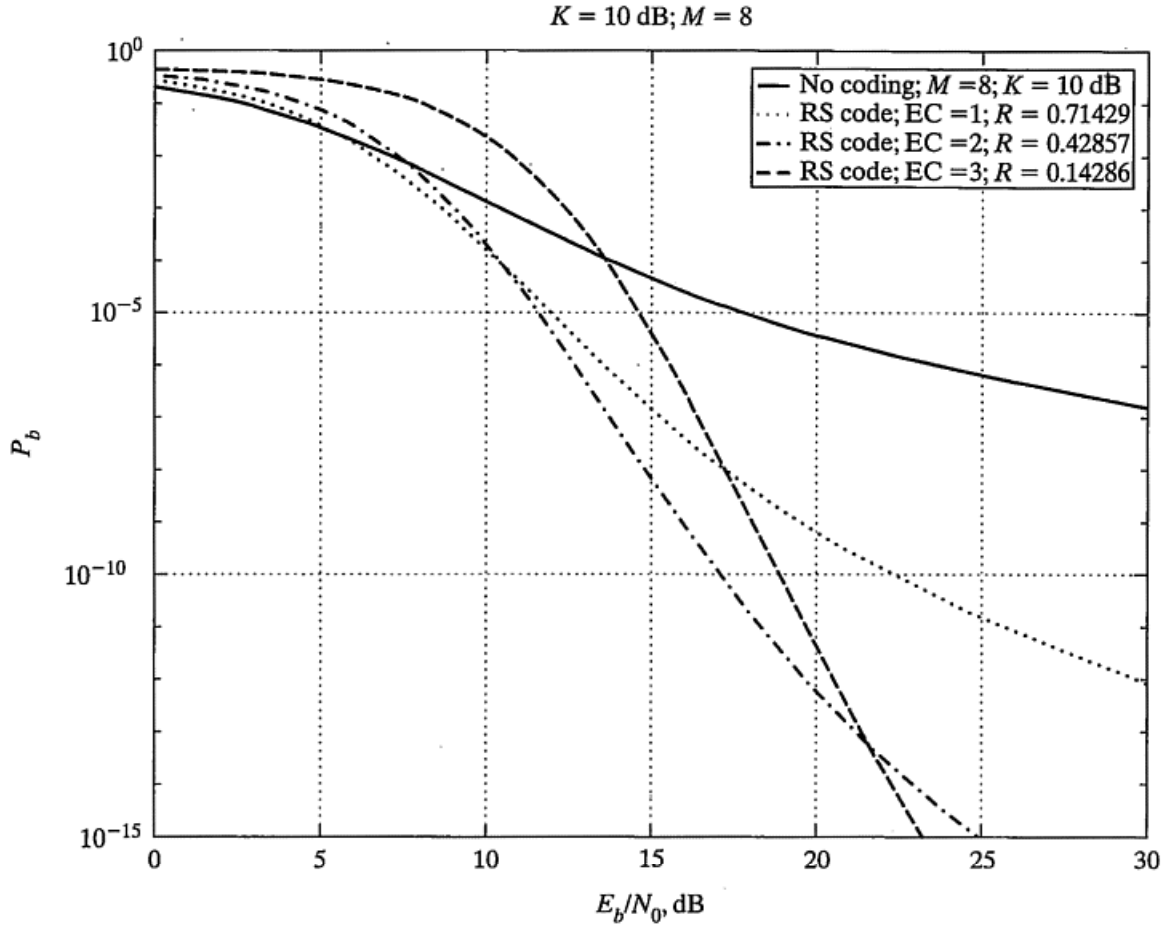


FIGURE 6–34 BEP versus E_b/N_0 of Reed–Solomon-coded, noncoherent FSK in slow Ricean fading with $K = 10$ dB.

$$H(W) = -\sum_{j=0}^{q_w-1} Q_w(j) \log_2 Q_w(j) \quad \text{bits/symbol} \tag{6–3}$$

which is the average information content of the source. The source entropy is the minimum average number of binary bits per DMS output symbol that must be transmitted over the channel. Thus, the source information cannot be communicated using fewer average bits per DMS symbol than the entropy.

6. The Huffman procedure is one intuitively satisfying method of source coding. The procedure assigns longer binary sequences to low probability DMS symbols than to higher probability symbols. DMS symbols can be Huffman coded as single symbols or in groups. Huffman coding of groups of DMS symbols produces the best results.
7. The capacity C of a channel is the number of bits per second that can theoretically be transmitted without error. Capacity is a function of the channel bandwidth and received signal-to-noise ratio. The Shannon capacity is

