Coding: Consider a communication system w.r.t encoder - decoder as shown in figure.

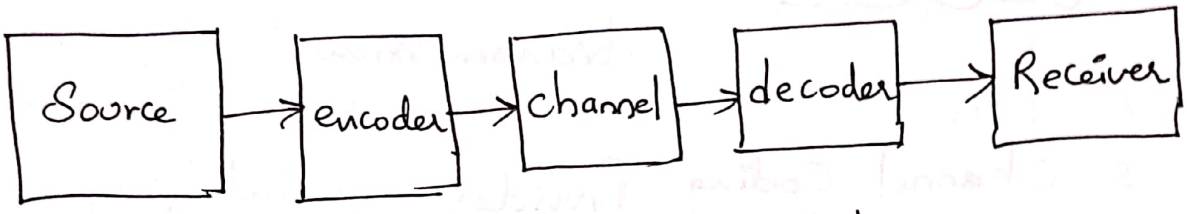


Fig: Communication System

→ the messages are first encoded by the encoder and then transmitted via channel.

→ At the receiving end, the received messages are first decoded and then the original messages are recovered.

→ Coding is defined as a procedure for mapping a given set of messages $[X] = [x_1\ x_2\ x_3 \cdots x_N]$ into a new set of encoded messages $[C] = [c_1\ c_2\ c_3 \cdots c_N]$ in such a way that transformation is one-to one. ie for each message there is only one encoded message.

→ Decoding is defined as a procedure of recovering original messages $[x] = [x_1\ x_2\ x_3 \cdots x_N]$ from encoded messages $[C] = [c_1\ c_2\ c_3 \cdots c_N]$

→ there are two types of Coding

Source Coding : Improves the efficiency of transmission. Ex: Shannon fano Coding and Huffman Coding

Channel Coding Provides special purposes such as Secrecy and minimum Pe by detecting & correcting errors. Ex: Hamming Code & cyclic code

→ Uniquely decipherable (or) Separable means the encoding and decoding correspondence of all possible sequence of words between the two languages is one to one when there is no space between the words.

→ Irreducibility (or) Prefix property means when no encoded words can be obtained from each other by the addition of more letters.

→ when a code is irreducible, it is also uniquely decipherible, but the reverse is not true. This can be explained from the following examples

Ex: 1 Let $C_1 = 0$, $C_2 = 10$, $C_3 = 110$

C_1	0
C_2	1 0
C_3	1 1 0

This code is irreducible and also decipherable.

If we receive encoded message as

$$\underbrace{0}\underbrace{1\,1\,0}\underbrace{1\,0}\underbrace{1\,0}\underbrace{0\,0}\underbrace{1\,0}\underbrace{1\,1\,0}\underbrace{1\,1\,0}\underbrace{1\,0}$$

then it is uniquely decoded as

$$C_1\ C_3\ C_2\ C_2\ C_1\ C_1\ C_2\ C_3\ C_3\ C_2$$

Hence if a code is irreducible, it is also uniquely decipherable.

Ex: 2 Let $C_1 = 0$, $C_2 = 01$, $C_3 = 011$

C_1	0
C_2	0 1
C_3	0 1 1

This code is decipherable. but not irreducible.

If we receive encoded message as

$$\underbrace{0}\underbrace{0}\underbrace{0}\underbrace{1\,1}\underbrace{0}\underbrace{1\,1}\underbrace{0}\underbrace{1}\underbrace{0}\underbrace{0}\underbrace{1\,0}\underbrace{1\,0}\underbrace{1\,0}\underbrace{1\,1}$$

then it is uniquely decoded as

$$C_1\ C_1\ C_3\ C_3\ C_2\ C_1\ C_2\ C_2\ C_2\ C_3$$

Hence if a code is decipherable, it is not necessarily irreducible.

→ English language is not uniquely decipherable i.e If FACTOR is received, it may be decoded either as a single word FACTOR (or) combination of two words FACT & OR.

Coding Efficiency (η) :

→ Let M be the number of symbols in an encoding alphabet, N be the number of messages in the message ensemble $[X] = [x_1\ x_2\ x_3 \dots x_N]$ with respective probabilities $[P] = [P_1\ P_2\ P_3 \dots P_N]$. No. of bits in encoded message $[C] = [C_1\ C_2\ C_3 \dots C_N]$ are given as $n = [n_1\ n_2\ n_3 \dots n_N]$.

→ Average length $L = \sum_{K=1}^{N} n_K P_K$ letter/message

minimum average length $L_{min} = \dfrac{H(x)}{\log_2 M}$

Entropy $H(x) = -\sum_{K=1}^{N} P_K \log_2 P_K$ bits/message

no. of bits per letter $= \log_2 M$ bits/letter

$$\text{Coding efficiency } \eta = \frac{L_{min}}{L}$$

$$= \frac{H(x) \text{ bits/message}}{L \text{ letters/message} \times \log_2 M \text{ bits/letter}}$$

$$\boxed{\%\ \eta = \frac{H(x)}{L \log_2 M} \times 100\ \%}$$

$$\boxed{\text{Redundancy} = 1 - \text{Efficiency}}$$

Shannon Encoding Algorithm (or) Shannon Fano Coding

→ This coding method constructs a reasonably efficient separable binary code

→ Consider [X] be the ensemble of messages to be transmitted and [P] be their corresponding probabilities.

→ The coding sequence C_k of binary numbers of the length n_k associated to each message x_k should fulfil the following two conditions

- No sequences of employed binary numbers C_k can be obtained from each other by adding more binary digits to the shortest sequence. This is called as prefix property.
- The transmission of an encoded message is reasonably efficient. ie 1 and 0 appear independently with almost equal probabilities.

The Procedure of Shannon Fano Coding via Source Partition is as follows:

→ The messages are first written in the order of non increasing probabilities.

→ The message set is then partioned into two most equiprobable subsets $[X_1]$ and $[X_2]$. A '0' is assigned to each message contained in one subset and a '1' is assigned to each message contained in another subset.

→ The above procedure is repeated for the subsets of $[X_1]$ and $[X_2]$ ie $[X_1]$ will be partitioned into two subsets $[X_{11}]$ & $[X_{12}]$ and $[X_2]$ will be partitioned into two subsets $[X_{21}]$ & $[X_{22}]$

→ The code words in $[X_{11}]$, $[X_{12}]$, $[X_{21}]$, $[X_{22}]$ will be starting from 00, 01, 10, 11 respectively.

→ The procedure is continued until each subset consists only one message.

→ It is noted that less no. of bits are assigned to more probable messages and more no. of bits are assigned to less probable messages.

P) Apply the Shannon-fano Coding procedure for the following message ensemble with probabilities as given. (07)

$$[x] = [x_1 \; x_2 \; x_3 \; x_4 \; x_5 \; x_6 \; x_7 \; x_8]$$

$$[P] = \left[\frac{1}{4} \; \frac{1}{8} \; \frac{1}{16} \; \frac{1}{16} \; \frac{1}{16} \; \frac{1}{4} \; \frac{1}{16} \; \frac{1}{8}\right]$$

Take $M = 2$.

Sol:→ Given message ensemble with probabilities

$$[x] = [x_1 \; x_2 \; x_3 \; x_4 \; x_5 \; x_6 \; x_7 \; x_8]$$

$$[P] = \left[\frac{1}{4} \; \frac{1}{8} \; \frac{1}{16} \; \frac{1}{16} \; \frac{1}{16} \; \frac{1}{4} \; \frac{1}{16} \; \frac{1}{8}\right]$$

$$= [0.25 \; 0.125 \; 0.0625 \; 0.0625 \; 0.0625 \; 0.25 \; 0.0625 \; 0.125]$$

no. of Symbols $M = 2$, No. of messages $N = 8$

→ In Shannon-fano Coding, first step is to arrange the messages in the decreasing order of their probabilities and second step is to assign Codes via source partition such that less no. of bits are assigned to more probable messages and more no. of bits are assigned to less probable messages

→ Coding Efficiency $\eta = \frac{H(x)}{L \log_2 M} \times 100\ \%$

where Entropy $H(x) = -\sum_{K=1}^{8} P_K \log_2 P_K$ bits/message

Average length $L = \sum_{K=1}^{8} n_K P_K$ letter/message

no. of bits per letter $= \log_2 M$ bits/letter

messages [X]	Probabilities [P]	Shannon fano Coding Via Source Partition	Code [C]	Length [n]
x_1	0.25	0 0	0 0	2
x_6	0.25	0 1	0 1	2
x_2	0.125	1 0 0	1 0 0	3
x_8	0.125	1 0 1	1 0 1	3
x_3	0.0625	1 1 0 0	1 1 0 0	4
x_4	0.0625	1 1 0 1	1 1 0 1	4
x_5	0.0625	1 1 1 0	1 1 1 0	4
x_7	0.0625	1 1 1 1	1 1 1 1	4

Here $L = \left(2 \times \frac{1}{4}\right) + \left(3 \times \frac{1}{8}\right) + \left(4 \times \frac{1}{16}\right) + \left(4 \times \frac{1}{16}\right) + \left(4 \times \frac{1}{16}\right) + \left(2 \times \frac{1}{4}\right) + \left(4 \times \frac{1}{16}\right) + \left(3 \times \frac{1}{8}\right)$

$= 2.75$ letters / menage

$$H = -\left[\frac{1}{4}\log\frac{1}{4} + \frac{1}{8}\log\frac{1}{8} + \frac{1}{16}\log\frac{1}{16} + \frac{1}{16}\log\frac{1}{16} + \frac{1}{16}\log\frac{1}{16} + \frac{1}{4}\log\frac{1}{4} + \frac{1}{16}\log\frac{1}{16} + \frac{1}{8}\log\frac{1}{8}\right]$$

$= 2.75$ bits / message

$\log_2 2 = 1$ bit / letter.

$$\% \text{ Efficiency} = \frac{2.75}{2.75 \times 1} \times 100\% = 100\%$$

$$\boxed{\% \eta = 100\%}$$

P) Apply the Shannon fano Coding procedure for the following message ensemble with probabilities as given

$$[x] = [x_1 \; x_2 \; x_3 \; x_4 \; x_5 \; x_6 \; x_7]$$

$$[P] = [0.4 \; 0.2 \; 0.12 \; 0.08 \; 0.08 \; 0.08 \; 0.04]$$

Take $M = 2$.

Sol. → Given message ensemble with probabilities

$$[x] = [x_1 \; x_2 \; x_3 \; x_4 \; x_5 \; x_6 \; x_7]$$

$$[P] = [0.4 \; 0.2 \; 0.12 \; 0.08 \; 0.08 \; 0.08 \; 0.04]$$

no. of symbols $M = 2$, no. of messages $N = 7$

→ In shannon fano Coding, first step is to arrange the messages in the decreasing order of their probabilities and second step is to assign codes via source partition such that less no. of bits are assigned to more probable messages and more no. of bits are assigned to less probable messages.

→ Coding Efficiency $\eta = \dfrac{H(x)}{L \log_2 M} \times 100\%$

where

Entropy $H(x) = -\sum_{K=1}^{7} P_K \log_2 P_K$ bits/message

Average Length $L = \sum_{K=1}^{7} n_K P_K$ letter/message

i) Partitioning Possibility 1st way

messages [x]	Probabilities [p]	Shannon fano Coding Via Source Partition	Code [c]	Length [n]
x_1	0.4	0 0	0 0	2
x_2	0.2	0 1	0 1	2
x_3	0.12	1 0 0	1 0 0	3
x_4	0.08	1 0 1	1 0 1	3
x_5	0.08	1 1 0	1 1 0	3
x_6	0.08	1 1 1 0	1 1 1 0	4
x_7	0.04	1 1 1 1	1 1 1 1	4

$$L = (2 \times 0.4) + (2 \times 0.2) + (3 \times 0.12) + (3 \times 0.08) + (3 \times 0.08) + (4 \times 0.08) + (4 \times 0.04)$$

$$= 2.52 \text{ letters / message}$$

ii) Partitioning possibility 2nd way

messages [x]	Probabilities [p]	Shannon fano Coding Via Source Partition	Code [c]	Length [n]
x_1	0.4	0	0	1
x_2	0.2	1 0 0	1 0 0	3
x_3	0.12	1 0 1	1 0 1	3
x_4	0.08	1 1 0 0	1 1 0 0	4
x_5	0.08	1 1 0 1	1 1 0 1	4
x_6	0.08	1 1 1 0	1 1 1 0	4
x_7	0.04	1 1 1 1	1 1 1 1	4

$$L = (1 \times 0.4) + (3 \times 0.2) + (3 \times 0.12) + (4 \times 0.08) + (4\ 0.08) + (4 \times 0.08) + (4 \times 0.04)$$

$$= 2.48 \text{ letters / message}$$

→ It can be seen that 2nd ... possible way is better to get high efficiency when compared to 1st way why because 2nd way has lower value of L.

$\therefore \quad L = 2.48$ letters / message

$$H(x) = -[(0.4 \log 0.4) + (0.2 \log 0.2) + (0.12 \log 0.12) + (0.08 \log 0.08) + (0.08 \log 0.08) + (0.08 \log 0.08 + 0.04 \log 0.04)]$$

$$= 2.42 \text{ bits / message}$$

no. of bits per letter $= \log_2 2 = 1$ bit / letter.

Coding efficiency $\eta = \dfrac{2.42}{2.48 \times 1} \times 100\%$

$= 97.6\%$

$\therefore \% \eta = 97.6\%$

Note: Sometimes Shannon fano Method is ambiguous due to the availability of more than one scheme of partitioning.

P) Apply the Shannon Fano Coding procedure for the following message ensemble with Probabilities as given

$[x] = [x_1 \; x_2 \; x_3 \; x_4 \; x_5 \; x_6 \; x_7]$

$[P] = [0.4 \; 0.2 \; 0.12 \; 0.08 \; 0.08 \; 0.08 \; 0.04]$

Take $M = 3$

Sol. → Given message ensemble with Probabilities

$[x] = [x_1 \; x_2 \; x_3 \; x_4 \; x_5 \; x_6 \; x_7]$

$[P] = [P_1 \; P_2 \; P_3 \; P_4 \; P_5 \; P_6 \; P_7]$

no. of symbols $M = 3$, no. of messages $N = 7$

→ In shannon fano Coding, first step is to arrange the messages in the decreasing order of their probabilities and second step is to assign codes via source partition (equal or approximately) such that less no. of bits are assigned to more probable messages and more no. of bits are assigned to less probable messages.

→ Coding Efficiency $\eta = \dfrac{H(x)}{L \log_2 M} \times 100\%$.

where Entropy $H(x) = -\sum_{K=1}^{7} P_K \log_2 P_K$ bits/message

Average length $L = \sum_{K=1}^{7} n_K P_K$ letter/message

As M = 3, Encoding alphabet be −1 0 1

messages [x]	Probabilities [P]	Shannon fano Coding via Source Partition	Code [c]	Length [n]
x_1	0.4	−1	−1	1
x_2	0.2	0 −1	0 −1	2
x_3	0.12	0 0	0 0	2
x_4	0.08	1 −1	1 −1	2
x_5	0.08	1 0	1 0	2
x_6	0.08	1 1 −1	1 1 −1	3
x_7	0.04	1 1 0	1 1 0	3

Here. $L = (1 \times 0.4) + (2 \times 0.2) + (2 \times 0.12) + (2 \times 0.08) + (2 \times 0.08) + (3 \times 0.08) + (3 \times 0.04)$

$= 1.72$ letters/message

$H(x) = -[(0.4 \log 0.4) + (0.2 \log 0.2) + (0.12 \log 0.12) + (0.08 \log 0.08) + (0.08 \log 0.08) + (0.08 \log 0.08) + (0.04 \log 0.04)]$

$= 2.42$ bits/message

$\log_2 3 = 1.58$ bits/letter

% Coding Efficiency $= \frac{2.42}{1.72 \times 1.58} 100\%$

$= 88.7\%$

$\% \eta = 88.7\%$

P) Apply the Shannon fano Coding procedure for the following message ensemble with probabilities given as $[X] = [x_1\ x_2\ x_3\ x_4]$ & $[P] = [\frac{1}{4}\ \frac{1}{8}\ \frac{1}{2}\ \frac{1}{8}]$ Take M = 2. find η. Also find P(0) and P(1) for natural binary Coding & Shannon fano Coding.

Sol: → Given message ensemble with probabilities

$[X] = [x_1\ x_2\ x_3\ x_4]$

$[P] = [\frac{1}{4}\ \frac{1}{8}\ \frac{1}{2}\ \frac{1}{8}] = [0.25\ 0.125\ 0.5\ 0.125]$

no. of symbols M = 2, no. of messages N = 4

→ In Shannon fano Coding first step is to arrange the messages in the decreasing order of their probabilities and second step is to assign Codes via source partition (equal or approximately) such that less no. of bits are assigned to more probable messages and more no. of bits are assigned to less probable messages.

→ Coding Efficiency $\eta = \frac{H(X)}{L \log_2 M} \times 100\ \%$

where Entropy $H(X) = -\sum_{K=1}^{4} P_K \log P_K$ bits/message

Average length $L = \sum_{K=1}^{4} n_K P_K$ letter/message

As M = 2, Encoding alphabet be 0, 1

messages [x]	Probabilities [P]	Shannon fano coding via Source Partition	Code [c]	Length [n]
x_3	0.5	0	0	1
x_1	0.25	1 0	1 0	2
x_2	0.125	1 1 0	1 1 0	3
x_4	0.125	1 1 1	1 1 1	3

Here $L = (1 \times 0.5) + (2 \times 0.25) + (3 \times 0.125) + (3 \times 0.125)$

$= 1.75$ letters / message

$= -[0.5 \log 0.5 + 0.25 \log 0.25 + 0.125 \log 0.125 + 0.125 \log 0.125] = 1.75$ bits / message

$\log_2^2 = 1$ bit / letter

coding efficiency $\eta = \frac{1.75}{1.75 \times 1} \times 100\%$

$= 100\%$

$\therefore \% \eta = 100\%$

messages [x]	Probabilities [P]	Natural Binary code [C]	Lengths [n]
x	0.5	0 0	2
x	0.25	0 1	2
x	0.125	1 0	2
x	0.125	1 1	2

Here $L = (2 \times \frac{1}{2}) + (2 \times \frac{1}{4}) + (2 \times \frac{1}{8}) + (2 \times \frac{1}{8}) = 2$ letters/message

Coding efficiency $\eta = \frac{1.75}{2 \times 1} \times 100\% = 87.5\%$

$$\therefore \% \eta = 87.5\%$$

Probabilities of '0' and '1' are

$$P(0) = \frac{\sum_{k=1}^{4} P_k C_{k0}}{\sum_{k=1}^{4} P_k n_k}, \quad P(1) = \frac{\sum_{k=1}^{4} P_k C_{k1}}{\sum_{k=1}^{4} P_k n_k}$$

Such that

$$P(0) + P(1) = 1$$

where C_{k0} and C_{k1} are the no. of '0's and '1's in the k^{th} code word respectively.

Natural Binary Coding :

$$P(0) = \frac{\left(\frac{1}{2}\times 2\right)+\left(\frac{1}{4}\times 1\right)+\left(\frac{1}{8}\times 1\right)+\left(\frac{1}{8}\times 0\right)}{\left(\frac{1}{2}\times 2\right)+\left(\frac{1}{4}\times 2\right)+\left(\frac{1}{4}\times 2\right)+\left(\frac{1}{8}\times 2\right)}$$

$$= \frac{\left(\frac{11}{8}\right)}{2} = \frac{11}{16}$$

$$P(1) = 1 - P(0) = 1 - \frac{11}{16} = \frac{5}{16}$$

$$P(0) = \frac{11}{16}, \quad P(1) = \frac{5}{16} \Rightarrow P(0) > P(1)$$

Shannon Fano Coding

$$P(0) = \frac{\left(\frac{1}{2}\times 1\right)+\left(\frac{1}{4}\times 1\right)+\left(\frac{1}{8}\times 1\right)+\left(\frac{1}{8}\times 0\right)}{\left(\frac{1}{2}\times 1\right)+\left(\frac{1}{4}\times 2\right)+\left(\frac{1}{8}\times 3\right)+\left(\frac{1}{8}\times 3\right)}$$

$$= \frac{\left(\frac{7}{8}\right)}{\left(\frac{7}{4}\right)} = \frac{4}{8} = \frac{1}{2}$$

$$P(1) = 1 - P(0) = 1 - \frac{1}{2} = \frac{1}{2}$$

$$P(0) = \frac{1}{2}, \quad P(1) = \frac{1}{2} \Rightarrow P(0) = P(1) = \frac{1}{2}$$

Hence it can be seen that Coding Efficiency increases when $P(0) = P(1) \Rightarrow$ L decreases

for natural binary coding $P(0) = \frac{11}{16}, P(1) = \frac{5}{16}$; $\eta = 87.5\%$.

& for Shannon fano Coding $P(0) = \frac{1}{2}, P(1) = \frac{1}{2}$, $\eta = 100\%$.

1. Prefix-Free Code

Definition:
A code is called prefix-free (or prefix code) if no codeword is a prefix of any other codeword in the set. In simpler terms, one codeword cannot appear as the beginning of another codeword.

Importance:
Prefix-free codes allow instantaneous decoding since the decoder can identify a codeword as soon as it is received without needing to check further bits.

Properties:

- Always uniquely decodable.
- Easy to represent using binary trees.
- Used in data compression (e.g., Huffman coding).

Example:
Let codewords be: A = 0, B = 10, C = 110, D = 111. None of these is a prefix of another. Hence, it's a prefix-free code.

Non-Example:
A = 0, B = 01. Here, A is a prefix of B, so it is not prefix-free.

2. Coding a Single Random Variable

Given a discrete memoryless source X with alphabet {x1, x2, ..., xn} and associated probabilities {P(x1), P(x2), ..., P(xn)}, the goal is to assign binary codewords such that the average codeword length is minimized.

Average codeword length:
$L = \sum P(x_i) * l_i$
Where l_i is the length of the codeword for symbol x_i.

Goal: Minimize L while maintaining uniquely decodable or prefix-free property.

3. Kraft's Inequality

Statement:
For any prefix-free code with codeword lengths l1, l2, ..., ln, the following inequality must hold:
$\sum 2^{(-li)} <= 1$

Implications:
- If the inequality is satisfied, then a prefix-free code exists for those lengths.
- If it’s violated, no prefix-free code is possible.

Example:
Lengths = {2, 3, 3}
2^(-2) + 2^(-3) + 2^(-3) = 1/4 + 1/8 + 1/8 = 1/2
Satisfies Kraft's inequality → A prefix-free code is possible.

4. Bounds on Optimal Code Length

To ensure efficient encoding, the codeword length l_i for symbol x_i should satisfy:
$\log_2(1/P(x_i)) \leq l_i < \log_2(1/P(x_i)) + 1$

Interpretation:
- The lower bound is the entropy contribution of the symbol.
- The upper bound ensures that the average length is within 1 bit of entropy.

Overall bound:
$H(X) \leq L < H(X) + 1$
Where $H(X) = -\sum P(x_i) \log_2 P(x_i)$

5. Rooted Tree with Probabilities

A rooted binary tree can represent prefix-free codes. Each leaf node corresponds to a symbol, and the path from the root to the leaf defines the codeword.

Construction:
- Start with the root node.
- Assign 0 to left branches and 1 to right branches.
- Traverse the tree to create codewords.

Example:
For symbols A (0.5), B (0.25), C (0.25):

```
      (*)
     /   \
   A(0)  (*)
        /   \
     B(10) C(11)
```

6. Shannon-Fano Coding

Steps:
1. List symbols in decreasing order of probability.
2. Divide list into two parts with approximately equal total probability.
3. Assign 0 to the first group, 1 to the second.
4. Repeat recursively for each group.

Example:

Symbol	Prob	Code
A	0.4	0

B | 0.3 | 10
C | 0.2 | 110
D | 0.1 | 111

Note: May not always give the optimal (shortest average length) code.

