Chapter 7

CHANNEL DECODING: ERROR DETECTION AND ERROR CORRECTION.

7.1 INTRODUCTION

This chapter deals with channel decoding, which involves error detection and correction of the received n bit code vectors transmitted from the channel encoders for the messages of Type I and Type II discrete information sources. The results presented in this chapter are the outcomes from Module 4 of the research work.

Two novel schemes are proposed for single bit error detection and correction of systematic linear block codes. The proposed methods in this chapter offer a more efficient way of detection and correction of single bit errors of a linear block code. The schemes involve the use of only Ex-OR operations and no AND operations. The error pattern present in the received n bit code can be used to generate a syndrome. For this syndrome the lookup table permanently created gives an error compensating vector which when Mod-2 added to the received vector gives corrected vector.

Though the single bit error correction Hamming codes are simple to generate, their decoding is much involved. The conventional scheme involving multiplication of parity check matrix $H^T$ with incoming code to find error syndrome $S$ is much involved which needs to do several multiplications and Modulo-2 additions. The schemes proposed here do
not involve any logical AND operations and the total number of logical operations also is less. The hardware required to implement the proposed algorithms also is less complex when compared with hardware realization of syndrome calculation in the conventional method. The basic principle and ideas of the algorithms introduced in this chapter are presented in the paper “Application of Code Lengths of Golden Sequence Terms in Error Correction of Linear Block Codes” in the National Conference NCCCIT-2010.

7.2 ORGANIZATION OF THE CHAPTER

The chapter is organized as follows:

In section 7.3, the notation adopted in this chapter is presented. The algorithm for construction of lookup table for the first proposed scheme is presented in section 7.4. In section 7.5, the algorithm for construction of lookup table for the second proposed scheme is presented. The step by step procedure for error detection and error correction by finding syndrome for the received code using both the schemes is presented in section 7.6. In section 7.7, the justification for the proposed algorithms is presented. Few examples for construction of lookup table and error detection and correction are presented in section 7.8. In section 7.9, the flow chart used for the simulation of the proposed algorithm using MATLAB 7.0 is presented. The results of the simulation of the proposed Algorithm 2 using MATLAB 7.0 is presented in
section 7.10. In section 7.11, the results and discussion are presented. The conclusions are presented in section 7.12.

7.3 NOTATION

(n, d) Linear block code:

d : Number of data bits.

n : The code length for any data and with (n-d) number of parity check bits.

Ci : Valid code vectors C₁, C₂, ....without errors. i = 1, 2, 3, ....2^d.

Always C₁ is the all zero n-bit valid code.

Cᵢⱼ : Single bit error vectors of any n-bit code Cᵢ¹, Cᵢ², Cᵢ³.....Cᵢⁿ.

j=1,2,3,.....n.

J, K : These are two numbers chosen such that J+K is a Fibonacci Number or code length of any Golden sequence number like 2, 3, 5, 8....etc.

Cₜ : Transmitted code vector. This is one of Cᵢ.

Cᵣ : The received code vector.

Cᵣᵢ : This is a code of n bits with the iᵗʰ bit same as the iᵗʰ bit of Cᵣ and all the remaining bits zero.

Cᵣₑ : The received code after single bit error correction.

Rₘ : For the first algorithm Rₘ is the n-tuple with all zeros in the d MSBs followed by the Golden Sequence term of length (n-d) bits.

For the second algorithm Rₘ is the n-tuple with 1’s in (J-1)ᵗʰ and (K-1)ᵗʰ position and 0’s in all remaining positions.
R : This is the code obtained by Mod-2 addition of C_r with R_m.

\[ R = C_r \oplus R_m = \text{The d-tuple } D_m \text{ followed by } (n-d) \text{ tuple } P_m \text{ which can be denoted as } D_m | P_m. \]

D_m : The d-tuple formed with d MSBs of R.

P_m : The (n-d) tuple formed with (n-d) LSBs of R.

R_{Ref} : Code word for the d-bit data D_m which can be found using the Generator matrix G. This is an n-tuple of the form \( R_{ref} = D_m | P_r. \)

D_m is a d-tuple followed by P_r which is a (n-d) tuple.

S : Syndrome \( S = P_m \oplus P_r. \)

S_{Di} : This is the syndrome for C_{ri}, \( i = 1, 2, \ldots, d. \)

S_D : This is the Modulo-2 addition of d number of syndromes for \( C_{r1}, C_{r2}, C_{r3}, \ldots, C_{rd}. \)

\[ S_D = S_{D1} \oplus S_{D2} \oplus S_{D3} \oplus S_{D4} \ldots \oplus S_{Dd}; \quad S_D = P_m. \]

S_{pi} : This is the syndrome for \( C_{r(d+i)}, \quad i = 1, 2, \ldots, (n-d). \)

S_p : This is the Modulo-2 addition of (n-d) number of syndromes.

\[ S_p = S_{p1} \oplus S_{p2} \oplus S_{p3} \oplus S_{p4} \ldots \oplus S_{p(n-d)}; \quad S_p = P_r. \]

S_c : This is the net syndrome for C_r.

This is same as S in the conventional method. \( S_c = S_{cd} \oplus S_{cp}. \)

S_R : This is the net syndrome for R_m. \( S_R = S_{Rd} \oplus S_{Rp}. \)

C_{ec} : The error compensating code which when Modulo-2 added to C_r gives error corrected code. These are the n-tuples stored in the lookup table as contents of the memory location with the (n-d) bit syndrome as its address.
7.4 CONSTRUCTION OF LOOKUP TABLE FOR ALGORITHM 1

In this algorithm the syndrome for no error code will be a Golden sequence term of length n-d bits. Typically it is 101, 1011, 10110……etc, the third (n - d = 3) , the fourth (n - d = 4) and the fifth (n - d = 5) golden sequence terms respectively.

The Lookup Table is constructed which gives an n bit error compensating code C_{ec} for each syndrome.

The syndrome for no error case i.e., the syndrome if C_r has no error is a binary Rabbit string or a Golden sequence term of length (n-d) bits. The Golden sequence terms are 0, 1, 10, 101, 10110, ..... etc.

PROCEDURE FOR CONSTRUCTION OF LOOKUP TABLE

The all zero vector of n bits, C_1 = 00000...0 is considered.

R is formed as, \( R = C_1 \oplus R_m \)

Where, \( R_m = n \) tuple with 0s in d MSBs followed by (n - d) bit Fibonacci Golden sequence term.

So, \( R = C_1 \oplus R_m = D_m|P_m \)

\( R_{ref} \) is found as the Code vector for the data \( D_m \). \( R_{ref} = D_m|P_r \)

The syndrome \( S \) is found as, \( S = P_m \oplus P_r \). This is the syndrome for no error. This contains (n-d) bits. In the lookup table the entry
against this syndrome is the all zero vector of length n. This procedure is repeated for \( C_1^i \), all zero vector with errors in positions 1,2,3...i...n. The respective entry in the lookup table for each of these syndromes is \( C_1^i \). This gives in all \((n+1)\) number of syndromes each of size \((n-k)\) bits and hence \((n+1)\) number of error compensating vectors each of \(n\)-bit length.

7.5 CONSTRUCTION OF LOOKUP TABLE FOR ALGORITHM 2

In this method a Fibonacci number \( F \) is chosen. The values of \( J \) and \( K \) are chosen such that \( J+k = F \). The value of \( R_m \) is the \(n\)-tuple with 1s in \((J-1)^{th}\) and \((K-1)^{th}\) positions and 0s in all other positions. But for this change in the form of \( R_m \), the rest of the procedure remains the same as in the first algorithm.

