- Introduction
- Few Terms Related to Error Control Coding
- Parity Coding
- Vertical Redundancy Check (VRC)
- Linear Block Codes
- Hamming Codes
- Encoder of (7, 4) Hamming Code
- Syndrome Decoding: Method of Correct Errors
- Error Correction using Syndrome Vector
- Syndrome Decoder for (n, k) Block Code
- Other Linear Block Code
- Cyclic Codes
- Generator and Parity Check Matrices of Cyclic Codes
- Encoders for Cyclic Codes
- BCH Codes (Bose-Chaudhri-Hocquenghem Codes)
- Convolutional Codes

9.1. Introduction

As discussed earlier that during the transmission process, the transmitted signal passes through some noisy channel. Due to noise interference, some errors are introduced in the received data. These errors can be detected and sometimes corrected using coding techniques. Generally, the error control methods are of two types:

(*i*) Error detection with retransmission, and

(*ii*) Forward acting error correction.

In case of first method, when an error is detected, the retransmission request (ARQ) is sent back to the transmitter. In the second method, the errors are detected and corrected by proper coding techniques at the receiver end. Whenever a single source transmits data to a number of receivers, the forward acting error correction is used. This is due to the fact that the retransmission is impossible in these cases.

In this chapter, we shall study some simple error detecting techniques such as parity checking. Some of the coding techniques which detect as well as correct errors will also be discussed in the subsequent sections. Coding techniques add some extra bits to the message bits and these bits are used to detect errors at the receiver end. Although these extra bits reduce the bit rate of the transmitter and its power but the advantage is that error probability is reduced.

Figure 9.1 shows the block diagram of the channel encoder and decoder.

A channel encoder adds extra bits to the message signal. On the other hand, the channel decoder identifies these redundant bits and uses them to detect and correct the errors in the message bits if there is any. Let us consider that the 'k' message bits are combined in one block for encoding. Some bits are added by the encoder to this block and total encoded bits become 'n'. Then such a code is known as **(n, k) *block code***. Then we can write,

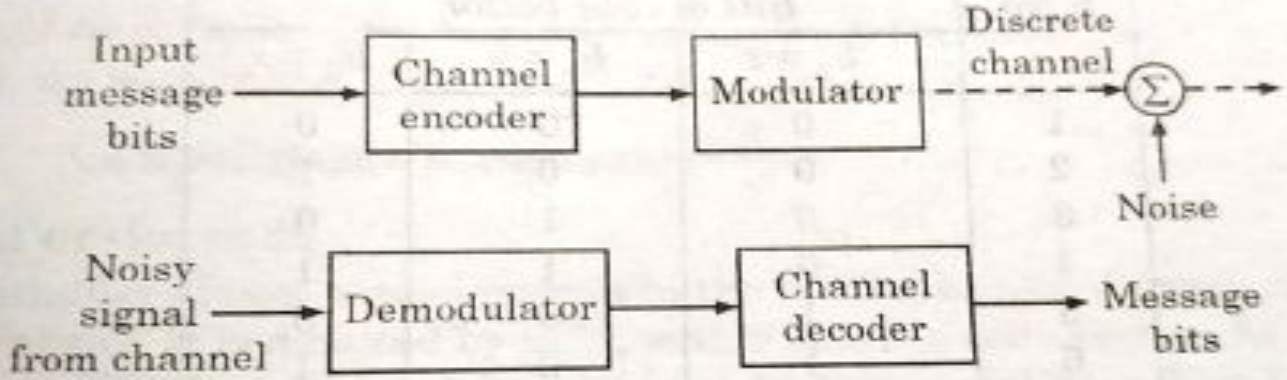


Fig. 9.1. *Digital communication system with channel encoding/decoding operations*

Redundant bits added to the block by encoder = (Bits at the output of encoder) – (Bits at the input of encoder)

$$= n - k \qquad \ldots(9.1)$$

These $(n - k)$ redundant bits (*i.e.*, code bits) are related to 'k' message bit depending on the coding technique used.

9.2. Few Terms Related to Error Control Coding

Some of the related terms are described here as under:

1. Codeword

The encoded block of 'n' bits is known as a **code word**. It consists of message bits and redundant bits.

2. Block Length

The number of bits 'n' after coding is known as the block length of the code.

3. Code Rate

The ratio of message bits (k) and the encoder output bits (n) is known as **code rate**. Usually, code rate is denoted by 'r' *i.e.*,

$$r = \frac{k}{n} \qquad ...(9.2)$$

we find that $0 < r < 1$.

4. Channel Data Rate

Channel data rate is the bit rate at the output of encoder. If the bit rate at the input of encoder is R_s, then channel data rate will be,

$$\text{Channel data rate } (R_0) = \frac{n}{k} R_s \qquad ...(9.3)$$

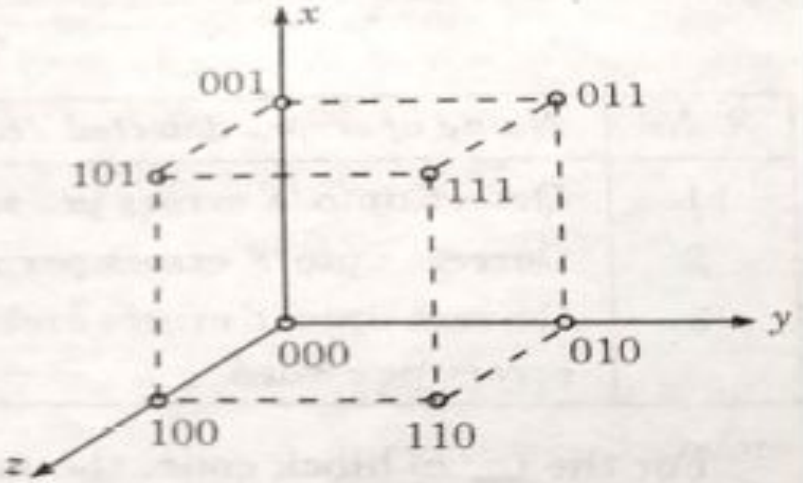


Fig. 9.2. *Code vectors representing 3-bit code words*

5. Code Vectors

An 'n' bit code word can be visualized in an n-dimensional space as a vector whose elements or co-ordinates are the bits in the code word. Let us visualize the 3-bit code words as it is quite simpler. Figure 9.2 shows the 3-bit code vectors. There will be distinct '8' code words (since number of code words = 2^k). If we let bits b_0 on x-axis, b_1 on y-axis and b_2 on z-axis, then the following Table 9.1 shows various points as code vectors in the 3-dimensional space.

Table 9.1

S.No	Bits of code vector		
	$b_2 = z$	$b_1 = y$	$b_0 = x$
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

6. Hamming Distance

The Hamming distance between the two code vectors is equal to the number of elements in which they differ.

As an example, let $X = 101$ and $Y = 110$. Since the two code vectors differ in second and third bits, therefore, Hamming distance between X and Y is 'two'. Hamming distance is denoted by $d(X, Y)$ or simply 'd'. *i.e*,

$$d(X, Y) = d = 2$$

Hence, it may be observed from figure 9.2 that the hamming distance between (100) and (011) is maximum, *i.e.* 3. This has been indicated by the vector diagram also.

7. Minimum Distance (d_{min})

Minimum distance (d_{min}) is the smallest Hamming distance between the valid code vectors.

Note: *It may be noted that the error detection is possible if the received vector is not equal to some other code vector. This shows that the transmission errors in the received code vector must be less than minimmum distance* d_{min}. *Table 9.2 shows some of the requirements of error capability of the code.*

7. Minimum Distance (d_{min})

Minimum distance (d_{min}) is the smallest Hamming distance between the valid code vectors.

Note: *It may be noted that the error detection is possible if the received vector is not equal to some other code vector. This shows that the transmission errors in the received code vector must be less than minimmum distance d_{min}. Table 9.2 shows some of the requirements of error capability of the code.*

Table 9.2.

S. No.	*Name of errors detected / corrected*	*Distance requirement*
1	Detect upto 's' errors per word	$d_{min} \geq s + 1$
2	Correct upto 't' errors per word	$d_{min} \geq 2t + 1$
3	Correct upto 't' errors and detect $s > t$ errors per word	$d_{min} \geq t + s + 1$

For the (n, k) block code, the minimum distance is expressed as

$$d_{min} \leq n - k + 1 \qquad ...(9.4)$$

9. Code Efficiency

The code efficiency is expressed as the ratio of message bits in a block to the transmitted bits for that particular block by the encoder *i.e.*,

$$\text{Code efficiency} = \frac{\text{Message bits in a block}}{\text{Transmitted bits for the block}}$$

Further, since for an (n, k) block code, there are 'k' message bits and 'n' transmitted bits, therefore code efficiency will be

$$\text{Code efficiency} = \frac{k}{n} \qquad \ldots(9.5)$$

Now, if we compare the above expression with the code rate (r) given in equation (9.2), we note that,

$$\text{Code efficiency} = \text{code rate} = \frac{k}{n} \qquad \ldots(9.6)$$

9. Weight of the code

The number of non-zero elements in the transmitted code vector is known as **vector weight**. It is denoted by $\omega(X)$, where X is the code vector. As an example, if $X = 01110101$, then the weight of this code vector will be $\omega(X) = 5$.

9.3. Parity Coding

The simplest method of coding is a parity coding. In this method, parity bits are added to the message at the time of transmission. At the reception end, receiver checks these parity bits. Errors are detected if the expected pattern of parity bits is not received.

It may be noted that parity coding is used only to detect the errors.

Parity may be of two types:

(*i*) Even Parity

The parity of a binary word is known as 'even' if it contains even number of 1's. For example, 01101001 has even parity since there are four number of 1's in the word.

(*ii*) Odd Parity

The parity of a binary word is known as 'odd' if it contains odd number of 1's. For example, 10010001 has odd parity because there are three number of 1s in

9.4. Vertical Redundancy Check (VRC)

In this method, the parity check is done on each character separately. This method is always used to ASCII code. In ASCII code, each character has seven binary digits. Also, one redundant digit is added to each character to make the parity odd or even. As an example, let us consider the coding shown in Table 9.3.

Table 9.3. Vertical redundancy check, even parity

Character	*ASCII code of the character*								*Additional parity bit*
	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8	
K	1	1	0	1	0	0	1	0	← Even number of 1s in ASCII code, therefore parity bit is zero.
O	1	1	1	1	0	0	1	1	↙ Odd number of 1s in ASCII code, therefore parity bit is one.
I	1	0	0	1	0	0	1	1	
A	1	0	0	0	0	0	1	0	

9.4.1. Few Points

From Table 9.3, let us note the following points:

(*i*) As depicted in the Table 9.3, the message consists of total 8 bits, *i.e.*, b_1 to b_16. Bits b_1 to b_7 are used for ASCII code of the character and the bit b_8 is used for parity bit.

(*ii*) For example, the ASCII code of *K* is 1101001. This consists of even number of 1s. Therefore, the 8th parity bit has '0' value. Hence, the transmitted message for *K* will be 11010010, which has even parity.

(*iii*) The ASCII code for 0 is 1111001. This contains odd (*i.e*, five) number of 1's. Hence, the parity bit has value of 1. Hence, the transmitted message for 0 is 11110011, which has even parity.

(*iv*) During the transmission, if single error is created, it will make number of 1's odd. For example, let us consider that the received message for character *K* is as shown below:

	b_1	b_2	b_3	b_4	b_5	b_6	b_7	b_8
Transmitted message for character K by Even Parity(see Table 16.3) →	1	1	0	1	0	0	1	0
Received message It contains odd number of 1s. It means that there is an error →	1	1	0	1	(1)	0	1	0

This digit is an error

Hence, as shown above, bit b_5 is in error and it makes number of 1s odd (*i.e.*, five). The receiver detects this error.

(*v*) However, it is not possible for the receiver to detect exactly which digit is in error. The receiver will only detect that *there are odd number of 1s and thus received message contains some error.*

(*vi*) Thus, the receiver has to send repeat request to the transmitter, to retransmit the message.

(*vii*) If two digits are in error, then number of 1's in the received message is even. In this case, receiver will detect no error. In fact, this is the major drawback of Vertical Redundancy Check (VRC).

9.4.2. Longitudinal Redundancy Check (LRC)

In the last subarticle, we have observed that VRC checks only in one direction. Hence, only single errors can be detected. But in LRC, large message block is divided into several characters. Now, parity check is applied in both row and columns. Thus, it is possible to detect upto tripplen errors. It is also possible to correct single errors with LRC. In LRC for checking the message block, a complete character known as '*Block Check Character*' (BCC) is added at the end of block of information. The block check character can be of even or odd parity. An even parity block check character (BCC) may be obtained by performing an exclusive OR operation on all the characters.

Figure 9.3 illustrates the structure of information block transmitted with block check character.