7. Fix-Free Code

Definition:
A fix-free code is a code in which no codeword is a prefix or suffix of another codeword.

Properties:
- Allows both forward and backward decoding.
- More restrictive than prefix-free codes.

Example:
Codewords: A = 01, B = 10, C = 111. None is a prefix or suffix of any other.

8. Coding an Information Source

Objective: Assign binary codewords to each symbol such that:
- Code is prefix-free.
- Average length is minimized.
- Use Huffman coding or similar techniques.

Steps:
1. Get source probabilities.
2. Use Huffman/Shannon-Fano algorithm.
3. Verify Kraft's inequality.
4. Compute average codeword length.

9. Huffman Coding

Definition:
Huffman coding is an algorithm that provides an optimal prefix-free binary code for a set of symbols based on their probabilities.

Algorithm Steps:
1. List all symbols with their probabilities.
2. Combine two symbols with the smallest probabilities.
3. Replace them with a node whose probability is the sum of the two.
4. Repeat until one node remains.
5. Assign 0 to one branch and 1 to the other recursively.

Example:
Symbols: A(0.4), B(0.2), C(0.2), D(0.1), E(0.1)
Step-by-step:
- Combine D and E → 0.1 + 0.1 = 0.2

- Combine B and C → 0.2 + 0.2 = 0.4
- Combine DE and BC → 0.2 + 0.4 = 0.6
- Combine A and DEBC → 0.4 + 0.6 = 1.0

Codes (example):
A = 0, B = 100, C = 101, D = 110, E = 111

Average Codeword Length:
L = 0.4(1) + 0.2(3) + 0.2(3) + 0.1(3) + 0.1(3) = 2.2 bits

VARIABLE TO BLOCK LENGTH CODING

➢ INTRODUCTION

In previous coding schemes (like Huffman coding), we convert **blocks of symbols → variable length codewords**.
In contrast, **Variable to Block Length Coding** works as:
Variable length input sequences → Fixed length output codewords
This technique is mainly used for:

- **Source coding (data compression)**
- Improving **coding efficiency**
- Used in **Tunstall coding**

➢ PROPER MESSAGE SET

Definition
A **Proper Message Set** is a collection of **source sequences (strings of symbols)** such that:

1. **No message is a prefix of another message**
2. The set is **complete and uniquely decodable**
3. It satisfies **prefix-free property**

Explanation
Instead of coding single symbols, we group symbols into **variable-length sequences** called **messages**.
Example:
Let source symbols be:
$\{A, B\}$

A possible proper message set:
$\{A, BA, BB\}$

Here:

- No message is a prefix of another
- Hence it is **prefix-free**

Important Property
A proper message set ensures:
Unique decodability

➢ K-ARY ROOTED TREE REPRESENTATION

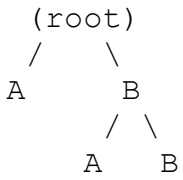
Concept
A **K-ary tree** is used to represent source symbols.

- If source has **K symbols → K-ary tree**
- Each branch corresponds to one symbol

Construction

1. Root represents **start**
2. Each branch represents a **symbol**
3. Each path represents a **message (sequence)**

Example (Binary Source: A, B)



Messages:

- A
- BA
- BB

➢ ASSIGNING PROBABILITIES TO K-ARY TREE

Rule

The probability of a node is:

$P(\text{node}) = \prod P(\text{symbols along path})$

Example

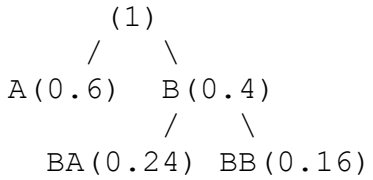
Let:

$P(A) = 0.6, P(B) = 0.4$

Then:

- P(A) = 0.6
- P(BA) = 0.4 × 0.6 = 0.24
- P(BB) = 0.4 × 0.4 = 0.16

Tree with Probabilities



➢ PREFIX-FREE CODING OF PROPER MESSAGE SET

Definition

Assign **fixed-length binary codewords** to each message such that:

- Codes are **prefix-free**
- Each message has **same code length**

Why Fixed Length?

Because:

Variable input → Fixed output

Example

Messages:

$\{A, BA, BB\}$

Assign 2-bit codewords:

Message	Code
A	00
BA	01
BB	10

Observation

- All codes have equal length
- Prefix-free condition satisfied

➢ TUNSTALL MESSAGE SET

Definition

A **Tunstall message set** is a **proper message set constructed optimally** such that:
Expected number of input symbols per codeword is maximized

Goal

Maximize:
Compression efficiency

Key Idea

- Expand **most probable node repeatedly**
- Generate **maximum number of leaves**
- Leaves form **message set**

➢ TUNSTALL CODING

Definition

Tunstall coding is a **variable-to-fixed length coding technique** that:

- Uses **fixed-length codewords**
- Maximizes **average input symbols per codeword**

➢ TUNSTALL CODING ALGORITHM (STEP-BY-STEP)

Step 1: Start with Root

Initialize tree with root node

Step 2: Expand Root

Create branches equal to number of symbols

Step 3: Find Most Probable Node

Select leaf node with **highest probability**

Step 4: Expand That Node

Replace it with K children

Step 5: Repeat

Continue until required number of leaves reached

Step 6: Assign Fixed-Length Codes

If number of leaves = 2^n, assign n-bit codewords

- **COMPLETE EXAMPLE OF TUNSTALL CODING**

Given:

$P(A) = 0.7, P(B) = 0.3$

Step 1: Initial Tree

```
(root)
```

Step 2: First Expansion

```
          (1)
         /   \
     A(0.7) B(0.3)
```

Step 3: Expand Highest Probability Node (A)

```
          (1)
         /   \
    A         B
   / \
AA    AB
```

Probabilities:

- AA = 0.49
- AB = 0.21
- B = 0.3

Step 4: Expand Highest Node (AA)

```
             (1)
            /   \
       A         B
      / \
    AA    AB
   / \
AAA   AAB
```

Probabilities:

- AAA = 0.343
- AAB = 0.147
- AB = 0.21
- B = 0.3

Step 5: Stop when desired leaves obtained

Assume 4 leaves needed:

$\{AAA, AAB, AB, B\}$

Step 6: Assign Fixed-Length Codes (2-bit)

Message	**Code**
AAA	00
AAB	01
AB	10
B	11

➢ ADVANTAGES OF TUNSTALL CODING

- Higher **compression efficiency**
- Fixed-length output → easier storage
- Suitable for **memory-based decoding**

➢ COMPARISON WITH HUFFMAN CODING

Feature	Huffman	Tunstall
Input	Fixed	Variable
Output	Variable	Fixed
Tree	Binary	K-ary
Goal	Minimize L	Maximize symbols/code

➢ IMPORTANT EXAM POINTS

- Proper message set → **prefix-free**
- Probability of sequence → **product rule**
- Tunstall expands → **highest probability node**
- Output code → **fixed length**

➢ SHORT NUMERICAL PROBLEM

Given:

$P(A) = 0.5, P(B) = 0.5$

Construct Tunstall code for 4 messages.

Solution (Outline):

- Expand root → A, B
- Expand A → AA, AB
- Leaves: {AA, AB, B}
- Expand B → BA, BB

Final set:

$\{AA, AB, BA, BB\}$

Assign 2-bit codes.

Asymptotic Equipartition Property (AEP)

Statement:
The **Asymptotic Equipartition Property (AEP)** states that for a large number of independent and identically distributed (i.i.d.) random variables, the sequence behaves in a predictable manner. Specifically, as the sequence length $n \to \infty$ most of the probability mass is concentrated on a small subset of all possible sequences (the **typical set**), and each sequence in this set has approximately equal probability.

For i.i.d. random variables $X_1, X_2, ..., X_n$ with entropy $H(X)$,

$$-\frac{1}{n} \log P(X_1, X_2, ..., X_n) \to H(X) \quad \text{with high probability as } n \to \infty$$

Properties of AEP:

1. **Concentration of probability:** As n increases, the probability distribution becomes sharply concentrated on the typical set.
2. **Approximate equiprobability:** All sequences in the typical set are nearly equally likely (equiprobable).
3. **Efficient representation:** Only about $2^{nH(X)}$ sequences have significant probability, out of $|\mathcal{X}|^n$ total possible sequences.
4. **Foundation for source coding theorem:** AEP provides the theoretical basis for data compression, showing we can represent data using roughly $nH(X)$ bits instead of $n \log |\mathcal{X}|$.

Hence AEP explains why **typical set** exists and why most sequences are concentrated

ii) Typical Set

Definition:
The **typical set** is the collection of sequences that occur with high probability and whose empirical information content is close to the source entropy.

the **strongly typical set** $A_\epsilon^{(n)}$ is defined as:

$$A_\epsilon^{(n)} = \left\{ x^n : \left| -\frac{1}{n} \log P(x^n) - H(X) \right| < \epsilon \right\}$$

where:

- $x^n = (x_1, x_2, \ldots, x_n)$ is a sequence of length n,
- $H(X)$ is the entropy,
- $\epsilon > 0$ is a small tolerance.

Properties of Typical Set:

1. **High Probability:**
   For large n, the typical set contains almost all the probability mass:
   $$P(A_\epsilon^{(n)}) \geq 1 - \delta, \quad \text{for small } \delta.$$
2. **Cardinality:**
   The number of sequences in the typical set is approximately:
   $$|A_\epsilon^{(n)}| \approx 2^{nH(X)}.$$
3. **Equiprobability:**
   Each sequence in the typical set has probability roughly:
   $$P(x^n) \approx 2^{-nH(X)}.$$
4. **Compression Basis:**
   Since almost all probability is concentrated in $2^{nH(X)}$ sequences, source coding can achieve compression rate close to $H(X)$.

Hence **Typical set** provides the practical structure for efficient data compression.

Weak Law of Large Numbers (WLLN)

Statement:

The **Weak Law of Large Numbers** states that the sample average of a large number of independent and identically distributed (i.i.d.) random variables converges **in probability** to the expected value (mean) of the random variable.

if $X_1, X_2, \ldots, X_n$ are i.i.d. random variables with finite mean $\mu = \mathbb{E}[X]$, then for every $\epsilon > 0$:

$$\lim_{n\to\infty} P\left(\left|\frac{1}{n}\sum_{i=1}^{n} X_i - \mu\right| \geq \epsilon\right) = 0$$

Interpretation:

As the number of trials n increases, the sample mean

$$\overline{X}_n = \frac{1}{n}\sum_{i=1}^{n} X_i$$

gets closer to the true mean μ with high probability.

"The average of many observations is close to the expected value."

- WLLN is a key foundation for the **Asymptotic Equipartition Property (AEP)**.
- In AEP, we apply WLLN to random variables of the form $Y_i = -\log P(X_i)$.
- By WLLN, the average information per symbol converges to the entropy $H(X)$.

✅ **Properties of WLLN (Weak Law of Large Numbers):**

1. The **sample mean** converges in probability to the true mean as $n \to \infty$.
2. The convergence depends only on the mean and variance of the random variable.
3. It holds for **i.i.d. random variables** with finite expected value.
4. Provides the **basis for AEP**, since the average information per symbol converges to entropy.

Example:

- Suppose we toss a fair coin ($X_i = 1$ for Head, 0 for Tail).
  - Expected value: $\mu = 0.5$.
  - If we toss $n = 10$ times, the average heads may not be exactly 0.5.
  - As $n \to 1000$, the fraction of heads will be very close to 0.5 with high probability.

Hence, WLLN Guarantees convergence in probability of sample mean to expectation

Chebyshev's Inequality

Statement:
Chebyshev's inequality gives an **upper bound** on the probability that a random variable deviates from its mean by more than a specified amount.

Let X be a random variable with **finite mean** $\mu = \mathbb{E}[X]$ and **finite variance** $\sigma^2 = \mathrm{Var}(X)$. For any $k > 0$:

$$P\left(|X - \mu| \geq k\sigma\right) \leq \frac{1}{k^2}$$

Interpretation:

- The probability that X lies more than k standard deviations away from its mean is at most $1/k^2$.
- In other words, **most values of a random variable are close to its mean.**

Example:

Suppose exam scores of students have mean $\mu = 50$ and standard deviation $\sigma = 10$.

- If we take $k = 2$:

$$P(|X - 50| \geq 20) \leq \frac{1}{2^2} = 0.25$$

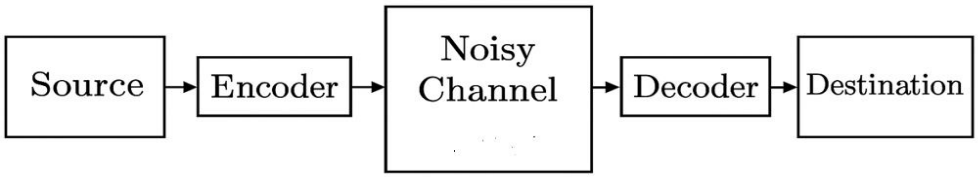
This means at most 25% of students can have scores outside the range $[30, 70]$.

Hence Chebyshev's inequality Bounds probability of deviation from mean and is the main tool to prove Weak Law of Large Numbers WLLN and Asymptotic Equipartition Property (AEP).

Noisy Channel Coding Theorem

Statement:

The **Noisy Channel Coding Theorem** gives the fundamental limit on reliable communication over a noisy channel.



- If information is transmitted over a channel with capacity C, then:
    1. **Achievability:** For any transmission rate R<C, there exists an encoding/decoding scheme such that the probability of error can be made **arbitrarily small** (as block length n→∞).
    2. **Converse:** For any transmission rate R>C, no coding scheme can achieve arbitrarily small error probability i.e errors are unavoidable.

Mathematical Form:

- Let the channel have capacity:

$$C = \max_{p(x)} I(X;Y)$$

  where $I(X;Y)$ = mutual information between input X and output Y.
- Then:
    - If $R < C$, reliable communication is possible.
    - If $R > C$, reliable communication is impossible.

Interpretation:

- The channel capacity CCC is the **maximum rate of information (bits per symbol or per second)** that can be transmitted with negligible error.
- It means **noise does not completely destroy communication** — as long as the coding rate is below capacity, we can design efficient error-correcting codes.

Example:

Binary Symmetric Channel (BSC) with crossover probability p:

- Capacity:

$$C = 1 - H(p)$$

  where $H(p) = -p \log_2 p - (1-p) \log_2 (1-p)$.
- If $p = 0.1$, then $C \approx 0.531$ bits per channel use.
- This means we can reliably transmit **up to 0.531 bits per symbol**.

Importance:

- Provides the **theoretical foundation of error-correcting codes**.
- Explains why coding methods (like convolutional codes, LDPC, turbo codes, polar codes) are useful.
- Forms the basis of **modern digital communication systems** (wireless, satellite, 5G/6G).

Hence The Noisy Channel Coding Theorem says: **Reliable communication over a noisy channel is possible if the coding rate is less than the channel capacity.**

Channel Capacity

Definition:

The channel capacity (C) is the maximum rate of information that can be transmitted through a channel with arbitrarily low probability of error.

for a discrete memoryless channel (DMC) with input X and output Y:

$$C = \max_{p(x)} I(X;Y)$$

where

I(X;Y) = **mutual information** between channel input X and output Y

p(x) = input probability distribution.

Interpretation:

- Channel capacity is like the "information carrying ability" of a channel.
- If transmission rate R<C: reliable communication is possible.
- If R>C: reliable communication is impossible (errors unavoidable).

Units:

- For discrete channels: **bits per channel use**.
- For continuous channels (like AWGN channel): **bits per second per Hertz (bps/Hz)**.

Examples:

1. **Binary Symmetric Channel (BSC):**
   - Crossover probability = p.
   - Capacity:

$$C = 1 - H(p)$$

     where $H(p) = -p \log_2 p - (1-p) \log_2 (1-p)$.
   - Example: If $p = 0.1$, $C \approx 0.531$ bits/use.

2. **Binary Erasure Channel (BEC):**
   - Erasure probability = ϵ.
   - Capacity:

$$C = 1 - \epsilon$$

   - Example: If $\epsilon = 0.2$, then $C = 0.8$ bits/use.

3. **Additive White Gaussian Noise (AWGN) Channel:**
   - With bandwidth B, signal power P, noise power spectral density N_0.
   - Capacity (Shannon formula):

$$C = B \log_2 \left(1 + \frac{P}{N_0 B}\right) \quad \text{bits/sec}$$

     where $\frac{P}{N_0 B}$ = **SNR (Signal-to-Noise Ratio)**.

Hence, channel capacity is the **upper bound** of reliable transmission rate and depends on channel type, noise characteristics, and coding scheme. It also provides foundation for the **Noisy Channel Coding Theorem**.

Rate–Distortion Theory

Definition:

Rate–Distortion Theory is a fundamental concept in Information Theory that deals with **lossy data compression**.

- It studies the **trade-off** between the **bit rate** (number of bits per symbol used to represent data) and the **distortion** (loss of fidelity or error due to compression).
- In simple words: *How much can we compress a source while still keeping the distortion below a certain level?*

Key Idea:

- Not all applications require exact reproduction (lossless).
- For example:
  - In audio, small noise is acceptable.
  - In images, minor quality loss is tolerable.
- Rate–Distortion Theory tells us the **minimum rate (bits per symbol)** needed to achieve a desired average distortion DDD.

Mathematical Formulation:

- Let the source random variable be X with distribution $p(x)$.
- The reproduction variable is $\hat{X}$.
- A distortion measure is defined as $d(x, \hat{x})$ (e.g., mean square error).
- The **Rate–Distortion Function** is:

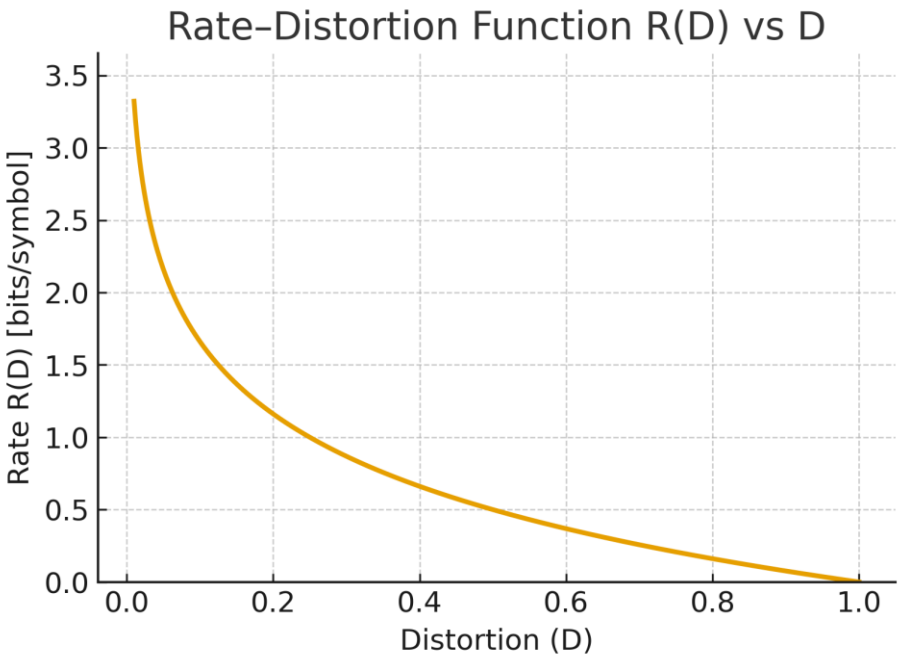
$$R(D) = \min_{p(\hat{x}|x):\, \mathbb{E}[d(X,\hat{X})] \le D} I(X;\hat{X})$$

where:

- $R(D)$ = minimum rate (bits per symbol) required for distortion ≤ D,
- $I(X;\hat{X})$ = mutual information between input and reproduction.

Properties

1. $R(D)$ is a **non-increasing function** of distortion D.
   - More distortion allowed → fewer bits needed.
2. If $D = 0$ (perfect reproduction), then $R(D) = H(X)$ (entropy).
3. If D is very large (don't care about accuracy), then $R(D) = 0$.



Example: Gaussian Source with MSE Distortion

- For a Gaussian source $X \sim \mathcal{N}(0, \sigma^2)$ and distortion measure = Mean Squared Error (MSE):

$$R(D) = \frac{1}{2} \log_2 \left(\frac{\sigma^2}{D} \right), \quad 0 < D < \sigma^2$$

Applications:

- **Image compression** (JPEG, WebP)
- **Audio compression** (MP3, AAC)
- **Video compression** (MPEG, H.264, HEVC)
- Wireless communication (rate–distortion optimized transmission)

Hence Rate–Distortion Theory says Minimum number of bits needed for compressing data while keeping distortion ≤ D. It provides the **fundamental limit of lossy compression**.

Differential Entropy

Definition:

Differential entropy is the continuous version of Shannon entropy, used for continuous random variables. It measures the "average uncertainty" of a continuous random variable. Unlike discrete entropy H(x), **differential entropy h(x) can be negative**. Its Units is bits if log base is 2.

For a continuous random variable X with probability density function (pdf) $f(x)$:

$$h(X) = -\int_{-\infty}^{\infty} f(x) \log f(x)\, dx$$

Example

If $X \sim \mathcal{N}(0, \sigma^2)$ (Gaussian with mean 0 and variance σ^2):

$$h(X) = \frac{1}{2} \log \left(2\pi e \sigma^2\right)$$

This shows Gaussian distribution has the **maximum entropy** among all distributions with the same variance.

Properties of Differential Entropy

1. **Continuous Case** – Defined for continuous random variables (unlike discrete entropy).
2. **Can be Negative** – Unlike discrete entropy, differential entropy may take negative values.
3. **Units** – Measured in **bits** if the logarithm is base 2, and in **nats** if natural log is used.
4. **Invariance to Translation** – If $Y = X + c$, then $h(Y) = h(X)$.
5. **Scaling** – If $Y = aX$, then $h(Y) = h(X) + \log |a|$.
6. **Gaussian Maximizes Entropy** – Among all distributions with the same variance, the Gaussian distribution has the maximum differential entropy.

Gaussian Channel

Definition:

A Gaussian channel is a communication channel where the noise is modeled as **Additive White Gaussian Noise (A**

The output of the channel is:

$$Y = X + N$$

where:

- X = input signal,
- $N \sim \mathcal{N}(0, N_0/2)$ = Gaussian noise with variance,
- Y = received signal.

Capacity of Gaussian Channel:

For bandwidth B, signal power P, and noise power spectral density N_0:

$$C = B \log_2 \left(1 + \frac{P}{N_0 B}\right) \quad \text{bits/sec}$$

where $\frac{P}{N_0 B}$ = **Signal-to-Noise Ratio (SNR)**.

Hence Capacity Additive White Gaussian Channel (AWGN) is

$$C = B \log_2(1 + SNR)$$

Blahut–Arimoto Algorithm

Blahut–Arimoto (BA) Algorithm is one of the most important algorithms in **Information Theory**, mainly used for: Computing **channel capacity** of a Discrete Memoryless Channel (DMC) and Solving R**ate–Distortion problems R(D)**

Computing Channel Capacity

The Blahut-Arimoto algorithm provides a way to find the channel capacity, C, of a DMC. This is the maximum mutual information, I(X;Y), between the input and output of a channel. The algorithm iteratively optimizes the input probability distribution, P(x), to maximize I(X;Y). Starting with an initial guess for the input distribution, the algorithm calculates a new, improved distribution and repeats the process until the values converge. The final converged value of I(X;Y) gives the channel capacity. The algorithm is guaranteed to converge to the optimal solution.

Solving Rate-Distortion Problems

The Blahut-Arimoto algorithm also finds the **rate-distortion function, R(D)**, for a source. The rate-distortion function represents the minimum number of bits per symbol (rate) required to encode a source with an average distortion no more than D. This is the reverse of the channel capacity problem, where you want to maximize information transfer. Here, the goal is to minimize the information rate while keeping the distortion below a certain threshold. The algorithm iteratively finds the optimal conditional probability distribution, P(y|x), that minimizes the mutual information I(X;Y) subject to a constraint on the average distortion. This function is crucial for understanding the fundamental limits of data compression with loss.