PROCEDURE FOR CONSTRUCTION OF LOOKUP TABLE

The all zero vector of \(n\) bits, \( C_1 = 00000...0 \) is considered.

\( R_m = n \) tuple with 1s in \((J-1)^{th}\) and \((K-1)^{th}\) positions and 0s in all other positions.

\( R \) is found as, \( R = C_1 \oplus R_m \)

So, \( R = C_1 \oplus R_m = D_m \| P_m \)

\( R_{ref} \) is found as \( R_{ref} = \text{Code vector for the data } D_m \). \( R_{ref} = D_m \| P_r \)

The syndrome \( S \) is found as \( S = P_m \oplus P_r \). This is the syndrome for the received code with no error. This contains \((n-d)\) bits. In the lookup table the entry against this syndrome is the all zero vector of length \(n\).

This procedure is repeated for \( C_1^i \), all zero vector with errors in positions 1,2,3...i...n. The respective entry in the lookup table for each of
these syndromes is \( C_i \). This gives in all \((n+1)\) number of syndromes each of size \((n-d)\) bits.

**Special case**: If \( J = K = 1 \), the syndromes obtained from the above procedure will be similar to the rows of the matrix \( H^T \). Each syndrome is given by \( S = C_r \cdot H^T \), Where \( C_r \) is the received vector and \( H^T \) is transpose of the Parity Check Matrix, \( H \). It can be verified that the Mod-2 addition of syndrome for zero error case obtained from the proposed method with the syndrome for the error in \( i^{th} \) bit gives the \( i^{th} \) row of \( H^T \) of the conventional method, which is the syndrome for error in the \( i^{th} \) bit.

**CHOICE OF J AND K AND THE PROCEDURE TO FIND R FOR ANY RECEIVED CODE \( C_r \)**

In the second proposed method of this chapter, a Fibonacci number \( F_n \) is chosen which is sum of two numbers \( J \) and \( K \).

\[
i.e., \ J + K = F_n \quad \ldots \ldots \ldots \ldots (7.1)
\]

The values of \( J \) and \( K \) are chosen such that \( J + K \) is a predetermined Fibonacci number. The possible values of \( J \) and \( K \) for few different values of \( F_n \) are listed in the following table.

**Table 7.1 : \( F_n \) VERSUS \( J \) AND \( K \)**

<table>
<thead>
<tr>
<th>( F_n = J + K )</th>
<th>( (J, K) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>(1,2), (2,1)</td>
</tr>
<tr>
<td>5</td>
<td>(1,4), (2,3), (3,2), (4,1)</td>
</tr>
<tr>
<td>8</td>
<td>(1,7), (2,6), (3,5), (5,3), (6,2), (7,1)</td>
</tr>
<tr>
<td>13</td>
<td>(1,12), (2,11), (3,10), (4, 9), (5,8), (6,7), (7,6), (8,5), (9, 4), (10,3), (11, 2), (12,1)</td>
</tr>
<tr>
<td>21</td>
<td>(1,20), (2,19), (3,18), (4,17), (5,16), (6,15), (7,14), (8,13), (9,12), (10,11), (11,10), (12,9), (13,8), (14,7), (15,6), (16,5), (17,4), (18,3), (19,2), (20,1)</td>
</tr>
</tbody>
</table>
Procedure to find R:
The (J-1)th and (K-1)th bits of Cr are inverted to obtain R. If J = 1, then, only (K-1)th bit is inverted. If K = 1, then, only (J-1)th bit is inverted. This is same as Mod-2 addition of Cr with Rm, where Rm is the n-tuple with 1s in the (J-1)th and (K-1)th positions and with zeros in all remaining positions.

7.6 STEP BY STEP PROCEDURE FOR DETECTION AND CORRECTION OF SINGLE BIT ERRORS USING LOOKUP TABLE

1. For the received vector Cr,

   \[ R = C_r \oplus R_m = D_m \mid P_m \]  \hspace{1cm} (7.2)

2. Rref is found as Rref = The valid code for data Dm.

   This is in the form \( D_m \mid P_r \) \hspace{1cm} (7.3)

3. The syndrome is found as \( S = P_m \oplus P_r \) \hspace{1cm} (7.4)

4. From the Lookup Table the error compensating vector Cec is identified for the computed syndrome.

5. The vector after error correction \( C_c = C_r \oplus C_{ec} \) \hspace{1cm} (7.5)

7.7 JUSTIFICATION

a) Interpretation of calculation of syndrome for a received code Cr:

1. The received code Cr is the linear Mod-2 addition of n number of n-tuples.

   \[ C_r = C_{r1} \oplus C_{r2} \oplus C_{r3} \oplus \ldots \oplus C_{rn} \] \hspace{1cm} (7.6)

   Each of these n-tuples is of weight 1 or 0. The components of Cr with weight 1 may be viewed as single bit error vectors of the all zero error free code.
Example:  \( n = 6 \)
\[
C_r = 101 101
\]
\[= 100000 \oplus 000000 \oplus 001000 \oplus 000100 \oplus 000000 \oplus 000001\]

2. The Syndrome \( S \) for \( C_r \) is given by,
\[
S = C_r \cdot H^T = (C_{r1} \oplus C_{r2} \oplus C_{r3} \oplus \ldots \oplus C_{rn}).H^T
\]
\[= C_{r1} \cdot H^T \oplus C_{r2} \cdot H^T \oplus C_{r3} \cdot H^T \oplus \ldots \oplus C_{rn} \cdot H^T \ldots (7.7)\]
\[
S = S_D \oplus S_P \ldots \ldots \ldots (7.8)
\]
where \( S_D = S_{D1} \oplus S_{D2} \oplus S_{D3} \oplus \ldots \ldots \oplus S_{Dd} \) ; \ldots \ldots (7.9)
\( S_D1 \), \( S_D2 \), \( S_D3 \), \ldots \( S_{Dd} \) are the syndromes for \( C_{r1} \), \( C_{r2} \), \( C_{r3} \) \ldots \( C_{rd} \) respectively, and
\[
S_P = S_{p1} \oplus S_{p2} \oplus S_{p3} \oplus \ldots \ldots \oplus S_{p(n-d)} \) ; \ldots \ldots (7.10)
\( S_{p1} \), \( S_{p2} \), \( S_{p3} \), \ldots \( S_{p(n-d)} \) are the syndromes for \( C_{r(d+1)} \), \( C_{r(d+2)} \), \( C_{r(d+3)} \) \ldots \( C_{rn} \) respectively.

3. The Mod-2 addition of \( d \) number of syndromes for \( C_{r1} \), \( C_{r2} \), \ldots \( C_{rd} \)

is the \((n-d)\) tuple formed with the \((n-d)\) parity bits of the error free valid code for the \( d \) number of data bits of \( C_r \). This is \( P_m \).

4. The Mod-2 addition of \((n-d)\) syndromes for \( C_{r(d+1)} \), \( C_{r(d+2)} \) \ldots \( C_{rn} \) is

same as parity portion of \( C_r \). This is the \((n-d)\) tuple formed with the \((n-d)\) Least Significant Bits of \( C_r \). This is \( P_r \).