$$C = B \cdot \log_2\left(1 + \frac{P}{N_0 B}\right) \tag{6–6}$$

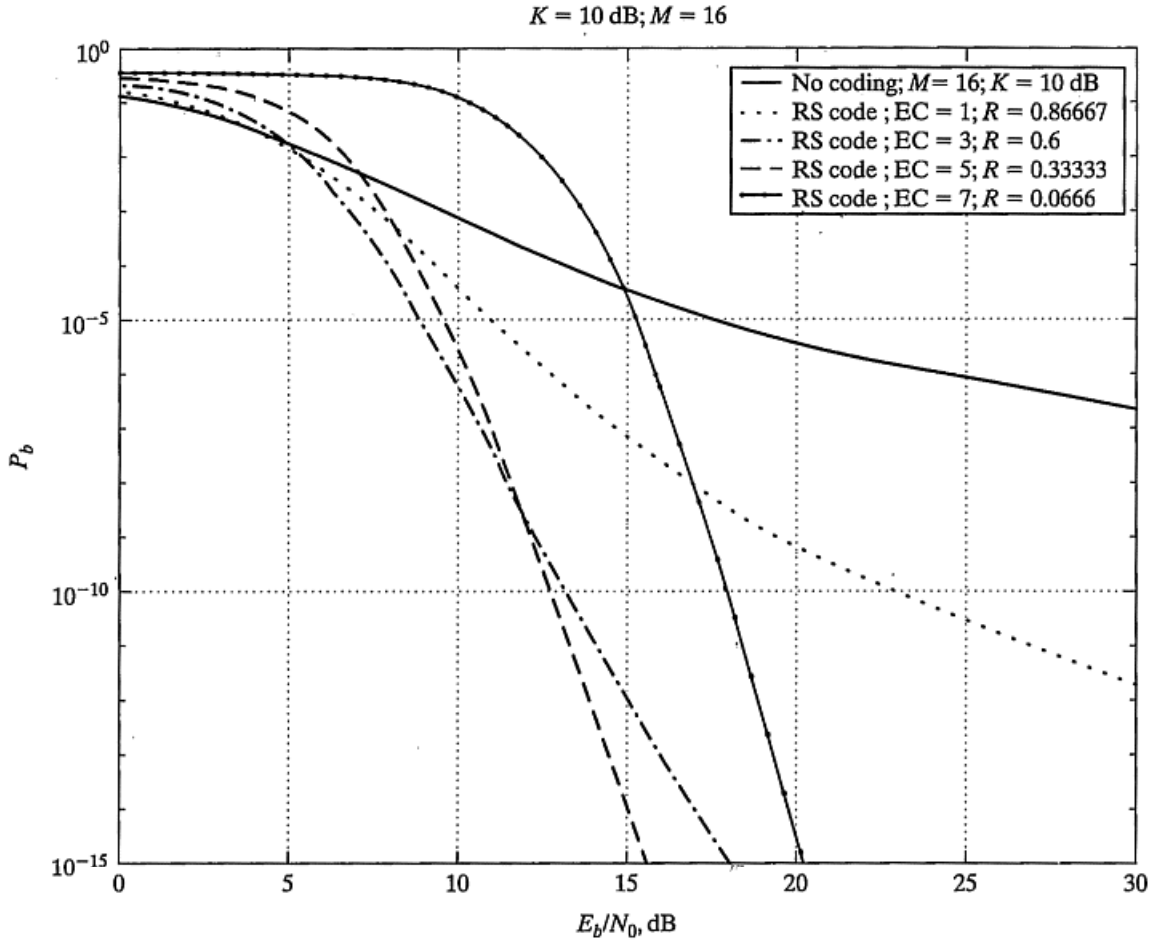


FIGURE 6–34 *Continued*

If the desired transmission rate R_m is less than C, arbitrarily low error rate can be achieved using forward error correction without increasing transmitter power. For R_m very near C, forward error correction complexity can be considerable.

8. Received E_b/N_0 must be higher than − 1.6 dB for arbitrarily reliable communications to be possible. At lower E_b/N_0 no amount of coding complexity can help the communicator.
9. The capacity of a discrete memoryless channel is denoted C_N and represents the maximum number of bits of information communicated for each DMC symbol transmitted.
10. The capacity of a binary symmetric channel with error probability p is

$$C_N = 1 + p \log_2 p + (1 - p) \log_2(1 - p) \tag{6–20}$$

11. The computational cutoff rate R_0 for a channel is another measure of capacity. It is, in fact, a practical limit on capacity that can be achieved with reasonable complexity. For the binary symmetric channel

$$R_0 = -\log_2[\tfrac{1}{2} + \sqrt{p(1 - p)}] \tag{6–21}$$

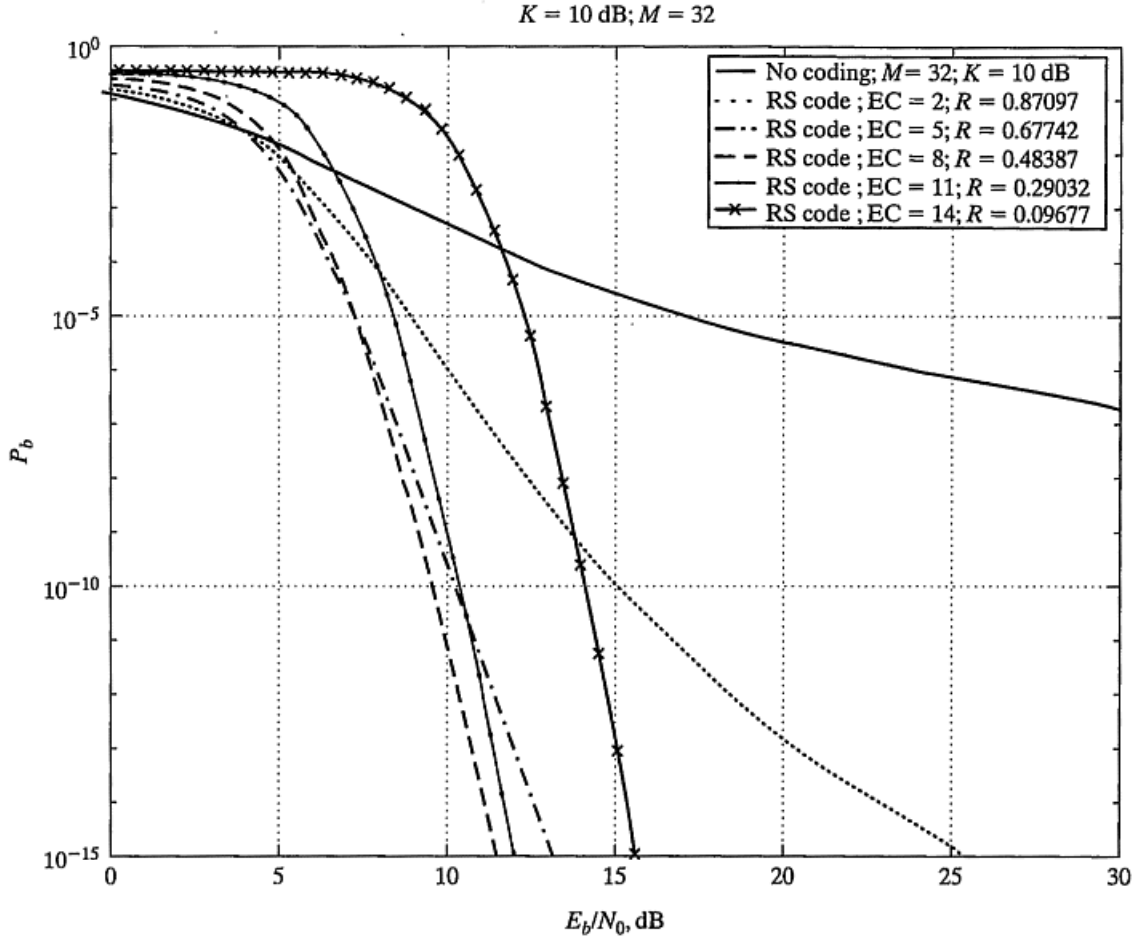


FIGURE 6–34 *Continued*

The units of R_0 are bits of information per channel use.

12. The coding gain of a forward error correction technique is the difference between E_b/N_0 required to achieve a specified bit error probability without coding and the E_b/N_0 required with coding. Coding gain is a function of the modulation technique, for example, BPSK, the coding technique, and the specified error probability.
13. A binary block code is a smart mapping of a block of k information bits called a message into a block of n binary channel symbols called a codeword. The code rate $R = k/n$ is always less than 1.0 for binary codes so that there are many more possible codewords (2^n) than possible messages (2^k).
14. The number of 1s in a binary n-vector $\mathbf{x}$ is called the Hamming weight of the vector denoted $w_H(\mathbf{x})$. The number of positions in which two binary vectors $\mathbf{x}_m$ and $\mathbf{x}_{m'}$ differ is called the Hamming distance between the vectors denoted $d_H(\mathbf{x}_m, \mathbf{x}_{m'})$. It was shown that $d_H(\mathbf{x}_m, \mathbf{x}_{m'}) = w_H(\mathbf{x}_m + \mathbf{x}_{m'})$.
15. The minimum Hamming distance between any two codewords in a code is the minimum distance of the code denoted $d_{\min}$. The Hamming distance between codewords

makes error correction possible since a number of error must occur before codewords will be confused with one another.