Problem Setup

Given:

- Input alphabet $X = \{x_1, x_2, \ldots, x_m\}$
- Output alphabet $Y = \{y_1, y_2, \ldots, y_n\}$
- Channel transition probabilities $P(y_j|x_i)$

We want to find the **capacity** $C = \max_{P_X} I(X;Y)$, where $I(X;Y)$ is the mutual information.

Steps of the Algorithm

1. **Initialize** input probabilities $p_i^{(0)} = P(x_i)$ (often uniform).
2. **Iterate** until convergence:

   a. Compute "output distribution" given input:

$$q_j^{(k)} = \sum_{i=1}^{m} p_i^{(k)} P(y_j|x_i)$$

   b. Compute the "log-likelihood ratio" for each input:

$$d_i^{(k)} = \sum_{j=1}^{n} P(y_j|x_i) \log \frac{P(y_j|x_i)}{q_j^{(k)}}$$

c. Update input probabilities:

$$p_i^{(k+1)} = \frac{p_i^{(k)} \cdot 2^{d_i^{(k)}}}{\sum_{i=1}^{m} p_i^{(k)} \cdot 2^{d_i^{(k)}}}$$

3. **Check convergence:**
   If $p_i^{(k+1)} \approx p_i^{(k)}$ (or mutual information change is small), stop.
4. **Compute capacity:**

$$C = \sum_{i=1}^{m} \sum_{j=1}^{n} p_i P(y_j|x_i) \log \frac{P(y_j|x_i)}{\sum_i p_i P(y_j|x_i)}$$

Example

Consider a **binary symmetric channel (BSC)** with crossover probability $p = 0.2$:

- $X = \{0, 1\}, Y = \{0, 1\}$
- Transition probabilities:

$$P(Y|X) = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}$$

1. Initialize $p_0 = p_1 = 0.5$
2. Compute q_j:

$$q_0 = 0.5 * 0.8 + 0.5 * 0.2 = 0.5, \quad q_1 = 0.5$$

3. Compute d_i:

$$d_0 = 0.8 \log_2(0.8/0.5) + 0.2 \log_2(0.2/0.5) \approx 0.2781$$

$$d_1 = 0.2 \log_2(0.2/0.5) + 0.8 \log_2(0.8/0.5) \approx 0.2781$$

4. Update p_i:

$$p_0^{new} = p_1^{new} = \frac{0.5 * 2^{0.2781}}{0.5 * 2^{0.2781} + 0.5 * 2^{0.2781}} = 0.5$$

5. **Convergence reached**, capacity:

$$C = 1 - H(p) = 1 - [-(0.8 \log_2 0.8 + 0.2 \log_2 0.2)] \approx 0.722 \text{ bits/channel use}$$

Example: Binary Symmetric Channel (BSC)

Given:

- Input: $X \in \{0, 1\}$
- Output: $Y \in \{0, 1\}$
- Crossover probability: p

Transition matrix:

$$P(y|x) = \begin{bmatrix} 1-p & p \\ p & 1-p \end{bmatrix}$$

Applying Blahut–Arimoto Algorithm

1. **Initialize input probabilities:**

$$p^{(0)}(x = 0) = p^{(0)}(x = 1) = 0.5$$

2. **Compute output distribution:**

$$q^{(0)}(y) = \sum_x p^{(0)}(x) P(y|x)$$

For uniform input:

$$q^{(0)}(0) = q^{(0)}(1) = 0.5$$

3. **Update input distribution:**

$$p^{(1)}(x) = \frac{p^{(0)}(x) \cdot 2^{d_x}}{\sum_x p^{(0)}(x) \cdot 2^{d_x}}$$

> Due to channel symmetry, the distribution stays uniform:

$$p(x) = 0.5, 0.5$$

4. **Compute mutual information:**

$$I(X;Y) = 1 - H(p)$$

where

$$H(p) = -p \log_2 p - (1-p) \log_2 (1-p)$$

Thus, the channel capacity for BSC is:

$$C = 1 - H(p)$$

Hence BA Algorithm is an iterative method to find **optimal input distribution** for maximizing mutual information.

- For symmetric channels (like BSC), the optimal distribution is uniform, so BAA confirms the known formula.

Universal Source Coding:

Universal Source Coding is a data compression technique that can efficiently compress data from a source without prior knowledge of the source's statistical properties or distribution. It aims to achieve optimal compression rates for a wide range of sources, making it a versatile and robust approach**.**

Properties:

- **No prior knowledge**: It does not require knowledge of the source's probability distribution or statistical properties.
- **Adaptability**: It can adapt to different types of data and sources.
- **Asymptotic optimality**: This schemes asymptotically achieves optimal compression rates as the amount of data increases.
-

Purpose

The primary purpose of universal source coding is to provide efficient data compression for a broad range of applications, without the need for specific knowledge about the data source. This makes it particularly useful in scenarios where the data characteristics are unknown, complex, or varying over time.

Examples:

- Lempel–Ziv family (LZ77, LZ78, LZW, LZSS, etc.)
- Arithmetic Coding (adaptive versions)
- Context Tree Weighting (CTW)

Applications

- General-purpose file compression (ZIP, gzip, 7z).
- Image formats (GIF, PNG, TIFF).
- Text/document compression.
- Basis for theoretical results in Information Theory.

Lempel–Ziv Family of Algorithms

Abraham **Lempel** and Jacob **Ziv**, two Israeli researchers, introduced the **Lempel–Ziv (LZ)** algorithm, which became the foundation for many modern **lossless data compression methods**. Their work established the principle of **dictionary-based, incremental parsing**, where an input sequence is parsed into non-overlapping phrases, each being the **shortest string not seen before**. They later developed improved versions: **LZ77** and **LZ78** are the two other foundational algorithms published in 1977 and 1978, respectively.

- **LZ (1977–78)** → *Dictionary-based, incremental parsing*
- **LZ77 (1977)** → *Sliding-window based compression*
- **LZ78 (1978)** → *Explicit dictionary-based coding*

Lempel–Ziv–Welch(LZW) Algorithm

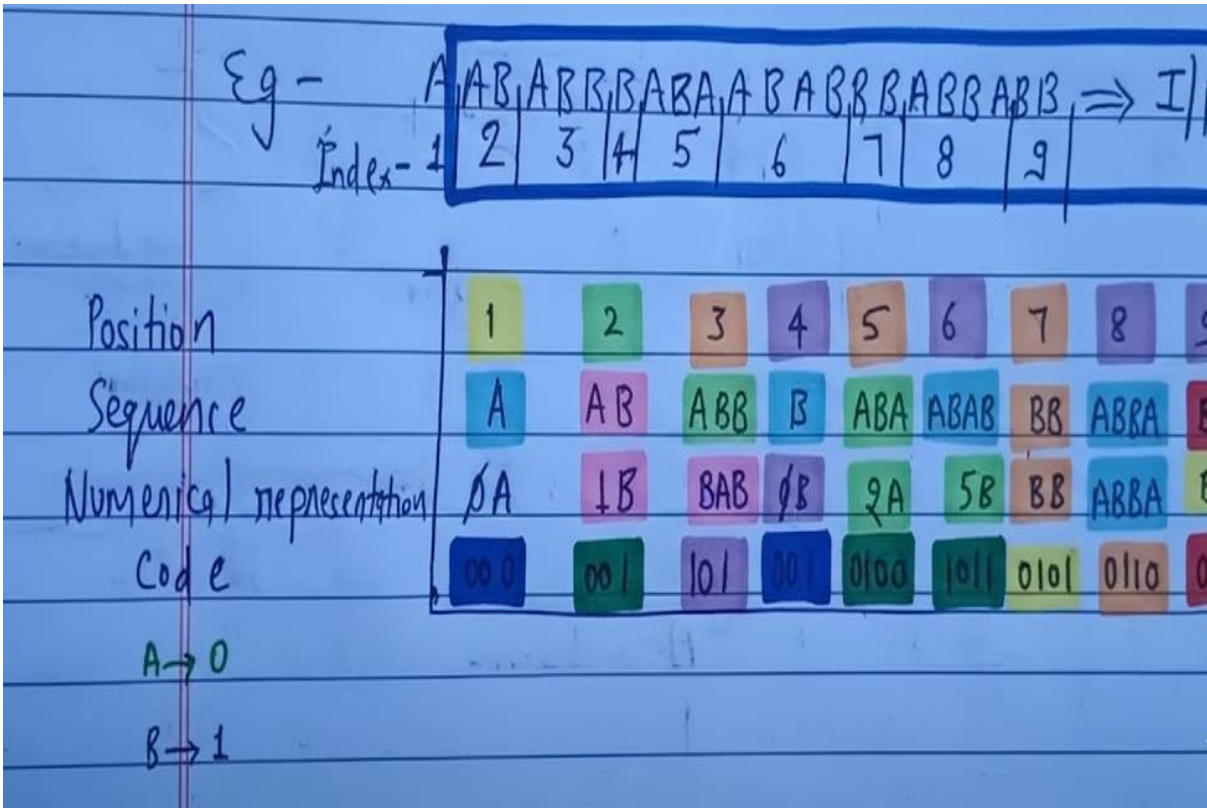
In 1984, **Terry Welch** extended **LZ78** and proposed the **Lempel–Ziv–Welch (LZW)** algorithm. It is a **universal, lossless compression algorithm**. LZW works on the principle of replacing repeated strings with dictionary indexes, while dynamically building the dictionary during compression.

- **LZW (1984)** → *Universal dictionary-based lossless compression*

LZ Algorithm:

Steps in General LZ Algorithm

1. **Initialize dictionary**
    - Start with an empty dictionary (or just the basic alphabet).
2. **Parsing the input sequence**
    - Read the input sequence or series from left to right.
    - Parse it into **non-overlapping phrases**, where each phrase is the **shortest string not seen before** in the dictionary.
3. **Dictionary update**
    - Add each new phrase to the dictionary as soon as it is discovered.
    - This ensures the dictionary grows adaptively with new patterns.
4. **Encoding**
    - Instead of writing the phrase itself, output a **reference (index/pointer) to the previous phrase + the next symbol**.
5. **Continue until end of input**
    - Repeat parsing, dictionary updating, and encoding until the entire input sequence is processed.
6. **Decoding** (reverse process)
    - Start with the same initial dictionary.
    - Reconstruct phrases from the encoded references.
    - Update dictionary in the same way as the encoder.



LZ77 Algorithm:

Steps in LZ77 Algorithm

1. **Initialize window**
    - Define a search buffer (past data) and look-ahead buffer (upcoming data).
2. **Parsing input**
    - Find the longest match between look-ahead buffer and search buffer.
3. **Encoding**
    - Output a triple: (Offset, Length of the match, Code word) i.e (O, L, C< >)
        - **Offset** → distance of the pointer from the look ahead buffer.
        - **Length of match**: the number of consecutive symbols in search buffer that match the consecutive symbols in the look ahead buffer starting with the first symbol
        - **Code word**: C is the code word corresponding to the symbol in the look ahead buffer that follows the match.
4. **Update window**
    - Slide the window forward by the length of the encoded phrase + 1.
5. **Repeat until end of input**
    - Continue matching, encoding, and sliding until input is finished.
6. **Decoding (reverse process)**
    - Rebuild sequence using triples by copying from previously decoded output.

LZW Algorithm

Steps in LZW Algorithm

Initialize dictionary
- Start dictionary with all possible single-character symbols

Parsing input
- Read the longest sequence (phrase) already in the dictionary.

Encoding
- Output the dictionary index of that sequence.

Dictionary update
- Add new entry = (sequence + next symbol) to dictionary.

Continue until end of input
- Repeat parsing, encoding, and dictionary updating for entire input.

Decoding (reverse process)
- Start with same initial dictionary.
- Replace each index with its dictionary entry.
- Update dictionary in same way as encoder.

Comparison of Universal Coding Schemes

Feature	LZ (General)	LZ77	LZ78	LZW
Year	1977–78	1977	1978	1984
Dictionary type	General concept	Sliding window (implicit)	Explicit dictionary (table)	Explicit dictionary (table)
Encoding output	General references	(Offset, Length, Symbol)	(Index, Symbol)	Index only
Dictionary growth	Adaptive parsing	Window slides (old data replaced)	Grows as new phrases are added	Grows as new phrases are added
Storage	No explicit dictionary	No explicit dictionary	Explicit dictionary needed	Explicit dictionary needed
Used in	Theory foundation	gzip, ZIP, PNG	Rarely used (basis for LZW)	GIF, TIFF, UNIX compress

LZ Encoding

Given that A=0 B=1

Input: Sequence	A	A	B	A	B	B	B	A	B	A

Parsing : Dividing Sequence into	A	AB	ABB	B	ABA
Giving Positions	1	2	3	4	5
Dictionary: Nemerical Representation	ϕA	1B	2B	ϕB	2A
Output: Digital Code (Prefix last bit)	0000	0011	0101	0001	0100

LZ Decoding

Given that A=0 B=1

Input: Digital Code (Prefix last bit)	0000	0011	0101	0001	0100
Dictionary: Nemerical Representation	ϕA	1B	2B	ϕB	2A
Giving Positions	1	2	3	4	5
Unparsing: Combining Segments for Sequence	A	AB	ABB	B	ABA

Output: Sequence	A	A	B	A	B	B	B	A	B	A

A	B	A	B	B	A	B	B	A	B	B

ABAB	BB	ABBA	BB
6	7	8	9=7
5B	4B	3A	4B
1011	1001	0110	1001

1011	1001	0110	1001
5B	4B	3A	4B
6	7	8	7=9
ABAB	BB	ABBA	BB

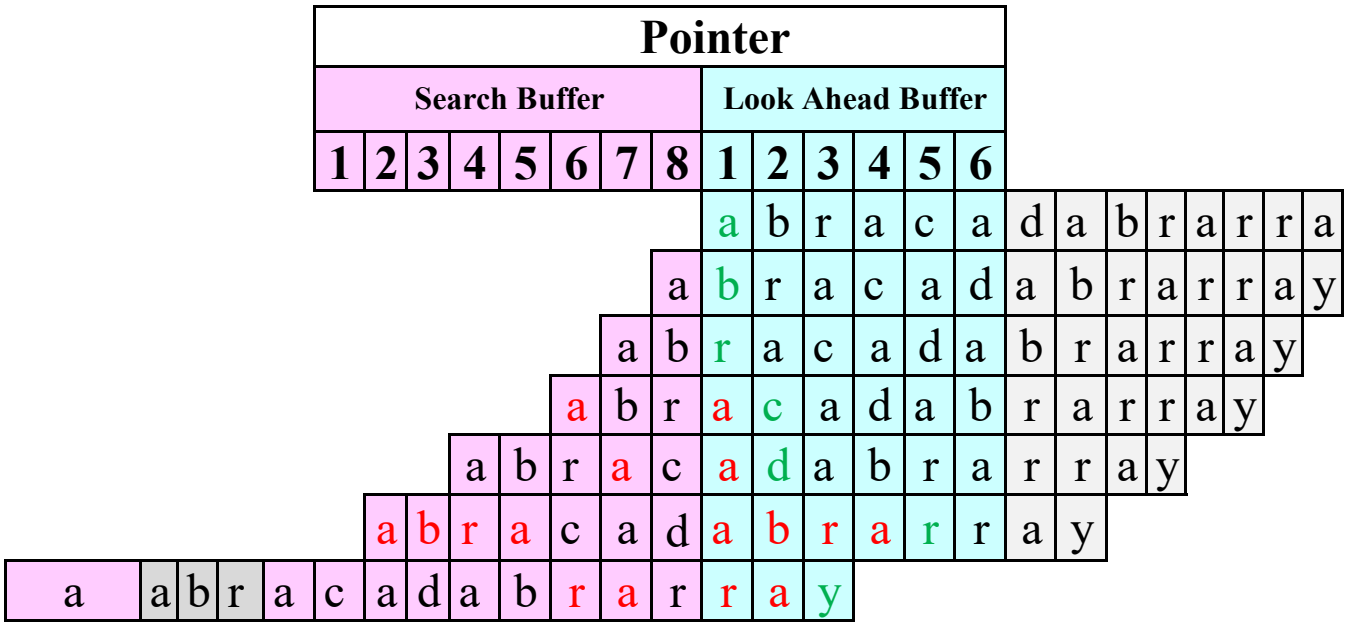
A	B	A	B	B	A	B	B	A	B	B

LZ7 Encoding

Given : Window Size=14, Look Ahead Buffer Size=6 and

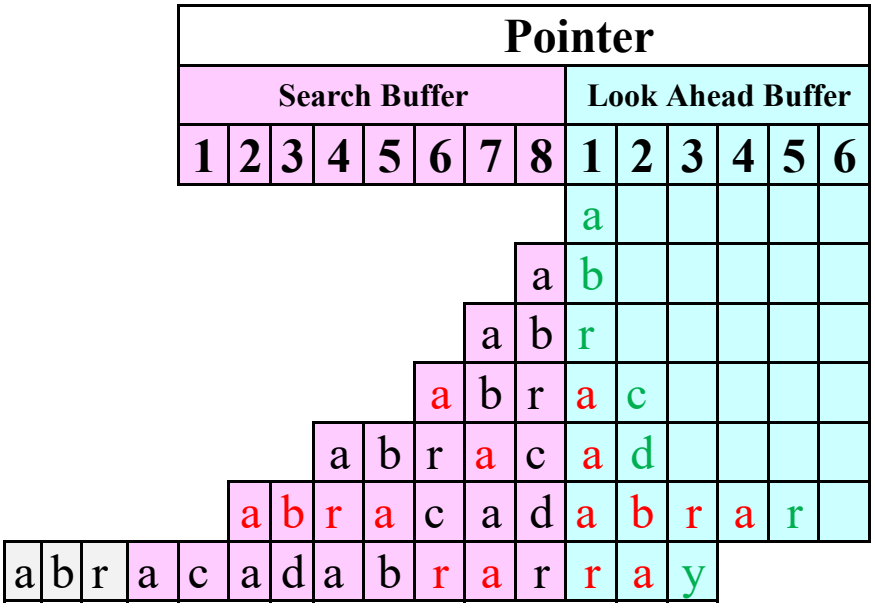
Input:

a	b	r	a	c	a	d	a	b	r	a	r	r	a	y



LZ7 Decoding

Given : Window Size=14, Look Ahead Buffer Size=6 and Code <(



Output:

a	b	r	a	c	a	d	a	b	r	a	r	r	a	y

Output: Code

<O,l,C(?)>

y

<0,0,C(a)>
<0,0,C(b)>
<0,0,C(r)>
<3,1,C(c)>
<2,1,C(d)>
<7,4,C(r)>
<3,2,C(y)>

O,l,C( ) >

Given

Input: Code

<O,l,C(?)>

<0,0,C(a)>
<0,0,C(b)>
<0,0,C(r)>
<3,1,C(c)>
<2,1,C(d)>
<7,4,C(r)>
<3,2,C(y)>

LZ Encoding

Given that

Input: Sequence			0	0	0	1	0	1	1

Parsing : Dividing Sequence into Segements	0	1	00	01	011
Giving Positions	1	2	3	4	5
Dictionary: Nemerical Representation	1	2	10	11	41
Output: Digital Code (Prefix last bit)	0000	0001	0010	0011	1001

LZ Decoding

Given that

Input: Digital Code (Prefix last bit)	0000	0001	0010	0011	1001
Dictionary: Nemerical Representation	1	2	10	11	41
Giving Positions	1	2	3	4	5
Unparsing: Combining Segments for Sequence	0	1	00	01	011

Output: Sequence	0	0	0	1	0	1	1

1	0	0	1	0	1	0	0	1	0	1

10	010	100	101
6	7	8	9
20	40	60	61
0100	1000	1100	1101

0100	1000	1100	1101
20	40	60	61
6	7	8	9
10	010	100	101

1	0	0	1	0	1	0	0	1	0	1

LZ Encoding

Given that

Input: Sequence	0	0	0	1	0	1	1	1	0
Parsing : Dividing Sequence into Segements	0	00	1	01	11				
Giving Positions	1	2	3	4	5				
Dictionary: Nemerical Representation	ϕ0	10	ϕ1	11	31				
Output: Digital Code (Prefix last bit)	0000	0010	0001	0011	0111				

LZ Decoding

Given that

Input: Digital Code (Prefix last bit)	0000	0010	0001	0011	0111				
Dictionary: Nemerical Representation	ϕ0	10	ϕ1	11	31				
Giving Positions	1	2	3	4	5				
Unparsing: Combining Segments for Sequence	0	00	1	01	11				
Output: Sequence	0	0	0	1	0	1	1	1	0

0	1	0	1	0	0	1	0	1

001	010	0101
6	7	8
21	40	71
0101	1000	1111

0101	1000	1111
21	40	71
6	7	8
001	010	0101

0	1	0	1	0	0	1	0	1

Q) Encode the given data by using LZW Algorithm

“w a b b a b w a b b a b w a b b a b w a b b a b w”

Ans. LZW Encoding: From the given data **Initial dictionary is {w, a, b}.**

Step	Current Symbol CS	Next Symbol NS	Is CS+NS in Dictionary?	Dictionary Entry (code → string)	Output code
–	-	–	–	1:w, 2:a, 3:b	–
1	w	a	wa no	4 → wa	1
2	a	b	ab no	5 → ab	2
3	b	b	bb no	6 → bb	3
4	b	a	ba no	7 → ba	3
	a	b	ab yes	Repeated	–
5	ab	w	abw no	8 → abw	5
	w	a	wa yes	Repeated	–
6	wa	b	wab no	9 → wab	4
	b	b	bb yes	Repeated	–
7	bb	a	bba no	10 → bba	6
	a	b	ab yes	Repeated	–
	ab	w	abw yes	Repeated	–
8	abw	a	abwa no	11 → abwa	8
	a	b	ab yes	Repeated	–
9	ab	b	abb no	12 → abb	5
	b	a	ba yes	Repeated	–
10	ba	b	bab no	13 → bab	7
11	b	w	bw no	14 → bw	3
	w	a	wa yes	Repeated	–
	wa	b	wab yes	Repeated	–
12	wab	b	wabb no	15 → wabb	9
	b	a	ba yes	Repeated	–
	ba	b	bab yes	Repeated	–
13	bab	w	babw no	16 → babw	13
End	w	–	–	–	1

The LZW encoded sequence for the given data is
“w a b b a b w a b b a b w a b b a b w a b b a b w” is
{1, 2, 3, 3, 5, 4, 6, 8, 5, 7, 3, 9, 13, 1}

LZW Decoding: For Encoded Sequence {1, 2, 3, 3, 5, 4, 6, 8, 5, 7, 3, 9, 13, 1} with **Initial dictionary is {w, a, b}.**

Encoded Sequence Entry	Dictionary Entry
1.	w
2.	a
3.	b
4.	wa
5.	ab
6.	bb
7.	ba
8.	abw
9.	wab
10.	bba
11.	abwa
12.	abb
13.	bab

The LZW decoded data for given encoded sequence {1, 2, 3, 3, 5, 4, 6, 8, 5, 7, 3, 9, 13, 1}is
"w a b b a b w a b b a b w a b b a b w a b b a b w" is

Error Probability with Repetition in the Binary Symmetric Channel (BSC)

In a digital communication system, the **Binary Symmetric Channel (BSC)** is the simplest model used to represent the effect of random noise.
Each transmitted bit has a probability **p** of being received in error and a probability **(1 – p)** of being received correctly.

- ## Concept of Repetition Coding

To reduce the probability of error, each bit of the message is **repeated M times** before transmission.
At the receiver, **majority voting** is applied—if more than half of the received bits are 1, the output is decided as 1; otherwise, 0.