As shown in the figure 9.3, start of text (STX) is one character in ASCII and implies that information block follows this character. After the information block, one character indicating End of Transmitted Block (ETB) is transmitted. It is

also ASCII character. The Block Check Character (BCC) is transmited at the end. As an example, let us consider the transmitted and received message blocks as shown in Table 9.4 which uses LRC.

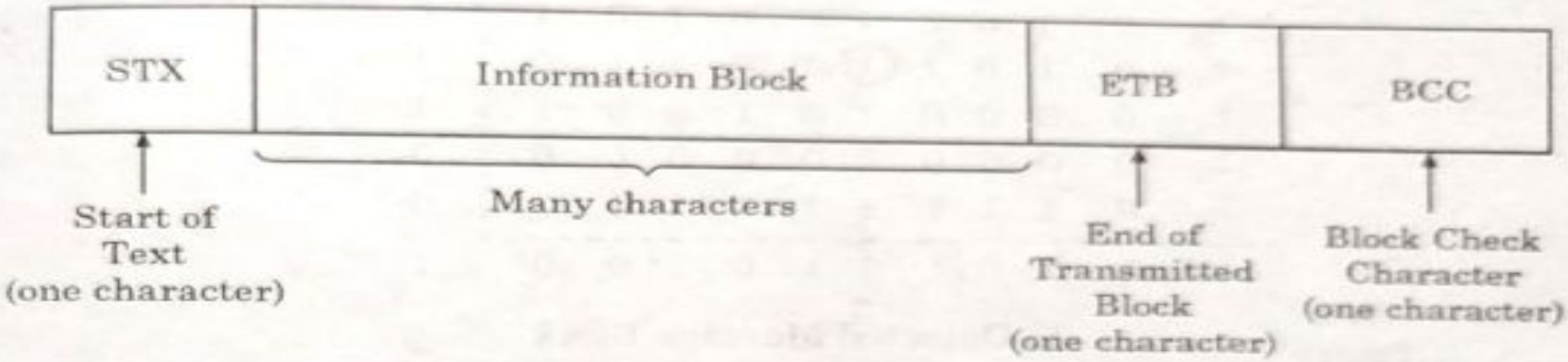


Fig. 9.3. *Structure of information block*

9.4.3. Few Points

(*i*) This method (*i.e.*, LRC) is used with ASCII codes only. Tables 9.4(*a*) shows the transmitted message block. The first row of this table shows characters which are transmitted.

(*ii*) It may be observed that the first character transmitted is STX. The ASCII value of STX is shown in its column. The ASCII value of STX is $(b_1 b_2 b_3 b_4 b_5 b_6 b_7) = (0100000)$.

(*iii*) The last row (for b_8) shows the even parity bits to be added for each character. Hence, $b_8 = 1$ for STX since ASCII code of STX contains only one (*i.e.* odd) binary 1. Hence, the transmitted character for STX is $(b_1 b_2 \ldots . b_8) = (01000001)$. This character is the first transmitted character according to figure 9.3.

(*iv*) As shown in Table 9.4 (*a*), the sequence of characters N, A, I, T,.... etc., is transmitted following STX. This is known as information block.

(*v*) In Table 9.4(*a*) it may be noted that the second last character transmitted is ETB (End of Transmitted Block). The ASCII value of ETB $(b_1, b_2, \ldots b_7)$

Table 9.4. LRC Using Block Check Character

Characters	STX	N	A	I	T	I	S	O	K	ETB	BCC ← Column Parity bits
Message Bits b_1	0	0	1	1	0	1	1	1	1	1	1
b_2	1	1	0	0	0	0	1	1	1	1	0
b_3	0	1	0	0	1	0	0	1	0	1	0
b_4	0	1	0	1	0	1	0	1	1	0	1
b_5	0	0	0	0	1	0	1	0	0	1	1
b_6	0	0	0	0	0	0	0	0	0	0	0
b_7	0	1	1	1	1	1	1	1	1	0	0
Character parity Row Parity Bits b_8	1	0	0	1	1	1	0	1	0	0	1

(a) Transmitted Message Block

	N	A	I	–	I	S	O	K		
b_1	0	0	1	1	0	1	1	1	1	1 ← Column Parity Check
b_2	1	1	0	0	0	0	1	1	1	1 0
b_3	0	1	0	0	1	0	0	1	0	1 0
b_4	0	1	0	1	(1)	1	0	1	1	0 1 ×
b_5	0	0	0	0	1	0	1	0	0	1 1
b_6	0	0	0	0	0	0	0	0	0	0 0
b_7	0	1	1	1	1	1	1	1	1	0 0
b_8	1	0	0	1	1	1	0	1	0	0 1
					×					

Row Parity Bits

(b) Detected Message Block

X implies an error in the Parity Check

(*vi*) After ETB, BCC is transmitted. BCC is shown in last column of Table 9.4(*a*). The value of BCC in particular row is obtained by performing exclusive OR operation on all the digits in that row.

(*vii*) As an example, let us consider the row for b_1. This is,

	STX	N	A	I	T	I	S	O	K	ETB	BCC
$b_1 \equiv$	0	0	1	1	0	1	1	1	1	1	1

(*viii*) At this point, it may be observed that the row for b_1 contains odd (*i.e.*, seven) number of 1*s*. Thus, exclusive OR operation will provide a value of 1. Hence, in the row of b_1, BCC has value of 1. Note that the complete row of b_1 has even parity.

(*ix*) Similarly, in the row of b_2, BCC is zero. It has also got even parity. Block check character (BCC) is thus obtained by parity coding. Hence, the last row *i.e.*, row of b_8 gives parity check in vertical direction and BCC provides parity check in horizontal direction.

(*x*) Now, let us consider the received message block as shown in Table 9.4(*b*). This message block is received in response to that shown in Table 9.4(*a*) As shown in Table 9.4(*b*), the encircled digit is in error. This error is easily located in row and column paity checks.

(*xi*) Hence, single error can be detected and corrected due to LRC method. Double and tripple errors can also be detected.

9.4.4. Advantages of LRC Over VRC

Now let us summarize advantages of LRC method over VRC method as under:

(*i*) In LRC method, single errors can be detected and corrected.

(*ii*) In LRC method, double and tripple errors can be detected.

The drawback of LRC is that complexity in the method is increased.

Check Sum

The bits corresponding to row b_8 are character parity bits. They are also called row parity bits. These bits are appended to the data bits as an error check group. These bits are also called as ***checksum*** combinely.

9.5. Linear Block Codes

For the block of k message bits, $(n - k)$ parity bits or check bits are added. This means that the total bits at the output of channel encoder are 'n'. Such types of codes are known as **(n,k) block codes**. Figure 9.4 illustrates this concept.

1. Systematic Codes

In the systematic block code, the message bits appear at the beginning of the code word. Thus, as shown in figure 9.4, the message bits appear first and then check bits are transmitted in a block. This type of code is known as the **systematic code**.

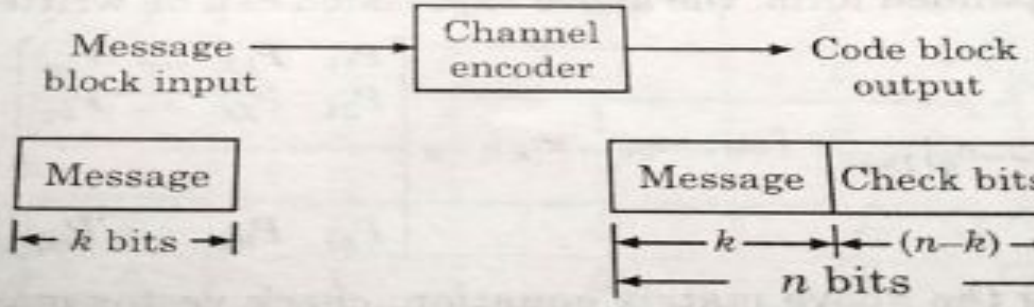


Fig 9.4. *Functional block diagram of a block coder*

In this section, we shall consider binary codes *i.e.*, in which all transmitted digits are binary.

2. Linear Code

A code is known as 'linear' if the sum* of any two code vectors produces anothe code vector. This shows that any code vector may be expressed as a linear combination of other code vectors.

9.5.1. Steps for Determination of all the Codewords for a Linear Block Code

(*i*) Let us consider that the particular code vector consists of $m_1, m_2, m_3, \ldots m_k$ message bits and $c_1, c_2, c_3 \ldots\ldots c_q$ check bits. Then this code vector may be written as under:

$$X = (m_1, m_2, \ldots m_k\, c_1, c_2, \ldots c_q) \quad \ldots(9.7)$$

where $q = n - k$ ...(9.8)

(*ii*) This means that q are the number of redundant bits added by the encoder. The above code vector may also be written as,

$$X = (M \mid C) \quad \ldots(9.9)$$

where $M = k$-bit message vector and

and $C = q$-bit check vector

(*iii*) Here, the check bits play the role of error detection and correction. The function of the linear block code is to generate these 'check bits'. The code vector can be represented as,

$$X = MG \quad \ldots(9.10)$$

where $X =$ Code vector of $1 \times n$ size or n bits

$M =$ Message vector of $1 \times k$ size or k bits

and $G =$ Generator matrix of $k \times n$ size.

(*iv*) Hence, equation (9.10) represents matrix form *i.e.*,

$$[X]_{1\times n} = [M]_{1\times k}\, [G]_{k\times n} \quad \ldots(9.11)$$

(*v*) The generator matrix depends upon the linear block code used. Normally, it is represented as,

$$G = \left[I_k \mid P_{k\times q}\right]_{k\times n} \quad \ldots(9.12)$$

Here $I_k = k \times k$ identity matrix, and

$P = k \times q$ submatrix

(*vi*) Now, the check vector may be obtained as,

$$C = MP \qquad \ldots(9.13)$$

Hence, in the expanded form, the above expression can be written as

$$[c_1, c_2, \ldots\ldots c_q]_{1\times q} = [m_1, m_2, \ldots m_k]_{1\times k} \begin{bmatrix} P_{11} & P_{12} & \ldots. & P_{1q} \\ P_{21} & P_{22} & \ldots. & P_{2q} \\ \vdots & & & \\ \vdots & & & \\ P_{k1} & P_{k2} & \ldots. & P_{kq} \end{bmatrix}_{k\times q} \qquad \ldots(9.14)$$

(*vii*) On solving the above matrix equation, check vector may be obtained as under:

$$\left.\begin{aligned} c_1 &= m_1 P_{11} \oplus m_2 P_{21} \oplus m_3 P_{31} \oplus \ldots. \oplus m_k P_{k1} \\ c_2 &= m_1 P_{12} \oplus m_2 P_{22} \oplus m_3 P_{32} \oplus \ldots. \oplus m_k P_{k2} \\ c_3 &= m_1 P_{13} \oplus m_2 P_{23} \oplus m_3 P_{33} \oplus \ldots. \oplus m_k P_{k3} \\ &\ldots.\text{and so on} \end{aligned}\right\} \qquad \ldots(9.14a)$$

Here, it may be noted that all the additions are mod-2 additions.

Example 9.1. The generator matrix for a (6, 3) block code is shown below. Obtain all code of words of this code.

$$G = \left[\begin{array}{ccc:ccc} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right]$$

Solution: We know that generation matrix is given by

$$G = \left[I_k : P_{k\times q}\right]_{k\times n} \qquad \ldots(i)$$

Given generator matrix is

$$G = \left[\begin{array}{ccc:ccc} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{array}\right] \quad \ldots(ii)$$

Comparing equation (*i*) with equation (*ii*), we get

$$I_k = I_{3\times 3} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

and $$P_{k\times q} = P_{3\times 3} = \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Here $k = 3$, $q - 3$ and $n = 6$.

This means that the block size of the message vector is 3 bits. Thus, there will be total 8 possible message vectors as shown in the Table 9.5.

S. No.	Bits of message vector in one block		
	m_1	m_2	m_3
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
6	1	0	1
7	1	1	0
8	1	1	1

The P submatrix is given in the example which is reproduced here i.e.

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

For the check bit vector, there would be three bits. These may be obtained according to equation (9.13) as under:

$$C = MP$$

$$[c_1 c_2 c_3] = [m_1\ m_2\ m_3] \begin{bmatrix} 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

Thus, from the above matrix multiplication, we get

$$c_1 = (0 \times m_1) \oplus (m_2) \oplus (m_3)$$
$$c_2 = (m_1) \oplus (0 \times m_2) \oplus (m_3)$$

or
$$c_3 = (m_1) \oplus (m_2) \oplus (0 \times m_3)$$

From the last three equations, we get

$$\left.\begin{aligned} c_1 &= m_2 \oplus m_3 \\ c_2 &= m_1 \oplus m_3 \\ c_3 &= m_1 \oplus m_2 \end{aligned}\right\} \quad \ldots(i)$$

There three equations give check bits for each block of m_1, m_2, m_3 message bits. For example, for first block of $(m_1, m_2, m_3) = 000$, we have

$$c_1 = 0 \oplus 0 = 0$$
$$c_2 = 0 \oplus 0 = 0$$
$$c_3 = 0 \oplus 0 = 0$$

i.e., $(c_1, c_2, c_3) = 0\ 0\ 0$

For second block of $(m_1, m_2, m_3) = 001$, we have

$$c_1 = 0 \oplus 1 = 1$$
$$c_2 = 0 \oplus 1 = 1$$
$$c_3 = 0 \oplus 0 = 0$$

i.e., $(c_1, c_2, c_3) = 1\ 1\ 0$

The Table 9.6 shows all the message bits, their check bits and code vectors calculated as above.