5. Hence the overall syndrome, \( S = P_m \oplus P_r \) \ldots \ldots (7.11)

b) Extension of the above interpretation for the proposed methods :
\[
R = C_r \oplus R_m \ldots \ldots (7.12)
\]
\[
S = R \cdot H^T = [ C_r \oplus R_m ] \cdot H^T \ldots \ldots (7.13)
\]
\[
S = C_r \cdot H^T \oplus R_m \cdot H^T = S_C \oplus S_R \quad \ldots \ldots \ldots \ldots (7.14)
\]

\[
S_C = S_{cD} \oplus S_{cp} = \text{Net syndrome for } C_r \quad \ldots \ldots \ldots \ldots (7.15)
\]

\[
S_R = S_{RD} \oplus S_{Rp} = \text{Net syndrome for } R_m \quad \ldots \ldots \ldots \ldots (7.16)
\]

Since \( R_m \) is uniquely decided by the \((n-d)\) bit Golden sequence term as in method 1 or the values of \( J \) and \( K \) chosen as in method 2, the net syndrome for \( R_m \) is unique. Therefore \( S = S_C \oplus S_R \) is unique.

Thus each syndrome in the proposed methods is nothing but Mod-2 addition of the corresponding syndrome for the conventional method with the syndrome of \( R_m \).

If \( R_m \) is the all zero vector, the syndromes for the proposed methods result in the corresponding syndromes for conventional method.

**7.8 EXAMPLES**

**EXAMPLE ON CONSTRUCTION OF LOOK UP TABLE FOR THE PROPOSED ALGORITHM 1**

Example 7.1: The Lookup table for error correction is to be constructed for a \((6,3)\) code with the following Generator matrix \( G \), which is used in Example 6.13 for channel coding for the symbols of the Type II source of Example 5.1.

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

The codes in systematic form are derived by the formula \( C = DG \), where \( D \) is data vector and \( G \) is the generator matrix.
The codes are given below:
000000, 001011, 010110, 011101, 100101, 101110, 110011, 111000.
The all zero vector $C^1$ without any error is considered as the received vector $C_r$.
$C_r = C^10 = 000000$.
Therefore $R_m = 000 101$.
$R = C_r \oplus R_m = 000101$. $D_m = 000$, $P_m = 101$
$R_{ref} =$ The code for data 000 = 000000. $P_r = 000$.
Syndrome $S = P_m \oplus P_r = 101 \oplus 000 = 101$.
This is the first row of the Lookup table.

The vector $C^1_1$ is considered as the received vector $C_r$.
Therefore $C_r = 100000$.
$R = C_r \oplus R_m = 100101$. $D_m = 100$, $P_m = 101$
$R_{ref} =$ The code for data 100 = 100101. $P_r = 101$
Syndrome $S = P_m \oplus P_r = 101 \oplus 101 = 000$.
This is the second row of the Lookup table.
The vector $C^1_2$ is considered as the received vector $C_r$.
Therefore $C_r = 010000$.
$R = C_r \oplus R_m = 010101$. $D_m = 010$, $P_m = 101$
$R_{ref} =$ The code for data 010 = 010110. $P_r = 110$
Syndrome $S = P_m \oplus P_r = 101 \oplus 110 = 011$.
This is the third row of the Lookup table.
The vector $C_1^3$ is considered as the received vector $C_r$.

Therefore $C_r = 001000$.

$R = C_r \oplus R_m = 001101$. $D_m = 001$, $P_m = 101$

$R_{ref} =$ The code for data 001 = 001011. $P_r = 011$

Syndrome $S = P_m \oplus P_r = 101 \oplus 011 = 110$.

This is the fourth row of the Lookup table.

The vector $C_1^4$ is considered as the received vector $C_r$.

Therefore $C_r = 000100$.

$R = C_r \oplus R_m = 000001$. $D_m = 000$, $P_m = 001$

$R_{ref} =$ The code for data 000 = 000000. $P_r =$000

Syndrome $S = P_m \oplus P_r = 001 \oplus 000 = 001$.

This is the fifth row of the Lookup table.

The vector $C_1^5$ is considered as the received vector $C_r$.

Therefore $C_r = 000010$.

$R = C_r \oplus R_m = 000111$. $D_m = 000$, $P_m = 111$

$R_{ref} =$ The code for data 000 = 000000. $P_r =$000

Syndrome $S = P_m \oplus P_r = 111 \oplus 000 = 111$.

This is the sixth row of the Lookup table.

The vector $C_1^6$ is considered as the received vector $C_r$.

Therefore $C_r = 000001$.

$R = C_r \oplus R_m = 000100$. $D_m = 000$, $P_m = 100$
R_{ref} = The code for data 000 = 000000. P_r = 000

Syndrome S = P_m \oplus P_r = 100 \oplus 000 = 100.

This is the seventh row of the Lookup table.

Shown below in Table 7.2 is the complete Lookup Table.

Table 7.2: LOOKUP TABLE FOR A (6,3) CODE FOR METHOD 1

<table>
<thead>
<tr>
<th>Syndrome, S</th>
<th>Error Compensating Vector, C_{ec}</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>000000</td>
</tr>
<tr>
<td>000</td>
<td>100000</td>
</tr>
<tr>
<td>011</td>
<td>010000</td>
</tr>
<tr>
<td>110</td>
<td>001000</td>
</tr>
<tr>
<td>001</td>
<td>000100</td>
</tr>
<tr>
<td>111</td>
<td>000010</td>
</tr>
<tr>
<td>100</td>
<td>000001</td>
</tr>
</tbody>
</table>

Check for special case:

Syndrome for error free case: 101

This syndrome when Mod-2 added with syndrome of \( i \)th bit error gives the \( i \)th row of \( H^T \).

\[ 101 \oplus 101 = 000 \]
\[ 000 \oplus 101 = 101 \]
\[ 011 \oplus 101 = 110 \]
\[ 110 \oplus 101 = 011 \]
\[ 001 \oplus 101 = 100 \]
\[ 111 \oplus 101 = 010 \]
\[ 100 \oplus 101 = 001 \]

Now for the chosen generator matrix,

The rows of \( H^T \) are found to be same as found from the proposed method.
EXAMPLES ON CONSTRUCTION OF LOOK UP TABLE FOR THE PROPOSED ALGORITHM 2

Example 7.2: The Lookup table for error correction is to be constructed for a (7,4) code with the following Generator matrix $G$, which is used in Example 6.14 for channel coding for the symbols of the Type II source of Example 5.2.

$F_n = 5 = 2 + 3 = J + K$ ; $J = 2, K = 3$

One possible Generator Matrix:

$$G = \begin{bmatrix} 67 \\ 37 \\ 22 \\ 15 \end{bmatrix} = \begin{bmatrix} 1000 011 \\ 0100 101 \\ 0010 110 \\ 0001 111 \end{bmatrix}$$

The sixteen valid codes are:

0000 000, 0001 111, 0010 110, 0011 001, 0100 101, 0101 010, 0110 011, 0111 100, 1000 011, 1001 100, 1010 101, 1011 010, 1100 110, 1101 001, 1110 000, 1111 111.

The all zero vector $C_1$ without any error is considered as the received vector $C_r$.

$C_r = C_1^0 = 0000000$. $J - 1 = 1$, $K - 1 = 2$

Therefore $R_m = 1100000$.

$R = C_r \oplus R_m = 1100000$. $D_m = 1100$, $P_m = 000$

$R_{ref} = The code for data 1100 = 1100110$. $P_r = 110$

Syndrome, $S = P_m \oplus P_r = 000 \oplus 110 = 110$.