- ## Error Probability Derivation

Let the transmitted bit be '1'.
An error occurs when the receiver decides '0', which happens if more than half of the bits are received incorrectly.

$$P_e = \sum_{i=\frac{M+1}{2}}^{M} \binom{M}{i} p^i (1-p)^{M-i}$$

For **M = 3**,

$$P_e = 3p^2(1-p) + p^3 = 3p^2 - 2p^3$$

- ## Example Calculation

If $p = 0.1$:

$$P_e = 3(0.1)^2 - 2(0.1)^3 = 0.028$$

Thus, only 2.8% error probability instead of 10% without coding.

- ## Figures

Figure 1: *Binary Symmetric Channel*

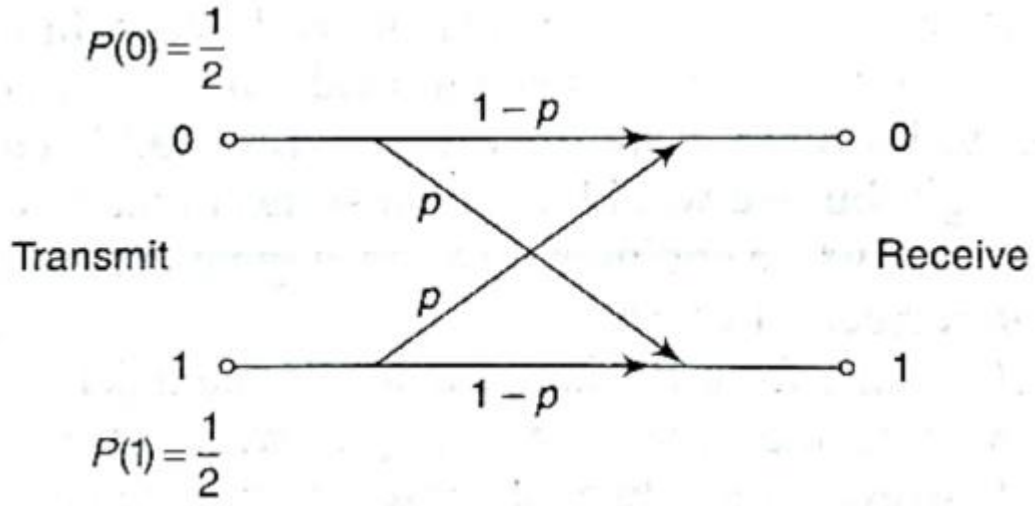


Fig. Binary Symmetric Channel

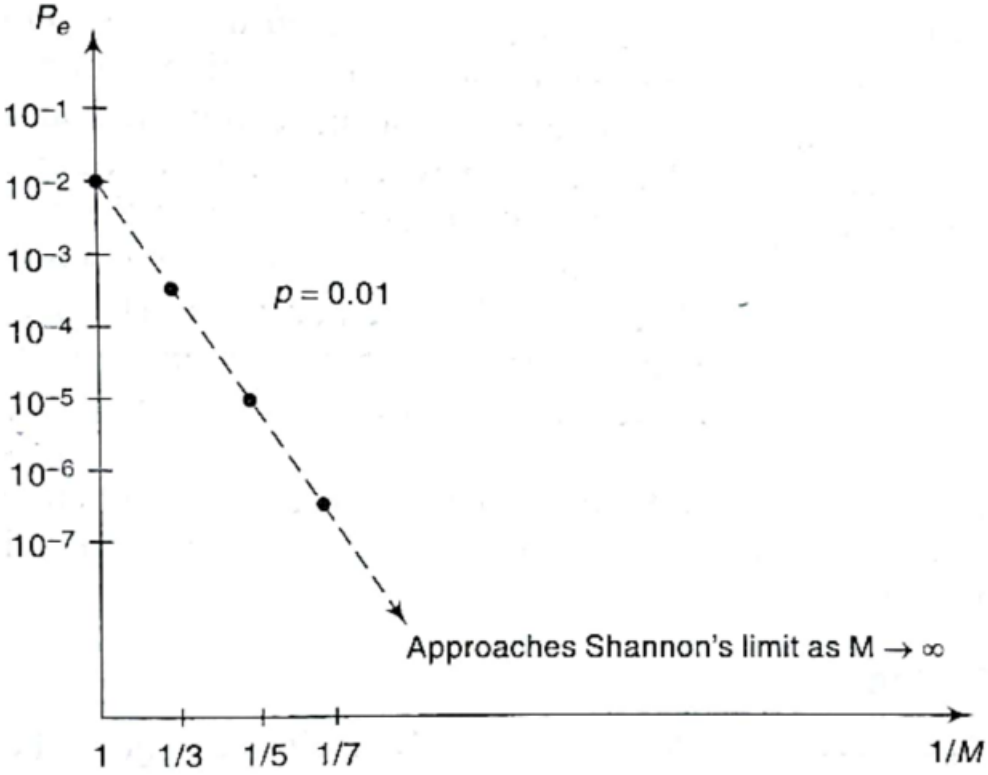
As each bit flips with probability **p**. then *Repetition Code (M=3)*

```
Message: 1 → Codeword: 111
Received: 110 → Majority → 1 (correct)
```

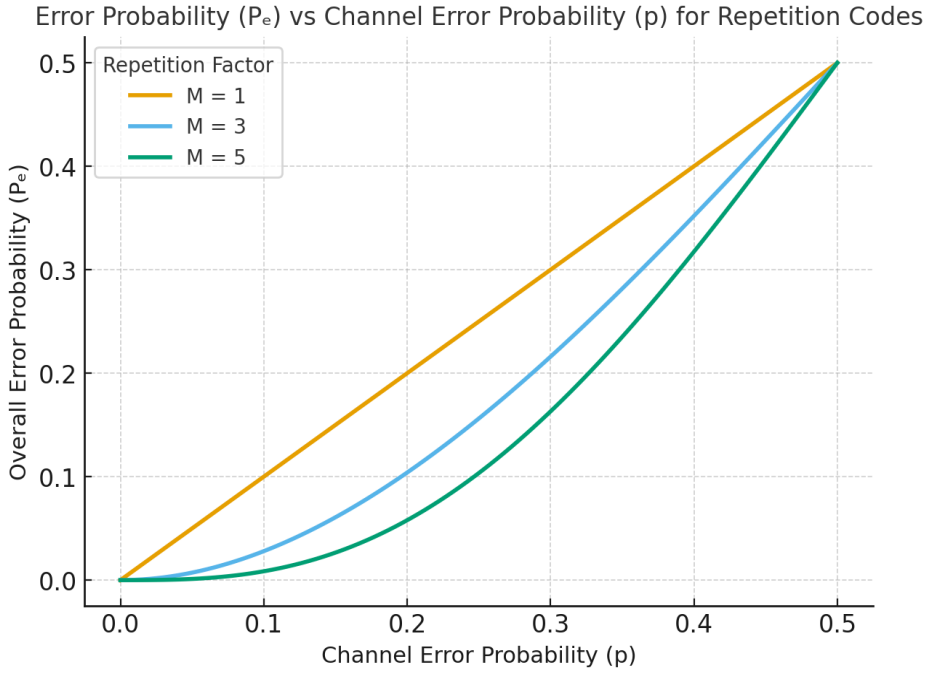
Figure 3: *Error Probability vs p Graph*
A graph of $\mathbf{P_e}$ (vertical) vs **p** (horizontal) for M = 1, 3, 5 shows that

- For small p, curves with higher M lie much lower.

- Increasing M reduces error probability exponentially.



Plot of P_e of BSC as a function of number of repetitions M.



Here's the **colored graph** showing the relationship between the **channel error probability (p)** and the **overall error probability (P_e)** for different repetition factors (**M = 1, 3, 5**).

It can clearly observe that as **M increases**, the error probability decreases sharply —

demonstrating how repetition coding improves reliability in a **Binary Symmetric Channel (BSC)**.

- ## Conclusion

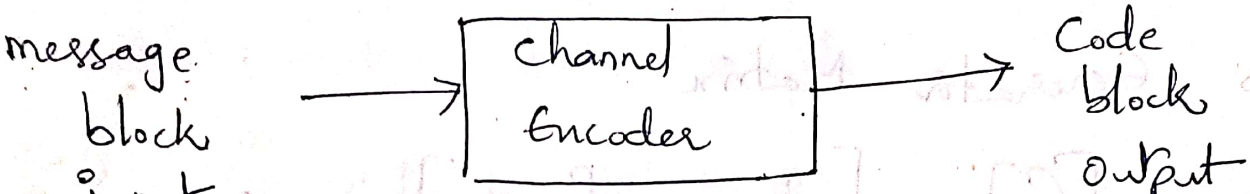
Repetition coding is a simple and effective method to reduce bit errors in the BSC. Although it improves reliability, it decreases **bandwidth efficiency** since more bits are transmitted per message bit.

Here's the **colored graph** showing the relationship between the **channel error probability (p)** and the **overall error probability (P_e)** for different repetition factors (**M = 1, 3, 5**). You can clearly observe that as **M increases**, the error probability decreases sharply — demonstrating how repetition coding improves reliability in a **Binary Symmetric Channel (BSC)**.

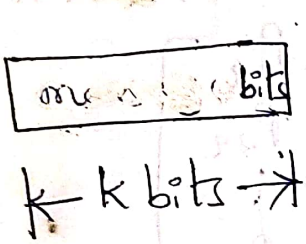
Would you like me to place this graph neatly into a **formatted PDF version** of your 300-word notes (with figures, captions, and color chart)?

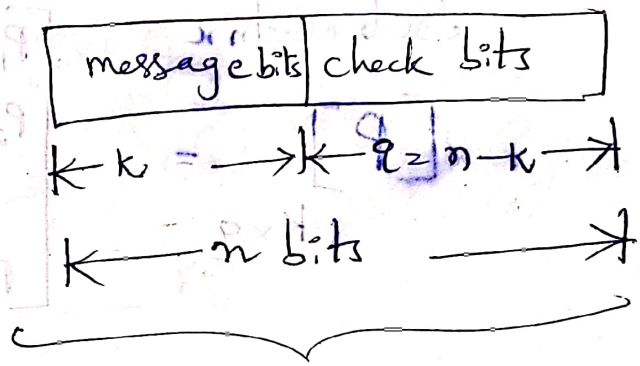
UNIT- V

Linear Block Codes



Block diagram





Various representations

⇒ message Vector

$$[M]_{1\times k} = [m_1\ m_2\ m_3\ \cdots\ m_k]_{1\times k}$$

Note: $[d_1\ d_2\ d_3\ \cdots\ d_k]_{1\times k}$ Sometimes

⇒ Check Vector.

$$[C]_{1\times q} = [c_1\ c_2\ c_3\ \cdots\ c_q]_{1\times q}$$

⇒ Code Vector Transmitted

$$[X]_{1\times n} = \left[\begin{array}{c|c} M & C \\ 1\times k & k\times q \end{array}\right]_{1\times n}$$

$$= [m_1, m_2, m_3, \cdots m_k, c_1, c_2\ c_3, \cdots c_q]_{1\times n}$$

$\Rightarrow$ $[x]_{1\times n} = [M]_{1\times k} \cdot [G]_{k\times n}$ where

$\Rightarrow$ Generator Matrix

$$[G]_{k\times n} = [\,I_{k\times k} \mid P_{k\times q}\,]_{k\times n} \quad \because n = k+q$$

Here $I_{k\times k}$ = Identity matrix of size $k\times k$ and

Sub-matrix

$$[P]_{k\times q} = \begin{bmatrix} P_{11} & P_{12} & P_{13} & \cdots & P_{1q} \\ P_{21} & P_{22} & P_{23} & \cdots & P_{2q} \\ \vdots & \vdots & \vdots & & \vdots \\ P_{k1} & P_{k2} & P_{k3} & \cdots & P_{kq} \end{bmatrix}_{k\times q}$$

$\Rightarrow$ Check Vector

$$[C]_{1\times q} = [M]_{1\times k}\,[P]_{k\times q}$$

$$[C_1\ C_2\ C_3\ \cdots\ C_q]_{1\times q} = [m_1\ m_2\ m_3\ \cdots\ m_k]_{1\times k} \cdot \begin{bmatrix} P_{11} & P_{12} & P_{13} & \cdots & P_{1q} \\ P_{12} & P_{22} & P_{23} & \cdots & P_{2q} \\ \vdots & \vdots & \vdots & & \vdots \\ P_{k1} & P_{k2} & P_{k3} & \cdots & P_{kq} \end{bmatrix}_{k\times q}$$

$\therefore$ that is

$$C_1 = m_1 P_{11} \oplus m_2 P_{12} \oplus m_3 P_{13} \oplus \cdots \oplus m_k P_{k1}$$
$$C_2 = m_1 P_{12} \oplus m_2 P_{22} \oplus m_3 P_{23} \oplus \cdots \oplus m_k P_{k2}$$
$$C_3 = m_1 P_{13} \oplus m_2 P_{32} \oplus m_3 P_{33} \oplus \cdots \oplus m_k P_{k3}$$
$$C_q = m_1 P_{1q} \oplus m_2 P_{2q} \oplus m_3 P_{3q} \oplus \cdots \oplus m_k P_{kq}$$

$\rightarrow$ Here all additions are modulo-2 (or mod 2 additions. ie XOR operation.

→ modulo 2 additions ⇒

$0 \oplus 0 = 0 \qquad 1 \oplus 1 = 0$

$0 \oplus 1 = 1 \qquad 1 \oplus 0 = 1$

→ Parity Check Matrix

$$[H]_{q \times n} = \left[P^T_{q \times k} \;\vdots\; I_{q \times q} \right]_{q \times n} \quad \because n = k+q$$

where P^T is Transpose of Submatrix P i.e

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

and

$I_{q \times q}$ is Identity Matrix of size $q \times q$

→ for a Systematic Linear Block Code

$$[G]_{k \times n} [H^T]_{n \times q} = [0]_{k \times q} \quad \text{and}$$

$$[H]_{q \times n} [G^T]_{n \times k} = [0]_{q \times k}.$$

i.e $G H^T = H G^T = 0$

→ Weight of Code Vector

W(x) is Numbe of nonzero elements in x

→ minimum distance between code Vectors

$$d_{min} = [W(x)]_{min} \; ; \; x \neq [0\ 0\ 0 - 0]$$

→ error detection

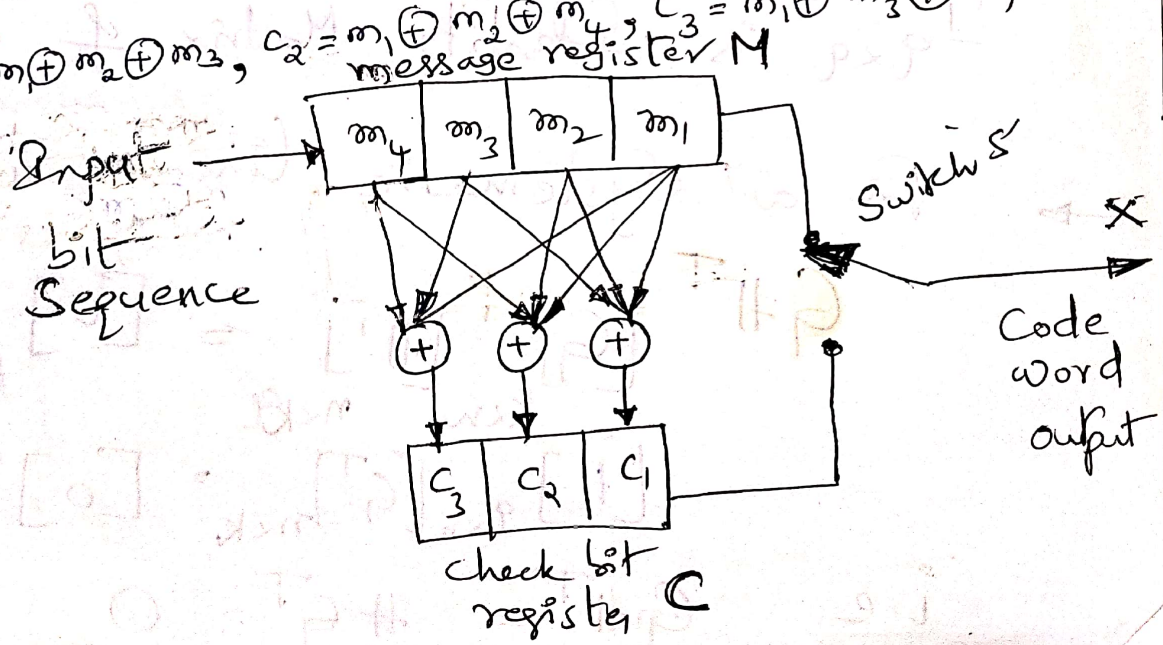
If $d_{min} \geq S+1$ then 's' errors will be detected

→ error detection and Correction

If $d_{min} \geq 2t+1$ then 't' errors will be corrected

→ Encoder for (7,4) Hamming Code (or) Linear block code

Ex: $c_1 = m_1 \oplus m_2 \oplus m_3$, $c_2 = m_1 \oplus m_2 \oplus m_4$, $c_3 = m_1 \oplus m_3 \oplus m_4$



Syndrome decoding

If $[X]_{1\times n}$ is transmitted code vector and $[Y]_{1\times n}$ is Received Code Vector then

$X = Y$ if there are no transmission errors

$X \neq Y$ if there are transmission errors ie 0 as 1 & 1 as 0

for a linear block code,

Parity check matrix is

$$[H]_{q\times n} = \left[P^T_{q\times k} \mid I_{q\times q} \right]_{q\times n}$$ and

its Transpose matrix is

$$\left[H^T\right]_{n\times q} = \left[\begin{array}{c} P_{k\times q} \\ \hline I_{q\times q} \end{array}\right]_{n\times q}$$

Important Property used in Syndrome decoding is

$$[X]_{1\times n}\,[H^T]_{n\times q} = [0]_{1\times q}$$

$$= [0\ 0\ 0\ \cdots\ 0]_{1\times q}$$

Note: $XH^T = 0$ is true for all Code Vectors.

→ Syndrome(S) : when some errors are present in received Code Vector Y then $Y H^T \neq 0$. and

Syndrome Vector is $[S]_{kq} = [Y]_{1xn} \cdot [H^T]_{nxq}$

→ Let E be the error vector then

$$[E]_{1xn} = [X] \oplus [Y]_{1xn} \Rightarrow E = X \oplus Y$$

Ex:

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

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

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

Now $[X]_{1xn} = [Y]_{1xn} \oplus [E]_{1xn} \Rightarrow X = Y \oplus E$

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

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

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

→ Relationship between Syndrome Vector S and Error Vector E.

$$S = Y H^T$$

$$= [X \oplus E] H^T$$

$$= X H^T \oplus E H^T$$

$$= 0 \oplus E H^T \quad \because X H^T = 0$$

Hence $\boxed{S = E H^T}$

S: $1 \times q$, E: $1 \times n$, H^T: $n \times q$

ie $[S]_{1\times q} = [E]_{1\times n} [H^T]_{n\times q}$

ie 'q' bits of Syndrome represents 2^q Syndrome Vectors and depends only on Error Pattern but not upon message bits.

→ Error Pattern for Single bit errors

Ex: If $[H]_{3\times7} = \left[\begin{array}{cccc:ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right]_{3\times7}$

If $[H]_{q\times n} = [H]_{3\times7} \Rightarrow$ $n = 7$, $q = 3$, $k = 4$

As $q = 3$, Syndrome Vector Size is $1\times q = 1\times3$ and no. of Syndrome vectors is $2^q - 1 = 2^3 - 1 = 7$

S.No	Bit in error	Error Vector [E]
1	1st	1 0 0 0 0 0 0
2	2nd	0 1 0 0 0 0 0
3	3rd	0 0 1 0 0 0 0
4	4th	0 0 0 1 0 0 0
5	5th	0 0 0 0 1 0 0
6	6th	0 0 0 0 0 1 0
7	7th	0 0 0 0 0 0 1

here

$$[H^T]_{7\times3} = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{7\times3}$$

Now $S = EH^T$

Syndrome Vector for first bit in error

$$S = [1\ 0\ 0\ 0\ 0\ 0\ 0]_{1\times 7} \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{7\times 3} = [1\ 0\ 1]_{1\times 3}$$

Syndrome Vector for Second bit in error

$$S = [0\ 1\ 0\ 0\ 0\ 0\ 0]_{1\times 7} \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{7\times 3} = [1\ 1\ 1]_{1\times 3}$$

Similarly Syndrome Vectors for remaining bits in error are

$[1\ 1\ 0]$, $[0\ 1\ 1]$, $[1\ 0\ 0]$, $[0\ 1\ 0]$, $[0\ 0\ 1]$.

$$\therefore S = \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{7\times 3} = H^T \Rightarrow$$

Syndrome Vectors are rows of H^T & Vice Versa.

→ Error correction using Syndrome Vector.

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

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

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

$S = YH^T$

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

$= [1\oplus0\oplus1\oplus0\oplus1\oplus0\oplus0 \quad 0\oplus0\oplus1\oplus1\oplus0\oplus1\oplus0$

$1\oplus0\oplus0\oplus1\oplus0\oplus0\oplus0] = [1\ 1\ 0]$

→ As $[1\ 1\ 0]$ is 3rd row in H^T, 3rd bit is in error

$\therefore E = [0\ 0\ 1\ 0\ 0\ 0\ 0]$ (11)

Now Corrected Vector

$$X = Y \oplus E$$

$$= [1\ 0\ 1\ 1\ 1\ 1\ 0] \oplus [0\ 0\ 1\ 0\ 0\ 0\ 0]$$

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

→ If double errors occurs in Y.

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

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

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

gives false indication that error is in 1st bit

Hence unable to detect and correct double errors. For this extended Hamming codes are used by adding one more bit extra.

Hamming Bound

For (n, k) block code there are $2^q - 1$ distinct non-zero syndromes. There are

$n C_1 = n$ single error patterns

$n C_2$ double error patterns

$n C_3$ tripple error patterns and so on.