Table 9.6. Code Vectors of (6, 3) Block Code.

S. No.	*Bits of message vector in one block*			*Check bits*			*Complete code vector or Complete codeword*					
	m_1	m_2	m_3	$c_1 = m_2 \oplus m_3$	$c_2 = m_1 \oplus m_3$	$c_3 = m_1 \oplus m_2$	m_1	m_2	m_3	C_1	C_2	C_3
1	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	1	1	1	0	0	0	1	1	1	0
3	0	1	0	1	0	1	0	1	0	1	0	1
4	0	1	1	0	1	1	0	1	1	0	1	1
5	1	0	0	0	1	1	1	0	0	0	1	1
6	1	0	1	1	0	1	1	0	1	1	0	1
7	1	1	0	1	1	0	1	1	0	1	1	0
8	1	1	1	0	0	0	1	1	1	0	0	0

9.5.2. Concept of Parity Check Matrix (*H*) for Linear Block Code

As a matter of fact, for each block code, there is a $q \times n$ parity check matrix (H). This is described as under:

$$H = \left[P^T : I_q\right]_{q \times n} \quad \ldots(9.15)$$

Here, P^T is the transpose of P submatix. The P submatrix is defined as

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} & \cdots & P_{1q} \\ P_{21} & P_{22} & P_{23} & \cdots & P_{2q} \\ P_{31} & P_{32} & P_{33} & \cdots & P_{3q} \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ P_{k1} & P_{k2} & P_{k3} & \cdots & P_{kq} \end{bmatrix}_{k \times q} \quad \ldots(9.16)$$

The transpose of this submatrix will become by changing rows to columns i.e.,

$$P^T = \begin{bmatrix} P_{11} & P_{21} & P_{31} & \cdots & P_{k1} \\ P_{12} & P_{22} & P_{32} & \cdots & P_{k2} \\ P_{13} & P_{23} & P_{33} & \cdots & P_{k3} \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ P_{1q} & P_{2q} & P_{3q} & \cdots & P_{kq} \end{bmatrix}_{q \times k} \quad \ldots(9.17)$$

With the above equation, we can write equation (9.15) as under:

$$H_{q\times n}=\left[\begin{array}{ccccc:ccccc} P_{11} & P_{21} & P_{31} & \cdots & P_{k1} & 1 & 0 & 0 & \cdots & 0 \\ P_{12} & P_{22} & P_{32} & \cdots & P_{k2} & 0 & 1 & 0 & \cdots & 0 \\ P_{13} & P_{23} & P_{33} & \cdots & P_{k3} & 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots & \vdots & \cdots & \vdots \\ P_{1q} & P_{2q} & P_{3q} & \cdots & P_{kq} & 0 & 0 & 0 & \cdots & 1 \end{array}\right]_{q\times n} \quad \ldots(9.18)$$

Note: *It may be noted that if generator matrix (G) is given, then parity check matrix (H) can be obtained and vice-versa.*

9.6. Hamming Codes

Hamming codes are defined as the (n, k) linear block codes.

These codes satisfy the following conditions:

(*i*) Number of check bits $q \geq 3$

(*ii*) Block length $n = 2^q - 1$

(*iii*) Number of message bits $k = n-q$

(*iv*) Minimum distance $d_{min} = 3$

We know that the code rate is expressed as,

$$r = \frac{k}{n} = \frac{n-q}{n} \quad \text{(for Hamming code } k = n - q\text{)}$$

or
$$r = 1 - \frac{q}{n} \quad \ldots(9.19)$$

Now, substituting the value of $n = 2^q - 1$ in last equation, we get

$$r = 1 - \frac{q}{2^q - 1} \quad \ldots(9.20)$$

From this equation, we may observe that $r \equiv 1$ if $q >> 1$.

9.6.1. Error Detection and Correction Capabilities of Hamming Codes

Because the minimum distance (d_{min}) of Hamming code is 3, it can be used to detect double errors or correct single errors. This can also be obtained form the generalized Table 9.2.

For detecting double (2) errors $\Rightarrow d_{min} \geq 2 + 1$ *i.e.* $d_{min} \geq 3$

and for correcting upto one (1) errors $\Rightarrow d_{min} \geq 2(1) + 1$ *i.e.* $d_{min} > 3$

Example 9.2. The parity check matrix of a particular (7, 4) linear blok code is expressed as

$$[H] = \begin{bmatrix} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 0 & 0 & 1 \end{bmatrix} \quad \ldots(i)$$

(*i*) Obtain the generator matrix (G)

(*ii*) List all the code vectors

(*iii*) What will the minimum distance between code vector?

(*iv*) How many errors can be detected? How many errors can be corrected?

Solution: Here, given that $n = 7$ and $k = 4$

Number of check bits are $n - k = 7 - 4$ *i.e.* $q = 3$

Hence, $n = 2^q - 1 = 2^3 - 1 = 7$

This indicates that the given code is a Hamming code. The parity check matrix is of $q \times n$ size and is given by equation (9.18). It can be written as (with $q = 3$ and $n = 7$ and $k = 4$)

$$[H]_{3 \times 7} = \begin{bmatrix} P_{11} & P_{21} & P_{31} & P_{41} & \cdots & 1 & 0 & 0 \\ P_{12} & P_{22} & P_{32} & P_{42} & \cdots & 0 & 1 & 0 \\ P_{13} & P_{23} & P_{33} & P_{43} & \cdots & 0 & 0 & 1 \end{bmatrix} \quad \ldots(ii)$$

or $$[H]_{3 \times 7} = \left[P^T : I_3\right] \quad \ldots(iii)$$

On comparing parity check matrices of equations (*ii*) and (*iii*), we have

$$P^T = \begin{bmatrix} P_{11} & P_{21} & P_{31} & P_{41} \\ P_{12} & P_{22} & P_{32} & P_{42} \\ P_{13} & P_{23} & P_{33} & P_{43} \end{bmatrix} \quad \ldots(iv)$$

Now, according to equations (*i*) and (*iv*), we have

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

Hence, the P submatrix may be obtained as,

$$P = \begin{bmatrix} P_{11} & P_{12} & P_{13} \\ P_{21} & P_{22} & P_{23} \\ P_{31} & P_{32} & P_{33} \\ P_{41} & P_{42} & P_{43} \end{bmatrix} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{bmatrix}_{4 \times 3} \quad \ldots(v)$$

Further, we know that generator matrix G is expressed as under:

$$G = \left[I_k : P_{k \times q}\right]_{k \times n}$$

with $k = 4$, $q = 3$ and $n = 7$, the above equation becomes,

$$G = \left[I_4 : P_{4 \times 3}\right]_{4 \times 7}$$

Substituting the identity matrix I_4 of size 4×4 and parity submatrix $P_{4\times3}$ of size as obtained in equation (*v*), in above equation, we get

$$G = \left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]_{4 \times 7} \quad \ldots(vi)$$

($I_{4\times4}$ and $P_{4\times3}$)

This is the required generator matrix.

To find all the Codewords

The check bits can be obtained using equation (9.13), *i.e.*,

$$c = MP$$

In the more general form we can use equation (9.14) *i.e.*, (with $q = 3$, and $k = 4$), we have

$$[c_1\ c_2\ c_3]_{1\times 3} = [m_1\ m_2\ m_3\ m_4]_{1\times 4}\,[P]_{4\times 3}$$

$$[c_1\ c_2\ c_3] = [m_1\ m_2\ m_3\ m_4]\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 0\\ 1 & 0 & 1\\ 0 & 1 & 1\end{bmatrix}_{4\times 3}$$

Solving the above equation with mod-2 addition, we obtain,

$$c_1 = (1 \times m_1) \oplus (1 \times m_2) \oplus (1 \times m_3) \oplus (0 \times m_4)$$
$$c_2 = (1 \times m_1) \oplus (1 \times m_2) \oplus (0 \times m_3) \oplus (1 \times m_4)$$

and
$$c_3 = (1 \times m_1) \oplus (0 \times m_2) \oplus (1 \times m_3) \oplus (1 \oplus m_4)$$

Thus the above equation may be written as,

$$\left.\begin{aligned} c_1 &= m_1 \oplus m_2 \oplus m_3\\ c_2 &= m_1 \oplus m_2 \oplus m_4\end{aligned}\right\} \quad \text{and} \quad c_3 = m_1 \oplus m_3 \oplus m_4 \qquad \ldots(vii)$$

For example, if $(m_1\ m_2\ m_3\ m_4) = 1\ 0\ 1\ 1$, we obtain,

$$c_1 = 1 \oplus 0 \oplus 1 = 0$$
$$c_2 = 1 \oplus 0 \oplus 1 = 0$$

and
$$c_3 = 1 \oplus 1 \oplus 1 = 1$$

Hence, for message vector of (1 0 1 1) the check bits are $(c_1\ c_2\ c_3) = 001$. These, the systematic block code of the code vector (code word) may be written as,

$(m_1\ m_2\ m_3\ m_4\ c_1\ c_2\ c_3) = (1\ 0\ 1\ 1 : 0\ 0\ 1)$

Using the same procedure as given, we may obtain the other code sords or code vectors. Table 9.6 lists all the code vectors (code words). This table also lists the weight of each code word.

The Table 9.7 lists $2^k = 2^4 = 16$ code vectors along with their weights. The smallest weight of any non-zero code vector is 3. We know that the minimum distance is $d_{min} = 3$.

Hence we can write

The minimum distance of a linear block code is equal to the minimum weigth of any non-zero code vector. This means that

$$d_{\min} = [w\,(X)]_{\min} ; X = (0\ 0 \ldots 0)$$

(*iii*) Minimum distance between code words is

$$d_{\min} = 3$$

(*iv*) Now, since

$$d_{\min} = 3.$$

We know that

$$d_{\min} \geq s + 1$$

or
$$3 \geq s + 1$$

or
$$s \leq 2$$

Hence, two errors will be detected.

Table 9.7.

S. No.	Message vector M				Check bits by eq. (v) c			Code vector orcode word X							Weight of code vector
	m_1	m_2	m_3	m_4	c_1	c_2	c_3	m_1	m_2	m_3	m_4	c_1	c_2	c_3	$w(X)$
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	1	1	0	0	0	1	0	1	1	3
3	0	0	1	0	1	0	1	0	0	1	0	1	0	1	3
4	0	0	1	1	1	1	0	0	0	1	1	1	1	0	4
5	0	1	0	0	1	1	0	0	1	0	0	1	1	0	3
6	0	1	0	1	1	0	1	0	1	0	1	1	0	1	4
7	0	1	1	0	0	1	1	0	1	1	0	0	1	1	4
8	0	1	1	1	0	0	0	0	1	1	1	0	0	0	3
9	1	0	0	0	1	1	1	1	0	0	0	1	1	1	4
10	1	0	0	1	1	0	0	1	0	0	1	1	0	0	3
11	1	0	1	0	0	1	0	1	0	1	0	0	1	0	3
12	1	0	1	1	0	0	1	1	0	1	1	0	0	1	4
13	1	1	0	0	0	0	1	1	1	0	0	0	0	1	3
14	1	1	0	1	0	1	0	1	1	0	1	0	1	0	4
15	1	1	1	0	1	0	0	1	1	1	0	1	0	0	4
16	1	1	1	1	1	1	1	1	1	1	1	1	1	1	7

Also we have

$$d_{min} \geq 2t + 1$$

or $$3 \geq 2t + 1$$

or $$t \leq 1$$

Therefore one error will be corrected.

Hence, with the help of Hamming code ($d_{min} = 3$), two errors can be detected and single error can be corrected by its property

9.7. Encoder of (7, 4) Hamming Code

Figure 9.5 shows the encoder of (7, 4) Hamming code. This encoder is implemented for generator matrix of the example 9.2. The lower register contains check bits c_1, c_2, and c_3. These bits are obtained from the message bits by mod-2 additions. These additions are performed according to equation (*vii*). The mod-2 addition operation is nothing but exclusive-OR operation.

The switch '*S*' is connected to message register first and all message bits are transmitted. This switch is then connected to the check bit register and check bits are transmitted. This forms a block of '7' bits. After this input bits are taken for next block.