This is the first row of the Lookup table.
The vector $\mathbf{C}_1^1$ is considered as the received vector $\mathbf{C}_r$. Therefore $\mathbf{C}_r = 1000000$.

$\mathbf{R} = \mathbf{C}_r \oplus \mathbf{R}_m = 0100000$. $\mathbf{D}_m = 0100$, $\mathbf{P}_m = 000$

$\mathbf{R}_{\text{ref}} = \text{The code for data 0100 = 0100 101. } \mathbf{P}_r = 101$

Syndrome, $S = \mathbf{P}_m \oplus \mathbf{P}_r = 000 \oplus 101 = 101$.

This is the second row of the Lookup table.

The vector $\mathbf{C}_1^2$ is considered as the received vector $\mathbf{C}_r$. Therefore $\mathbf{C}_r = 0100000$.

$\mathbf{R} = \mathbf{C}_r \oplus \mathbf{R}_m = 1000000$. $\mathbf{D}_m = 1000$, $\mathbf{P}_m = 011$

$\mathbf{R}_{\text{ref}} = \text{The code for data 1000 = 1000 011. } \mathbf{P}_r = 011$

Syndrome, $S = \mathbf{P}_m \oplus \mathbf{P}_r = 000 \oplus 011 = 011$.

This is the third row of the Lookup table.

The vector $\mathbf{C}_1^3$ is considered as the received vector $\mathbf{C}_r$. Therefore $\mathbf{C}_r = 0010000$.

$\mathbf{R} = \mathbf{C}_r \oplus \mathbf{R}_m = 1110000$. $\mathbf{D}_m = 1110$, $\mathbf{P}_m = 000$

$\mathbf{R}_{\text{ref}} = \text{The code for data 1110 = 1110 000. } \mathbf{P}_r = 000$

Syndrome, $S = \mathbf{P}_m \oplus \mathbf{P}_r = 100 \oplus 110 = 000$.

This is the fourth row of the Lookup table.

The vector $\mathbf{C}_1^4$ is considered as the received vector $\mathbf{C}_r$. Therefore $\mathbf{C}_r = 0001000$. 


\[ R = C_r \oplus R_m = 1101000. \quad D_m = 1101, \quad P_m = 000 \]
\[ R_{ref} = \text{The code for data 1101 = 1101 001.} \quad P_r = 001 \]
Syndrome, \[ S = P_m \oplus P_r = 000 \oplus 001 = 001. \]
This is the fifth row of the Lookup table.

The vector \( C_1^5 \) is considered as the received vector \( C_r \).
Therefore \( C_r = 0000100. \)
\[ R = C_r \oplus R_m = 1100100. \quad D_m = 1100, \quad P_m = 100 \]
\[ R_{ref} = \text{The code for data 1100 = 1100 110.} \quad P_r = 110 \]
Syndrome, \[ S = P_m \oplus P_r = 100 \oplus 110 = 010. \]
This is the sixth row of the Lookup table.

The vector \( C_1^6 \) is considered as the received vector \( C_r \).
Therefore \( C_r = 0000010. \)
\[ R = C_r \oplus R_m = 1100010. \quad D_m = 1100, \quad P_m = 010 \]
\[ R_{ref} = \text{The code for data 1100 = 1100 110.} \quad P_r = 110 \]
Syndrome, \[ S = P_m \oplus P_r = 010 \oplus 110 = 100. \]
This is the seventh row of the Lookup table.

The vector \( C_1^7 \) is considered as the received vector \( C_r \).
Therefore \( C_r = 0000001. \)
\[ R = C_r \oplus R_m = 1100001. \quad D_m = 1100, \quad P_m = 001 \]
\[ R_{ref} = \text{The code for data 1100 = 1100 110.} \quad P_r = 110 \]
Syndrome, \[ S = P_m \oplus P_r = 001 \oplus 110 = 111. \]
This is the eighth row of the Lookup table.
Table 7.3 shown below is the complete Lookup Table.

Table 7.3 : LOOKUP TABLE FOR A (7,4) CODE FOR METHOD 2

<table>
<thead>
<tr>
<th>Syndrome , S</th>
<th>Error compensating Vector, $C_{cc}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>0000000</td>
</tr>
<tr>
<td>101</td>
<td>1000000</td>
</tr>
<tr>
<td>011</td>
<td>0100000</td>
</tr>
<tr>
<td>000</td>
<td>0010000</td>
</tr>
<tr>
<td>001</td>
<td>0001000</td>
</tr>
<tr>
<td>010</td>
<td>0000100</td>
</tr>
<tr>
<td>100</td>
<td>0000010</td>
</tr>
<tr>
<td>111</td>
<td>0000001</td>
</tr>
</tbody>
</table>

**Check for special case :**

Syndrome for error free case : 110

This syndrome when Mod-2 added with syndrome of $i^{th}$ bit error gives the $i^{th}$ row of $H^T$.

\[
101 \oplus 110 = 011
\]

\[
011 \oplus 110 = 101
\]

\[
000 \oplus 110 = 110
\]

\[
001 \oplus 110 = 111
\]

\[
010 \oplus 110 = 100
\]

\[
100 \oplus 110 = 010
\]

\[
111 \oplus 110 = 001
\]
Now for the chosen generator matrix,

\[ G = \begin{bmatrix} I_k \mid P_{(n-k)} \end{bmatrix} = \begin{bmatrix} 1000 & 011 \\ 0100 & 101 \\ 0010 & 110 \\ 0001 & 111 \end{bmatrix} \]

\[ H^T = \begin{bmatrix} P \\ I_{(n-k)} \end{bmatrix} = \begin{bmatrix} 011 \\ 101 \\ 110 \\ 111 \\ 100 \\ 010 \\ 001 \end{bmatrix} \]

The rows of \( H^T \) are found to be same as found from the proposed method for the special case of \( J = K = 1 \).

Example 7.3: The Lookup table for error correction is to be constructed for a (6,3) code with the following Generator matrix \( G \), which is used in Example 6.12 for channel coding for the symbols of the Type I source of Example 4.11.

\[ F_n = 8 = 2 + 6 = J + K \quad ; \quad J = 2, K = 6 \]

Generator matrix chosen \( G \) :

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

The codes in systematic form are derived by the formula \( C = DG \), where \( D \) is data vector and \( G \) is the generator matrix.

The codes are given below:

000000, 001011, 010110, 011101, 100101, 101110, 110011, 111000.
The all zero vector $C_1$ without any error is considered as the received vector $C_r$.

$C_r = C_1^0 = 000000$. $J-1=1$, $K-1=5$

Therefore $R_m = 100010$.

$R = C_r \oplus R_m = 100010$. $D_m = 100$, $P_m = 010$

$R_{ref} = $ The code for data 100 = 100101. $P_r = 101$

Syndrome $S = P_m \oplus P_r = 010 \oplus 101 = 111$.

This is the first row of the Lookup table.

The vector $C_1^1$ is considered as the received vector $C_r$.

Therefore $C_r = 100000$.

$R = C_r \oplus R_m = 000010$. $D_m = 000$, $P_m = 010$

$R_{ref} = $ The code for data 000 = 000000. $P_r = 000$

Syndrome $S = P_m \oplus P_r = 010 \oplus 000 = 010$.

This is the second row of the Lookup table.

The vector $C_1^2$ is considered as the received vector $C_r$.

Therefore $C_r = 010000$.