16. The probability of a particular sequence of e' errors occurring in n transmissions on a BSC is

$$\Pr\left[\begin{array}{c} e' \text{ errors in} \\ \text{specific locations} \\ \text{in codeword} \end{array}\right] = p^{e'}(1-p)^{n-e'} \tag{6–28}$$

The probability of some sequence (without regard to order) of e' errors occurring in n transmissions on a BSC is

$$\Pr\left[\begin{array}{c} e' \text{ errors in} \\ \text{block of } n \end{array}\right] = \binom{n}{e'} p^{e'}(1-p)^{n-e'} \tag{6–31}$$

The probability of receiving n-vector $\mathbf{y}$ having transmitted codeword $\mathbf{x}$ is

$$\Pr(\mathbf{y} \mid \mathbf{x}) = p^{d_H}(1-p)^{n-d_H} \tag{6–29}$$

17. Given prior knowledge of the code and channel, the optimum decoding rule is simply to estimate the transmitted codeword to be that codeword that was most likely to have been transmitted. For the BSC this decoding rule is equivalent to estimating the transmitted codeword to be the codeword closest to the received vector in terms of Hamming distance.
18. With the decoding rule known, decoding can be visualized as partitioning the space of all possible received n-vectors into nonoverlapping decoding regions Λ_m. Each region is associated with a unique codeword $\mathbf{x}_m$ and includes all n-vectors closer in Hamming distance to $\mathbf{x}_m$ than to any other message.
19. Given the optimum decoding regions Λ_m, a block decoding error results if $\mathbf{x}_m$ is transmitted and $\mathbf{y}$ is within $\Lambda_{m'}$ for some $m' \neq m$. The average block decoding error probability is

$$P_B = \sum_{m=0}^{2^k-1} Q_M(m) P_{B_m} \tag{6–39}$$

where

$$P_{B_m} = \sum_{y \in \bar{\Lambda}_m} \Pr(\mathbf{y} \mid \mathbf{x_m}) \tag{6–38}$$

20. Exact calculation of bit error probability requires knowledge of the number of bit errors which occur with each block decoding error. Exact bit error probability is

$$P_b = \frac{1}{k} \sum_{m=0}^{2^k-1} Q_M(m) \sum_{\substack{m'=0 \\ m' \neq m}}^{2^k-1} B(m, m')\, P_B(m, m') \tag{6–41}$$

where

$$P_B(m, m') = \sum_{\mathbf{y} \in \Lambda_{m'}} \Pr(\mathbf{y} \mid \mathbf{x}_m) \tag{6–40}$$

and $B(m, m')$ is the number of bit errors that occur when transmitted message m is decoded as message m'.

21. The modulo-2 sum of two binary n-vectors is the term-by-term modulo-2 sum of the vectors. The scalar product of a binary vector and binary scalar is the term-by-term modulo-2 product of the scalar with the vector components. The dot product of two binary vectors is

$$\mathbf{a} \cdot \mathbf{b} = (a_0 \cdot b_0) + (a_1 \cdot b_1) + (a_2 \cdot b_2) + \cdots + (a_{n-1} \cdot b_{n-1}) \tag{6–42}$$

Two vectors are orthogonal if their dot product is zero.

22. A linear combination of vectors $\mathbf{g}_i$ is

$$(w_0 \cdot \mathbf{g}_0) + (w_1 \cdot \mathbf{g}_1) + (w_2 \cdot \mathbf{g}_2) + \cdots + (w_{k-1} \cdot \mathbf{g}_{k-1}) \tag{6–43}$$

The set of vectors $\mathbf{g}_i$ is linearly independent if no set of nonzero scalars w_i exist that causes this result to be zero. Otherwise, the set is linearly dependent.

23. A binary vector space is a set of vectors satisfying (a) modulo-2 sum of any two vectors is another vector in the set, (b) scalar product of 0 or 1 and any vector is in the set, (c) there is a distributive law, and (d) there is an associative law. The set of all binary n-vectors is a vector space.

24. A subspace of a vector space is a subset of the elements of the space satisfying (a) modulo-2 sum of any two vectors in the subset is another vector in the subset, and (b) the scalar product of 0 or 1 and any vector in the set is in the set.

25. The vector set formed by calculating all possible linear combinations of any subset of a linear vector space is a subspace of the vector space. The smallest set of vectors whose linear combinations form all vectors of a space or subspace are called a basis of the space or subspace. The number of vectors in the basis is called the dimension of the space or subspace.

26. The codewords of an (n, k) binary linear block code are a k-dimensional subspace of the space of all binary n-vectors.

27. The modulo-2 sum of any two codewords of an (n, k) linear block code is another codeword in the code.

28. The all-zero n-vector is a codeword in every (n, k) linear binary block code.

29. The set of n-vectors that form the basis of an (n, k) binary linear block code are called the generator vectors of the code. The generator matrix $\mathbf{G}$ of the code is an n column by k row binary matrix whose rows are $\mathbf{g}$. Given the generator vectors $\mathbf{g}$ the codewords are calculated from

$$\begin{aligned} \mathbf{x}_m &= \mathbf{w}_m \times \mathbf{G} \\ &= [w_{m0}\, w_{m1} \cdots w_{m,k-1}] \times \begin{bmatrix} g_{00} & g_{01} & \cdots & g_{0,n-1} \\ g_{10} & g_{11} & \cdots & g_{1,n-1} \\ \vdots & \vdots & & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{bmatrix} \end{aligned} \tag{6–48}$$

30. The null space of an (n, k) linear block code is an $(n, n-k)$ block code whose generator matrix is denoted **H.** The matrix **H** is called the parity check matrix for the original (n, k) block code. An n-vector **y** is a codeword of the (n, k) linear block code if and only if

$$\mathbf{0} = \mathbf{y} \times \mathbf{H}^T \tag{6–52}$$

31. Systematic linear block codes are codes for which the message vector appears directly in each codeword. For any linear block code there is a systematic linear block code with equivalent performance. Thus, the system designer may limit consideration to systematic codes without fear of degrading system performance.