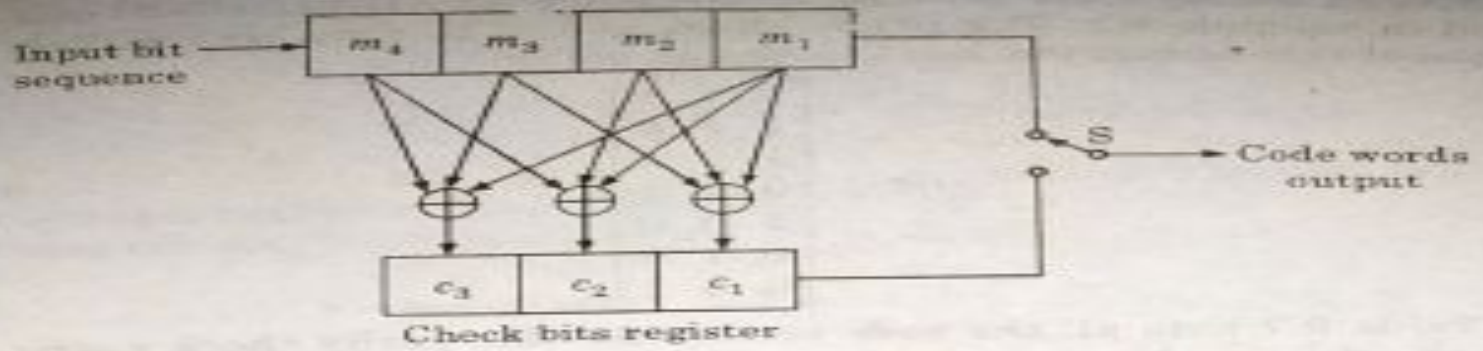


Fig. 9.5. *Encoder for (7, 4) Hamming code or (7, 4) linear block code for example 9.2.*

9.8. Syndrome Decoding: Method to Corect Errors

In this article, we shall discuss the method to correct errors in linear block coding. Let the transmitted code vector be 'X' and corresponding received code vector be represented by 'Y'.

Then, we can write,

$X = Y$ if there are no transmission errors

and $X \neq Y$ if there are errors produced during transmission

The decoder detects or corrects these errors in Y by using the stored bit pattern in the decoder about the code. For larger block lengths, more and more bits are needed to be stored in the decoder. This increases the memory requirement and adds to the complexity and cost of the system. Therefore, to avoid these problems, synchrome decoding is used in linear block codes. This method is shown in the paragraphs to follow:

(*i*) We know that with every (n, k) linear block code, there exists a parity check matrix (H) of size $q \times n$. It is defined as,

$$H = [P^T : I_q]_{q \times n}$$

The transpose of the above matrix may be obtained by interchanging the rows and the columns.

Thus, we have

$$H^T = \begin{bmatrix} P \\ \cdots \\ I_q \end{bmatrix}_{n \times q}$$

Here P is the submatrix of size $k \times q$ and I_q is the identity matrix of size $q \times q$. We have defined P submatrix earlier.

(*ii*) The transpose of parity check matrix (H^T) has very important property as under:

$$XH^T = (0\ 0\ 0\ \ldots 0) \quad \ldots(9.21)$$

or

$$[H]_{1 \times n} [H^T]_{n \times q} = (0\ 0\ 0\ \ldots 0)_{1 \times q} \quad \ldots(9.22)$$

(*iii*) As an example, let us consider the parity check matrix and code vectors obtained in example 9.2. The parity check matrix is given by equation (*i*). The transpose of this matrix can be obtained as under:

$$H^T = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{7 \times 3 \ (n=7 \text{ and } q=3)} \quad \ldots(9.23)$$

(*iv*) Table 9.7 lists all the code vectors for this parity check matrix. Let us consider the third code vector in this table *i.e.*,

$$X = (0\ 0\ 1\ 0\ 1\ 0\ 1) \quad \ldots(9.24)$$

Now, let us apply the property of equation (9.21) *i.e.*,

$$XH^T = [0010101]_{1 \times 7} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{7 \times 3} \quad \ldots(9.25)$$

(*v*) On solving the above two matrices with the rules of mod-2 addition (Exclusive - OR operation), we obtain,

$$\begin{aligned} XH^T &= (0 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \\ &\quad \oplus 0 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1) \\ &= (0 \quad 0 \quad 0) \end{aligned}$$

This proves the property.

(*vi*) It can also be proved for other code vectors also. Hence, X belongs to the valid code vector at the transmitter end. At the receiver end the received code is Y.

Then, we can write,

$YH^T = (0\ 0\ \ldots .0)$, if $X = Y$ *i.e.*, no errors or Y is valid code vector

and $YH^T =$ Non-zero, if $X \neq Y$ *i.e.*, some errors.

(*vii*) Also, when some errors are present in received vector Y, then it will not be from valid code vectors and it will not satisfy the property of equation (9.21). This shows that whenever YH^T is non-zero, some errors are present in Y. The non-zero output of the product YH^T is called *syndrome* and it is used to detect the errors in Y. Syndrome is represented by 'S' and may be written as,

$$S = YH^T \quad \ldots(9.26)$$

or

$$[S]_{1 \times q} = [Y]_{1 \times n} [H^T]_{n \times q} \quad \ldots(9.27)$$

(*viii*) The non-zero elements of 'S' represent error in the output. When all elements of 'S' are zero, then the two cases possible:

(*a*) No error in the output and $Y = X$

(*b*) Y is some other valid code vector other than X. This means that the transmission errors are undetectable.

(*ix*) Now, let us consider an n-bit error vector E. Let, this vector represent the position of transmission errors in Y. As an example, consider,

$X = (1\ 0\ 1\ 1\ 0)$ be a transmitted vector

and $Y = (1\ 0\ 0\ 1\ 1)$ be a received vector

Then $E = (0\ 0\ 1\ 0\ 1)$ represents the error vector

The non-zero entries represent errors in Y.

(*x*) Using the mod-2 addition rules, we can write:

$$Y = X \oplus E \qquad \ldots(9.28)$$

or $Y = (1 \oplus 0 \quad 0 \oplus 0 \quad 1 \oplus 1 \quad 1 \oplus 0 \quad 0 \oplus 1)$

Bit by bit mod-2 addition

or $Y = (1\ 0\ 0\ 1\ 1)$

or we can write $X = Y \oplus E$

$= (1 \oplus 0\ 0 \oplus 0\ 0 \oplus 1\ 1 \oplus 0\ 1 \oplus 1)$

or $X = (1\ 0\ 1\ 1\ 0)$

or we can write, $X = Y \oplus E$

$= (\oplus 0 \quad 0 \oplus 0 \quad 0 \oplus 1 \quad 1 \oplus 0 \quad 1 \oplus 1)$

or $X = (1\ 0\ 1\ 1\ 0)$

From equation (9.26), we know that syndrome vector is expressed as,

$$S = YH^T$$

Substituting the value of Y = X ⊕ E from equation (9.28) in this equation, we get

$$S = (X \oplus \mathrm{E})\ H^T$$

or $$S = XH^T \oplus EH^T$$

From the property of equation (9.21) we know that $XH^T = 0$, then above equation will be

$$S = EH^T \qquad \ldots(9.29)$$

(*xi*) This relation indicates that syndrome depends on the error pattern only. It does not depend upon a particular message. Syndrome vector 'S' is of size $1 \times q$. Hence, q bits of syndrome can only represent 2^q syndrome vectors. Each syndrome vector corresponds to a particular error pattern.

Example 9.3. The parity check matrix of a (7, 4) Hamming code is expressed as under:

$$H = \left[\begin{array}{cccc:ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right]_{3 \times 7}$$

Evaluate the syndrome vector for single bit errors.

Solution: This is a (7, 4) linear block code.

This means $n = 7$ and $k = 4$

$$q = n - k = 3$$

We know that syndorme vector is a q bit vector. For this example, syndrome will be a 3 bit vector. Hence, there will be $2^3 - 1 = 7$ non-zero syndromes. This shows that '7' single bit error patterns will be represented by these '7' non-zero syndromes. Error vector E is a n bit vector representing error pattern. For this example, E is '7' bit vector. Following Table 9.8 shows the single error patterns in a 7 bit error vector (Note that only single bit error patterns have been shown).

Table 9.8. Single error pattern of 7 bit error vector

S. No.	Bit in error	Bits of Error vector (E), Non-zeros bit shows error						
1	1^{st}	1	0	0	0	0	0	0
2	2^{nd}	0	1	0	0	0	0	0
3	3^{rd}	0	0	1	0	0	0	0
4	4^{th}	0	0	0	1	0	0	0
5	5^{th}	0	0	0	0	1	0	0
6	6^{th}	0	0	0	0	0	1	0
7	7^{th}	0	0	0	0	0	0	1

Calculation of Syndromes

From equation (9.29) the syndrome vector is given as,

$$S = EH^T$$

S is q bit, E is n bit and H^T is $n \times q$ bits size. For this example $n = 7$, $q = 3$, we can write above equation as,

$$[S]_{1\times 3} = [E]_{1\times 7}[H]_{7\times 3} \quad \ldots(i)$$

Form the given parity check matrix H, we can obtain its transpose (H^T) by interchanging rows to column, *i.e.*,

$$H^T = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \ldots(ii)$$

Let us calculate syndrome for first bit error vector *i.e.*,

$$S = EH^T = [1\,0\,0\,0\,0\,0\,0] \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= (1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \quad 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \quad 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0) = [1 \quad 0 \quad 1]$$

This is the syndrome vector for first bit in error.

Let us evaluate syndrome for 2^{nd} bit in error. In Table 9.7, the error vector for error in 2^{nd} bit is given. We may write,

$$S = EH^T = [0\,1\,0\,0\,0\,0\,0] \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

or $S = (0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \quad 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0$
$0 \oplus 1 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0)$

or $S = [1 \quad 1 \quad 1]$

Hence, this is the syndrome vector for second bit in error. The Table 9.9 lists the error vector with single bit error and corresponding syndromes. Other syndromes can be calculated using the same procedure as above. The table also lists the syndrome for no error vector *i.e.*, $E = (0\ 0\ 0\ 0\ 0\ 0\ 0)$. It may be observed that the corresponding syndrome is $S = (0\ 0\ 0)$.

Table 9.9. Syndromes for (7, 4) Hamming code of single bit error

S. No.	*Error vector 'E' showing single bit error patterns*							*Syndrome Vector 'S'*			
1	0	0	0	0	0	0	0	0	0	0	
2	1	0	0	0	0	0	0	1	0	1	← 1st row of HT
3	0	1	0	0	0	0	0	1	1	1	← 2nd row of H^T
4	0	0	1	0	0	0	0	1	1	0	← 3rd row of H^T
5	0	0	0	1	0	0	0	0	1	1	← 4th row of H^T
6	0	0	0	0	1	0	0	1	0	0	← 5th row of H^T
7	0	0	0	0	0	1	0	0	1	0	← 6th row of H^T
8	0	0	0	0	0	0	1	0	0	1	← 7th row of H^T

This Table (9.8) shows that error in the first bit corresponds to a syndrome vector of $S = (101)$. This syndrome vector is same as the first row of H^T. The syndrome vector ($S = 111$) for error in seccond bit is same as second row of H^T. This is same for remaining syndromes.

9.9. Error Correction Using Syndrome Vector

(*i*) Let us observe how single bit errors can be corrected using syndrome decoding. We will observe this for (7, 4) block code. Let the transmitted code vector be,

$$X = (1\ 0\ 0\ 1\ 1\ 1\ 0)$$

(*ii*) Let there be error created in the 3rd bit in the received code vector Y. Then Y will be

$$Y = (1\ 0\ \textcircled{1}\ 1\ 1\ 1\ 0)$$ encircled bit shows that it is in error.

(*iii*) Let us use the parity check matrix and syndrome vectors of, example 9.2 for this illustration.

The receiver calculates $S = YH^T$ *i.e.*,

$$S = YH^T = \underbrace{[1\ 0\ 1\ 1\ 1\ 1\ 0]}_{X} \underbrace{\begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}}_{H^T}$$ (from equation (9.26))

or $S = (1 \oplus 0 \oplus 1 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \quad 0 \oplus 0 \oplus 1 \oplus 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \quad 1 \oplus 0 \oplus 0 \oplus 1 \oplus 0 \oplus 0 \oplus 0)$

or $S = [1\ \ 1\ \ 0]$

(*iv*) Now, from equations (9.26) and (9.29), we can write

$$S = YH^T = EH^T$$

Here $S = YH^T = EH^T = (1\ 1\ 0)$.

(*v*) On comparing this syndrome with H^T, we observe that ($S = 1\ 1\ 0$) is the 3rd row of H^T. From Table 9.8, we can get the error pattern corresonding to this syndrome as under:

$$E = (0\ 0\ 1\ 0\ 0\ 0\ 0)$$

(*vi*) This shows that there is an error in the third bit of Y. We have also verifed that if syndrome vector is equal to 3rd row of H^T, then third bit of Y is in error. The correct vector can be obtained from equation (9.28) as under:

$$X = Y \oplus E$$

or $X = [1\ 0\ 1\ 1\ 1\ 1\ 0) \oplus [0\ 0\ 1\ 0\ 0\ 0\ 0] = (1\ 0\ 0\ 1\ 1\ 1\ 0)$

which is same as transmitted code vector.