Hence to correct t errors per word the following relation should be satisfied. ie

$$2^q - 1 \geq n C_1 + n C_2 + n C_3 + \cdots + n C_t$$

Now

$$2^q \geq 1 + n C_1 + n C_2 + n C_3 + \cdots + n C_t$$

$$2^{n-k} \geq \sum_{i=0}^{t} n C_i \qquad \because n C_0 = 1$$

& $q = n - k$

By taking logarithm to base 2 on both sides

$$\log_2 2^{n-k} \geq \log_2 \sum_{i=0}^{t} n C_i$$

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

dividing with n on both sides

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

$$1 - r \geq \frac{1}{n} \log_2 \sum_{i=1}^{t} n C_t$$

This eq is Hamming Bound

→ All error pattern that differ by a codeword have the same : Syndrome

Let X_1 & X_2 be two Code Vectors and error be introduced in first bit (MSB) ie for 7 bit vector Error Vector

$$E = [1\ 0\ 0\ 0\ 0\ 0\ 0].$$

Let Y_1 & Y_2 be Received Code Vectors

then Syndrome for Y_1 is

$$S_1 = Y_1 H^T$$

$$= (X_1 \oplus E) H^T = X_1 H^T \oplus E H^T$$

$$= E H^T \quad \because X_1 H^T = 0$$

and Syndrome for Y_2 is

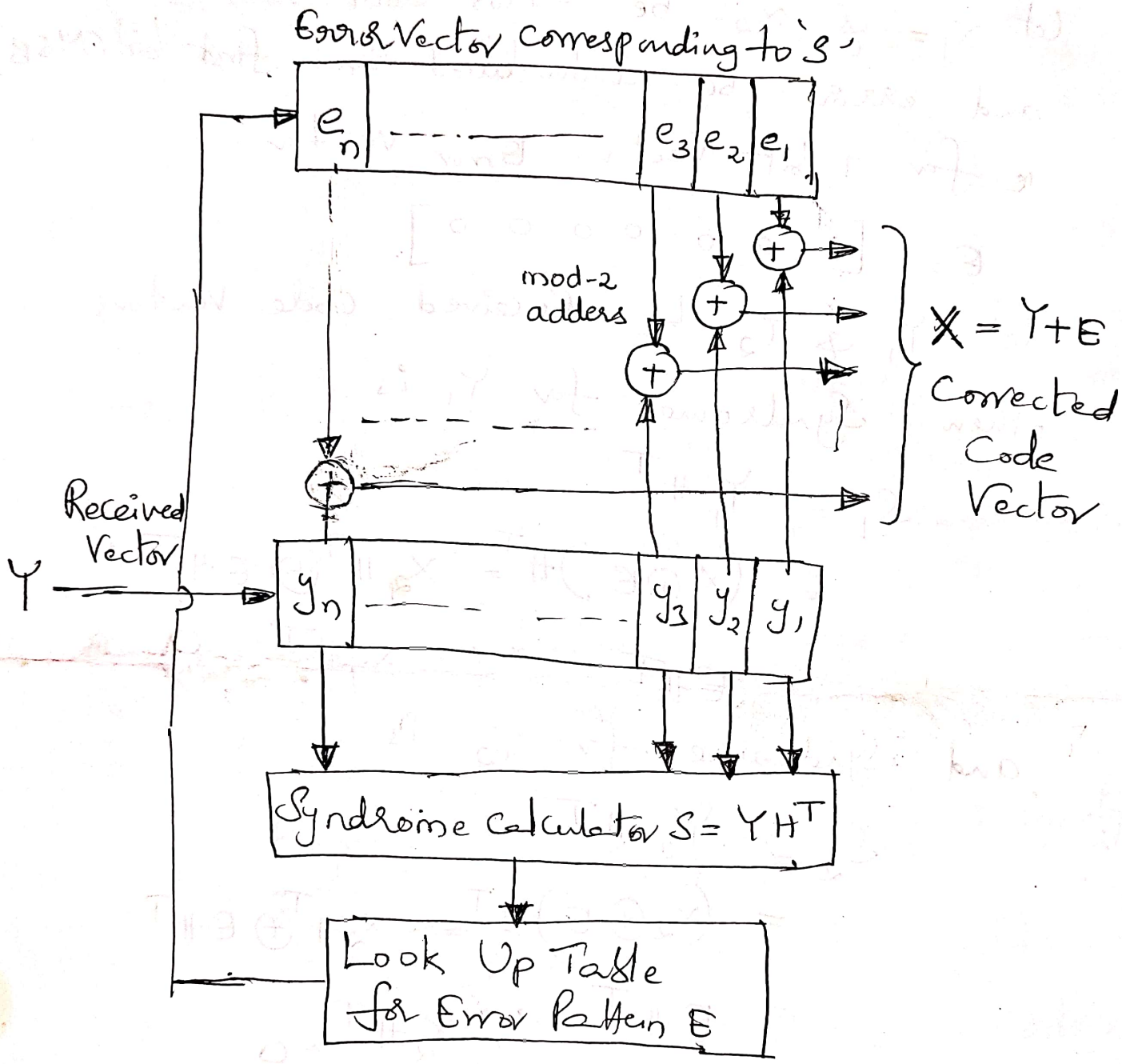
$$S_2 = Y_2 H^T$$

$$= (X_2 \oplus E) H^T = X_2 H^T \oplus E H^T$$

$$= E H^T \quad \because X_2 H^T = 0$$

It is observed that as $S_1 = S_2 = E H^T$ Error patterns differs by the codeword have the same Syndrome. This confirms that

Syndrome decoder for linear block codes



Cyclic Codes

→ Sub-class of Linear block codes

→ Systematic (or) Non-systematic form

→ In systematic form $[x] = [M \mid C]$

where $x \rightarrow$ Code Vector, $M \rightarrow$ message bits
$C \rightarrow$ check bits

→ A linear code is called cyclic code if every cyclic shift of the code Vector produces some other code Vector.

→ Two fundamental properties are included in cyclic code.

→ **Property 1 : Linearity Property**

$$X_3 = X_1 \oplus X_2$$

that is modulo-2 sum of any two code-words is also a valid codeword

→ **Property 2 : Cyclic Property**

Consider n bit code Vector

$$X = [x_{n-1}, x_{n-2}, x_{n-3}, \ldots, x_3, x_2, x_1, x_0]$$

MSB LSB

Then one cyclic left shift of x is

$$X' = [x_{n-2}, x_{n-3}, x_{n-4}, \ldots, x_2, x_1, x_0, x_{n-1}]$$

MSB LSB

Now one more cyclic left shift of x is

$$X'' = [x_{n-3}, x_{n-4}, x_{n-5}, \ldots, x_1, x_0, x_{n-1}, x_{n-2}]$$

MSB LSB

Cyclic shift of Valid code vector produces another Valid code vector. ie x' & x'' of x are valid code words

Algebraic structures of Cyclic Codes

→ Consider n bit code word

$$X = [x_{n-1}, x_{n-2}, x_{n-3}, \ldots, x_2, x_1, x_0]$$

→ Codeword can be represented by a polynomial of degree less than & equal to $(n-1)$ as

$$X(P) = x_{n-1}P^{n-1} + x_{n-2}P^{n-2} + x_{n-3}P^{n-3} + \cdots$$

$$\cdots + x_2P^2 + x_1P + x_0$$

where → $X(P)$ is the polynomial of degree $(n-1)$

→ P is the arbitary variable of the polynomial

→ The power of P represents the position of the codeword bits P^{n-1} represents MSB, P^0 represents LSB P^1 represents second bit from LSB.

→ why to represent codeword by a polynomial?

Polynomial representation is due to

i) these are algebraic codes. Hence operations such as addition, multiplication, division, subtraction etc. becomes very simple

ii) Position of the bits are represented with the help of powers of 'P' in a polynomial.

Generation of Code Vectors in Non Systematic form

→ Let $M = [m_{k-1}, m_{k-2}, m_{k-3}, \ldots m_2, m_1, m_0]$ be k bits of message vector. Then it can be represented by the polynomial as

$$M(P) = m_{k-1}P^{k-1} + m_{k-2}P^{k-2} + m_{k-3}P^{k-3} + \cdots$$
$$\cdots + m_2 P^2 + m_1 P + m_0$$

→ Code word polynomial $X(P)$ is given as

$$X(P) = M(P) \cdot G(P)$$

where $G(P)$ is the generating polynomial of degree q.

→ Hence for (n, k) cyclic code $q = n-k$ represents number of parity bits.

→ Generating polynomial $G(P)$ is given as

$$G(P) = P^q + g_{q-1}P^{q-1} + g_{q-2}P^{q-2} + \cdots$$
$$\cdots + g_2 P^2 + g_1 P + 1$$

Here $g_{q-1}, g_{q-2}, g_{q-3}, \cdots g_2, g_1$ are the parity bits.

→ If M_1, M_2, M_3 -- etc are the other message vectors, then the corresponding code vectors can be calculated as

$$X_1(P) = M_1(P) \cdot G(P)$$

$$X_2(P) = M_2(P)\, G(P)$$

$$X_3(P) = M_3(P)\, G(P)$$

and so on. All the above code vectors X_1, X_2, X_3 - - -etc are all in the nonsystematic form. and they satisfy cyclic property.

Note that the generator polynomial $G(P)$ remains the same for all code vectors.

Ex. The generator polynomial of a (7, 4) cyclic code is $G(P) = P^3 + P + 1$. find all the code vectors for the cyclic code in nonsystematic form.

Sol. For the given (7,4) cyclic code ⇒

$n = 7,\ K = 4$ & $q = n - k = 7 - 4 = 3$

no. of bits in message Vector $K = 4$

Message Vector $M = [m_3\ m_2\ m_1\ m_0]$

message Polynomial

$$M(P) = m_3 P^3 + m_2 P^2 + m_1 P^1 + m_0$$

no. of bits in Code Vector $n = 7$

Code Vector $X = [x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0]$

Code word Polynomial

$$X(P) = x_6 P^6 + x_5 P^5 + x_4 P^4 + x_3 P^3 + x_2 P^2 + x_1 P^1 + x_0$$

no. of Parity bits $q = n - k = 7 - 4 = 3$

Generator Polynomial $G(P) = P^3 + P + 1$

Non systematic form of cyclic code word Polynomial is

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

As $K=4$ and $n=7$, we have $2^K = 2^4 = 16$ message vectors of size 4 and 16 corresponding Code vectors of size 7.

Consider $M = [0\ 1\ 0\ 1] \Rightarrow$

$$M(P) = 0 \times P^3 + 1 \times P^2 + 0 \times P^1 + 1 = P^2 + 1$$

$$X(P) = M(P) \cdot G(P)$$

$$= (P^2+1)(P^3+P+1)$$

$$= P^5 + P^3 + P^2 + P^3 + P + 1$$

$$= P^5 + P^3 + P^3 + P^2 + P + 1$$

$$= P^5 + (1 \oplus 1) P^3 + P^2 + P + 1$$

$$= P^5 + P^2 + P + 1 \quad \because 1 \oplus 1 = 0$$

$$X(P) = 0P^6 + 1 \cdot P^5 + 0 \cdot P^4 + 0 \cdot P^3 + 1P^2 + 1P^1 + 1$$

Corresponding Code vector for above polynomial is $X = [0\ 1\ 0\ 0\ 1\ 1\ 1]$.

Similarly other Code vectors can be obtained by using the same procedure and are in Table below

S.No	Message bits $M = m_3\ m_2\ m_1\ m_0$	message Polynomial $M(p) = m_3p^3 + m_2p^2 + m_1p + m_0$	code word Polynomial $X(p) = M(p).G(p) = M(p)(p^3+p+1)$	Non systematic cyclic code Vectors $X = x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0$
1	0 0 0 0	0	0	0 0 0 0 0 0 0
2	0 0 0 1	1	p^3+p+1	0 0 0 1 0 1 1
3	0 0 1 0	p	p^4+p^2+p	0 0 1 0 1 1 0
4	0 0 1 1	$p+1$	$p^4+p^3+p^2+1$ $\because (1\oplus 1)p=0$	0 0 1 1 1 0 1
5	0 1 0 0	p^2	$p^5+p^3+p^2$	0 1 0 1 1 0 0
6	0 1 0 1	p^2+1	p^5+p^2+p+1 $\because (1\oplus 1)p^3=0$	0 1 0 0 1 1 1
7	0 1 1 0	p^2+p	$p^5+p^4+p^3+p$ $\because (1\oplus 1)p^2=0$	0 1 1 1 0 1 0
8	0 1 1 1	p^2+p+1	p^5+p^4+1 $\because (1\oplus 1)(p^4+p^2+p)=0$	0 1 1 0 0 0 1
9	1 0 0 0	p^3	$p^6+p^4+p^3$	1 0 1 1 0 0 0
10	1 0 0 1	p^3+1	p^6+p^4+p+1 $\because (1\oplus 1)p^3=0$	1 0 1 0 0 1 1
11	1 0 1 0	p^3+p	$p^6+p^3+p^2+p$ $\because (1\oplus 1)p^4=0$	1 0 0 1 1 1 0
12	1 0 1 1	p^3+p+1	p^6+p^2+1 $\because (1\oplus 1)(p^4+p^3+p)=0$	1 0 0 0 1 0 1
13	1 1 0 0	p^3+p^2	$p^6+p^5+p^4+p^2$ $\because (1\oplus 1)p^3=0$	1 1 1 0 1 0 0
14	1 1 0 1	p^3+p^2+1	$p^6+p^5+p^4+p^3+p^2+p+1$ $\because (1\oplus 1\oplus 1)p^3=p^3$	1 1 1 1 1 1 1
15	1 1 1 0	p^3+p^2+p	p^6+p^5+p $\because (1\oplus 1)(p^4+p^3+p^2)=0$	1 1 0 0 0 1 0
16	1 1 1 1	p^3+p^2+p+1	$p^6+p^5+p^3+1$ $\because (1\oplus 1)(p^4+p^2+p)=0$ & $(1\oplus 1\oplus 1)p^3=p^3$	1 1 0 1 0 0 1

Verification of two fundamental Properties

Linearity Property

Consider $X_7 \oplus X_9$

ie

$X_7 = 0\ 1\ 1\ 1\ 0\ 1\ 0$

$X_9 = 1\ 0\ 1\ 1\ 0\ 0\ 0$

$1\ 1\ 0\ 0\ 0\ 1\ 0 = X_{15}$

---

ie modulo-2 sum of any two code vectors is another valid code vector

Cyclic Property

Let $X = X_5 = [0\ 1\ 0\ 1\ 1\ 0\ 0]$

cyclic left shift of x is x' ie

$X' = [1\ 0\ 1\ 1\ 0\ 0\ 0] = X_9$

one more cyclic left shift of x is x'' ie

$X'' = [0\ 1\ 1\ 0\ 0\ 0\ 1] = X_8$

cyclic shift of one code vector produces another valid code vector. the above two properties can be verified for all other code vectors also.

Hence two properties are verified

Generation of Code Vectors in Systematic form

→ Let $M = [m_{K-1}\ m_{K-2}\ m_{K-3} \cdots m_2\ m_1\ m_0]$ be k bits of message vector. Then it can be represented by the polynomial as

$$M(P) = m_{K-1}P^{K-1} + m_{K-2}P^{K-2} + m_{K-1}P^{K-3} + \cdots + m_2P^2 + m_1P^1 + m_0$$

→ Let $C = [C_{q-1}\ C_{q-2}\ C_{q-3} \cdots C_2\ C_1\ C_0]$ be q bits of Check Vector. Then it can be represented by the polynomial as

$$C(P) = C_{q-1}P^{q-1} + C_{q-2}P^{q-2} + C_{q-3}P^{q-3} + \cdots + C_2P^2 + C_1P^1 + C_0$$

→ Systematic form of Cyclic code Vector is $[X]_{1\times n} = [\underset{1\times K}{M} \mid \underset{1\times q}{C}]_{1\times n}$ $\because n = K+q$

ie $X = [K \text{ message bits} \mid q \text{ check bits}]$

→ Let $X = [m_{K-1}\ m_{K-2}\ m_{K-3} \cdots m_2\ m_1\ m_0 : C_{q-1}\ C_{q-2}\ C_{q-3}, \cdots C_2\ C_1\ C_0]$ be n bits of Code Vector.

→ check bit Polynomial is obtained by

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

where rem → remainder and G(P) is Generator Polynomial of degree 'q'.

→ that is first multiply message polynomial M(P) by P^q, then divide $P^q M(P)$ by generator Polynomial in modulo-2 series & then remainder of the division is check bit Polynomial

→ If M_1, M_2, M_3 -- etc are the other message vectors, then corresponding check bit polynomial can be calculated as

$$C_1(P) = \text{rem}\left[\frac{P^q M_1(q)}{G(P)}\right], \quad C_2 = \text{rem}\left[\frac{P^q M_2(q)}{G(P)}\right]$$

$$C_3(P) = \text{rem}\left[\frac{P^q M_3(P)}{G(P)}\right] \text{ and so on}$$

All the above message vectors M_1, M_2 M_3 --- etc along with respective check bit vectors C_1, C_2, C_3 --- etc will give systematic form of Cyclic Code as

$$X_1 = [M_1 \mid C_1]$$

$$X_2 = [M_2 \mid C_2]$$

$$X_3 = [M_3 \mid C_3]$$ --- and so on.

and they satisfy cyclic property.

Note that generator polynomial remain same for all check bits.

Ex: The generator polynomial of (7,4) cyclic code is $G(P) = P^3 + P + 1$. find all the code vectors for the cyclic code in systematic form.

Sol. For the given (7, 4) cyclic code ⇒

$n = 7$, $K = 4$, & $q = n - k = 7 - 4 = 3$

no. of bits in message vector $K = 4$

message vector $M = [m_3 \; m_2 \; m_1 \; m_0]$

message polynomial

$$M(P) = m_3 P^3 + m_2 P^2 + m_1 P^1 + m_0$$

no. of bits in check vector $q = n - k$
$= 7 - 4 = 3$

check vector $C = [c_2 \; c_1 \; c_0]$

check bit polynomial

$$C(P) = c_2 P^2 + c_1 P^1 + c_0$$

no. of bits in Code vector $n = 7$

Systematic form of cyclic

Code vector $X = [m_3 \; m_2 \; m_1 \; m_0 : c_2 \; c_1 \; c_0]$

Generator polynomial $G(P) = P^3 + P + 1$

Check bit polynomial for systematic cyclic code is

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

As $k=4$ and $n=7$, we have $2^k = 2^4 = 16$ message vectors of size 4 and 16 corresponding Code vectors of size 7

Consider $M = [0\ 1\ 0\ 1] \Rightarrow$

$$M(P) = 0P^3 + 1\cdot P^2 + 0\cdot P + 1 = P^2 + 1$$

Now $$C(P) = \text{rem}\left[\frac{P^3 \cdot M(P)}{G(P)}\right]$$

$$= \text{rem}\left[\frac{P^3(P^2+1)}{P^3+P+1}\right] = \text{rem}\left[\frac{P^5+P^3}{P^3+P+1}\right]$$

P^2

$P^3+P+1 \;\big)\; P^5 + P^3$

$P^5 + P^3 + P^2$

P^2 — remainder.

$\because\ P^5 \oplus P^5 = 0$

$P^3 \oplus P^3 = 0$

$0 \oplus P^2 = P^2$

Hence $C(P) = P^2 = 1\cdot P^2 + 0P^1 + 0$

Corresponding check bit vector $C = [1\ 0\ 0]$

and hence Code vector for above message vector is $X = [0\ 1\ 0\ 1\ 1\ 0\ 0]$

Similarly other code vectors can be obtained by using the same procedure and are given in below table.

(29)

S.No	message bits M = $m_3\ m_2\ m_1\ m_0$	message Polynomial $M(P) = m_3P^3 + m_2P^2 + m_1P + m_0$	Check bit Polynomial $C(P) = \text{rem}\left[\frac{P^3 M(P)}{G(P)}\right]$	Systematic cyclic 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
2	0 0 0 1	1	$P+1$	0 0 0 1 0 1 1
3	0 0 1 0	P	P^2+P	0 0 1 0 1 1 0
4	0 0 1 1	$P+1$	P^2+1	0 0 1 1 1 0 1
5	0 1 0 0	P^2	P^2+P+1	0 1 0 0 1 1 1
6	0 1 0 1	P^2+1	P^2	0 1 0 1 1 0 0
7	0 1 1 0	P^2+P	1	0 1 1 0 0 0 1
8	0 1 1 1	P^2+P+1	P	0 1 1 1 0 1 0
9	1 0 0 0	P^3	P^2+1	1 0 0 0 1 0 1
10	1 0 0 1	P^3+1	P^2+P	1 0 0 1 1 1 0
11	1 0 1 0	P^3+P	$P+1$	1 0 1 0 0 1 1
12	1 0 1 1	P^3+P+1	0	1 0 1 1 0 0 0
13	1 1 0 0	P^3+P^2	P	1 1 0 0 0 1 0
14	1 1 0 1	P^3+P^2+1	1	1 1 0 1 0 0 1
15	1 1 1 0	P^3+P^2+P	P^2	1 1 1 0 1 0 0
16	1 1 1 1	P^3+P^2+P+1	P^2+P+1	1 1 1 1 1 1 1

Verification of two fundamental properties

Linearity Property

Consider $X_2 \oplus X_5$

$X_2 = 0\ 0\ 0\ 1\ 0\ 1\ 1$

$X_5 = 0\ 1\ 0\ 0\ 1\ 1\ 1$

$= 0\ 1\ 0\ 1\ 1\ 0\ 0 = X_6$

ie modulo-2 sum of any two code vectors is another valid code vector

Cyclic Property

Let $X = X_{10} = [1\ 0\ 0\ 1\ 1\ 1\ 0]$

cyclic left shift of X is X' ie

$X' = [0\ 0\ 1\ 1\ 1\ 0\ 1] = X_4$

One more cyclic left shift X is X'' ie

$X'' = [0\ 1\ 1\ 1\ 0\ 1\ 0] = X_8$

cyclic shift of one code vector produces another valid code vector

The above two properties can be verified for all other code vectors also.

Hence two properties are verified

Ex: The generator polynomial of (7,4) cyclic code is $G(P) = P^3 + P + 1$. find the code word for the message (1101) both in systematic and Nonsystematic forms

Sol. For the given (7,4) cyclic code $\Rightarrow$

$n = 7$, $K = 4$ and $q = n - k = 7 - 4 = 3$

no. of bits in message vector $K = 4$

message Vector $M = [m_3 \; m_2 \; m_1 \; m_0]$

Message polynomial $M(P) =$

$$m_3 P^3 + m_2 P^2 + m_1 P + m_0$$

Here As $M = [1 \; 1 \; 0 \; 1]$

$$M(P) = 1 \times P^3 + 1 \times P^2 + 0 \times P + 1 = P^3 + P^2 + 1$$

Generator Polynomial $G(P) =$

$$= P^3 + P + 1$$

Systematic and Nonsystematic forms of cyclic Code Vector X is to be found as follows:

Systematic form of Cyclic Code :

no. of bits in check vector q =

$$n-k = 7-4 = 3$$

check bit vector $C = [C_2 \; C_1 \; C_0]$

check bit Polynomial $C(P) =$

$$C_2P^2 + C_1P + C_0$$

no. of bits in Code vector $n = 7$

Systematic form of Code Vector $x =$

$$[m_3 \; m_2 \; m_1 \; m_0 \; C_2 \; C_1 \; C_0]$$

Check bit Polynomial for systematic cyclic code is

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

Here $C(P) = \text{rem}\left[\frac{P^3(P^3+P^2+1)}{P^3+P+1}\right]$

$$C(P) = \text{Rem}\left[\frac{P^6+P^5+P^3}{P^3+P+1}\right]$$

$$P^3+P+1 \;\Big|\; P^6+P^5+P^3$$

P^3+P^2+P+1

$P^6+P^4+P^3$ (+)(+)

P^5+P^4

$P^5+P^3+P^2$ (+)

$P^4+P^3+P^2$

P^4+P^2+P (+)(+)

P^3+P

P^3+P+1 (+)(+)

$1 \rightarrow$ Remainder

$(1\oplus 1)P^6$, $(1\oplus 1)P^5$, $(1\oplus 1)P^4$, $(1\oplus 1)P^3$, $(1\oplus 1)P^2$ $= 0$

$\therefore C(P) = 1 = 0\times P^2 + 0\times P + 1 \Rightarrow$

Now $C = [0\ 0\ 1]$

Systematic form of cyclic code vector is

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

Non Systematic form of cyclic code

no. of bits in Code Vector $n = 7$

Non Systematic form of Code Vector $X =$

$$[x_6 \quad x_5 \quad x_4 \quad x_3 \quad x_2 \quad x_1 \quad x_0]$$

Codeword Polynomial $X(P) =$

$$x_6P^6 + x_5P^5 + x_4P^4 + x_3P^3 + x_2P^2 + x_1P + x_0$$

No of Parity bits $q = n - k = 7 - 4 = 3$

Code word Polynomial for Non systematic cyclic code is:

$$X(P) = M(P) \cdot G(P)$$

Here $X(P) = (P^3 + P^2 + 1)(P^3 + P + 1)$

$$= P^6 + P^4 + P^3 + P^5 + P^3 + P^2 + P^3 + P + 1$$

$$= P^6 + P^5 + P^4 + (1 \oplus 1 \oplus 1)P^3 + P^2 + P + 1$$

$$= P^6 + P^5 + P^4 + P^3 + P^2 + P + 1 \quad \because (1 \oplus 1 \oplus 1)P^3 = P^3$$

Now

$X(P) = 1 \times P^6 + 1 \times P^5 + 1 \times P^4 + 1 \times P^3 + 1 \times P^2 + 1 \times P + 1 \Rightarrow$

Non systematic form of cyclic code Vector is

$$X = [1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1]$$

Ex: The generator polynomial for (7, 4) cyclic code is $G(P) = P^3 + P^2 + 1$. find systematic and Nonsystematic Code word for the message (1 1 1 0)

Sol:

Ans:

Systematic Code Vector

$X = [$ $]$

Nonsystematic Code Vector

$X = [$ $]$,

Ex: Show that $P^7 + 1 = (P+1)(P^3+P^2+1)(P^3+P+1)$ in mod-2 addition.