$R = C_r \oplus R_m = 110010$. $D_m = 110$, $P_m = 010$

$R_{ref} = $ The code for data 110 = 110011. $P_r = 011$

Syndrome $S = P_m \oplus P_r = 010 \oplus 011 = 001$.

This is the third row of the Lookup table.
The vector $C_1^3$ is considered as the received vector $C_r$.
Therefore $C_r = 001000$.

$R = C_r \oplus R_m = 101010$. $D_m = 101$, $P_m = 010$

$R_{ref}$ = The code for data $101 = 101110$. $P_r = 110$

Syndrome $S = P_m \oplus P_r = 010 \oplus 110 = 100$.
This is the fourth row of the Lookup table.

The vector $C_1^4$ is considered as the received vector $C_r$.
Therefore $C_r = 000100$.

$R = C_r \oplus R_m = 100110$. $D_m = 100$, $P_m = 110$

$R_{ref}$ = The code for data $100 = 100101$. $P_r = 101$

Syndrome $S = P_m \oplus P_r = 110 \oplus 101 = 011$.
This is the fifth row of the Lookup table.

The vector $C_1^5$ is considered as the received vector $C_r$.
Therefore $C_r = 000010$.

$R = C_r \oplus R_m = 100000$. $D_m = 100$, $P_m = 000$

$R_{ref}$ = The code for data $100 = 100101$. $P_r = 101$

Syndrome $S = P_m \oplus P_r = 000 \oplus 101 = 101$.
This is the sixth row of the Lookup table.

The vector $C_1^6$ is considered as the received vector $C_r$.
Therefore $C_r = 000001$.

$R = C_r \oplus R_m = 100011$. $D_m = 100$, $P_m = 011$

$R_{ref}$ = The code for data $100 = 100101$. $P_r = 101$
Syndrome $S = P_m \oplus P_r = 011 \oplus 101 = 110$.

This is the seventh row of the Lookup table.

Table 7.4 shown below is the complete Lookup Table.

<table>
<thead>
<tr>
<th>Syndrome, $S$</th>
<th>Error Compensating Vector, $C_{ec}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>000000</td>
</tr>
<tr>
<td>010</td>
<td>100000</td>
</tr>
<tr>
<td>001</td>
<td>010000</td>
</tr>
<tr>
<td>100</td>
<td>001000</td>
</tr>
<tr>
<td>011</td>
<td>000100</td>
</tr>
<tr>
<td>101</td>
<td>000010</td>
</tr>
<tr>
<td>110</td>
<td>000001</td>
</tr>
</tbody>
</table>

Check for special case:

Syndrome for error free case: 111

This syndrome when Mod-2 added with syndrome of $i^{th}$ bit error gives the $i^{th}$ row of $H^T$.

$010 \oplus 111 = 101$

$001 \oplus 111 = 110$

$100 \oplus 111 = 011$

$011 \oplus 111 = 100$

$101 \oplus 111 = 010$

$110 \oplus 111 = 001$

Now for the chosen generator matrix,

The rows of $H^T$ are found to be same as found from the proposed method for the special case of $J = K = 1$. 
EXAMPLE ON ERROR DETECTION AND CORRECTION

FOR THE PROPOSED ALGORITHM 1

Example 7.4 : The (6, 3) linear block code of example 7.1 is considered for checking the result of applying Algorithm 1 for error correction.

The eight valid codes are given below:

000000, 001011, 010110, 011101
100101, 101110, 110011, 111000.

The lookup table of Table 7.2 constructed for this code is used for error correction.

1. The transmitted code \( C_t = 111000 \) Error in third bit is considered.
   Therefore the received code \( C_r = 110000 \).

\[
R = 110101
\]
\[R_{ref} = \text{The code for data 110} = 110011.
\]
\[P_m = 101, P_r = 011.
\]
\[S = P_m \oplus P_r = 101 \oplus 011 = 110
\]
From lookup table \( C_{ec} = 001000 \) and \( C_c = 111000 \).

2. The transmitted code \( C_t = 001011 \) Error in fifth bit is considered.
   Therefore the received code \( C_r = 001001 \).

\[
R = 001100
\]
\[R_{ref} = \text{The code for data 001} = 001011.
\]
\[P_m = 100, P_r = 011.
\]
\[S = P_m \oplus P_r = 100 \oplus 011 = 111
\]
From lookup table \( C_{ec} = 000010 \) and \( C_c = 001011 \).
Example 7.5: The transmission of symbols of Type II source of example 6.13 is considered in this example. The lookup table constructed for the decoding of the (6,3) code of example 7.1 is used for error correction. Original sequence of symbols from this source: \(s_2 \ s_3 \ s_4\) 

Original binary data of symbols after source coding: 111 101 100 

This binary data is arranged into blocks of three bits each:

111 101 100

For the same Generator matrix, \(G\) as used in example 7.1 the transmitted bit stream of Type II source with channel coding:

111000 101110 100101

These three code vectors are used as received codes for testing the error correction scheme proposed in the present chapter.

From the example 7.1 the lookup table constructed is shown below as Table 7.5.

Table 7.5: LOOKUP TABLE FOR A (6,3) CODE

<table>
<thead>
<tr>
<th>Syndrome, (S)</th>
<th>Error Compensating Vector, (C_{ec})</th>
</tr>
</thead>
<tbody>
<tr>
<td>101</td>
<td>000000</td>
</tr>
<tr>
<td>000</td>
<td>100000</td>
</tr>
<tr>
<td>011</td>
<td>010000</td>
</tr>
<tr>
<td>110</td>
<td>001000</td>
</tr>
<tr>
<td>001</td>
<td>000100</td>
</tr>
<tr>
<td>111</td>
<td>000010</td>
</tr>
<tr>
<td>100</td>
<td>000001</td>
</tr>
</tbody>
</table>
1. The transmitted code $C_t = 111\ 000$ Error in second bit is considered. Therefore the received code $C_r = 101\ 000$.

$R = 101\ 101$.

$R_{ref} =$ The code for data 101 =101 110.

$P_m =101, \ P_r = 110$.

$S = P_m \oplus P_r = 101 \oplus 110 = 011$

From lookup table $C_{ec} = 010\ 000$ and $C_c =111\ 000$.

2. The transmitted code $C_t = 101\ 110$ Error in fourth bit is considered. Therefore the received code $C_r = 101\ 010$.

$R = 101\ 111$

$R_{ref} =$ The code for data 101 =101 110.

$P_m = 111, \ P_r = 110$.

$S = P_m \oplus P_r = 111 \oplus 110 = 001$

From lookup table $C_{ec} = 000\ 100$ and $C_c = 101\ 110$.

3. The transmitted code $C_t = 100\ 101$ Error in fifth bit is considered. Therefore the received code $C_r = 100\ 111$.

$R = 100\ 010$

$R_{ref} =$ The code for data 100 =100 101.

$P_m = 010, \ P_r = 101$.

$S = P_m \oplus P_r = 010 \oplus 101 = 111$

From lookup table $C_{ec} = 000\ 010$ and $C_c = 100\ 101$. 
EXAMPLES ON ERROR DETECTION AND CORRECTION

FOR THE PROPOSED ALGORITHM 2

Example 7.6: The (6,3) linear block code of example 7.3 is considered for channel coding and the proposed Algorithm 2 for error correction.

Since the lookup table is constructed $F_n = 8$, $J = 2$, $K = 6$,
the vector $R_m = 100\ 010$.