Thus, a single bit errors can be corrected using syndrome decoding.

9.9.1. What Happens if Double Error Occurs in Y

Let us observe the case of double error in Y. Let us consider the same message vector X, i.e.,

$$X = 1\ 0\ 0\ 1\ 1\ 1\ 0$$

Let us consider that the error is present in 3rd and 4th bits. Then, Y will be

$$Y = 1\ 0\ \textcircled{1}\ \textcircled{0}\ 1\ 1\ 0$$

encircled bits are in error.

Then $S = YH^T$ gives,

$$S = YH^T = [1\ 0\ 1\ 0\ 1\ 1\ 0] \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Therefore, we have $S = 1\ 0\ 1$

From Table 9.8, we observe that the syndrome $S = 101$ corresponds to an error pattern of $E = 1000000$. This shows that there is an error in the first bit. Thus, the error detection and correction goes wrong. The probability of occurrence of multiple errors is less compared to single error. To correct multiple errors extended Hamming codes are used. In these codes, one more extra bit is provided to correct double errors.

We know that for (n, k) block code, there are $2^q - 1$ distinct non-zero syndromes. There are ${}^nC_1 = n$ single error patterns, ${}^nC_2$ double error patterns 9.7.2, ${}^nC_3$ tripple error patterns and so on. Therefore, to correct 't' errors per word, the following relation should be satisfied.

$$2^q - 1 \geq {}^nC_1 + {}^nC_2 + {}^nC_3 + \ldots + {}^nC_t \qquad \ldots(9.30)$$

9.9.2. Hamming Bound

The last equation (*i.e.*, equation (9.30)) can be written as

$$2^q \geq 1 + {}^nC_1 + {}^nC_2 + \ldots + {}^nC_t \geq \sum_{i=0}^{t} {}^nC_i$$

We know that $q = n - k$. Therefore, the above equation becomes,

$$2^{n-k} \geq \sum_{i=0}^{t} {}^nC_i$$

By taking logarithm to base 2 on both sides, we get

$$n - k \geq \log_2 \sum_{i=0}^{t} {}^nC_i$$

Dividing both sides by n, we get

$$1 - \frac{k}{n} \geq \frac{1}{n} \log_2 \sum_{i=0}^{t} {}^nC_i$$

Since code rate $r = \frac{k}{n}$, the above equation would be,

$$1 - r \geq \frac{1}{n} \log_2 \sum_{i=0}^{t} {}^nC_i \qquad \ldots(9.31)$$

As indicated, this equation relates code rate 'r' with the error correction capability of 't' errors per code vector in a block of 'n' bits. We know that the error correcting capability of the code (i.e., 't' errors per code vector) is related to the minimum distance. This minimum distance is also called ***Hamming distance.*** *Equation (9.31) gives the relation between code rate, number of errors to be corrected and number of bits in a block. This equation is also known as* ***Hamming bound.***

9.10. Syndrome Decoder for (n, k) Block Code

Figure 9.6 illustrates the block diagram of a syndrome decoder for linear block code to correct errors. The received n-bit vector 'Y' is stored in an n-bit register. From this vector, a syndrome is calculated using,

$$S = YH^T \qquad \ldots(9.32)$$

Hence, H^T is stored in the syndrome calculator. The q-bit syndrome vector is then applied to a look-up table of error patterns. Depending upon the particular

syndrome, an error pattern is selected. This error pattern's added (mod-2 addition) to the vector Y. Hence, the output will be

$$Y \oplus E = X \qquad \ldots(9.33)$$

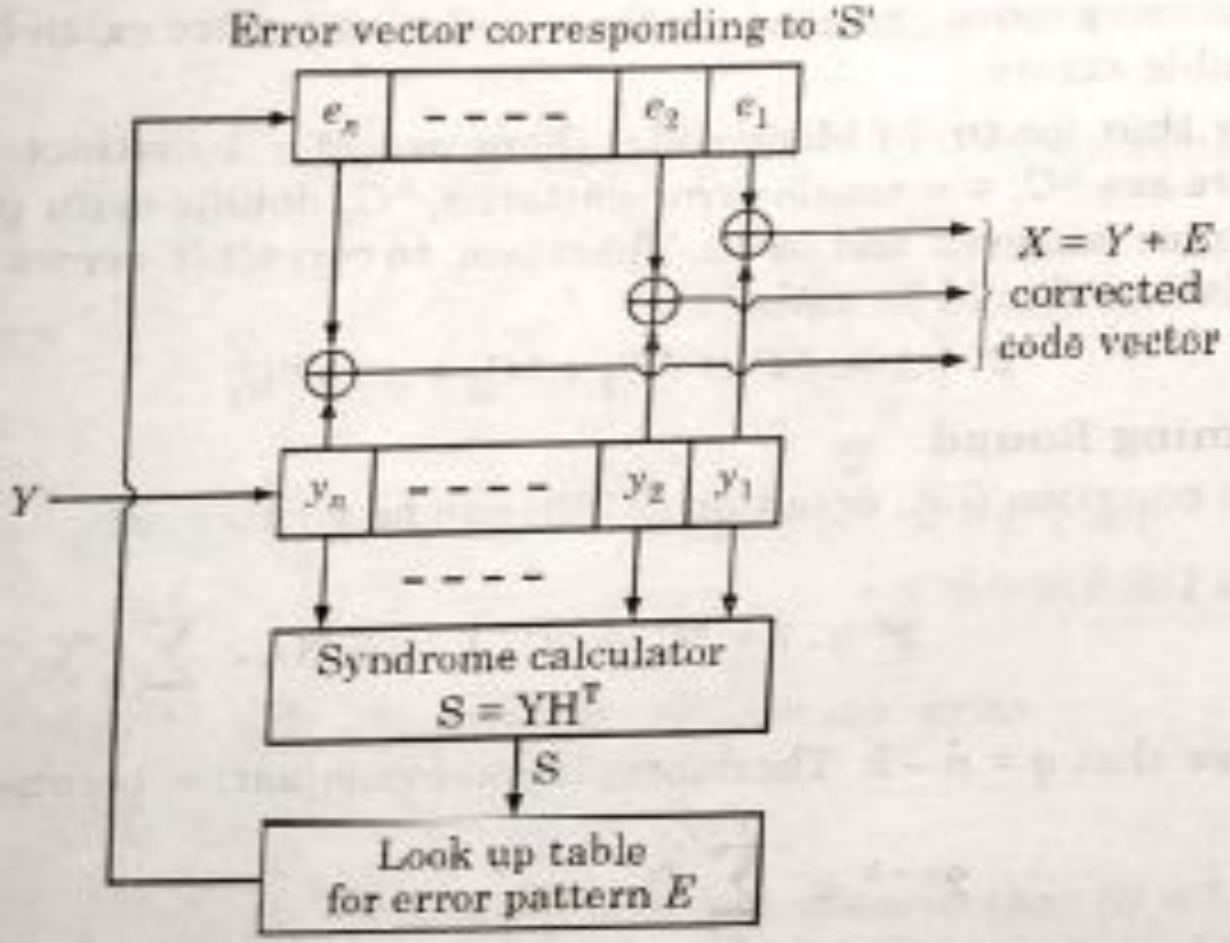


Fig. 9.6. *Syndrome decoder for linear block code*

The block diagram shown in figure 9.6 can correct only single errros in the above vectors.

Example 9.4. For a (6, 3) code, the parity check matrix is given by

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

Determine whether a received code vector is erroneous. Received code vector is 100101.

Solution: We have

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

The received code vector is $Y = 100101$.

Then,
$$S = YH^T = [100101] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [0\,0\,0]$$

Because, syndrome vector is zero, there are no errors in the received code vector.

Example 9.5. A(6, 3) linear block code is generated according to the following generator matrix:

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

For a particular code word transmitted, the received code word is 100011. Find the corresponding data word transmitted.

Solution: We have

Given $$G = [I_k : P] = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

$$H = \left[P^T : I_{n-k}\right] = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 0 & 1 & 1 \end{bmatrix}$$

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

Here $n = 6, k = 3, n - k = 6 - 3 = 3$

$Y = [1\ 0\ 0\ 0\ 1\ 1]$

We have

$$S = Y \cdot H^T = [1\ 0\ 0\ 0\ 1\ 1] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$S = [1\ 1\ 0]$

Also, $S = EH^T$

where $E = [e_1, e_2, e_3, e_4, e_5, e_6]$

i.e., $$S = [e_1, e_2, e_3, e_4, e_5, e_6] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [1, 1, 0]$$

On solving, we get

$$\therefore \quad e_1 + e_3 + e_4 = 1$$
$$e_2 + e_3 + e_5 = 1$$
$$e_1 + e_2 + e_6 = 0$$

By trial and error method, let us find all the combination of e's that satisfy these relations.

e_1	e_2	e_3	e_4	e_5	e_6
0	0	0	1	1	0
0	0	1	0	0	0
0	1	0	1	0	1
0	1	1	0	1	1
1	0	0	0	1	1
1	0	1	1	0	1
1	1	0	0	0	0
1	1	1	1	1	0

Thus there are $2^k (k = 3) = 8$ possible error vectors that satisfy the relation

$$S = EH^T$$

Among all the available error vectors, the vectors with minimum weight is selected (*i.e.*, the one having minimum number of 1's)

$$E = 0\ 0\ 1\ 0\ 0\ 0$$
$$Y = 1\ 0\ 0\ 0\ 1\ 1$$

$$X = Y + E = \begin{array}{cccccc} 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ \underbrace{1\ 0\ 1}_{M} & & & \underbrace{0\ 1\ 1}_{C} & & \end{array}$$

Data word transmitted is 101 or simply the row in H^T corresponding to S = (110) is the 3rd row. Hence, the error is in the 3rd bit of the received vector.

Therefore, received codeword is 101011.

The first there bit are the transmitted data bits *i.e.*, 101.

Example 9.6. For a (7, 4) block code generated by [G] below, explain how the error syndrome helps in correcting a single error.

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

Solution: We have $H = [P^T : I] = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ \underbrace{0 \quad 1 \quad 1 \quad 1}_{P_T} & & & & \underbrace{0 \quad 0 \quad 1}_{I} & & \end{bmatrix}$

Let message be 1 1 0 1

$$X = MG = [1\ 1\ 0\ 1\ 0\ 1\ 0]$$

For this X, $S = XH^T = [0\ 0\ 0]$

Let $Y = 1\ 1\ 1\ 1\ 0\ 0\ 1\ 0$ (3rd bit in error)

$$Y = X \oplus E = [1\ 1\ 0\ 1\ 0\ 1\ 0] + [0\ 0\ 1\ 0\ 0\ 0\ 0]$$

$$S = YH^T$$

$$= [1111\ 010] \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [1\ 0\ 1] = EH^T$$

i.e., S for an error in the third bit = third row of H^T.

Also, for this code a single error in the i^{th} bit of X leads to a syndrome vector *i.e.*, = i^{th} row of H^T.

Thus single error can be corrected at the receiver by comparing S with rows of H^T and correcting the i^{th} received bit if $S = i^{th}$ row of H^T. When the syndrome vector = 0, it means no error has occured.

Example 9.7. *A* (7, 4) linear block code is generated according to the following parity check matrix *H*:

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

The received codeword *Y* is 1000011 for a transmitted code word *X*. Find the corresponding data transmitted.

Solution: We have $H^T = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$

$$Y = [1\,0\,0\,0\,0\,1\,1]$$

$$S = YH^T = [1\,0\,0\,0\,0\,1\,1] \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [1\,0\,0]$$ fifth row

$$S = EH^T = [e_1\ e_2\ e_3\ e_4\ e_5\ e_6\ e_7] \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [100]$$

On solving, we get

$$e_1 + e_2 + e_3 + e_5 = 1$$
$$e_1 + e_2 + e_4 + e_6 = 0$$
$$e_1 + e_3 + e_4 + e_7 = 0$$

There will be $2^4 = 16$ possible error vectors that satisfy the above equations.

Among them all, the error vector with minimum of weight is 0 0 0 0 1 0 0 1 in the fifth column (or the fifth row in H^T indicates that this is the error vector.

Therefore, transmitted codeword is

$$X = Y \oplus E = 1\ 0\ 0\ 0\ 0\ 1\ 1$$
$$0\ 0\ 0\ 0\ 1\ 0\ 0$$
$$= 1\ 0\ 0\ 0\ 1\ 1\ 1$$

Hence, data word will be $M = 1\ 0\ 0\ 0$

Example 9.8. Given a (6, 3) linear block code with the following parity-check matrix H:

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

(*i*) Find the generator matrix G.

(*ii*) Find the code word for the data bit 101.