Sol: Given that, $P^7+1 = (P+1)(P^3+P^2+1)(P^3+P+1)$

RHS $= (P+1)(P^3+P^2+1)(P^3+P+1)$

$= (P^4+P^3+P+P^3+P^2+1)(P^3+P+1)$

$= (P^4+(1\oplus 1)P^3+P^2+P+1)(P^3+P+1)$

$= (P^4+P^2+P+1)(P^3+P+1)$ $\because 1\oplus 1 = 0$

$= P^7+P^5+P^4+P^5+P^3+P^2+P^4+P^2+P+P^3+P+1$

$= P^7+(1\oplus 1)P^5+(1\oplus 1)P^4+(1\oplus 1)P^3+(1\oplus 1)P^2+(1\oplus 1)P+1$

$= P^7+1 =$ LHS.

Ex: Find out the possible Generator Polynomials for (7,4) cyclic code.

Sol:

→ For the given (7,4) cyclic code,

$n = 7$, $K = 4$, and $q = n - k = 7 - 4 = 3$.

→ It is known that, Generator Polynomial $G(P)$ is the factor of $P^n + 1$.

→ Here $G(P)$ is the factor of $P^7 + 1$

→ $P^7 + 1 = (P+1)(P^3 + P^2 + 1)(P^3 + P + 1)$

→ Its order of the $G(P)$ is q

ie $G(P) = P^q + g_{q-1} P^{q-1} + g_{q-2} P^{q-2} + \cdots + g_2 P^2 + g_1 P + 1$.

→ Here $q = 3$ Hence as degree of $(P+1)$ is 1 it is not a valid Polynomial and as degree of $(P^3 + P^2 + 1)$ & $(P^3 + P + 1)$ is 3, these two will be Valid Generator Polynomials

ie $G_1(P) = P^3 + P^2 + 1$ and

$G_2(P) = P^3 + P + 1$

Generator Matrix (G) and Parity check Matrix (H) of cyclic Codes:

Since cyclic codes are subclass of linear block codes, Generator Matrix and Parity check Matrix can also be defined for cyclic codes.

Nonsystematic Generator Matrix (G):

for (n, k) nonsystematic cyclic code

$q = n - k$ is no. of Parity bits

k is no. of bits in message Vector M. such that $M = [m_{k-1}\ m_{k-2}\ m_{k-3} \cdots m_2\ m_1\ m_0]$

n is no. of bits in nonsystematic Code Vector X such that

$$X = [x_{n-1}\ x_{n-2}\ x_{n-3} \cdots x_2\ x_1\ x_0]$$

G is the Nonsystematic form of Generator Matrix of size kxn ie G_{kxn} has k rows & n columns

Consider Generator Polynomial $G(P)$ of order q as (4)

$$G(P) = P^q + g_{q-1}P^{q-1} + g_{q-2}P^{q-2} + \cdots + g_2P^2 + g_1P + 1$$

To find Generator Matrix G from Generator Polynomial $G(P)$ multiply P^i on both sides and then substitute $i = K-1, K-2, K-3, \ldots, 2, 1, 0$ to find $1^{st}, 2^{nd}, 3^{rd}, 4^{th} \ldots K^{th}$ rows Polynomials respectively

$$P^i G(P) = P^i\left(P^q + g_{q-1}P^{q-1} + g_{q-2}P^{q-2} + \cdots + g_2P^2 + g_1P + 1\right)$$

that is

$$P^i G(P) = P^{i+q} + g_{q-1}P^{i+q-1} + g_{q-2}P^{i+q-2} + \cdots + g_2P^{i+2} + g_1P^{i+1} + P^i;$$

for $i = K-1, K-2, K-3, \ldots, 2, 1, 0$

Once all row Polynomials are obtained, transform them to Matrix to find non-systematic form of Generator Matrix $G_{K \times n}$

Ex: Obtain the Generator matrix corresponding to $G(P) = P^3 + P^2 + 1$ for a non-systematic (7, 4) cyclic code and then find code vectors.

Sol: For the given non-systematic (7, 4) cyclic code, $n = 7$, $k = 4$, and

$q = n - k = 7 - 4 = 3$.

Generator Polynomial $G(P) = P^3 + P^2 + 1$

i) Generator Matrix $G = ?$

To find Generator matrix $G_{k \times n}$ from $G(P)$, multiply P^i on both sides and then substitute

$i = k-1, k-2, k-3, \ldots 2, 1, 0$ to find $1^{st}, 2^{nd}, 3^{rd}, 4^{th}, \ldots k^{th}$ rows respectively.

Here

$P^i G(P) = P^i (P^3 + P^2 + 1)$; for $i = 3, 2, 1, 0$ as $k = 4$

$$P^i G(P) = P^{i+3} + P^{i+2} + P^i \text{ ; for } i = 3, 2, 1, 0$$

For row 1, $i=3 \Rightarrow P^3 G(P) = P^6+P^5+P^3 = 1.P^6+1.P^5+0.P^4+1.P^3+0.P^2+0.P+0$

for row 2, $i=2 \Rightarrow P^2 G(P) = P^5+P^4+P^2 = 0.P^6+1.P^5+1.P^4+0.P^3+1.P^2+1.P+0$

for row 3, $i=1 \Rightarrow P^1 G(P) = P^4+P^3+P = 0.P^6+0.P^5+1.P^4+1.P^3+0.P^2+1.P+0$

for row 4, $i=0 \Rightarrow P^0 G(P) = P^3+P^2+1 = 0.P^6+0.P^5+0.P^4+1.P^3+1.P^2+0.P+1$

On transforming four polynomials into a matrix of size 4x7 gives

nonsystematic form of Generator Matrix

$$G_{4\times7} = \begin{array}{c} \\ \text{row 1} \\ \text{row 2} \\ \text{row 3} \\ \text{row 4} \end{array} \begin{array}{c} \begin{array}{ccccccc} P^6 & P^5 & P^4 & P^3 & P^2 & P^1 & P^0 \end{array} \\ \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}_{4\times7} \end{array}$$

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}_{4\times7}$$

ii) Nonsystematic Code Vector X

no. of bits in message Vector K = 4

Message Vector $M = [m_3\ m_2\ m_1\ m_0]$

no. of bits in Code Vector $n = 7$

Nonsystematic Code Vector $X = [x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0]$

As cyclic code is a subclass of linear block code, its code vectors can be obtained as

$$X_{1\times n} = M_{1\times K} \cdot G_{K\times n}$$

Here

$$X_{1\times 7} = M_{1\times 4} \cdot G_{4\times 7}$$

Let $M = [1\ 0\ 0\ 1]$ then

$$[x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0] = [1\ 0\ 0\ 1] \cdot \begin{bmatrix} 1&1&0&1&0&0&0 \\ 0&1&1&0&1&0&0 \\ 0&0&1&1&0&1&0 \\ 0&0&0&1&1&0&1 \end{bmatrix}$$

Perform matrix multiplications and then additions by mod-2, ie $0\oplus 0=0$, $0\oplus 1=1$, $1\oplus 0=1$, $1\oplus 1=0$

$[x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0] = [1\ 0\ 1\ 0\ 0\ 1\ 1]$

Hence for $M = [1\ 0\ 0\ 1]$, $X = [1\ 0\ 1\ 0\ 0\ 1\ 1]$

As $K = 4$, we have $2^k = 2^4 = 16$ message vectors and corresponding 16 code vectors of each size 7 bits.

In the same way remaining 15 nonsystematic cyclic code vectors can be found by using Generator Matrix G.

Ex: obtain the Generator matrix corresponding to $G(P) = P^3 + P + 1$ for a nonsystematic $(7, 4)$ cyclic code and then find code vectors.

Sol:

Ans:

i) $G_{4\times 7} = \begin{bmatrix} 1 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}_{4\times 7}$

ii) For $M = [1\ 0\ 0\ 1]$, $X = [1\ 0\ 1\ 0\ 0\ 1\ 1]$

Note: Parity check Matrix can not be obtained for a nonsystematic cyclic code using direct method

Systematic form of Generator Matrix

for (n, k) Systematic cyclic code

$q = n - k$

k is no. of bits in message vector M

such that $M = [m_{k-1}\ m_{k-2}\ m_{k-3} \cdots m_2\ m_1\ m_0]$

q is no. of bits in check vector C

such that $C = [c_{q-1}\ c_{q-2}\ c_{q-3} \cdots c_2\ c_1\ c_0]$

n is no. of bits in systematic Code vector X

such that $X = [M \mid C]$ i.e

$X = [m_{k-1}\ m_{k-2}\ m_{k-3} \cdots m_2\ m_1\ m_0\ c_{q-1}\ c_{q-2}\ c_{q-3} \cdots c_2\ c_1\ c_0]$

G(P) is Generator Polynomial of order 'q' as

$G(P) = P^q + g_{q-1}P^{q-1} + g_{q-2}P^{q-2} + \cdots + g_2P^2 + g_1P + 1$

G is the systematic form of Generator Matrix of size kxn

ie G_{kxn} has k rows & n Columns.

The Systematic form of Generator Matrix is

$$G_{k\times n} = \left[I_{k\times k} \mid P_{k\times q} \right]_{k\times n} \quad \because n = k+q$$

where $I_{k\times k}$ is Identity Matrix of size $k\times k$ and $P_{k\times q}$ is Submatrix

The t^{th} row polynomial of Systematic generator matrix $G_{k\times n}$ is given as $= P^{n-t} + R_t(P)$ where $t = 1, 2, 3, \ldots k$.

Divide P^{n-t} by $G(P)$ and express in terms of Quotient Polynomial $Q_t(P)$ and Remainder Polynomial $R_t(P)$ whose degrees are always less than q.

$$\frac{P^{n-t}}{G(P)} = Q_t(P) + \frac{R_t(P)}{G(P)} \Rightarrow$$

$$P^{n-t} = Q_t(P)\,G(P) + R_t(P)$$

It is known that in mod-2 addition if $a = b \oplus c$ then $b = a \oplus c$ and $c = a \oplus b$

Therefore, To find t^{th} row polynomial of Generator Matrix $G_{k \times n}$

$$P^{n-t} + R_t(P) = Q_t(P) \cdot G(P);$$ where

$R_t(P)$ and $Q_t(P)$ are Remainder Polynomial and Quotient Polynomial respectively obtained in $\frac{P^{n-t}}{G(P)}$ & then substitute $t = 1, 2, 3, \ldots k$ for 1st row, 2nd row 3rd row ... k^{th} row polynomials respectively.

Once all row polynomials are obtained transform them to matrix to find systematic form of Generator Matrix $G_{k \times n}$

Systematic form of Parity check Matrix H

The systematic form of parity check Matrix $H_{q \times n} = \left[P^T_{q \times k} \mid I_{q \times q} \right]_{q \times n}$ $\because n = k + q$

where $P^T_{q \times k}$ is Transpose of Submatrix $P_{k \times q}$ and $I_{q \times q}$ is Identity Matrix of size $q \times q$

Ex: Find out the Generator matrix for a systematic (7,4) cyclic code if $G(P) = P^3 + P + 1$. Also find out the Parity check matrix

Sol: For the given (7,4) systematic cyclic code, $n=7$, $k=4$, $q = 7-4 = 3$

Generator Polynomial $G(P) = P^3 + P + 1$

Let G be the systematic form of Generator Matrix of size $k \times n$.

To find t^{th} row polynomial of $G_{k \times n}$

$P^{n-t} + R_t(P) = Q_t(P)\, G(P)$; where $R_t(P)$ & $Q_t(P)$ are Remainder Polynomial and Quotient polynomial obtained in $\frac{P^{n-t}}{G(P)}$ and then substitute $t = 1, 2, 3, \ldots k$ for row 1, row 2, row 3, ---- row k polynomials respectively.

Here To find t^{th} row Polynomial of $G_{4\times 7}$ from given $G(P)$

$$P^{7-t} + R_t(P) = Q_t(P)(P^3+P+1) \text{ for } t=1,2,3,4$$

For row 1 Polynomial, $t=1 \Rightarrow$

$$P^6 + R_1(P) = Q_1(P)(P^3+P+1)$$

$P^3+P+1 \longrightarrow Q_1(P)$

$P^3+P+1 \;\big|\; P^6$

$\oplus\; P^6 + P^4 + P^3$

$P^4 + P^3$

$\oplus\; P^4 + P^2 + P$ $\quad \because 1 \oplus 1 = 0$

$P^3 + P^2 + P$

$\oplus\; P^3 + P + 1$

$P^2 + 1 \longrightarrow R_1(P)$

Now $P^6 + P^2 + 1 = (P^3+P+1)(P^3+P+1)$

$= P^6 + P^4 + P^3 + P^4 + P^2 + P + P^3 + P + 1$

$= P^6 + (1\oplus 1)P^4 + (1\oplus 1)P^3 + P^2 + (1\oplus 1)P + 1 \quad \because 1\oplus 1 = 0$

$= P^6 + P^2 + 1 \Rightarrow$ LHS = RHS Hence Row 1 Polynomial is

$$P^6 + P^2 + 1 = 1\cdot P^6 + 0\cdot P^5 + 0\cdot P^4 + 0\cdot P^3 + 1\cdot P^2 + 0\cdot P + 1 \longrightarrow (i)$$

For row 2 Polynomial, $t=2 \Rightarrow$

$$P^5 + R_2(P) = Q_2(P).(P^3+P+1)$$

$$P^3+P+1 \;\big|\; P^5 \qquad \longrightarrow Q_2(P) = P^2+1$$

$$\oplus\; P^5 + P^3 + P^2$$

$$\overline{P^3 + P^2}$$

$$\oplus\; P^3 + P + 1$$

$$\overline{P^2 + P + 1} \longrightarrow R_2(P)$$

$\because 1 \oplus 1 = 0$

Now

$$P^5 + P^2 + P + 1 = (P^2+1)(P^3+P+1)$$

$$= P^5 + P^3 + P^2 + P^3 + P + 1$$

$$= P^5 + (1 \oplus 1)P^3 + P^2 + P + 1$$

$\because 1 \oplus 1 = 0$

$$= P^5 + P^2 + P + 1 \Rightarrow \text{LHS} = \text{RHS}$$ Hence

Row 2 Polynomial is $P^5 + P^2 + P + 1 =$

$$0.P^6 + 1.P^5 + 0.P^4 + 0.P^3 + 1.P^2 + 1.P + 1 \longrightarrow (2)$$

For row 3 Polynomial, $t=3$

$$P^4 + R_3(P) = Q_3(P)(P^3+P+1)$$

$$P^3+P+1 \;\big|\; P^4 \qquad \longrightarrow Q_3(P) = P$$

$$\oplus\; P^4 + P^2 + P$$

$$\overline{P^2 + P} \longrightarrow R_3(P)$$

Now → $P^4+P^2+P = P(P^3+P+1) = P^4+P^2+P \Rightarrow$ LHS = RHS

Hence Row 3 Polynomial is $P^4+P^2+P =$

$0.P^6 \; P^5 + 1.P^4 + 0.P^3 + 1.P^2 + 1.P + 0 \longrightarrow (3)$

for Row 4 Polynomial, $t = 4 \Rightarrow$

$$P^3 + R_4(P) = Q_4(P)(P^3+P+1)$$

$$P^3+P+1 \;\Big)\; P^3 \;(1 \longrightarrow Q_4(t)$$

$$P^3+P+1$$

$$P+1 \longrightarrow R_4(t)$$

Now $P^3+P+1 = 1(P^3+P+1) \Rightarrow$ LHS = RHS

Hence Row 4 Polynomial is $P^3+P+1 =$

$0.P^6 + 0.P^5 + 0.P^4 + 1.P^3 + 0.P^2 + 1.P + 1 \longrightarrow (4)$

On transforming four Row polynomials into a matrix of size 4x7 gives Systematic form of Generator Matrix

$$G_{4\times7} = \begin{array}{c} \\ \text{Row 1} \\ \text{Row 2} \\ \text{Row 3} \\ \text{Row 4} \end{array} \begin{array}{c} \begin{array}{ccccccc} P^6 & P^5 & P^4 & P^3 & P^2 & P^1 & P^0 \end{array} \\ \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}_{4\times7} \end{array}$$

$$G = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}_{4\times7}$$

Systematic form of Generator matrix

$$G_{k\times n} = \left[\, I_{k\times k} \mid P_{k\times q} \,\right]_{k\times n}$$

Here $G_{4\times 7} = \left[\, I_{4\times 4} \mid P_{4\times 3} \,\right]_{4\times 7}$

where $I_{4\times 4}$ is Identity Matrix of size 4×4 $= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}_{4\times 4}$

and $P_{4\times 3}$ is Submatrix $= \begin{bmatrix} 1 & 0 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \end{bmatrix}_{4\times 3}$

Parity Check Matrix $H_{q\times n} = \left[\, P^T_{q\times k} \mid I_{q\times q} \,\right]_{q\times n}$

here Transpose

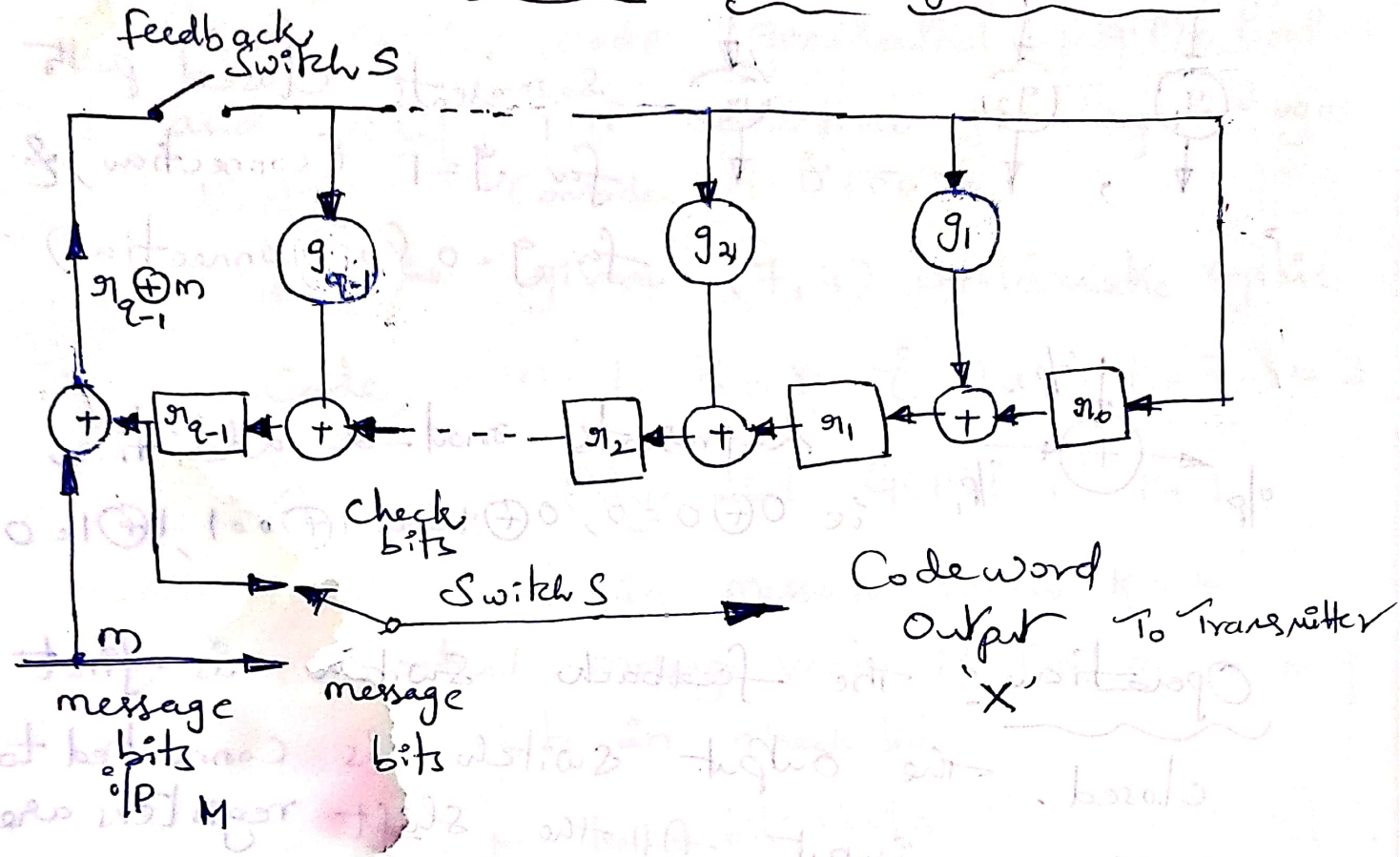
$$H_{3\times 7} = \left[\, P^T_{3\times 4} \mid I_{3\times 3} \,\right]_{3\times 7}$$

where $P^T_{3\times 4}$ is Transpose of $P_{4\times 3} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 1 & 1 \\ 1 & 1 & 0 & 1 \end{bmatrix}_{3\times 4}$

and $I_{3\times 3}$ is Identity Matrix of size 3×3 $= \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_{3\times 3}$

$$\therefore \boxed{H = \left[\begin{array}{cccc|ccc} 1 & 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{array}\right]}$$

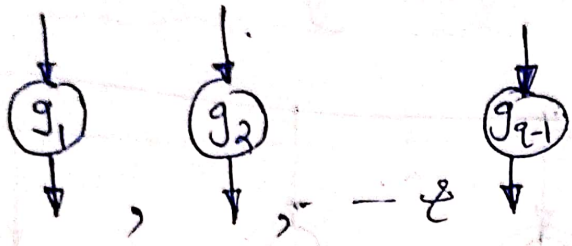
Encoder for Systematic (7,4) cyclic code:



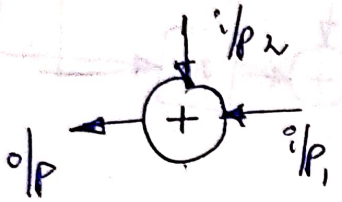
Block diagram of a Generalized Systematic (7,4) cyclic code encoder is shown in above figure where different symbols/blocks are

r_0, r_1, r_2, ... & r_{q-1}

are all 'q' flip flops are all connected in sequential order to make a shift register. ie for every clock pulse contents of the register are shifted from input to output



represents closed path for $g=1$ (Connection) & for $g=0$ (no connection)



represents mod-2 addition
ie $0 \oplus 0 = 0$, $0 \oplus 1 = 1$, $1 \oplus 0 = 1$, $1 \oplus 1 = 0$

Operation: The feedback switch is first closed. The output switch is connected to message input. All the shift registers are initialized to all zero state. The k message bits are shifted to the transmitter as well as shifted into the shift registers.

After the shift of k message bits the registers contain 'r' check bits. The feedback switch is now opened and output switch is connected to check bits position. With every shift, the check bits are then shifted to the transmitter.

It is observed that encoder performs division operation and generates the remainder i.e check bits which are stored in shift register after all message bits are shifted out.

Ex: Design the systematic encoder for the (7,4) cyclic code generated by $G(P) = P^3 + P + 1$ and verify its operation for any message vector. Consider $M = [1\ 0\ 0\ 1]$

Sol: For the given (7,4) systematic cyclic code, $n = 7$, $K = 4$ & $q = n - k = 7 - 4 = 3$.

Generator Polynomial $G(P) = P^3 + P + 1$

no. of bits in message vector $K = 4$

Such that message vector $M = [m_3\ m_2\ m_1\ m_0]$

no. of bits in check bit

Such that check bit vector $C = [c_2\ c_1\ c_0]$

no. of bits in codeword $n = 7$

So that systematic code word

$$X = [M \mid C] = [m_3\ m_2\ m_1\ m_0\ c_2\ c_1\ c_0]$$

The general form of $G(P)$ whose order q is

$$G(P) = P^q + g_{q-1}P^{q-1} + g_{q-2}P^{q-2} + \cdots + g_2P^2 + g_1P + 1$$

If $q = 3 \Rightarrow G(P) = P^3 + g_2P^2 + g_1P + 1$

Given $G(P)$ is $G(P) = P^3 + 0 \cdot P^2 + 1 \cdot P + 1 \Rightarrow$

$g_2 = 0$ (no connection) and $g_1 = 1$ (connection)

Now encoder for (7,4) cyclic code for $G(P) = P^3 + P + 1$ is shown in fig.