1. The transmitted code $C_t = 001011$ Error in third bit is considered. Therefore the received code $C_r = 000011$.

$R = 100001$

$R_{ref} =$ The code for data 100 =100 101.

$P_m = 001$, $P_r = 101$.

$S = P_m \oplus P_r = 001 \oplus 101 = 100$

From lookup table $C_{ec} = 001000$ and $C_c = 001011$.

2. The transmitted code $C_t = 011101$ Error in fifth bit is considered. Therefore the received code $C_r = 011111$.

$R = 111101$

$R_{ref} =$ The code for data 111 =111 000.

$P_m = 101$, $P_r = 000$.

$S = P_m \oplus P_r = 101 \oplus 000 = 101$

From lookup table $C_{ec} = 000010$ and $C_c = 011101$. 
3. The transmitted code $C_t = 111000$ Error in first bit is considered. Therefore the received code $C_r = 011000$.

$R = 111 010$

$R_{ref} = $ The code for data $111 = 111 000$.

$P_m = 010, P_r = 000$.

$S = P_m \oplus P_r = 010 \oplus 000 = 010$

From lookup table $C_{ec} = 100000$ and $C_c = 111000$.

Example 7.7: Using the lookup table created in example 7.3 errors in the channel codes for the transmitted symbols of Type I source of example 6.12 are to be corrected.

Original sequence of symbols from this source: $s_2 \ s_3 \ s_4$

Original binary data of symbols after source coding: $11 \ 100 \ 1011$

This binary data arranged into blocks of three bits each as:

$111 \ 001 \ 011$

The chosen Generator matrix, $G$

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

The codes in systematic form are derived by the formula $C = DG$, where $D$ is data vector and $G$ is the generator matrix.

The codes are given below:

000000, 001011, 010110, 011101, 100101, 101110, 110011, 111000.
The transmitted bit stream of type I source with channel coding:

\[111000\quad 001011\quad 011101.\]

These three code vectors are used as received codes for testing the error correction scheme proposed in the work of present chapter.

The lookup table constructed in example 7.3 is shown in Table 7.6 below.

Table 7.6 : LOOKUP TABLE FOR A (6,3) CODE

<table>
<thead>
<tr>
<th>Syndrome, S</th>
<th>Error Compensating Vector, ( C_{cc} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>111</td>
<td>000000</td>
</tr>
<tr>
<td>010</td>
<td>100000</td>
</tr>
<tr>
<td>001</td>
<td>010000</td>
</tr>
<tr>
<td>100</td>
<td>001000</td>
</tr>
<tr>
<td>011</td>
<td>000100</td>
</tr>
<tr>
<td>101</td>
<td>000010</td>
</tr>
<tr>
<td>110</td>
<td>000001</td>
</tr>
</tbody>
</table>

1. The transmitted code \( C_t = 111\ 000 \). It is considered that the error has occurred in third bit. Therefore the received code \( C_r = 110\ 000 \).

\[ R = 010\ 010 \]

\( R_{ref} = \) The code for data 010 = 010 110.

\( P_m = 010, \ P_r = 110. \)

\[ S = P_m \oplus P_r = 010 \oplus 110 = 100 \]

From lookup table \( C_{cc}= 001\ 000 \) and \( C_c=111\ 000 \).

2. The transmitted code \( C_t = 001\ 011 \). It is considered that the error has occurred in fifth bit. Therefore the received code \( C_r = 001\ 001 \).

\[ R = 101\ 011 \]

\( R_{ref} = \) The code for data 101 =101 110.
\[ P_m = 011, P_r = 110. \]
\[ S = P_m \oplus P_r = 011 \oplus 110 = 101 \]
From lookup table \( C_{ec} = 000 \ 010 \) and \( C_c = 001 \ 011 \).

3. The transmitted code \( C_t = 011 \ 101 \). It is considered that the error has occurred in first bit. Therefore the received code \( C_r = 111 \ 101 \).
\[ R = 011 \ 111 \]
\[ R_{ref} = \text{The code for data} \ 011 = 011 \ 101. \]
\[ P_m = 111, P_r = 101. \]
\[ S = P_m \oplus P_r = 111 \oplus 101 = 010 \]
From lookup table \( C_{ec} = 100 \ 000 \) and \( C_c = 011 \ 101 \).

Example 7.8: In this example the transmission of symbols of the English text message of example 6.14 is considered.

For the transmitted message “source coding. ”, the binary data is arranged in blocks of four bits each. The first three data blocks of four bits each and their channel codes are taken in this example which are shown below.

\[
\begin{align*}
1000 & & 1001 & & 0101 \\
1000 \ 011 & & 1001 \ 100 & & 0101 \ 010 \\
\end{align*}
\]
\[ F_n = 5 = 2 + 3 = J + K \quad J = 2, K = 3 \]
From example 6.14 the sixteen valid codes are: \( 0000 \ 000, 0001 \ 111, 0010 \ 110, 0011 \ 001, 0100 \ 101, 0101 \ 010, 0110 \ 011, 0111 \ 100, \)
1000 011, 1001 100, 1010 101, 1011 010, 1100 110, 1101 001, 1110 000, 1111 111.

The lookup table constructed for this code in example 7.2 is reproduced in the Table 7.7 below which can be used for error detection.

Table 7.7 : LOOKUP TABLE FOR A (7,4) CODE

<table>
<thead>
<tr>
<th>Syndrome, S</th>
<th>Error compensating Vector, Cec</th>
</tr>
</thead>
<tbody>
<tr>
<td>110</td>
<td>0000000</td>
</tr>
<tr>
<td>101</td>
<td>1000000</td>
</tr>
<tr>
<td>011</td>
<td>0100000</td>
</tr>
<tr>
<td>000</td>
<td>0010000</td>
</tr>
<tr>
<td>001</td>
<td>0001000</td>
</tr>
<tr>
<td>010</td>
<td>0000100</td>
</tr>
<tr>
<td>100</td>
<td>0000010</td>
</tr>
<tr>
<td>111</td>
<td>0000001</td>
</tr>
</tbody>
</table>

1. The transmitted code C_t = 1000 011. It is considered that the error has occurred in sixth bit. Therefore the received code C_r = 1000 001.

R_m = 0100 001

R_ref = The code for data 0100 = 0100 101.

P_m = 001, P_r = 101.

S = P_m ⊕ P_r = 001 ⊕ 101 = 100

From lookup table C_{ec} = 0000 010 and C_c = 1000 011.

2. The transmitted code C_t = 1001 100. It is considered that the error has occurred in second bit. Therefore the received code C_r = 1101 100.

R_m = 0001 100
R_{ref} = The code for data 0001 = 0001 111.

P_m = 100, P_r = 111.

S = P_m \oplus P_r = 100 \oplus 111 = 011

From lookup table C_{cc} = 0100 000 and C_c = 1001 100.

3. The transmitted code C_t = 0101 010. It is considered that the error has occurred in fifth bit. Therefore the received code C_r = 0101 110.

R_m = 1001 110

R_{ref} = The code for data 1001 = 1001 100.

P_m = 110, P_r = 100.

S = P_m \oplus P_r = 110 \oplus 100 = 010

From lookup table C_{cc} = 0000 100 and C_c = 0101 010.
7.9 FLOW CHART:

start

Enter n, k values
Enter kxn Generator matrix

Find kxn-k parity part matrix

Enter required Fibonacci number and j, k values

Consider valid code 000…n bits

i = 0

Reverse j-1, k-1 bits to get Rₘ

Find Rₙ from Generator matrix

Find Syndrome

\[ S[i] = \text{XOR} \{(n-k) \text{ LSB bits of } Rₘ, (n-k) \text{ LSB bits of } Rₙ\} \]

Store the value in Syndrome Lookup Table

i ≤ n

false

Display Syndrome and lookup table

true

i++

Reverse \( i^{th} \) bit of valid code
Continuation of the flowchart …..