Solution: Given Parity-check matrix is

$$H = \left[\begin{array}{ccc|ccc} 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 \end{array}\right] = \left[\begin{array}{c|c} P & I_k \end{array}\right]$$

(*i*) Since, H is a 6×3 matrix, $n = 6$, $k = 3$, then,

The transpose of the given parity matrix is given by

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

Then, the generator matrix will be

$$G = \left[I_k \; P^T\right] = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

(*ii*) Codeword can be obtained as under:

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

or $$X = [1\ 0\ 1\ 0\ 1\ 0] \quad \text{Ans.}$$

Example 9.9. For the (7, 4) Hamming code, the parity-check matrix H is given by

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

(*i*) Construct the Generator matrix.
(*ii*) The code word that begins with 1010.
(*iii*) If the received codeword Y is 0111100, then decode this received codeword.

Solution: (*i*) Since given parity check matrix $[H]$ is 7×3 matrix, therefore, we have $n = 7$, $k = 3, q = 4$

then $$P^T = \begin{bmatrix}1 & 1 & 0\\ 0 & 1 & 1\\ 1 & 0 & 1\\ 1 & 1 & 1\end{bmatrix}$$

We can obtain the generator matrix as under:

$$G = [I \quad P^T] = \begin{bmatrix}1 & 0 & 0 & 0 & 1 & 1 & 0\\ 0 & 1 & 0 & 0 & 0 & 1 & 1\\ 0 & 0 & 1 & 0 & 1 & 0 & 1\\ 0 & 0 & 0 & 1 & 1 & 1 & 1\end{bmatrix}$$

(*ii*) Codeword can be found by the following expression:

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

or $$X = [1\ 0\ 1\ 0\ 0\ 1\ 1]$$

(*iii*) Given $$Y = [0\ 1\ 1\ 1\ 1\ 0\ 0]$$

Then $$\begin{bmatrix}1 & 1 & 0\\ 0 & 1 & 1\\ 1 & 0 & 1\end{bmatrix}$$

9.12. Cyclic Codes

Cyclic codes may be described as the subclass of linear block codes. They have the property that a cyclic shift of one code word produces another code word. As an example, let us consider an n-bit code vector as shown below:

$$X = (x_{n-1}, x_{n-2}, \cdots, x_1, x_0) \quad \ldots(9.49)$$

Here, $x_{n-1}, x_{n-2}, \ldots, x_1, x_0$ etc. represent the individual bits of the code vector 'X'. If we shift the above code word cyclically, then we will get another code vector X' *i.e.*,

$$X' = (x_{n-2}, x_{n-3}, \cdots, x_1, x_0, x_{n-1}) \quad \ldots(9.50)$$

Here, it may be observed that every bit is shifted to left by one bit position. Previously x_{n-1} was MSB but after cyclic shift it is at LSB position. One more cyclic shift provides another code vecdtor X', *i.e.*,

$$X' = (x_{n-3}, x_{n-4}, \cdots, x_1, x_0, x_{n-1}, x_{n-2}) \quad \ldots(9.51)$$

Definition of Cyclic Code

A linear code is known as cyclic code if every cyclic shift of the code vector produces some other valid code vector.

The code vector may be defined with the help of polynomial of degree $< n - 1$.

Thus, we can define code vector of equation (9.45). 1 in terms of polynomial as under:

$$X(p) = x_{n-1}\, p^{n-1} + x_{n-2}\, p^{n-2} + \ldots + x_1 p + x_0 \quad \ldots(9.52)$$

where p is an arbitrary variable and power of p represents the position of the codeword bits. i.e. $(n - 1)$ power of p represents MSB and p^0 represents LSB, p^1 represents bit at 1st position from LSB side and so on.

Now, multiplying the above equation (9.48) by p, we obtain

$$pX(p) = x_{n-1}p^n + x_{n-2}p^{n-1} + \ldots + x_1 p^2 + x_0 p \quad \ldots(9.53)$$

Let us form the polynomial of code vector gven by equation (9.46) *i.e.*,

$$X'(p) = x_{n-2}p^{n-1} + x_{n-3}p^{n-2} + \ldots + x_1 p^2 + x_0 p + x_{n-1} \quad \ldots(9.54)$$

Let us add equatons (9.49) and (9.50) by mod-2 addition rules to obtain

$$pX(p) + X'(p) = x_{n-1}p^n + (x_{n-2} \oplus x_{n-2})p^{n-1} + \ldots + (x_1 \oplus x_1)p^2 + (x_0 \oplus x_0)p + x_{n-1} \quad \ldots(9.55)$$

We know that in mod-2 addition if both bits are same, then result is zero. This means that we have

$$x_{n-2} \oplus x_{n-2} = 0$$
$$x_1 \oplus x_1 = 0$$
$$x^0 \oplus x_0 = 0 \text{ and so on.}$$

Then, the above equation (9.55) will become,

$$pX(p) \oplus X'(p) = x_{n-1}p^n + x_{n-1} = x_{n-1}(p^n + 1) \quad \ldots(9.56)$$

We know that by mod-2 addition rules, there is no difference in addition and btraction. This means that if $x \oplus y = z$ then $x = y \oplus z$ or $y = x \oplus z$. This is because od-2 addition and subtraction is the same operation.

Applying this rule to above equation (9.56) and taking $pX(p)$ on RHS, we get

$$X'(p) = pX(p) \oplus x_{n-1}(p^n + 1) \quad \ldots(9.57)$$

Note that this is an iterative equaton. It gives another code vector $X'(p)$ from e code vector $X'(p)$. That is if we know one code vector polynomial $X'(p)$, then her code vector polynomial $X'(p)$ may be obtained by using above equation.

Now, let us represent $X(p)$ as,

$$X(p) = M(p)\,G(p) \quad \ldots(9.58)$$

Here, $M(p)$ is the message signal polynominal of degree $\leq k$. This is due to the ct that there are k message bits and $G(p)$is the generating polynomial of degree q.

For an (n, k) cyclic code, the generating polynomial is expressed as,

$$G(p) = p^q + g_{q-1}p^{q-1} + \ldots + g_1 p + 1 \quad \ldots(9.59)$$

Here $q = n - k$ *i.e.*, number of parity bits.

Also, the coefficients $g_{q-1}, g_{q-2}, \ldots, g_2, g_1$ are adjusted such that $G(p)$ is the ctor of $p^n + 1$.

The message polynominal $M(p)$ is defined as,

$$M(p) = m_{k-1}p^{k-1} + m_{k-2}p^{k-2} + \ldots + m_1 p + m_0 \quad \ldots(9.60)$$

This is the message polynomial of degree less than 'k' for 'k' message bits. utting for $M(p)$ and $G(p)$ in equation (9.54), we get

$$X(p) = M(p)\,G(p)$$

We know that $k + q = n$, then above equation becomes,

$$pX(p) = m_{k-1}p^n + (m_{k-2} + m_{k-1}g_{q-1})p^{n-1} + \ldots + m_0 p \quad \ldots(9.61)$$

Substituting this value of $p\,X(p)$ in equation (9.55), we obtain,

$$X'(p) = pX(p) \oplus x_{n-1}(p^n + 1)$$

or

$$X'(p) = m_{k-1}p^n + (m_{k-2} + m_{k-1}g_{q-1})p^{n-1} + \ldots + m_0 p \oplus x_{n-1}p^n \oplus x_{n-1}$$

Now, combining the coefficients of p^n together in above equation, we have

$$X'(p) = (m_{k-1} \oplus x_{n-1})p^n + (m_{K-2} + m_{k-1}g_{q-1})p^{n-1} + \ldots + m_0 p + x_{n-1}$$

In the above equation, $G(p)$ is selected such that is the factor of p^{n+1}, and under such condition $m_{k-1} = x_{n-1}$.

Hence $m_{k-1} \oplus x_{n-1} = 0$

Then the above equation will be,

$$X'(p) = (m_{k-2} + m_{k-1}g_{q-1})p^{n-1} + \ldots + m_0 p + x_{n-1} \quad \ldots(9.62)$$

When $G(p)$ is the factor of p^{n+1}, the above equation is equivalent to equation (9.50). Equation (9.50) gives the polynomial of the code word X' after one cyclic shift of X. Since both the equation are identical, we have proved that the code word polynomial obtained by product of equation (9.54) *i.e.*,

$$X(p) = M(p)\,G(p) \quad \ldots(9.63)$$

Readly represents a valid cyclic code vector. But $G(p)$ should be a factor of $p^n + 1$. Also, both $X(p)$ and $X'(p)$ satisfy the iterative equation for multiple shifts given by equation (9.53) as,

$$X'(p) = p\,X(p) \oplus x_{n-1}(p^{n+1}) \quad \ldots(9.64)$$

The other code vectors may be obtained from equation (9.58) as,

$$\begin{aligned} X_1(p) &= M_1(p)\,G(p) \\ X_2(p) &= M_2(p)\,G(p) \\ X_3(p) &= M_3(p)\,G(p) \end{aligned} \quad \ldots(9.65)$$

and so on.

Here, all the code vectors x_2, x_3 ... and so on satisfy the cyclic shift condition and their polynomials also satisfy equation (9.61). It may be noted that generator polynomial remains same for all code vectors.

The code vectors obtained in equation (9.61) above are not in systematic form. The systematic cyclic code vectors can be obtained using the following procedure.

The n-bit code vector 'X' and its polynomial $X(p)$ are given by equation, (9.45) and (9.48) respectively as,

$$X = (x_{n-1}x_{n-2}x_{n-3}\cdots x_3x_2x_1x_0) \quad \ldots(9.66)$$

and

$$X(p) = x_{n-1}p^{n-1} + x_{n-2}p^{n-2} + x_{n-3}p^{n-3} + \ldots + x_1 p + x_0 \quad \ldots(9.66a)$$

If we want to represent code vector 'X' to be in systematic form, then it will be,

The above code vector may be written in polynomial form as under:

$$X(p) = m_{k-1}p^{n-1} + m_{k-2}p^{n-2} + \ldots + m_1 p^{n-k+1} + m_0 p^{n-k} + c_{q-1}p^{n-k-1} + c_{q-2}p^{n-k-2} + \ldots + c_1 p + c_0$$

We know that $n - k = q$ or $n = k + q$. Putting those values of 'n' and '$n - k$' in above equation, we obtain,

$$X(p) = m_{k-1}p^{k+q-1} + m_{k-2}p^{k+q-2} + \ldots + m_1 p^{q+1} + m_0 p^q + c_{q-1}p^{q-1} + c_{q-2}p^{q-2} + \ldots + c_1 p + c_0$$

Let us rearrange the above equation as under,

$$X(p) = p^q \left[m_{k-1}p^{k-1} + m_{k-2}p^{k-2} + \ldots + m_1 p + m_0\right] + c_{q-1}p^{q-1} + c_{q-2}p^{q-2} + \ldots + c_1 p + c_0 \quad \ldots(9.68)$$

In the above equation, $m_{k-1}p^{k-1} + m_{k-2}p^{k-2} + \ldots + m_1 p + m_0 = M(p)$

As we have defined in equation (9.56) earlier. Therefore above equation becomes,

$$X(p) = p^q M(p) + (c_{q-1}c_{q-2}\ldots c_1 c_0) + p^{q-1} + \ldots + c_1 p + c_0 \quad \ldots(9.69)$$

Let us define the check bit polynomial of check bits

$C = (c_{q-1}c_{q-2}\ldots c_1 c_0)$ as

$$C(p) = c_{q-1}p^{q-1} + c_{q-2}p^{q-2} + \ldots + c_1 p + c_0 \quad \ldots(9.70)$$

The above equation is check bit polynomial of degree less than q. From equations (9.69) and (9.68), we get

$$X(p) = p^q M(p) + C(p) \quad \ldots(9.71)$$

This equation gives a code word polynomial in systematic form. For this code vector to be cyclic, the above equation must be same as equation (9.59). Thus, for a cyclic code vector, we can equate equations (9.71) and (9.61) *i.e.*

$$p^q M(p) + C(p) = M(p)G(p)$$

$$\frac{p^q M(p)}{G(p)} \oplus \frac{C(p)}{G(p)} = M(p)$$

The above equation has the form of $z \oplus t = l$. We know that mod-2 addition and subtraction operation is same *i.e.*, $z \oplus t = l$, then we can write $z = t \oplus l$ or $t = z \oplus l$ or $z \oplus t \oplus l = 0$. Thus, there is no mod-2 subtraction as such. Mod-2 addition and subtraction yield same result. With these conclusions, we can write above equation as under:

$$\frac{p^q M(p)}{G(p)} = M(p) \oplus \frac{C(p)}{G(p)}$$

Hence, the check bit polynominal is obtained as remainder in dividing $p^q\ M(p)$ by $G(p)$ *i.e.*,

$$C(p) = rem\left[\frac{p^q\ M(p)}{G(p)}\right] \qquad ...(9.72)$$

Here, $C(p)$ check bit polynomial for systematic code.
$M(p)$ message bit polynomial,
and $G(p)$ generating polynomial, which is the factor of p^{n+1}.