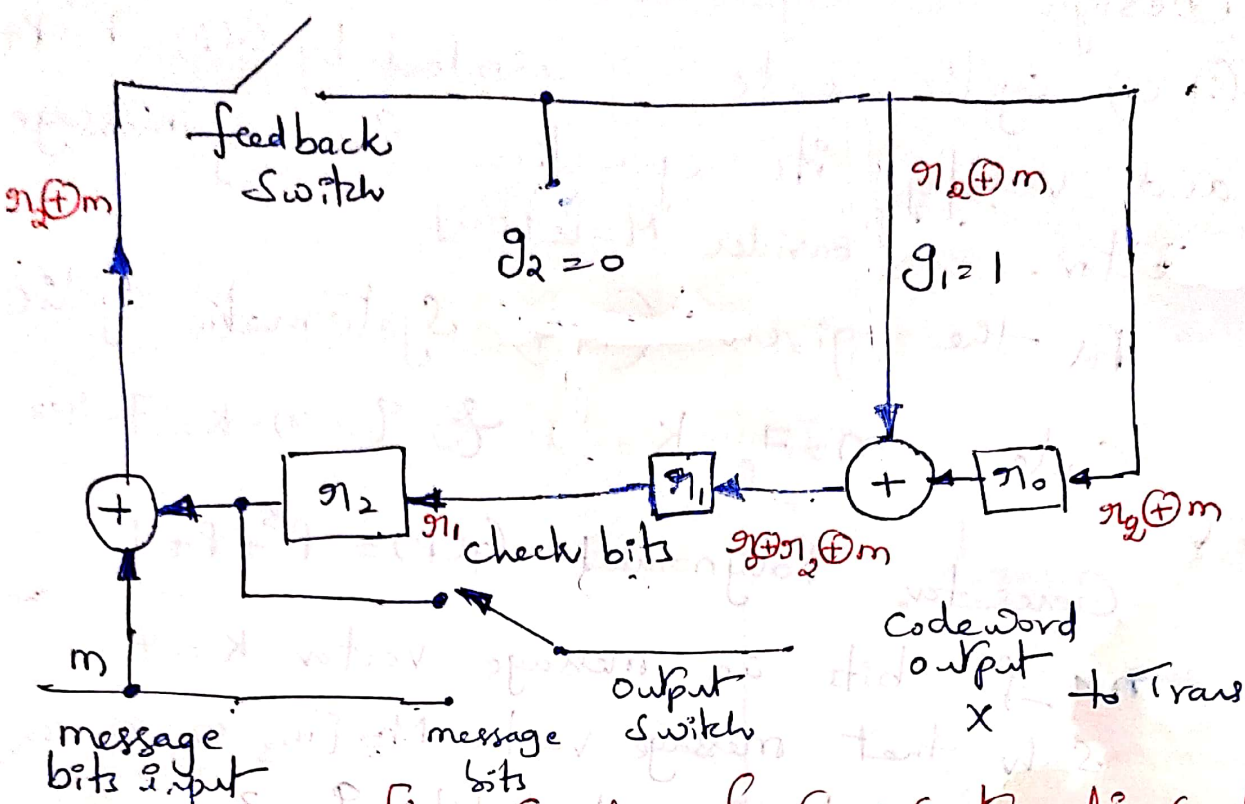


fig. Encoder for (7,4) systematic cyclic code

Table: Shift register bits positions for input message M = [1100]

Input message bits m	Register bit inputs before shift			Register bit outputs after shift		
	$r_2 = r_2'$	$r_1 = r_1'$	$r_0 = r_0'$	$r_2' = r_1$	$r_1' = r_0 \oplus r_2 \oplus m$	$r_0' = r_2 \oplus m$
	0	0	0	0	0	0
1	0	0	0	0	$0 \oplus 0 \oplus 1 = 1$	$0 \oplus 1 = 1$
1	0	1	1	1	$1 \oplus 0 + 1 = 0$	$0 \oplus 1 = 1$
0	1	0	1	0	$1 \oplus 1 \oplus 0 = 0$	$1 \oplus 0 = 1$
0	0	0	1	0	$1 \oplus 0 \oplus 0 = 1$	$0 \oplus 0 = 0$

Hence Check bits in C are $C_2 = 0$ $C_1 = 1$ $C_0 = 0$

Table 2. operation of (7,4) cyclic code encoder

Shift clock	message bit m	Shift register outputs r_2' r_1' r_0'	feedback switch On/off	output switch message/check bits	Transmitted bits X
1	1	0 1 1	On	message	1 (m_3)
2	1	1 0 1	On	message	1 (m_2)
3	0	0 0 1	On	message	0 (m_1)
4	0	0 1 0	On	message	0 (m_0)
5	–	0 1 0	off	check bits	0 (C_2)
6	–	1 0 0	off	check bits	1 (C_1)
7	–	0 0 0	off	check bits	0 (C_0)

Hence Transmitted code word X = [1 1 0 0 0 1 0]

Explanation: As $q = 3$ there are 3 flipflops r_2, r_1 & r_0 in shift register to hold check bits C_2, C_1 & C_0. As $g_2 = 0$ link is not connected and $g_1 = 1$ link is connected. Shown in fig. encoder whose operation for M = [1 1 0 0] is shown in Table.1.

The contents of the shift registers before and after shift are shown in Table 1.

Initially feedback switch is closed and output switch is connected to message. four message bits are shifted to transmitter as well as into the registers. in 1st, 2nd, 3rd, & 4th clock pulses. At the end of the last message bit (m_0) the register bits output are $r_2^1 = 0$, $r_1^1 = 1$ and $r_0^1 = 0$ ie check bits. C_2, C_1 & C_0.

Now feedback switch is opened and output switch is connected to check bits position. Then check bits are then shifted to Transmitter in 5th, 6th & 7th clock pulses. as shown in Table-2.

$\therefore X = [m_3\ m_2\ m_1\ m_0\ C_2\ C_1\ C_0] = [1\ 1\ 0\ 0\ 0\ 1\ 0]$

Ex: ① Design the systematic encoder for the (7,4) cyclic code generated by $G(P) = P^3 + P + 1$ and verify its operation for any message vector.

Consider $M = [1\ 0\ 0\ 1]$

Syndrome Decoding, Error Detection and Error Correction

Let X be the Transmitted Code Vector
& Y be the Received Code Vector.

$X = Y$ if there are no transmission errors

$X \neq Y$ if there are transmission errors.
ie 0 as 1 & 1 as 0.

Let E be the Error Vector 'E' then
X, Y & E Vectors are related as

$E = X \oplus Y$, $Y = X \oplus E$, $X = Y \oplus E$.

where $\oplus$ indicates mod-2 addition ie

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

Consider $Y = X \oplus E$,

Express in polynomial form

$Y(P) = X(P) \oplus E(P)$, as $X(P) = M(P)G(P)$,

$Y(P) = M(P) \cdot G(P) + E(P)$

where G(P) is Generator Polynomial of order 'r'

divide Y(P) by G(P)

$$\frac{Y(P)}{G(P)} = Q(P) + \frac{R(P)}{G(P)}$$ where

$Q(p)$ & $R(p)$ are Quotient and Remainder Polynomials respectively whose order is always less than 'q'.

If $T(p) = X(p) \Rightarrow$ no errors then above equation indicates

$$\frac{X(p)}{G(p)} = Q(p) + \frac{R(p)}{G(p)}$$

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

Above two equations shows that $R(p) = 0$ and $Q(p) = M(p)$

If $T(p) \neq X(p) \Rightarrow$ error is there

then $$\frac{T(p)}{G(p)} = Q(p) + \frac{R(p)}{G(p)}$$

multiply the above equation with $G(p)$ on both sides gives

$$T(p) = Q(p)G(p) + R(p) \quad \text{But}$$

$$T(p) = M(p)G(p) + E(p) \Rightarrow$$

$$E(p) + M(p) \cdot G(p) = Q(p)G(p) + R(p)$$

$$E(P) = M(P)G(P) + Q(P)G(P) + R(P)$$

$$= (M(P) \oplus Q(P))G(P) + R(P)$$

This equation indicates that for a fixed message vector and Generator Polynomial, Error Vector E depends on Remainder R. For every Remainder R there will be specific error vector. Hence remainder Vector R is called as Syndrome Vector S. ie $R(P) = S(P)$

$$\therefore \frac{Y(P)}{G(P)} = Q(P) + \frac{S(P)}{G(P)} \Rightarrow$$

Syndrome Vector can be calculated by:

$$S(P) = \text{rem}\left[\frac{Y(P)}{G(P)}\right]$$

Syndrome Calculator:

Block diagram of Syndrome calculator for (n, k) systematic cyclic code is shown in fig. [P.T.O]

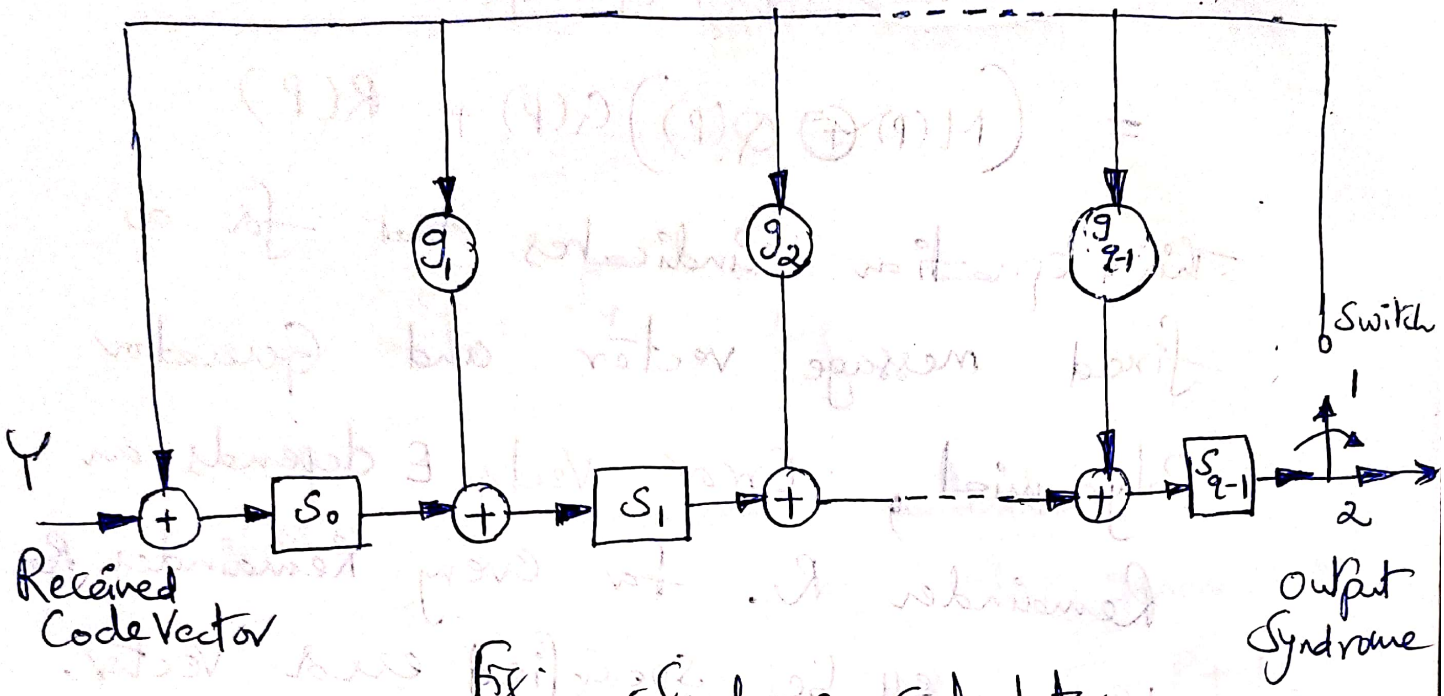


Fig: Syndrome Calculator for (n,k) Cyclic Code

Operation:

For (n,k) cyclic code, $q = n-k$. Hence there are q flip flops in shift register.

As $G(P) = P^q + g_{q-1}P^{q-1} + g_{q-2}P^{q-2} + \cdots + P^2 g_2 + P g_1 + 1$

here for any $g = \begin{cases} 1 & \text{Connection} \\ 0 & \text{no Connection} \end{cases}$

Initially all the shift registers are in zero state and switch is in position 1. The received vector Y is shifted bit by bit into the shift register. The contents of the flip flops goes on changing according to the values of $g_1, g_2, \cdots g_{q-1}$ and input bits of Y. After all the bits bits of Y are shifted, contents of the shift register contains q bit Syndrome Vector. The switch is then

moved to position 2 and clocks are applied to the shift register. to get output Syndrome Vector 'S' is

$$S = [S_{q-1}\ S_{q-2}\ S_{q-3} \ldots S_2\ S_1\ S_0]$$

Ex: Design a syndrome calculator for a (7,4) cyclic Hamming Code generated by the polynomial $G(P) = P^3 + P + 1$. Calculate the syndrome for $Y = [1001101]$

Sol. For the given (7,4) systematic cyclic Code, $n = 7$, $k = 4$, $q = n - q = 3$

Generator Polynomial $G(P) = P^3 + P + 1$

General form of $G(P)$ whose order q is

$$G(P) = P^q + P^{q-1} g_{q-1} + P^{q-2} g_{q-2} + \cdots + P^2 g_2 + P g_1 + 1$$

$$q = 3 \Rightarrow G(P) = P^3 + g_2 P^2 + g_1 P + 1$$

Given $G(P) = P^3 + 0 \cdot P^2 + 1 \cdot P + 1 \Rightarrow$

$g_2 = 0$ (no connection) & $g_1 = 1$ (connection).

No. of bits in received code vector Y is $n = 7$. and Given $Y = [1001101]$

As $q = 3$, Syndrome Vector $S = [S_2\ S_1\ S_0]$

Block diagram of Syndrome Calculator for Systematic (7,4) cyclic code is

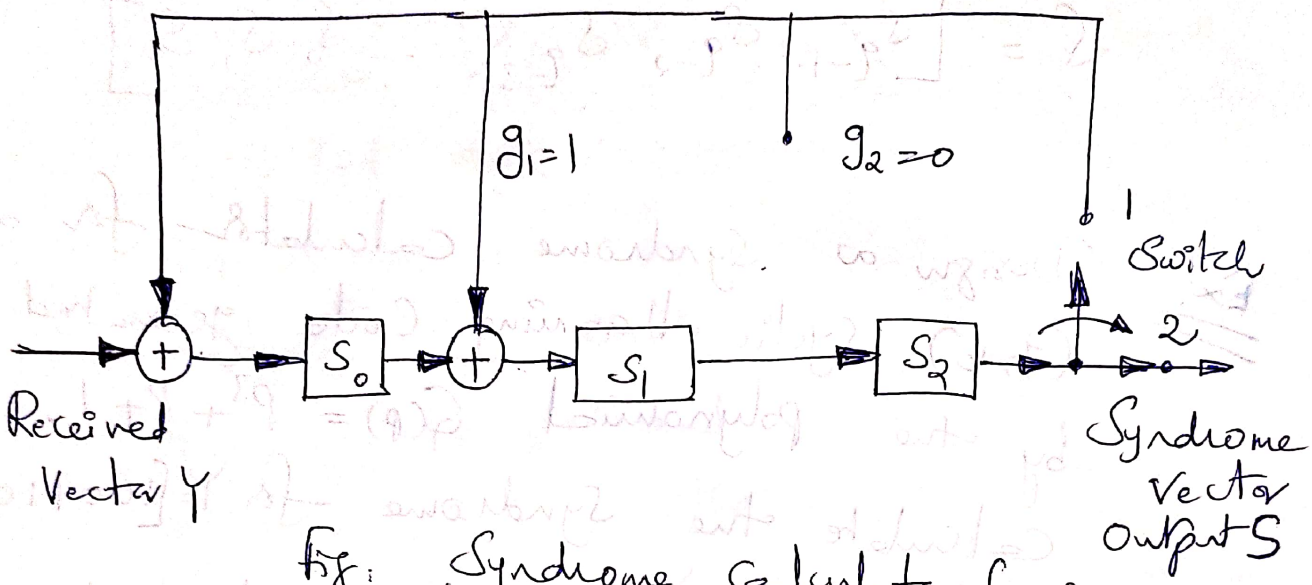


Fig: Syndrome Calculator for (7,4) cyclic code with $g(p) = p^3+p+1$

Shift clock	Received Vector bits Y	Contents of shift register $S_0=Y\oplus S_2$	$S_1=S_0\oplus S_2$	$S_2=S_1$
		0	0	0
1	1	$1\oplus0=1$	$0\oplus0=0$	0
2	0	$0\oplus0=0$	$1\oplus0=1$	0
3	0	$0\oplus0=0$	$0\oplus0=0$	1
4	1	$1\oplus1=0$	$0\oplus1=1$	0
5	1	$1\oplus0=1$	$0\oplus0=0$	1
6	0	$0\oplus1=1$	$1\oplus1=0$	0
7	1	$1\oplus0=1$	$1\oplus0=1$	0

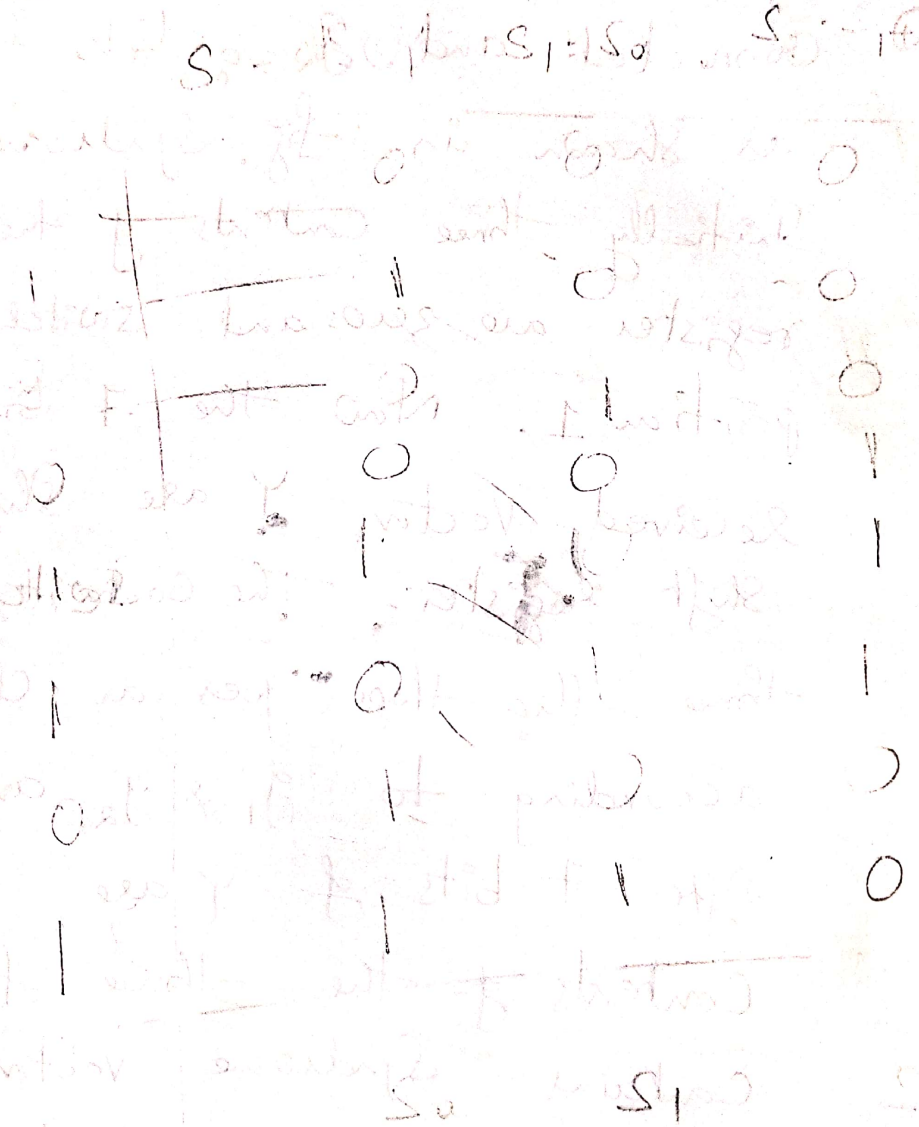
Hence as $S_0=1, S_1=1$ & $S_2=0$, Syndrome Vector $S=[S_2\ S_1\ S_0]=[0\ 1\ 1]$

Explanation:

As $q=3$, there are three flip flops S_0, S_1 & S_2 in shift register to hold Syndrome bits. As $g_1=1$ link is connected and $g_2=0$ link is not connected as shown in fig. Syndrome calculator. Initially three contents of the shift register are zeros and switch is in position 1. Now the 7 bits of the received vector Y are shifted into the shift register. The contents of the three flip flops goes on changing according to g_1 & g_2 and 7 bits of Y. After 7 bits of Y are shifted, contents of the three flip flops contains Syndrome vector bits. The Table shows the contents of shift register for all seven clock pulses and by the end of 7th clock pulse $S_0=1, S_1=1, \& S_2=0 \Rightarrow$

$$S=[S_2\ S_1\ S_0]=[0\ 1\ 1]$$

Ex: Design a Syndrome Calculator for (7,4) cyclic Hamming code generated by the polynomial $G(P) = P^3 + P^2 + 1$. Calculate the Syndrome for $Y_2 = [1001101]$



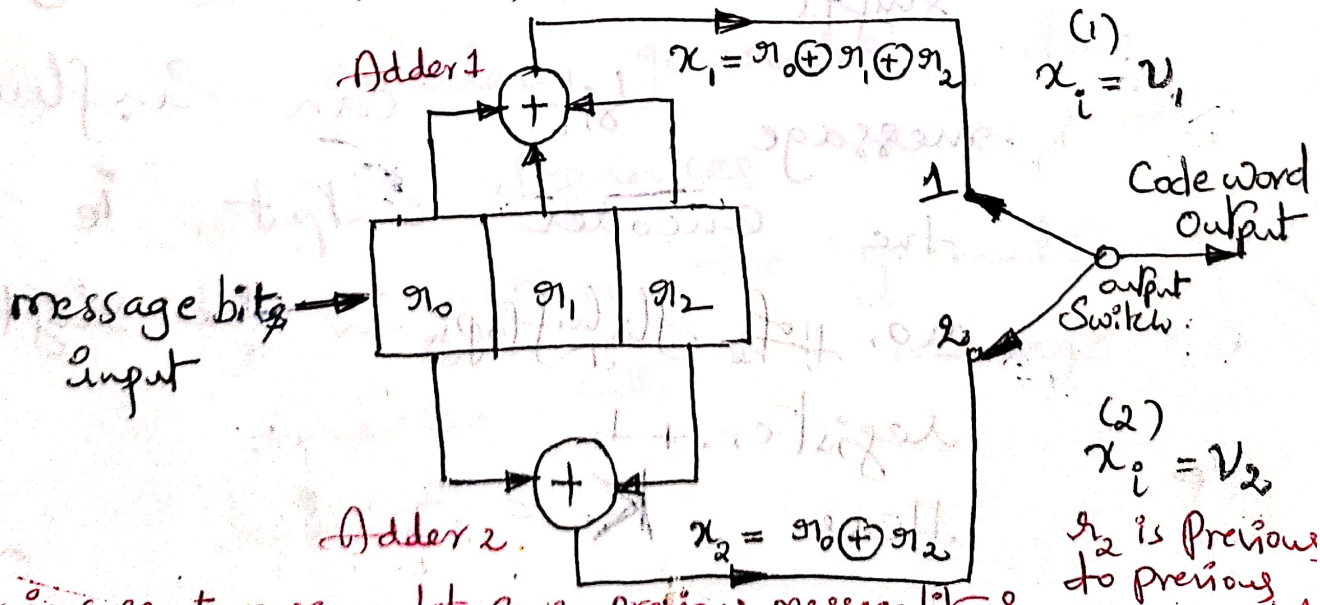
Convolutional Codes

→ Convolutional codes are error correcting codes

→ A Convolutional code is done by combing the fixed number of input message bits. The input bits are stored in the shift register where message bits can be shifted (slided) and then they are combined via mod-2 adders.