D

Enter the received code \( C_r \)

Reverse \( j-1, k-1 \) bits to get \( R_m \)

Find \( R_r \)

Calculate Syndrome

Find error position syndrome lookup table and name it as ‘e’

Find error vector \( C_e \)

Store the value in Syndrome Lookup Table

\( C_t = C_e \) xor \( C_r \)

Display Syndrome and lookup table

end
7.10 SIMULATION RESULTS:

Example 7.1: Construction of lookup table for a (6,3) code using Algorithm 1

```
MATLAB

% enter the n value 6
% enter the k value 3
% enter 3x6 generator matrix [1 0 0 1 0 1; 1 0 1 1 0; 0 0 1 0 1]
1 0 0 1 0 1
0 1 0 1 1 0
0 0 1 0 1 1

therefore 3x3 parity part of generator matrix
1 0 1
1 1 0
0 1 1
```

```
MATLAB

% let consider a valid code [0 0 0 0]
% vtemp =
% 0 0 0 0 1 0 0
% vtemp =
% 0 0 0 1 0 0
% vtemp =
% 0 0 0 1 0 0
% vtemp =
% 0 0 0 1 0 1
% Dm = [0 0 0 1 0 1]
% Rr = [0 0 0 0 1 0]
% syndrome = 1 0 1
```

Example 7.1 continued....
Example 7.5: Error correction using the lookup table of Example 7.1

```
% MATLAB

% Enter the received code [1 0 0 1 0 0]
% Rm = [1 0 1 1 0 1]
% Hr = [1 0 1 1 1 0]
% Syndrome S = [0 1 1]
% Error is in bit 2
% Error vector = 0 1 0 0 0 0
% Hence, transmitted code = 1 1 0 0 0 0

>>
```

```
% MATLAB

% Enter the received code [1 0 0 1 0 0]
% Rm = [1 0 1 1 1 1]
% Hr = [1 0 1 1 1 0]
% Syndrome S = [0 0 1]
% Error is in bit 4
% Error vector = 0 0 0 1 0 0
% Hence, transmitted code = 1 0 1 1 0 0

>>
```

```
% MATLAB

% Enter the received code [1 0 0 1 1 0]
% Rm = [1 0 0 0 1 0]
% Hr = [1 0 0 1 0 1]
% Syndrome S = [1 1 1]
% Error is in bit 5
% Error vector = 0 0 0 0 1 0
% Hence, transmitted code = 1 0 0 1 0 1

>>
```
Example 7.2: Construction of lookup table for a (7,4) code using Algorithm 2

```
MATLAB

>> enter the n value 7
>> enter the k value 4
>> enter 4x7 generator matrix:
    1 0 0 0 1 1 0
    0 1 0 0 1 0 1
    0 0 1 1 0 1 1
    0 0 0 1 1 1 1

Therefore 4x3 parity part of generator matrix:
    0 1 1
    1 1 0
    1 1 1
```

```
MATLAB

>> enter a syndrome no: 5
>> enter i value 2
>> enter k value 3

1,3 bits of received code are to be reversed.
Let consider a valid code=0 0 0 0 0 0
Rx = 1 1 0 0 1 0
syndrome S = 1 1 0
error in bit position 1
Rx = 0 1 0 0 0 0
Syndrome S = 1 0 1
error in bit position 2
Rx = 1 0 0 0 0 0
Syndrome S = 0 1 1
error in bit position 3
Rx = 1 1 0 0 0 0
Syndrome S = 0 0 0
```
Example 7.2 continued....

error in bit position 4
RM = 1 1 0 1 0 0 0
Rr = 1 1 0 1 0 0 1
syndrome S = 0 0 1
error in bit position 5
RM = 1 1 0 0 1 0 0
Rr = 1 1 0 0 1 1 0
syndrome S = 0 1 0
error in bit position 6
RM = 1 1 0 0 0 1 0
Rr = 1 1 0 0 1 1 0
syndrome S = 1 0 0
error in bit position 7
RM = 1 1 0 0 0 0 1
Rr = 1 1 0 0 1 1 0
syndrome S = 1 1 1
syndrome look up table:
1 1 0
1 0 1
0 1 1
0 0 0
0 0 1
0 1 0
1 0 0
1 1 1
Example 7.8: Error correction using the lookup table of Example 7.2

```
Example 7.8: Error correction using the lookup table of Example 7.2

```

```
Example 7.8: Error correction using the lookup table of Example 7.2

```

```
Example 7.8: Error correction using the lookup table of Example 7.2

```
Example 7.3: Construction of lookup table for a (6,3) code using Algorithm 2

---

**MATLAB**

File Edit Debug Desktop Window Help

Shortcuts How to Add Switch to New

enter the k value 6
enter the n value 3
enter 3x6 generator matrix:
\[
\begin{bmatrix}
1 & 0 & 0 & 1 & 0 & 1 \\
1 & 0 & 1 & 0 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 0 \\
0 & 1 & 0 & 1 & 1 & 1
\end{bmatrix}
\]

therefore 3x3 parity part of generator matrix:
\[
\begin{bmatrix}
1 & 0 & 1 \\
1 & 1 & 0 \\
0 & 1 & 1
\end{bmatrix}
\]

---

**MATLAB**

File Edit Debug Desktop Window Help

Shortcuts How to Add Switch to New

enter fibonacci no: 6
enter j value 2
enter k value 6
1,5 bits of interleaved code are to be reversed
let consider a valid code=0 0 0 0 0 0
\[ P_1 = 0 0 0 1 0 \]
\[ P_2 = 0 0 1 0 1 \]
syndrome s= 1 1 1

---

"Start"
Example 7.3 continued....
Example 7.7: Error correction using the lookup table of Example 7.3
7.11 RESULTS AND DISCUSSION

RESULTS:

The proposed algorithms are applied for correcting the single bit errors of the received codes and found to be working satisfactorily. Of particular interest, are the received codes for the data transmitted by the three sources considered in the examples 4.11, 5.1 and 5.2. Single bit errors are introduced in the channel codes of these sources and are corrected using the two proposed algorithms. It is found that the proposed schemes are working efficiently.

DISCUSSION:

Performance comparison with the conventional method:

a) The proposed methods are compared with conventional method from the view point of the theoretical requirements of memory and the total number of operations to be carried out. The values are computed for few (n, k) codes and the results are summarized in Table 7.8 and Table 7.9.