Example 9.11. The generator polynomal of a (7, 4) cyclic code is $G(p) = p^3 + p + 1$. Obtain all the code vectors for the code in non-systematic and systematic form. *(U.P. Tech., Sem. Examination, 2003-2004)*

Solution: Here $n = 7$ and $k = 4$ therefore $q = n - k = 3$.

There will be total $2^k = 2^4 = 16$ message vectors of 4 bits each. Let us consider any message vector as under:

$$M = (m_3\ m_2\ m_1\ m_0)$$

or $$M = (0\ 1\ 0\ 1)$$

Then, the message polynomial will be (by substituting $k = 4$ in equation (9.56),

$$M(p) = m_3p^3 + m_2p^2 + m_1p + m_0$$

$$\therefore \quad M(p) = p^2 + 1 \qquad ...(i)$$

Also, the given generator polynomial is,

$$G(p) = p^3 + p + 1 \qquad ...(ii)$$

To Obtain Non Systematic Code Vectors

The non-systematic cyclic code is given by (using equation (9.59)),

$$X(p) = M(p)\ G(p)$$

or $$X(p) = \left(p^2 + 1\right)\left(p^3 + p + 1\right).$$

or $$X(p) = p^5 + p^3 + p^2 + p^3 + p + 1$$

or $$X(p) = p^5 + p^3 + p^3 + p^2 + p + 1$$

or $$X(p) = p^5 + (1 \oplus 1)\ p^3 + p^2 + p + 1$$

or $$X(p) = p^5 + p^2 + p + 1 \qquad [\text{Since } (1 \oplus 1)\ p^3 = op^3 = 0)$$

(Since $(1 \oplus |\ \ |^3 = 0P^3 = 0)$

$$= 0p^6 + p^5 + 0p^4 + 0p^3 + p^2 + p + 1 \qquad ...(iii)$$

Note that the degree of above polynomial is $n - 1 = 6$. The code vector corresponding to above polynomial is,

$$C(p) = rem\left[\frac{p^q\, M(p)}{G(p)}\right]$$

Since, $q = 3$, $p^q\, M(p)$ will be,

$$p^q\, M(p) = p^3\, M(p)$$

or $$p^q\, M(p) = p^3(p^2 + 1)$$

or $$p^q\, M(p) = p^5 + q^3 \quad \text{(for message vector of 0101)}$$

or $$p^q\, M(p) = p^5 + 0p^4 + p^3 + 0p^2 + 0p + 0$$

and $$G(p) = p^3 + p + 1$$

or $$G(p) = p^3 + 0p^2 + p + 1$$

Now, we have $p^q\, M(p)$ and $G(p)$. Now let us perform the division to find remainder of this division as under:

$$\begin{array}{r} p^2 + 0 + 0 \quad \leftarrow \text{Quotient} \\ p^3 + 0p^2 + p + 1 \;\overline{)\; p^5 + 0p^4 + p^3 + 0p^2 + 0p + 0} \\ p^5 + 0p^4 + p^3 + p^2 \\ \hline \oplus \quad \oplus \quad \oplus \quad \oplus \\ 0 \quad +0 \quad +0 \quad + p^2 \quad +0p + 0 \end{array}$$

Mod-2 addition

Remainder

Hence, the remainder polynomial is $p^2 + 0p + 0$ in the division of $p^3\, M(p)$ by $G(p)$. Therefore, equation (v) may be written as,

$$C(p) = rem\left[\frac{p^3\, M(p)}{G(p)}\right]$$

or $$C(p) = p^2 + 0p + 0$$

With $q = 3$, the polynomial $C(p)$ will be (using equation (9.65))

$$C(p) = c_2\, p^2 + c_1 p + c_0$$

Thus $$c_2\, p^2 + c_1 p + c_0 = p^2 + 0p + 0$$

Hence, the check bits are

$$C = (c_2\, c_1\, c_0) = (1\ 0\ 0)$$

The code vector is written in systematic form as under: (using equation (9.62(a))

$$X = \left(m_{k-1} m_{k-2} \ldots m_1\, m_0 : c_{q-1} c_{q-2} \ldots c_1 c_0\right)$$

Table 9.10. Code vectors of a (7, 4) cyclic code for $G(p) = p^3 + p + 1$

S. No.	Message bits $M =$ m_3	m_2	m_1	m_0	Nonsystematic code vectors $X =$ x_6	x_5	x_4	x_3	x_2	x_1	x_0	Systematic code vectors $X =$ m_3	m_2	m_1	m_0	c_2	c_1	c_0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	1	0	0	0	1	0	1	1	0	0	0	1	0	1	1
3	0	0	1	0	0	0	1	0	1	1	0	0	0	1	1	1	0	1
4	0	0	1	1	0	0	1	1	1	0	1	0	0	1	1	1	0	1
5	0	1	0	0	0	1	0	1	1	0	0	0	1	0	0	1	1	1
6	0	1	0	1	0	1	0	0	1	1	1	0	1	0	1	1	0	0
7	0	1	1	0	0	1	1	1	0	1	0	0	1	1	0	0	0	1
8	0	1	1	1	0	1	1	0	0	0	1	0	1	1	1	0	1	0
9	1	0	0	0	1	0	1	1	0	0	0	1	0	0	0	1	0	1
10	1	0	0	1	1	0	1	1	0	0	0	1	0	0	0	1	0	1
11	1	0	1	0	1	0	0	1	1	1	0	1	0	1	0	0	1	1
12	1	0	1	1	1	0	0	0	1	0	1	1	0	1	1	0	0	0
13	1	1	0	0	1	1	1	0	1	0	0	1	1	0	0	0	1	0
14	1	1	0	1	1	1	1	1	1	1	1	1	1	0	1	0	0	1
15	1	1	1	0	1	1	0	0	0	1	0	1	1	1	0	1	0	0
16	1	1	1	1	1	1	0	1	0	0	1	1	1	1	1	1	1	1

9.13. Generator and Parity Check Matrices of Cyclic Codes

Because cyclic codes are subclass of linear block codes, generater and parity check matrices can also be defined for cyclic codes. The generator matrix has the size of $k \times n$. This means that there are 'k' rows and 'n' columns. Let the generator matrix $G(p)$ be given by equation (9.55) as,

$$G(p) = p^q + g_{q-1}p^{q-1} + \ldots + g_1 p + 1 \qquad \ldots(9.73)$$

Multiplying both the sides of this polynomial by p^i, we get

$$p^i G(p) = p^{i+q} + g_{q-1} p^{i+q-1} + \ldots + g_1 p^{i+1} + p^i \qquad \ldots(9.74)$$

and $(k-1)$, $(k-2)$, ..., 2,1.0.

The above equation gives the polynomials for the rows of a generating polynomial. This procedure will be clear after the discussion of next exmaple.

Example 9.12. Find the generator matrix corresponding to $G(p) = p^3 + p^2 + 1$ for a (7, 4) cyclic code.

Solution: Here $n = 7$, $k = 4$ and $q = 7 - 4 = 3$

For row 1: $i = 3 \Rightarrow$
For row 2: $i = 2 \Rightarrow$
For row 3: $i = 1 \Rightarrow$
For row 4: $i = 0 \Rightarrow$

$$\left.\begin{aligned} p^3\,G(p) &= p^6 + p^5 + p^3 \\ p^2\,G(p) &= p^5 + p^4 + p^2 \\ pG(p) &= p^4 + p^3 + p \\ G(p) &= p^3 + p^2 + 1 \end{aligned}\right\} \quad \ldots(i)$$

The generator matrix for (n, k) code is of size $k \times n$. For this (7, 4) cyclic code, the size will be 4×7. Corresponding to four rows, we have obtained four polynomials given by equation (i). Let us write each polynomial in the following manner:

Row 1 $\quad p^3\,G(p) = p^6 + p^5 + 0p^4 + p^3 + 0p^2 + 0\,p + 0$
Row 2 $\quad p^2\,G(p) = 0p^6 + p^5 + p^4 + 0p^3 + p^2 + 0p + 0$
Row 3 $\quad pG(p) = 0p^6 + 0p^5 + p^4 + p^3 + 0p^2 + p + 0$
Row 4 $\quad G(p) = 0p^6 + 0p^5 + 0p^4 + p^3 + p^2 + 0p + 1$ $\quad \ldots(ii)$

Let us transform the above set of polynomials into a matrix of 4×7

$$G_{4\times 7} = \begin{matrix} p^6 & p^5 & p^4 & p^3 & p^2 & p^1 & p^0 \end{matrix}$$
$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}_{4\times 7} \quad \ldots(iii)$$

This is the generator matrix for given generator polynomials. **Ans.**

Example 9.13. Find out the possible generator polynomials (7, 4) cyclic code. Obtain the code vectors corresponding to these generator polynomials.

Solution: For this (7, 4) cyclic code, we have

$$n = 7, \quad k = 4 \quad \text{and} \quad q = n - k = 7 - 4 = 3$$

We know that the generator polynomial is the factor of $p^n + 1$. For this example, generator polynomial is the factor of $p^7 + 1$. The factors of $p^7 + 1$ are

$$p^7 + 1 = (p \oplus 1)\,(p^3 \oplus p^2 \oplus 1)\,(p^3 \oplus p \oplus 1) \quad \ldots(i)$$

The valid generating polynomial is given by equation (9.55) *i.e.*,

$$G(p) = p^q + g_{q-1}\,p^{q-1} + \ldots + g_1\,p + 1$$

Hence, the degree of the generating polynomial should be q. For this example $q = 3$. Therefore the valid generator polynomials for $p^7 + 1$ will be $p^3 + p^2 + 1$ and $p^3 + p + 1$. Also, $p + 1$ will not be a generator polynomial. Since its degree is not q (*i.e.*, 3). Thus generator polynomials for (7, 4) cyclic code are,

$$G_1(p) = p^3 + p^2 + 1$$

and $\quad G_2(p) = p^3 + p + 1$ **Ans.**

Example 9.14. Obtain the generator matrix corresponding to $G(p) = p^3 + p + 1$ and find out the code vectors for (7, 4) cyclic code.

Solution: The row of a generator matrix are given by $p^i G(p)$.

Here, $\quad p^i\,G(p) = p^{i+3} + p^{i+1} + p^i$

The above set of polynomials is transformed into a generator matrix of size 4 × 7 (*i.e.*, $k \times n$) as shown below.

$$[G] = \begin{array}{c} \\ \text{Row 1} \\ \text{Row 2} \\ \text{Row 3} \\ \text{Row 4} \end{array} \begin{array}{c} \begin{array}{ccccccc} p^6 & p^5 & p^4 & p^3 & p^2 & p^1 & p^0 \end{array} \\ \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}_{4 \times 7} \end{array}$$

Since, cyclic is a subclass of linear block code, its code vectors can be obtained by

$$X = MG \qquad \ldots(ii)$$

Here, M is the $1 \times k$ message vector and G is generator matrix.

Here, $k = 4$. Let us consider any 4-bits message vector as under:

$$M = (m_3\ m_2\ m_1\ m_0) = (1\ 0\ 0\ 1)$$

The code vector corresponding to this message vector will be,

$$X = MG = [1001] \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix} = (1\ 0\ 1\ 0\ 0\ 1\ 1)$$

Note: *Here, we perform matrix multiplication and additions by mod-2 rules. i.e.,*

$$1 \oplus 1 = 0,$$
$$1 \oplus 0 = 1$$
$$0 \oplus 1 = 1$$

and $0 \oplus 0 = 0.$

This code vector we have already obtained in example 9.5, and is listed in Table 9.9. This code vector is in non-systematic form. Also note that generator matrix is also in non-systematic form. In a same way other code vectors for cyclic code may be obtained.

Systematic form of Generator Matrix

The systematic form of generator matrix is expressed as,

$$G = \left[I_k : P_{k \times q}\right]_{k \times n} \qquad \ldots(iii)$$

The t^{th} row of this matrix will be represented in the polynomial form as,

$$t^{th} \text{ row of } G = p^{n-t} + R_t(p)$$

where $t = 1, 2, 3 \ldots k$ $\qquad \ldots(iv)$

Let us divide p^{n-t} by a generator matrix $G(p)$. Then, we can express the result of this division in terms of quotient and remainder us under:

$$\frac{p^{n-t}}{G(p)} = \text{Quotient} + \frac{\text{Remainder}}{G(p)} \qquad \ldots(v)$$

$$\frac{p^{n-t}}{G(p)} = Q_1(p) + \frac{R_t(p)}{G(p)} \qquad ...(vi)$$

i.e., $$p^{n-t} = Q_t(p)G(p) \oplus R_t(p)$$

and $$t = 1, 2, \ldots k \qquad ...(vii)$$

Again, we know that if $z = y \oplus t$, then $z \oplus y = t$ or $z \oplus t = y$. This means that the mod-2 addition and subtraction yield same results. Then we can write equation (*vii*) as under:

$$p^{n-t} \oplus R_t(p) = Q_t(p)\ G(p) \qquad ...(viii)$$

Thus, the above equation represents t^{th} row of systematic generator matrix.