32. The generator and parity check matrices for systematic codes have the form

$$\mathbf{G} = \begin{bmatrix} g_{0,0} & g_{0,1} & \cdots & g_{0,n-k-1} & 1 & 0 & 0 & \cdots & 0 \\ g_{1,0} & g_{1,1} & \cdots & g_{1,n-k-1} & 0 & 1 & 0 & \cdots & 0 \\ g_{2,0} & g_{2,1} & \cdots & g_{2,n-k-1} & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & & & & & & & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-k-1} & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

and

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & g_{0,0} & g_{1,0} & \cdots & g_{k-1,0} \\ 0 & 1 & 0 & \cdots & 0 & g_{0,1} & g_{1,1} & \cdots & g_{k-1,1} \\ 0 & 0 & 1 & \cdots & 0 & g_{0,2} & g_{1,2} & \cdots & g_{k-1,2} \\ \vdots & \vdots & \vdots & & & & & & \\ 0 & 0 & 0 & \cdots & 1 & g_{0,n-k-1} & g_{1,n-k-1} & \cdots & g_{k-1,n-k-1} \end{bmatrix} \tag{6–54}$$

33. The Hamming distance between any two codewords of a linear block code is equal to the Hamming distance between the all-zero codeword and some nonzero codeword in the code.

34. A minimum distance decoder for a block code with a minimum distance $d_{\min}$ is able to correctly decode any transmission where $\lfloor (d_{\min} - 1)/2 \rfloor \doteq t$ or fewer errors occur. In some specific cases a minimum distance decoder is able to correct more than t errors.

35. The standard array for a linear code is

$$\begin{array}{cccc} \mathbf{x}_0 & \mathbf{x}_1 & \mathbf{x}_2 & \cdots & \mathbf{x}_{2^k-1} \\ \mathbf{e}_1 + \mathbf{x}_0 & \mathbf{e}_1 + \mathbf{x}_1 & \mathbf{e}_1 + \mathbf{x}_2 & \cdots & \mathbf{e}_1 + \mathbf{x}_{2^k-1} \\ \mathbf{e}_2 + \mathbf{x}_0 & \mathbf{e}_2 + \mathbf{x}_1 & \mathbf{e}_2 + \mathbf{x}_2 & \cdots & \mathbf{e}_2 + \mathbf{x}_{2^k-1} \\ \vdots & \vdots & \vdots & & \vdots \\ \mathbf{e}_{2^{n-k}-1} + \mathbf{x}_0 & \mathbf{e}_{2^{n-k}-1} + \mathbf{x}_1 & \mathbf{e}_{2^{n-k}-1} + \mathbf{x}_2 & \cdots & \mathbf{e}_{2^{n-k}-1} + \mathbf{x}_{2^k-1} \end{array} \tag{6–55}$$

Codewords appear in the first row. Correctable error patterns appear in the first column. This array is used for decoding in two different manners. (a) The decoder output codeword is the codeword at the head of the column in which the received vector

y is found. (b) The syndrome vector is precalculated for each correctable error pattern. The syndrome of the received vector **y** is calculated. The decoder estimated error pattern is that error pattern having the same syndrome as the received vector.

36. The block decoding error probability for standard array decoding is

$$P_B = 1 - \sum_{j=0}^{2^{n-k}-1} p^{w_H(\mathbf{ej})}(1-p)^{n-w_H(\mathbf{ej})} \tag{6–57}$$

where the sum is over all correctable error patterns.

37. The block and bit error probabilities for linear systematic codes are independent of which codeword was transmitted. Block error probability is union bounded by

$$P_B \leq \sum_{d=d_{\min}}^{n} a_d P'_B(d) \tag{6–68}$$

where a_d is the number of codewords with Hamming weight d and

$$P'_B(d) = \sum_{e=(d/2)+1}^{d} \binom{d}{e} p^e(1-p)^{d-e} + \frac{1}{2}\binom{d}{d/2} p^{d/2}(1-p)^{d/2} \tag{6–63}$$

for d even and

$$P'_B(d) = \sum_{e=(d+1)/2}^{d} \binom{d}{e} p^e(1-p)^{d-e} \tag{6–64}$$

for d odd. The bit error probability is overbounded by

$$P_b < \frac{1}{k} \sum_{d=d_{\min}}^{n} B_d P'_B(d) \tag{6–73}$$

where B_d is the total number of bit errors associated with all block decoding error events in which a Hamming weight d codeword is decoded rather than the correct **0** codeword.

38. When detailed code parameters a_d or B_d are not available, a bounded distance decoder may be assumed and the block error probability bound is

$$P_B = \sum_{i=M+1}^{n} \binom{n}{i} p^i(1-p)^{n-i} \tag{6–69}$$

where M is the maximum number of error corrected by the bounded distance decoder. The bit error probability bound for this same decoder is

$$P_b \leq \frac{1}{k} \sum_{i=M+1}^{n} \min[k, i+M] \binom{n}{i} p^i(1-p)^{n-i} \tag{6–74}$$

39. If the block error probability P_B for a code is known, the bit error probability P_b is bounded by

$$\frac{P_B}{k} \leq P_b \leq P_B$$

40. Modulo-2 polynomial arithmetic is the same as conventional polynomial arithmetic except that coefficients are added and multiplied using the modulo-2 rules for arithmetic. Feedback and feedforward shift registers can be used to perform both polynomial multiplication and division.
41. Cyclic codes are a subset of all linear codes. The end-around cyclic shift of any codeword of a cyclic code is another codeword in the code. Cyclic codes can be encoded and decoded using the general techniques for all linear codes; however, their structure permits simplified encoding and decoding.
42. An (n, k) cyclic code is defined by its generator polynomial $\mathbf{g}(D) = g_0 + g_1 D + g_2 D^2 + \cdots + g_{n-k} D^{n-k}$, which has degree $n - k$. The $k \times n$ generator matrix $\mathbf{G}$ is

$$\mathbf{G} = \begin{bmatrix} g_0 & g_1 & \cdots & g_{n-k-1} & g_{n-k} & 0 & 0 & & \cdots & 0 \\ 0 & g_0 & g_1 & \cdots & g_{n-k-1} & g_{n-k} & 0 & 0 & \cdots & 0 \\ 0 & 0 & g_0 & g_1 & \cdots & g_{n-k-1} & g_{n-k} & 0\,0 & \cdots & 0 \\ \vdots & \vdots & & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & g_0 & g_1 & \cdots & g_{n-k-1} & g_{n-k} \end{bmatrix}$$

The $(n - k) \times n$ parity check matrix for a cyclic code is

$$\mathbf{H} = \begin{bmatrix} h_k & h_{k-1} & \cdots & \cdots & h_1 & h_0 & 0 & 0 & . & . & . & 0 \\ 0 & h_k & h_{k-1} & \cdots & \cdots & h_1 & h_0 & 0 & 0 & \cdots & \cdots & 0 \\ 0 & 0 & h_k & h_{k-1} & \cdots & \cdots & h_1 & h_0 & 0 & 0 & \cdots & 0 \\ \vdots & \vdots & & & & & & & & & \vdots & \\ 0 & 0 & 0 & 0 & 0 & 0 & h_k & h_{k-1} & . & . & . & h_0 \end{bmatrix} \quad (6\text{–}78)$$

where $\mathbf{h}(D) = h_0 + h_1 D + h_2 D^2 + \cdots + h_k D^k$ is the result of the division of $D^n + 1$ by $\mathbf{g}(D)$.