Comparison for Memory requirement:

Conventional method:

1. Memory required to store each syndrome: (n-k) bits.

   Number of syndrome vectors: $2^{n-k}$

   Memory required to store syndromes: $2^{n-k}(n-k)$……..(I)

2. Memory required to store each error pattern: n

   Number of error patterns: $2^{n-k}$

   Memory required to store error patterns: $2^{n-k}(n)$……..(II)
Total memory required : (I) + (II) = $2^{n-k}(2n-k)$ ..........(7.17)

Proposed methods :

1. Memory required to store parity bits of each code : (n-k) bits

   Number of valid codes : $2^{n-k}$

   Memory required to store parity bits of all valid codes :
   $2^{n-k}$ (n-k) bits..............................................(III)

2. Memory required to store each error compensating vector : n bits

   Number of error compensating vectors : (n+1)

   Memory required to store error compensating vectors :
   n (n+1) bits...................................................(IV)

Total memory required to construct the lookup table :

   (IV)+(V) = $2^{n-k}$ (n-k) + n (n+1).............(7.18)

Comparison for total number of operations :

Conventional method :

1. To find syndrome, $S = C_r \cdot H^T$

   AND operations : n (n-k)......................(A)

   Ex-OR operations : (n-1)(n-k) ..........(B)

2. For final error correction, $C_{rc} = C_r \oplus C_{ec}$

   Ex-OR additions : n ...............................(C)

   Total number of operations : (A)+(B)+(C) = n(n-k)+(n-1)(n-k)+n

   = (2n-1)(n-k)+n..........(7.19)
Proposed methods:

1. To find R : C_r ⊕ R_m : (n-k) ......................(P)

   If (J-1) ≤ (d+1) ≤ n  ;  If (K-1) ≤ (d+1) ≤ n

2. To find syndrome : (n-k).............(Q)

3. For final error correction : n .................(R)

Total number of operations :

\( (P)+(Q)+(R) = (n-k) + (n-k) + n = 2(n-k) + n \)................(7.20)

Table 7.8 : COMPARISON WITH THE CONVENTIONAL METHOD FOR MEMORY REQUIREMENT

<table>
<thead>
<tr>
<th>(n, k) code</th>
<th>Conventional method (2^{n-k}(2n-k)) bits</th>
<th>Proposed methods (2^{n-k}(n-k) + n(n+1)) bits</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2,5)</td>
<td>(2^3(8) = 64)</td>
<td>(8(3)+5(6)=24+30=54)</td>
</tr>
<tr>
<td>(3,6)</td>
<td>(2^3(9) = 72)</td>
<td>(8(3)+6(7)=24+42=66)</td>
</tr>
<tr>
<td>(4,7)</td>
<td>(2^3(10) = 80)</td>
<td>(8(3)+7(8)=24+56=80)</td>
</tr>
<tr>
<td>(5,9)</td>
<td>(2^4(13) = 208)</td>
<td>(16(4)+9(10)=64+90=154)</td>
</tr>
<tr>
<td>(6,10)</td>
<td>(2^4(14) = 224)</td>
<td>(16(4)+10(11)=64+110=174)</td>
</tr>
<tr>
<td>(11,15)</td>
<td>(2^4(19) = 304)</td>
<td>(16(4)+15(16)=64+240=304)</td>
</tr>
<tr>
<td>(12,17)</td>
<td>(2^5(22) = 704)</td>
<td>(32(5)+17(18)=160+306=466)</td>
</tr>
<tr>
<td>(26,31)</td>
<td>(2^5(62-26)=1152)</td>
<td>(32(5)+31(32)=160+992=1152)</td>
</tr>
</tbody>
</table>

Table 7.9 : COMPARISON WITH THE CONVENTIONAL METHOD FOR TOTAL NUMBER OF OPERATIONS

<table>
<thead>
<tr>
<th>(n,k) code</th>
<th>Conventional method ((2n-1)(n-k)+n)</th>
<th>Proposed methods (2(n-k)+n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(2,5)</td>
<td>(9(3)+5=32)</td>
<td>(2(3)+5=11)</td>
</tr>
<tr>
<td>(3,6)</td>
<td>(11(3)+6=39)</td>
<td>(2(3)+6=12)</td>
</tr>
<tr>
<td>(4,7)</td>
<td>(13(3)+7=46)</td>
<td>(2(3)+7=13)</td>
</tr>
<tr>
<td>(5,9)</td>
<td>(17(4)+9=77)</td>
<td>(2(4)+9=17)</td>
</tr>
<tr>
<td>(6,10)</td>
<td>(19(4)+10=86)</td>
<td>(2(4)+10=18)</td>
</tr>
<tr>
<td>(11,15)</td>
<td>(29(4)+15=131)</td>
<td>(2(4)+15=23)</td>
</tr>
<tr>
<td>(12,17)</td>
<td>(33(5)+17=182)</td>
<td>(2(5)+17=27)</td>
</tr>
<tr>
<td>(26,31)</td>
<td>(61(5)+31=336)</td>
<td>(2(5)+31=41)</td>
</tr>
</tbody>
</table>
It is observed from the above tables that the proposed algorithms require far less memory and fewer total number of operations.

b) The proposed methods are compared with conventional method from the view point of hardware implementation of the schemes. The comparison is carried out with respect to the following parameters in the form of digital circuit components required for the hardware realization.

1. Number of gates.
2. Number of shift registers.
3. Number of D-flip flops.

The results of this parameter wise requirement comparison are shown in tables 7.10, 7.11 and 7.12.

Number of gates:

The results of this comparison are shown in the Table 7.10

Table 7.10 : COMPARISON WITH THE CONVENTIONAL METHOD FOR NUMBER OF GATES

<table>
<thead>
<tr>
<th></th>
<th>(6,3) code</th>
<th>(7,4) code</th>
<th>(9,5) code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional method</td>
<td>129</td>
<td>232</td>
<td>288</td>
</tr>
<tr>
<td>Proposed methods</td>
<td>129</td>
<td>187</td>
<td>288</td>
</tr>
</tbody>
</table>

Number of shift registers:

The results of this comparison are shown in the Table 7.11

Table 7.11 : COMPARISON WITH THE CONVENTIONAL METHOD FOR NUMBER OF SHIFT REGISTERS

<table>
<thead>
<tr>
<th></th>
<th>(6,3) code</th>
<th>(7,4) code</th>
<th>(9,5) code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional method</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Proposed methods</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Number of D flip flops:

The results of this comparison are shown in the Table 7.12

Table 7.12: COMPARISON WITH THE CONVENTIONAL METHOD FOR NUMBER OF D FLIP FLOPS

<table>
<thead>
<tr>
<th></th>
<th>(6,3) code</th>
<th>(7,4) code</th>
<th>(9,5) code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional</td>
<td>3</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Proposed methods</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

c) The proposed methods are compared with conventional method from the view point of Response time. The results of this comparison are shown in the Table 7.13

Table 7.13: COMPARISON WITH THE CONVENTIONAL METHOD FOR RESPONSE TIME IN NANO SECONDS

<table>
<thead>
<tr>
<th></th>
<th>(6,3) code</th>
<th>(7,4) code</th>
<th>(9,5) code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Conventional</td>
<td>5.596</td>
<td>5.5897</td>
<td>5.607</td>
</tr>
<tr>
<td>Proposed methods</td>
<td>6.679</td>
<td>7.687</td>
<td>8.692</td>
</tr>
</tbody>
</table>

It is observed from the above tables that the proposed algorithms require far less number of shift registers, D flip flops and comparable number of gates. But the response time of the proposed method is slightly more than the response time for the conventional method.
7.12 CONCLUSIONS

Two schemes for correction of single bit errors of linear block codes in systematic form are proposed in this chapter. The schemes have the following advantages over the conventional method.

1. The methods require less memory for the lookup table
2. The methods involve less number of operations
3. The implementation of the schemes requires less hardware in the form of shift registers and D-flip flops and almost equal number of gates.
4. Though the response time is slightly more the resulting advantages out beat this drawback.
5. The syndromes for the conventional methods can be easily computed by applying the proposed methods as a special case.