Ans.

Example 9.15. Obtain the generator matrix for a systematic (7, 4) cyclic code if $G(p) = p^3 + p + 1$. Also, obtain the partiy check matrix.

Solution: The t^{th} row of generator matrix is given as,

$$p^{n-t} + R_t(p) = Q_t(p)\ G(p) \quad \text{and} \quad t = 1, 2, \ldots k$$

We are given that $n = 7$, $k = 4$ and $q = n - k = 3$.

The above equation will be,

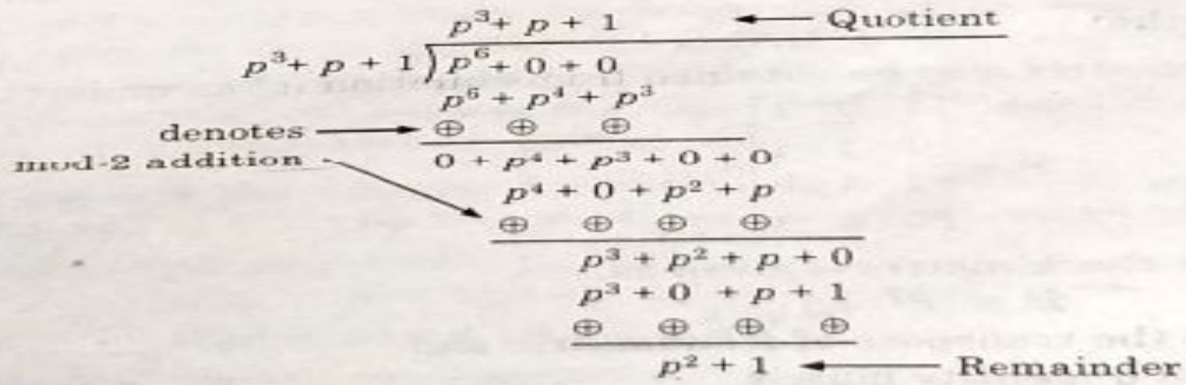
$$p^{7-t} + R_t(p) = Q_t(p)\ (p^3 + p + 1) \qquad ...(i)$$

and $$t = 1, 2, 3, 4$$

With $t = 1$, the above equation becomes,

$$p^6 + R_1(p) = Q_1(p)\ (p^3 + p + 1) \qquad ...(ii)$$

The RHS or LHS of this equation represents 1^{st} row of systematic generator matrix. We have to find $Q_1(p)$. From last example, we know that $Q_1(p)$ is obtained by dividing p^{n-t} by $G(p)$. Here to obtain $Q_1(p)$, we have to divide p^6 by $G(p) = p^3 + p + 1$.



Here $Q_1(p) = p^3 + p + 1$

and $R_1(p) = p^2 + 1$

9.16. Convolutional Codes

A convolutional coding is done by combining the fixed number of input bits. The input bits are stored in the fixed length shift register and they are combined with the help of mod-2 adders. This operation is equivalent to binary convolution and hence it is called **convolutional coding**. This concept is illustrated with the help of simple example given below.

Figure 9.12 illustrates a convolutional encoder.

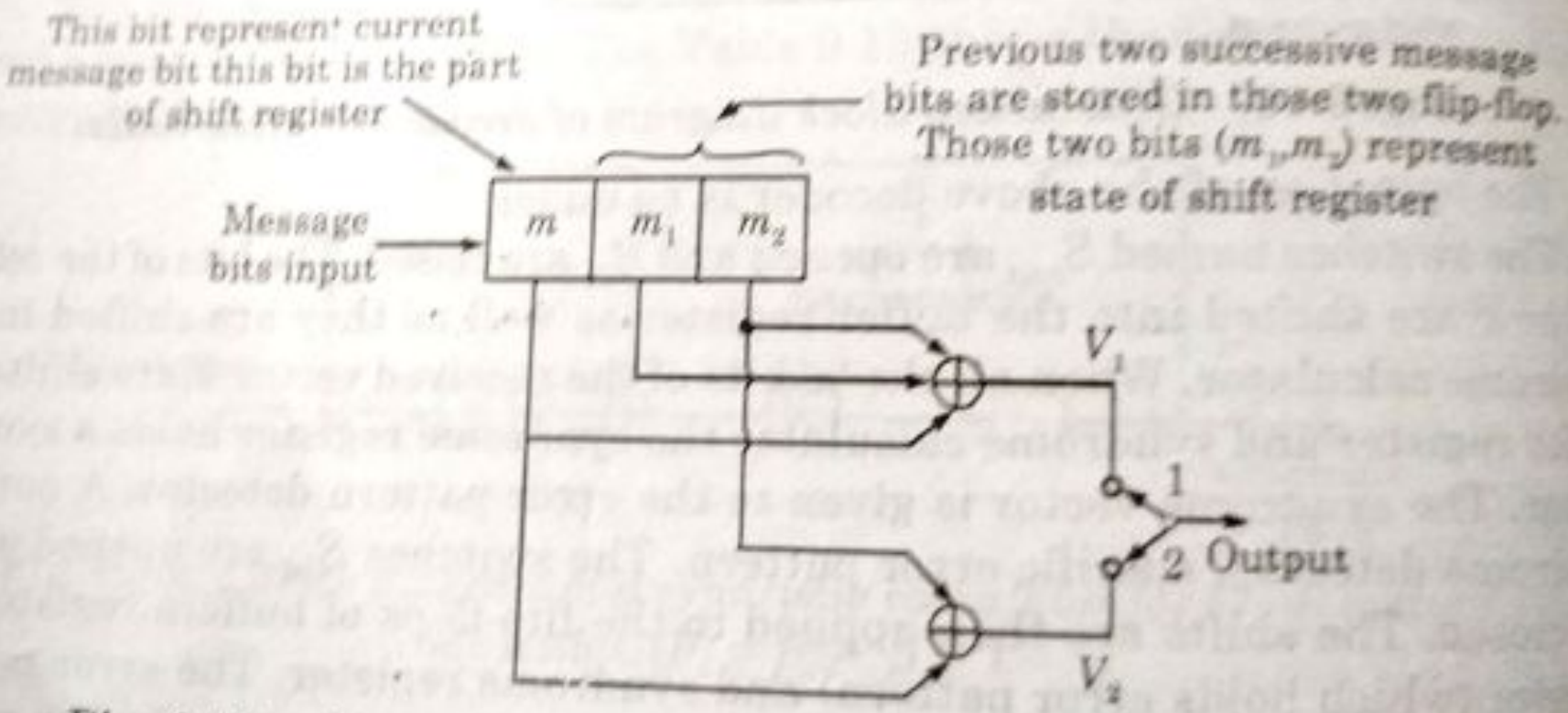


Fig. 9.12. *Convolutional encoder with $K = 3$, $k = 1$ and $n = 2$*

This convolutional encoder operates as under:

Operation

Whenever the message bit is shifted to position 'm', the new values of v_1 and v_2 are generated depending upon m, m_1 and m_2. m_1 and m_2 store the previous two message bits. The current bit is present in m. Hence, we can write,

$$v_1 = m \oplus m_1 \oplus m_2 \quad \ldots(9.85)$$

and

$$v_2 = m \oplus m_2 \quad \ldots(9.86)$$

The output switch first samples v_1 and then v_2. The shift register then shifts contents of m_1 to m_2 and contents of m to m_1. Next input bit is then taken and stored in m. Again v_1 and v_2 are generated according to this new combination of m, m_1 and m_2. Hence, output switch then samples v_1 then v_2. Hence, the output bit strem for successive input bits will be,

$$X = v_1\, v_2\, v_1\, v_2\, v_1\, v_2 \ldots \text{ and so on} \quad \ldots(9.87)$$

Here it may be noted that for every input message bit two encoded output bits v_1 and v_2 are transmitted. In other words, for a single message bit, the encoded code word is two bits *i.e.*, for this convolutional encoder,

Number of message bits, $k = 1$

Number of encoded output bits for on message bit, $n = 2$

Therefore the code rate of this encoder is,

$$r = \frac{k}{n} = \frac{1}{2} \quad \ldots(9.88)$$

In the encoder of figure 9.12, observe that whenever a particular message bit enters a shift register, it remains in the shift register for three shifts *i.e.*,

First shift	→	Message bit is entered in position 'm'.
Second shift	→	Message bit is shifted in position m_1.
Third shift	→	Message bit is shifted in position m_2.

And at the forth shift the message bit is discarded or simply lost by overwriting. We know that v_1 and v_2 are combinations of m, m_1, m_2. Since a single message bit remains in m during first shift, in m_1 during second shift and in m_2 during second shift; it influences output v_1 and v_2 for 'there' successive shifts.

Constraint Length (K):

The constraint length of a convolution code is defined as the number of shifts over which a single message bit can influence the encoder output. It is expressed in terms of message bits.

For the encoder of figure 9.13 constraint length $K = 3$ bits. This is because in this encoder, a single message bit influences encoder output for three successive shifts. At the fourth shift, the message bit is lost and it has no effect on the output.

9.16.1. Code Tree, Trellis and State Diagram for a Convolution Encoder

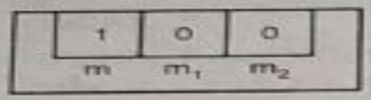
Now let us discuss the operation study the operation of the convolutional encoder with the help of code tree, trellis and state diagram. For this our illustration, we will consider the convolutional encoder of Fig. 9.12 as an example

Let the initial values of bits stored in m_1 and m_2 be zero. That is $m_1\ m_2 = 00$ initially and the encoder is in state 'a'.

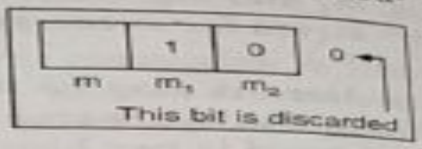
Table 9.14. States of the encoder of Figure 9.13.

m_2	m_1	*State of encoder*
0	0	a
0	1	b
1	0	c
1	1	d

Let the first message input be $m = 1$. With this input v_1 and v_2 will be calculated follows.



$$V_1 = 1 \oplus 0 \oplus 0 = 1$$
$$v_2 = 1 \oplus 0 = 1$$



The values of $v_1\ v_2 = 11$ are transmitted to the output and register contents are shifted to right by one bit position as shown.

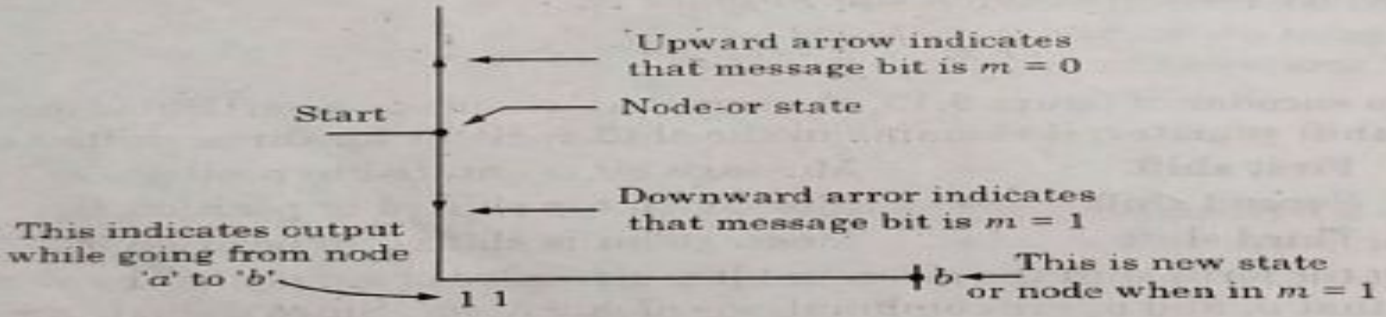


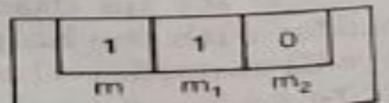
Fig. 9.13. *Code tree from node 'a' to 'b'.*

Hence, the new state of encoder is $m_2\ m_1 = 01$ or 'b' and output transmitted are $v_1\ v_2 = 11$. This shows that if encoder is in state 'a' and if input is $m = 1$ then the next state is 'b' and outputs are $v_1\ v_2 = 11$. The first row of Table 9.14 illustrates this operation.

The last column of this table shows the code free diagram. The code free diagram starts at node or state 'a'. This diagram is reproduced as shown in figure 9.13.

It may be observed that if $m = 1$, we go downward from node 'a'. Otherwise if $m = 0$. We go upward from node 'a'. It can be verifed that if $m = 0$ then next node (state) is 'a' only. Since $m = 1$ here we go downwards toward node b and output is 11 in this node (or state).

Now let the second message bit be 1. The contents of shift register with this input will be as shown below.



$$v_1 = 1 \oplus 1 \oplus 0 = 0$$