43. Calculation of the code polynomials in systematic form for cyclic codes is accomplished by premultiplying the message $\mathbf{w}(D)$ by D^{n-k} and obtaining the remainder $\rho(D)$ of the division of $D^{n-k}\mathbf{w}(D)$ by the generator $\mathbf{g}(D)$; the code polynomial is $\mathbf{x}(D) = \rho(D) + D^{n-k}\mathbf{w}(D)$.
44. Decoding of cyclic codes makes use of the syndrome polynomial, which is a function of only the error polynomial and is calculated from $\mathbf{s}(D) = R_{\mathbf{g}(D)}[\mathbf{y}(D)]$. Error locations are calculated from the syndrome polynomial.
45. The parity check matrix for an (n, k) Hamming code has n columns and $n - k$ rows. The columns are all nonzero $(n - k)$-vectors. Hamming codes have $d_{\min} = 3$ and can correct all single errors. The generator polynomials $\mathbf{g}(D)$ for Hamming codes are primitive polynomials.

46. Both the encoder and decoder for Hamming codes can be implemented using shift register multiplication and division circuits.

47. The block error probability for a Hamming code with length $n = 2^j - 1$ is

$$P_B = 1.0 - (1 - p)^n - \sum_{l=1}^{n} p(1 - p)^{n-1} \tag{6–81}$$

This result is exact since the Hamming codes are "perfect."

48. The weight structure for a code defines a_d which is the number of codewords with Hamming weight d. For the Hamming codes

$$\begin{aligned} A(z) &= \sum_{d=0}^{n} a_d z^d \\ &= \frac{1}{n+1}\left[(1+z)^n + n(1+z)^{(n-1)/2}(1-z)^{(n+1)/2}\right] \end{aligned} \tag{6–83}$$

49. BCH codes are linear cyclic codes having a wide range of code rates and error correction capability. They are defined by their generator polynomials many of which have been cataloged in advanced texts. Bit error rate performance can be estimated, assuming the bounded distance decoder results of (6–74). Decoding is accomplished using a three-step procedure that is well defined but beyond the scope of this presentation.

50. The Reed–Solomon codes are nonbinary cyclic block codes which are used in compact disc audio systems, in concatenated coding systems, and in the NASA space station. The coder input and output alphabets contain 2^m symbols which can be constructed from groups of m binary symbols if desired. These codes correct e_0 errors in a block of $n = 2^m - 1$ symbols. There are $n - k = n - 2e_0$ parity symbols and the code rate is $R = k/n$.

51. The encoder and decoders for Reed–Solomon codes is similar to the encoders and decoders for binary BCH codes except that all arithmetic is performed modulo-2^m. The bit error probability for Reed–Solomon coding is bounded above by

$$P_b \le \sum_{i=e_0+1}^{2^m-1} \frac{i}{2(2^m-2)} \binom{2^m-1}{i} p_s^i (1-p_s)^{2^m-1-i} \tag{6–85}$$

where p_s is the symbol error probability for the 2^m-ary communications link. The improvement provided by Reed–Solomon codes in slow Ricean fading channels was calculated and plotted for specific cases.

52. The Golay code is one of the few known "perfect codes." The Golay code is a (23, 12) binary cyclic code. The bit error probability for the Golay code is bounded above by

$$P_b \le \frac{1}{12} \sum_{i=4}^{23} \min[12, i+3] \binom{23}{i} p^i (1-p)^{23-i} \tag{6–90}$$

where a bounded distance decoder is assumed. The weight structure for the Golay code is

$$A(z) = 1 + 253z^7 + 506z^8 + 1288z^{11} + 1288z^{12} + 506z^{15} + 253z^{16} + z^{23} \quad (6\text{–}91)$$

There is an extended (24, 12) nonperfect Golay code having slightly improved performance relative to the (23, 12) code. The weight structure for the extended code is

$$A(z) = 1 + 759z^8 + 2576z^{12} + 759z^{16} + z^{24} \quad (6\text{–}92)$$

REFERENCES

[1] C. E. Shannon, "A Mathematical Theory of Communication," *Bell System Technical Journal,* Vol. 27, pp. 379–423, 623–656, 1948.

[2] C. E. Shannon, "Communication in the Presence of Noise," *Proceedings of the IRE,* Vol. 37, pp. 10–21, January 1949.

[3] N. Wiener, *Extrapolation, Interpolation, and Smoothing of Stationary Time Series* (Cambridge, MA: MIT Press, 1949).

[4] R. W. Hamming, "Error Detecting and Error Correcting Codes," *Bell System Technical Journal,* April 1950.

[5] M. J. E. Golay, "Notes on Digital Coding," *Proceedings of the IRE,* Vol. 37, p. 657, June 1949.

[6] R. G. Gallager, *Information Theory and Reliable Communication* (New York: John Wiley, 1968).

[7] J. M. Wozencraft and I. M. Jacobs, *Principles of Communication Engineering* (New York: John Wiley, 1965).

[8] S. Lin and D. J. Costello, Jr., *Error Control Coding: Fundamentals and Applications* (Englewood Cliffs, NJ: Prentice-Hall, 1983).

[9] G. C. Clark and J. B. Cain, *Error-Correction Coding for Digital Communications* (New York: Plenum, 1981).

[10] E. R. Berlekamp, *Algebraic Coding Theory* (New York: McGraw-Hill, 1968).

[11] R. Blahut, *Theory and Practice of Error Control Codes* (Reading, MA: Addisson-Wesley, 1983).

[12] A. J. Viterbi and J. K. Omura, *Principles of Digital Communication and Coding* (New York: McGraw-Hill, 1979).

[13] W. W. Peterson and E. J. Weldon, Jr., *Error-Correcting Codes* (Cambridge, MA: MIT Press, 1972).

[14] D. A. Huffman, "A Method for the Construction of Minimum Redundancy Codes," *Proceedings of the IRE,* Vol. 40, pp. 1098–1101, September, 1962.

[15] J. Ziv and A. Lempel, "A Universal Algorithm for Sequential Data Compression," *IEEE Trans. Infor. Theory,* Vol. IT-23, pp. 337–343, May 1977.

[16] J. Ziv and A. Lempel, "Compression of Individual Sequences Via Variable-Rate Coding," *IEEE Trans. Infor. Theory,* Vol. IT-24, pp. 530–536, Sept 1978.

[17] T. Welch, "A Technique for High-Performance Data Compression," *IEEE Computer,* pp. 8–19, June 1984.

[18] A. D. Wyner and J. Ziv, "The Sliding-Window Lempel-Ziv Algorithm Is Asymptotically Optimal," *Proc. IEEE,* Vol. 82, pp. 872–877, June 1984.
[19] P. E. Bender and J. K. Wolf, "New Asymptotic Bounds and Improvements on the Lempel-Ziv Data Compression Algorithm," *IEEE Trans. Infor. Theory,* Vol. 37, pp. 721–729, May 1991.
[20] A. D. Wyner and A. J. Wyner, "Improved Redundancy of a Version of the Lempel-Ziv Algorithm," *IEEE Trans. Infor. Theory,* Vol. 41, pp. 723–731, May 1995.
[21] A. J. Wyner, "The Redundancy and Distribution of the Phrase Lengths of the Fixed-Database Lempel-Ziv Algorithm," *IEEE Trans. Infor. Theory,* Vol. 43, pp. 1452–1464, Sept. 1997.
[22] A. D. Wyner, J. Ziv, and A. J. Wyner, "On the Role of Pattern Matching in Information Theory," *IEEE Trans. Infor. Theory,* Vol. 44, pp. 2045–2056, Oct. 1998.
[23] J. P. Oldenwalder, "Error Control," In *Data Communications, Networks, and Systems,* T. Bartree (ed.) (Indianapolis, IN: Howard W. Sams, 1984.)
[24] A. Papoulis, *Probability, Random Variables, and Stochastic Processes,* 2nd ed. (New York: McGraw-Hill, 1985).
[25] A. Hocquenghem, "Codes Corecteurs d'Erreurs," *Chiffres,* 1959.
[26] R. C. Bose and D. K. Ray-Chaudhuri, "On a Class of Error Correcting Binary Group Codes," *Infor. and Control,* Vol. 3, pp. 68–79, Mar. 1960.
[27] E. R. Berlekamp, "On Decoding Binary Bose-Chaudhuri-Hocquenghem Codes," *IEEE Trans. Infor. Theory,* Vol. IT-11, pp. 577–579, Oct. 1965.
[28] R. T. Chien, "Cyclic Decoding Procedure for the Bose-Chaudhuri-Hocquenghem Codes," *IEEE Trans. Infor. Theory,* Vol. IT-10, pp. 357–363, Oct. 1964.
[29] L. J. Weng, "Soft and Hard Decoding Performance Comparisons for BCH Codes, *Conf. Record: Intern. Conf. On Commun.,* pp. 25.5.1–25.5.5, June 1979.
[30] I. S. Reed and G. Solomon, "Polynomial Codes Over Certain Finite Fields," *Journ. of the Soc. For Ind. and Appl. Math.,* June 1960.
[31] M. K. Simon, J. K. Omura, R. A. Scholtz, and B. K. Levitt, *Spread Spectrum Communications* (Rockville, MD: Computer Science Press, 1985.)
[32] E. R. Berlekamp, "The Technology of Error-Correcting Codes," *Proc. IEEE,* Vol. 68, pp. 564–593, May 1980.
[33] B. Sklar, *Digital Communications, Fundamentals, and Applications* (Upper Saddle River, NJ: Prentice Hall, 1988).
[34] Consultative Committee for Space Data Systems, *Blue Book,* CCSDS 101.0B-2, 1987.
[35] T. Kasami, "A Decoding Procedure for Multiple-Error-Correcting Cyclic Codes," *IEEE Trans. Infor. Theory,* Vol. IT-10, pp. 134–138, April 1964.
[36] J. H. Yuen (ed.), *Deep Space Telecommunications Systems Engineering,* (New York: Plenum, 1983).

PROBLEMS

6–1. A discrete memoryless source outputs letters from the alphabet {0, 1, 2, 3} with probabilities {0.2, 0.4, 0.2, 0.2}.
(a) What is the information content of each letter?
(b) What is the average information content of the source output?
(c) How many bits per letter is required to transmit the DMS output without source coding?

(d) Use the Huffman procedure to assign binary codewords to the letters and calculate the resultant source coder average output bit rate.
(e) Use the Huffman procedure to encode the second extension of the source and calculate the resultant source coder average output bit rate.
(f) What is the bit sequence required to communicate the DMS output symbol sequence 1, 3, 0, 1, 3, 1, 2, 0, 2, 1 using each of the three (uncoded, Huffman, and second extension Huffman) coding procedures?

6–2. A communicator must select a source code for the transmission of English text. Assume that the probability of a vowel is three times the probability of a consonant. Compare the binary channel bit rates required communicate using a fixed number of bits to represent each letter to the rate required using a variation of Morse code. Assume that Morse code DIT and DAH are transmitted using "0" and "1," respectively, and that a single "0" separates each Morse codeword.

6–3. A discrete memoryless source outputs one symbol from the alphabet {A, B, C, D, E, F, G, H} each 1.0 millisecond. The probability of occurrence for these symbols are {0.01, 0.03, 0.35, 0.02, 0.15, 0.18, 0.19, 0.07}.
(a) What is the entropy of this source?
(b) Use the Huffman procedure to assign binary codewords to all symbols.
(c) What is the average source encoded output bit rate?
(d) What is the minimum binary channel bit rate required to communicate with and without source coding?

6–4. The receiver in a communications system has a received signal power of −134 dBm, a received noise power spectral density of −164 dBm/Hz, and a bandwidth of 2000 Hz. What is the maximum rate of error-free information transfer of this system?

6–5. A BPSK channel with a symbol rate of 1000 symbols per second is available for communication of binary information. The maximum transmitter output power is used resulting in a received signal power of −134 dBm. The received noise density is −164 dBm/Hz. What is the maximum number of bits of information per second which can be transmitted over this channel without error? What is the maximum rate of error-free information transfer possible if transmitter power were permitted to increase without bound?

6–6. Given a source alphabet $\mathbf{W} = \{a,b,c\}$, use the Lempel–Ziv–Welch procedure to encode the sequence $\{a\ b\ a\ b\ c\ b\ a\ b\ a\ b\ a\ a\ a\ a\ a\ a\ a\}$. This sequence is used in the example in [17].

6–7. Use the sliding window version of the Lempel–Ziv procedure with a window size $n_w = 12$ to encode the binary sequence {011100111010110100110001 101110000011}. Express the encoder output as a binary sequence. Decode the encoded sequence.

6–8. A Lempel–Ziv–Welch decoder knows that the source alphabet contains three symbols. Decode the received sequence {1, 2, 4, 3, 5, 8, 1, 10, 11}.

6–9. Use the sliding window Lempel–Ziv procedure to encode and then decode the following binary sequence {0010100100000110100110100101100010001}. Express the codewords in the form $C_l = [m_l, L_l]$, that is, do not convert the codewords to their binary form. Use a window size $n_w = 10$.

6–10. Use the Lempel–Ziv–Welch procedure to decode the sequence {122341387699}. The source encoder alphabet is binary. What is the complete dictionary?

6–11. A binary symmetric channel with an error probability of 0.01 and rate of 9600 symbols/second is available to a communicator. The information user requires the information error rate to be better than 1 error in 10^6 received bits.
(a) Calculate the capacity C_N and cutoff rate R_0 for this channel.
(b) What is the maximum rate of information transfer to the user that can be theoretically supported and that can be supported with some confidence that the coding system will have reasonable complexity?

6–12. Calculate and plot R_0 and C_N as a function of received E_b/N_0 for a binary phase-shift keyed communication system. What is the maximum value for each of these capacity measures and why does this maximum exist?

6–13. Can communications at a rate of 2500 bps and $P_b \leq 10^{-5}$ be supported by a BSC using a symbol rate of 10,000 symbols/second that has a symbol error rate of 9 errors in 100 transmissions?

6–14. Consider an error correction code that generates codewords by simply repeating each information symbol five times.
(a) What is the code rate?
(b) How many codewords are in the code?
(c) What is the minimum distance of the code?
(d) How many channels errors would have to occur to confuse one codeword with another?
(e) What fraction of all possible codewords are used in this code?

6–15. Consider the code of Table 6–4. Calculate the number of codewords with Hamming weight $w = 0, 1, \ldots, 7$. Calculate the Hamming distance between each codeword and codeword 1111111. Compare the number of codewords with Hamming distance $w = 0, 1, \ldots, 7$ with the number of codewords with Hamming weight w from 1111111.

6–16. The codeword $\mathbf{x} = 0110100$ is transmitted and the vector $\mathbf{y} = 0010101$ is received over a BSC with error probability $p = 0.01$. What is the probability of this event? What is the error vector that caused this event?

6–17. What is the probability that three or fewer errors occur during the transmission of twelve codeword symbols over a BSC with error probability p?

6–18. Consider the code of Table 6–4. The codeword $\mathbf{x}_0 = 0000000$ is transmitted over a BSC with error probability p. What is the probability that some other valid codeword is received?

6–19. List all possible error 5-vectors with Hamming weight 2. Compare the number of 5-vectors in your list to the binomial coefficient of (6–30).

6–20. Consider again the repeat code (Problem 6–14) that generates codewords by a 5-times repetition of the information bit. List all 2^n possible received 5-vectors and partition this vector space into the optimum decoding regions. How many transmission errors must occur to cause a decoding error?

6–21. Decode the following received sequences assuming the Hamming code of Table 6–4: 1011011, 1111100, 1000110, 1100011.

6–22. Consider the code of Table 6–4. What is the probability of receiving the vector 1101111 given that all messages are equally likely and given a BSC with transmission error probability $p = 0.03$?

6–23. Suppose that a BSC was available for which $p > 0.5$. Is the minimum distance decoding rule optimum for this channel? Explain. If not, what is a better decoding rule?

6–24. Generate the optimum decoding regions for the 5-times repeat code (Problem 6–14) when the message probabilities are $Q_w(1) = 0.7$ and $Q_w(0) = 0.3$ and the BSC has transmission error probability of 0.01.

6–25. Calculate the exact bit error probability for the 5-times repeat code as a function of the BSC transmission error probability p.

6–26. A vector space is generated by forming all linear combinations of the following set of vectors:

$$\mathbf{b}_0 = 1\,1\,0\,1\,0\,0\,0$$

$$\mathbf{b}_1 = 0\,1\,1\,0\,1\,0\,0$$

$$\mathbf{b}_2 = 0\,1\,0\,0\,0\,1\,1$$

$$\mathbf{b}_3 = 1\,1\,1\,0\,0\,1\,0$$

$$\mathbf{b}_4 = 1\,0\,1\,0\,0\,0\,1$$

(a) What is the dimension of the resulting vector space?
(b) Find a basis for the vector space.
(c) Is the set of vectors **b** linearly independent?

6–27. Perform the vector arithmetic requested using the following vector set:

$$\mathbf{b}_0 = 0\,0\,0\,0\,0\,0$$

$$\mathbf{b}_1 = 0\,0\,1\,1\,1\,0$$

$$\mathbf{b}_2 = 0\,1\,0\,1\,0\,1$$

$$\mathbf{b}_3 = 1\,0\,0\,0\,1\,1$$

$$\mathbf{b}_4 = 0\,1\,1\,0\,1\,1$$

$$\mathbf{b}_5 = 1\,0\,1\,1\,0\,1$$

$$\mathbf{b}_6 = 1\,1\,0\,1\,1\,0$$

$$\mathbf{b}_7 = 1\,1\,1\,0\,0\,0$$

(a) $\mathbf{b}_2 + \mathbf{b}_7 = ?$ (b) $\mathbf{b}_3 + \mathbf{b}_5 + \mathbf{b}_7 = ?$ (c) $0 \cdot \mathbf{b}_5 = ?$
(d) $1 \cdot \mathbf{b}_5 = ?$ (e) $\mathbf{b}_6 \cdot \mathbf{b}_2 = ?$ (f) $\mathbf{b}_2 \cdot \mathbf{b}_0 = ?$

6–28. Calculate all vectors in the linear vector space defined by the following vectors:

$$\mathbf{g}_0 = 0\,0\,1\,1\,1\,0$$

$$\mathbf{g}_1 = 0\,1\,0\,1\,0\,1$$

$$\mathbf{g}_2 = 1\,0\,0\,0\,1\,1$$

Are the vectors **g** a basis? What is the dimension of the vector space?

6–29. A (6, 3) binary linear block code is defined by the following generator vectors:

$$\mathbf{g}_0 = 0\,0\,1\,1\,1\,0$$

$$\mathbf{g}_1 = 0\,1\,0\,1\,0\,1$$

$$\mathbf{g}_2 = 1\,0\,0\,0\,1\,1$$

(a) What is the generator matrix **G** of the code?
(b) How many rows and columns are in the code's parity check matrix?

6–30. The parity check matrix for a linear code is

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

Determine whether the following vectors are codewords in the code

$$\mathbf{y}_1 = 0\,0\,1\,1\,0\,0\,0$$

$$\mathbf{y}_2 = 1\,0\,1\,0\,0\,0\,1$$

$$\mathbf{y}_3 = 1\,1\,1\,0\,1\,0\,0$$

$$\mathbf{y}_4 = 0\,1\,0\,1\,0\,1\,1$$

$$\mathbf{y}_5 = 1\,1\,0\,1\,0\,0\,0$$

6–31. The generator matrix for a (6, 3) systematic binary linear block code is

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

(a) What is the parity check matrix for this code?
(b) Generate the standard array for this code. Note: One double error pattern will be used to complete the array.
(c) Calculate the syndrome vector for all of the correctable error patterns.
(d) Decode (i.e., find the *message* vector) the received sequence $\mathbf{y} = 101101$.

6–32. The message $\mathbf{w} = 101$ is transmitted over a communications link using ideal BPSK modulation. The system achieves $E_b/N_0 = 9.0$ dB.
(a) What is the probability of correctly receiving this message without forward error correction?
(b) What is the probability of correctly receiving this message if the (6, 3) code of Problem 6–31 is used?

6–33. Reconsider the 5-times repeat code (Problem 6–14) whose two codewords are generated by simply repeating the information symbol.
(a) Develop an exact formula for block error probability as a function of BSC error probability p assuming a minimum distance decoder.
(b) Apply the overbound of (6–68) to find block error probability as a function of BSC error probability p and compare with the result of part (a). Explain your observations.

6–34. The generator matrix for a (6, 3) systematic binary linear block code is

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

Assume that the communication system uses BPSK modulation.

(a) Calculate a_d and B_d for this code for $0 \le d \le n$.

(b) Calculate both bounds for P_B of (6–68) as a function of E_b/N_0 and plot your results.

(c) Calculate P_B as a function of E_b/N_0 assuming a bounded distance decoder and plot on the same graph as part (b).

(d) Calculate bit error probability P_b as a function of E_b/N_0 using both (6–73) and (6–74) and plot your results on the same graph.

6–35. Using the decoding regions for the (6, 3) code of Problem 6–34, graphically illustrate why the last result of (6–62) is an overbound rather than an exact result.

6–36. Represent the following vectors as polynomials in D:

$$\mathbf{y}_1 = 0\,0\,1\,1\,0\,0\,0$$

$$\mathbf{y}_2 = 1\,0\,1\,0\,0$$

$$\mathbf{y}_3 = 1\,1\,1\,0\,1\,0\,0\,1\,0\,1\,1\,1$$

$$\mathbf{y}_4 = 0\,1\,0\,1\,1\,1$$

$$\mathbf{y}_5 = 1\,1\,0\,1$$

6–37. Calculate the following polynomial products and illustrate the feedforward shift register that could be used to perform the calculation. Demonstrate that the shift-register functions properly by listing the contents of the shift register and its output for the first five clock cycles.

(a) $\mathbf{x}(D) = (1 + D + D^3) \cdot (1 + D^2)$

(b) $\mathbf{x}(D) = (1 + D + D^4) \cdot (D^2 + D^3 + D^4)$

(c) $\mathbf{x}(D) = (D + D^6 + D^7) \cdot (1 + D + D^9)$

6–38. Divide the polynomial $\mathbf{x}(D) = 1 + D^3$ by the polynomial $\mathbf{g}(D) = 1 + D + D^3 + D^5$ to obtain both the quotient and remainder. Illustrate the shift-register circuit that is used to perform this calculation and demonstrate its complete operation.

6–39. Show that the generator polynomial for the (15, 11) cyclic Hamming code is a factor of $D^{15} + 1$.

6–40. Consider the (7, 4) cyclic Hamming code with generator polynomial $\mathbf{g}(D) = 1 + D + D^3$. Calculate any code polynomial and its fourth cyclic shift both manually and by division of $D^4\mathbf{x}(D)$ by $D^7 + 1$.

6–41. Find the generator and parity check matrices for the (7, 4) cyclic Hamming code in the form of (6–77) and (6–78). Generate all possible codewords from the generator matrix and compare the result with Table 6–4.

6–42. Using the three-step procedure of Section 6.3.3.4, generate the codewords for the (15, 11) Hamming code corresponding to message polynomials

(a) $\mathbf{w}(D) = 1 + D^4 + D^{10}$ and

(b) $\mathbf{w}(D) = D^3 + D^{10}$. Verify that the message polynomial appears within the code polynomial.

6–43. Illustrate the shift-register decoder for the (15, 11) Hamming code and decode the received polynomial $\mathbf{y}(D) = D^3 + D^4 + D^7 + D^9$.

6–44. Determine the parity check matrix for the (31, 26) Hamming code.

6–45. Calculate the weight structure for the (7, 4) Hamming code to verify the expansion following (6–83).

6–46. Calculate the block error probability for the (15, 11) Hamming code assuming a bounded distance decoder and compare your results to those of Figure 6–22.

6–47. Prove that, for the Hamming codes, (6–69) and (6–81) are identical.

6–48. Calculate the code polynomial corresponding to message polynomial $\mathbf{w}(D) = 1 + D^{12} + D^{21} + D^{50}$ for the (63, 51) two-error correcting BCH code.

6–49. Calculate and plot bit error probability as a function of E_b/N_0 assuming BPSK signaling for the (255, 99) 23-error correcting BCH code.

6–50. Calculate and plot the bit error probability as a function of E_s/N_0 for a Reed–Solomon code with rate $R = 1/4$ using 16–FSK modulation. How many symbol errors per codeword can be corrected by this code?

6–51. Calculate the minimum Hamming distance as a function of block length for Reed–Solomon codes with rate approximately 1/2.

6–52. Calculate the coding gain at $P_b = 10^{-5}$ for a Reed–Solomon code with rate 1/4 and $m = 5$. The physical channel is an ideal BPSK binary communications link.

6–53. Calculate and plot the block error rate for the (23, 12) Golay code using the bound of (6–68).

6–54. What fraction of all possible n-vectors is used by the Golay code?

6–55. **(a)** Calculate the (23, 12) Golay codewords for all Hamming weight 1 messages.
(b) Compare the Hamming weight of these codewords with the weight structure given in (6–91).
(c) Is the Hamming weight of these codewords consistent with the principles of good forward error correction code design in that the most likely error events are associated with a minimum number of bit errors?

6–56. Show that (6–86) and (4–124) are equivalent.

6–57. Write a MATLAB program to reproduce the curves shown in Figures 6–33 and 6–34.

6–58. (i) It is desired to communicate at 10 kbps and 10^{-5} bit error probability through a slow Ricean fading channel with $K = 0$ dB. Noncoherent MFSK with Reed–Solomon coding is to be used. Find the minimum E_b/N_0 required for the following channel bandwidths:
(a) 100 kHz; **(b)** 150 kHz; **(c)** 250 kHz.
(ii) Repeat for $K = 10$ dB
You may use the graphs shown in Figures 6–33 and 6–34.
Hint: Don't forget to include bandwidth expansion due to code rate.