→ As above operation is equivalent to binary convolution, this code is called as Convolutional code.

→ Example of Convolutional Encoder



→ no. of message bits that encoder can take at a time $k = 1$

no. of encoded output bits for one message bit $n = 2$ (i.e x_1 & x_2)

Code rate of the encoder is

$$r = \frac{k}{n} = \frac{1}{2}$$

Dimension of the code is

$$(n, k) = (2, 1)$$

Constrained length of the code is defined as the no. of shifts over which a single message bit can influence the encoder output.

Here $K = 3$

Analysis of Convolutional Encoder

Time Domain Approach:

→ Let the sequence $[g_0^{(1)}, g_1^{(1)}, g_2^{(1)}, \ldots g_m^{(1)}]$ denote the impulse response of the adder 1 which generates x_1

Let the sequence $[g_0^{(2)}, g_1^{(2)}, g_2^{(2)}, \ldots g_m^{(2)}]$ denote the impulse response of the adder 2 which generates x_2.

These impulses responses are also called generator sequences of the convolutional code.

Let the incoming message sequence

→ be $M = [m_0\ m_1\ m_2\ m_3 \ldots m_{L-1}]$

→ the encoder generates two output sequences x_1 and x_2 by convolving the generator sequences with the message sequence.

→ Hence By Time Domain Approach (Convolution Sum)

$$\therefore x_1 = x_i^{(1)} = \sum_{l=0}^{M} g_l^{(1)} g_{i-l} \quad ; \quad i = 0, 1, 2, \ldots$$

$$x_2 = x_i^{(2)} = \sum_{l=0}^{M} g_l^{(2)} g_{i-l} \quad ; \quad i = 0, 1, 2, \ldots$$

where $g_{i-l} = 0$ for all $l > i$

→ all additions in above summations are as per mod-2 addition rules

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

→ The two sequences: x_1 and x_2 are multiplexed by the output switch. Hence the output sequence of the Convolutional encoder is

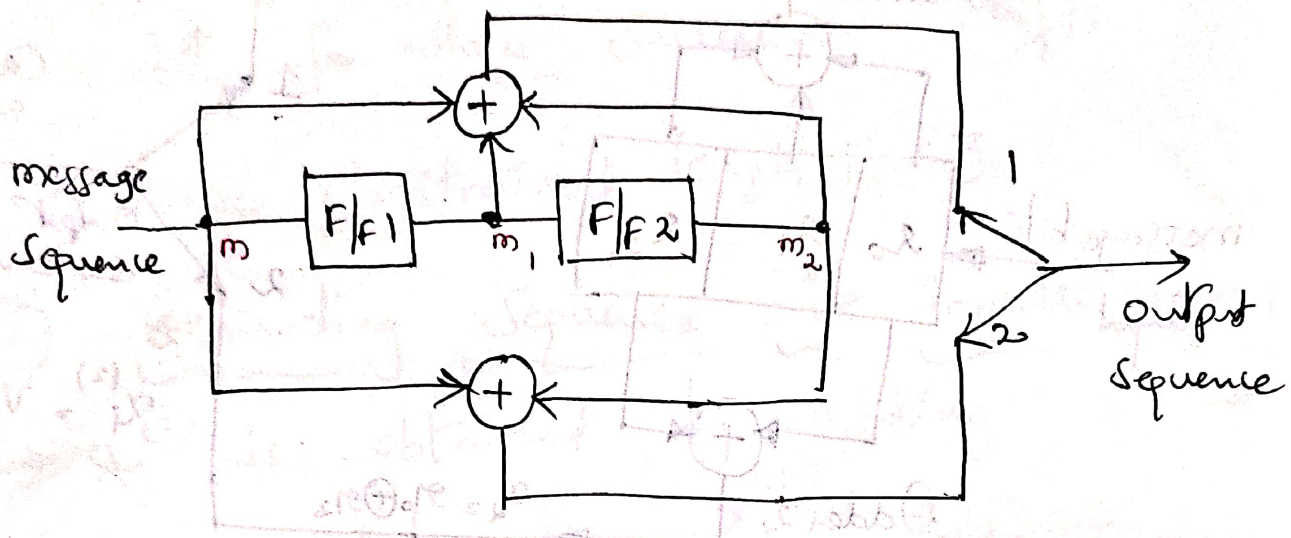
$$x_i = \left[x_0^{(1)} \; x_0^{(2)} \; x_1^{(1)} \; x_1^{(2)} \; x_2^{(1)} \; x_2^{(2)} \; x_3^{(1)} \; x_3^{(2)} \; \ldots\ldots \right]$$

→ Let $V_1 = x_i^{(1)} = \left[x_0^{(1)} \; x_1^{(1)} \; x_2^{(1)} \; x_3^{(1)} \; \ldots\ldots \right]$

and $V_2 = x_i^{(2)} = \left[x_0^{(2)} \; x_1^{(2)} \; x_2^{(2)} \; x_3^{(2)} \; \ldots\ldots \right]$

ie V_1 & V_2 are multiplexed by o/p switch.

Ex: For the Convolutional encoder shown in figure find the following



i) Dimension of the code

ii) Code rate

iii) Constraint length

iv) generating sequences (impulse responses)

v) output sequence for the message sequence $M = [1\ 0\ 0\ 1\ 1]$

Sol: Given Convolutional encoder can be redrawn as shown in fig. where x_0, x_1 & x_2 are representing current message bit, Previous message bit and previous to previous message bit respectively.

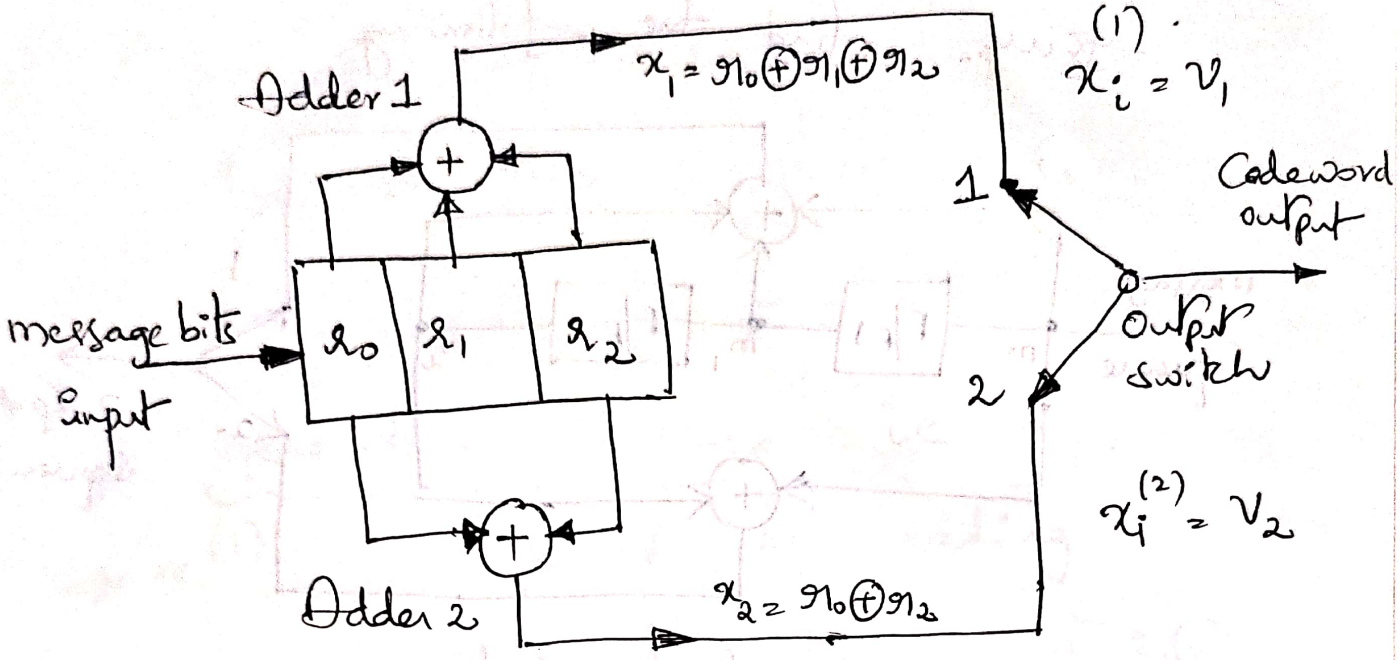


Fig: Convolutional encoder

No. of message bits that encoder can take at a time $k = 1$

No. of encoded output bits for one message bit $n = 2$ (ie x_1 & x_2)

i) Dimensions of the Code is $(n, k) = (2, 1)$

ii) Code rate of Convolutional Code is $r = \frac{k}{n} = \frac{1}{2}$

iii) It is noted that every message bit will influence output bits for three successive shifts

Hence Constrained length K = 3

iv) Generating Sequence i.e impulse response

$\Rightarrow x_i^{(1)}$ is obtained by adding all the three bits. i.e $x_i^{(1)} = s_0 \oplus s_1 \oplus s_2$

Generating Sequence $g_i^{(1)}$ is

$g_i^{(1)} = [1\ 1\ 1]$ where

$g_0^{(1)} = 1$ represents s_0 Connected

$g_1^{(1)} = 1$ represents s_1 Connected

$g_2^{(1)} = 1$ represents s_2 Connected

$\Rightarrow x_i^{(2)}$ is obtained by adding first and last bits. ie $x_i^{(2)} = s_0 \oplus s_2$

Generating Sequence $g_i^{(2)}$ is

$g_i^{(2)} = [1\ 0\ 1]$ where

$g_0^{(2)} = 1$ represents s_0 Connected

$g_1^{(2)} = 0$ represents s_1 not Connected

$g_2^{(2)} = 1$ represents s_2 Connected

v) message sequence $M = [m_0\ m_1\ m_2\ m_3\ m_4] = [1\ 0\ 0\ 1\ 1]$

output of adder 1 is

$$x_i^{(1)} = \sum_{l=0}^{M} g_l^{(1)} s_{i-l}$$

and output of adder 2 is

$$x_i^{(2)} = \sum_{l=0}^{M} g_l^{(2)} s_{i-l}; \text{ where } \text{for } i = 0, 1, \ldots 6.$$

Table: Operation of Encoder

i	s_0	s_1	s_2	$x_i^{(1)} = s_0 \oplus s_1 \oplus s_2$	$x_i^{(2)} = s_0 \oplus s_2$
0	m_0 |			1	1
1	m_1 0	m_0 |		1	0
2	m_2 0	m_1 0	m_0 |	1	1
3	m_3 1	m_2 0	m_1 0	1	1
4	m_4 1	m_3 1	m_2 0	0	1
5		m_4 1	m_3 1	0	1
6			m_4 1	1	1

$\therefore x_i^{(1)} = [1\ 1\ 1\ 1\ 0\ 0\ 1]$ & $x_i^{(2)} = [1\ 0\ 1\ 1\ 1\ 1\ 1]$

multiplexed sequence of x_1 and x_2 is $x_i = [1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1]$
i.e Output Code word

Analysis of Convolutional Encoder

Transform Domain Approach

→ Let the sequence $[g_0^{(1)}, g_1^{(1)}, g_2^{(1)}, \ldots g_m^{(1)}]$ denote the impulse response of the adder which generates x_1

Let the sequence $[g_0^{(2)}, g_1^{(2)}, g_2^{(2)}, \ldots g_m^{(2)}]$ denote the impulse response of the adder which generates x_2

These impulse responses are also called generator sequences of the convolutional code.

→ Let the incoming message sequence be $M = [m_0, m_1, m_2, m_3, \ldots m_{L-1}]$

→ Impulse responses and message sequence can be represented by Polynomials

$$g^{(1)}(P) = g_0^{(1)} + g_1^{(1)} P + g_2^{(1)} P^2 + g_3^{(1)} P^3 + \ldots + g_M^{(1)} P^M$$

$$g^{(2)}(P) = g_0^{(2)} + g_1^{(2)} P + g_2^{(2)} P^2 + g_3^{(2)} P^3 + \ldots + g_M^{(2)} P^M$$

and

$$m(P) = m_0 + m_1 P + m_2 P^2 + m_3 P^3 + \ldots + m_{L-1} P^{L-1}$$

Note: 'P' represents Unit delay operator

Convolution sums in Time Domain are converted into Polynomial Multiplications in Transform Domain. i.e.

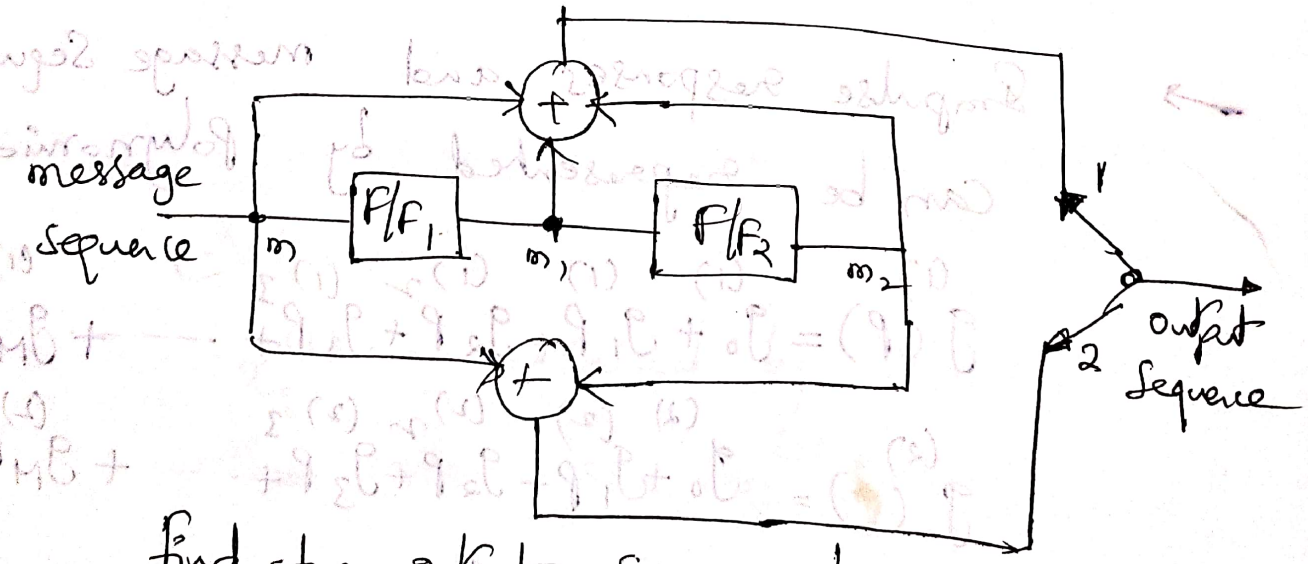
$x^{(1)}(p) = g^{(1)}(p) \cdot m(p)$ and

$x^{(2)}(p) = g^{(2)}(p) \cdot m(p)$ are the output polynomials of the sequences $x_i^{(1)}$ and $x_i^{(2)}$ respectively.

Note: All additions in above equations are as per mod 2 addition rules

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

Ex. For the convolutional encoder shown in figure



Find the output sequence by using Transform domain calculations (Polynomial multiplications)

Sol: Given Convolutional encoder can be redrawn as shown in figure where r_0, r_1 & r_2 are representing current message bit, previous message bit and previous to previous message bit respectively.

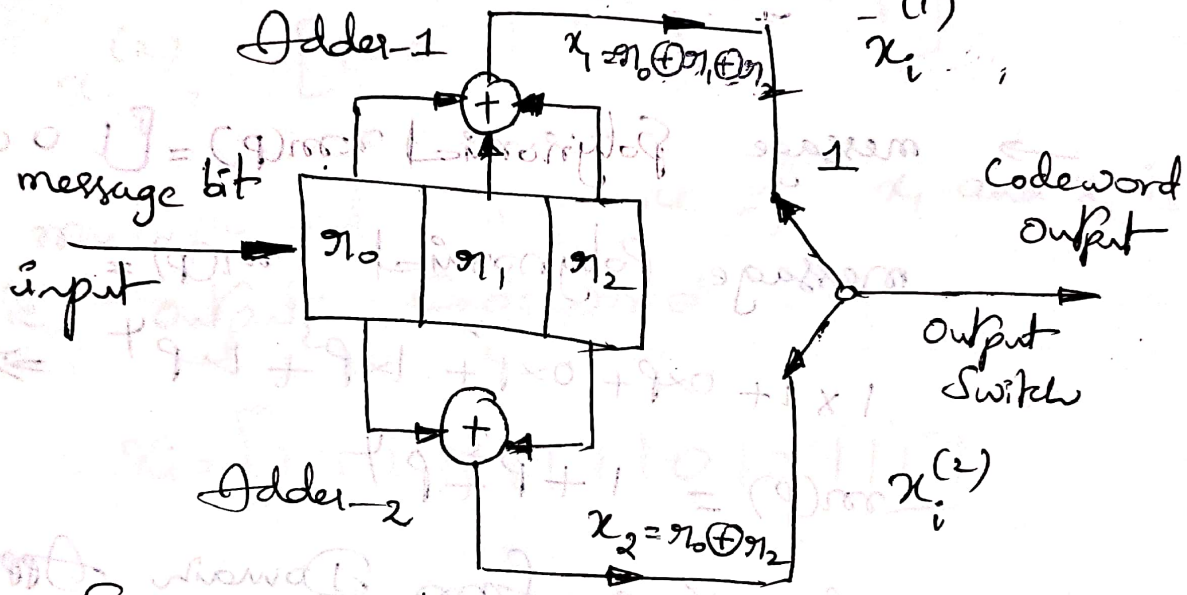


fig: Convolutional encoder

→ Generating Sequences ie impulse responses:

$x_i^{(1)}$ is obtained by adding all the three bits ie $x_i^{(1)} = r_0 \oplus r_1 \oplus r_2$

∴ generating sequence

$g_i^{(1)} = [1 \;\; 1 \;\; 1]$ and its Polynomial $g^{(1)}(p) = 1x1 + 1\alpha p + 1\alpha p^2 = 1 + p + p^2$

$x_i(2)$ is obtained by adding first and last bits.

ie $x_i^{(2)} = g_0 \oplus g_2$

$\therefore$ generating sequence $g_i^{(2)} = [1\ 0\ 1]$

and its polynomial $g^{(2)}(p) = 1\times1 + 0\times p + 1\times p^2 \Rightarrow$

$= 1 + p^2$

→ message sequence $m = [1\ 0\ 0\ 1\ 1]$

message Polynomial $m(p) =$

$1\times1 + 0\times p + 0\times p^2 + 1\times p^3 + 1\times p^4 \Rightarrow$

$m(p) = 1 + p^3 + p^4$

→ Hence By Transform Domain Approach (Polynomial Multiplications)

output of adder 1 $x^{(1)}(p) = g^{(1)}(p) \cdot m(p)$

Here $x^{(1)}(p) = (1 + p + p^2)(1 + p^3 + p^4)$

$= 1 + p^3 + p^4 + p + p^4 + p^5 + p^2 + p^5 + p^6$

$= 1 + p + p^2 + p^3 + (1 \oplus 1)p^4 + (1 \oplus 1)p^5 + p^6$

$= 1 + p + p^2 + p^3 + p^6 \quad \because 1 \oplus 1 = 0$

$= 1\times1 + 1\times p + 1\times p^2 + 1\times p^3 + 0\times p^4 + 0\times p^5 + 1\times p^6 \Rightarrow$

$x_i^{(1)} = [1\ 1\ 1\ 1\ 0\ 0\ 1]$

output of adder 2 : $x^{(2)}(p) = g^{(2)}(p) \cdot m(p)$ (11)

Here $x^{(2)}(p) = (1+p^2)(1+p^3+p^4)$

$= 1+p^3+p^4+p^2+p^5+p^6$

$= 1+p^2+p^3+p^4+p^5+p^6$

$= 1\times1 + 0\times p + 1\times p^2 + 1\times p^3 + 1\times p^4 + 1\times p^5 + 1\times p^6 \Rightarrow$

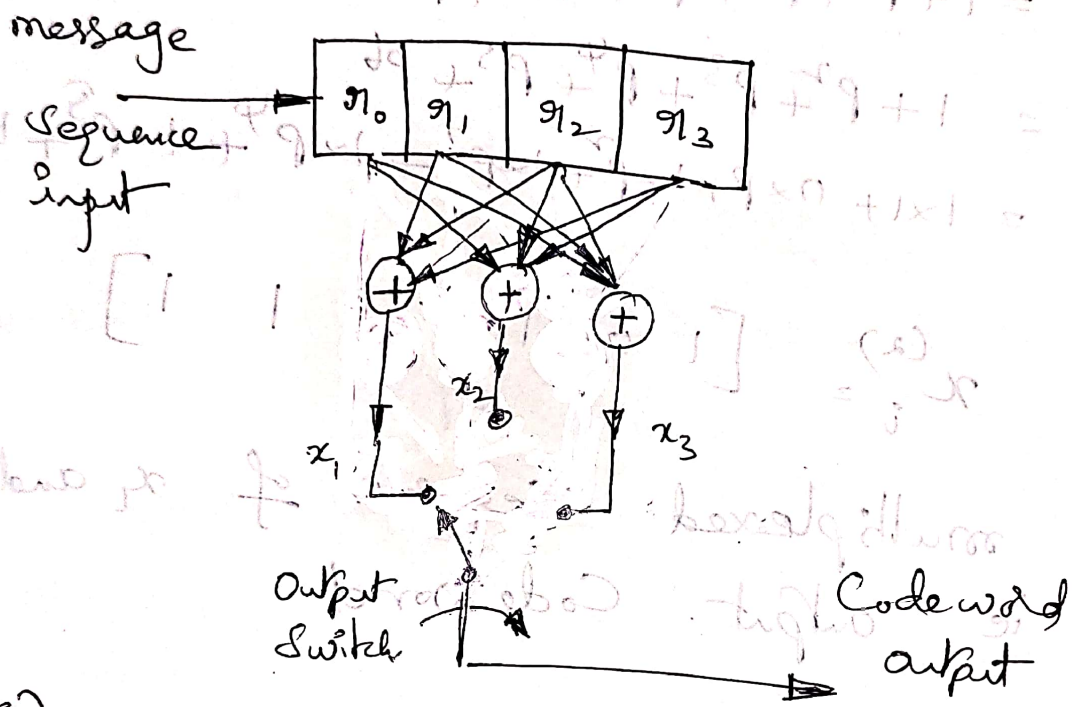
$x_i^{(2)} = [1\ 0\ 1\ 1\ 1\ 1\ 1]$

→ multiplexed sequence of x_1 and x_2 is ie output Codeword

$x_i = [1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1\ 0\ 1\ 1\ 1]$

Note: Transform Domain approachment of finding Codeword output of Convolutional encoder has few calculations when compared to Time Domain approach.

Ex: For the Convolution encoder shown in (88) figure. Find the following.



i) Code rate ii) Code Dimension

iii) Constrained length

iv) generating sequences (ie impulse responses) output sequence for the message

$M = [1\ 1\ 0\ 0\ 1\ 1]$

v) by using Time Domain Approach

vi) by using Transform Domain Approach

Operation of a Convolutional Encoder

Can be study by using

- Code Tree
- Code Trellis
- State Diagram

→ Consider the convolutional encoder shown in figure $x_1 = r_0 \oplus r_1 \oplus r_2$

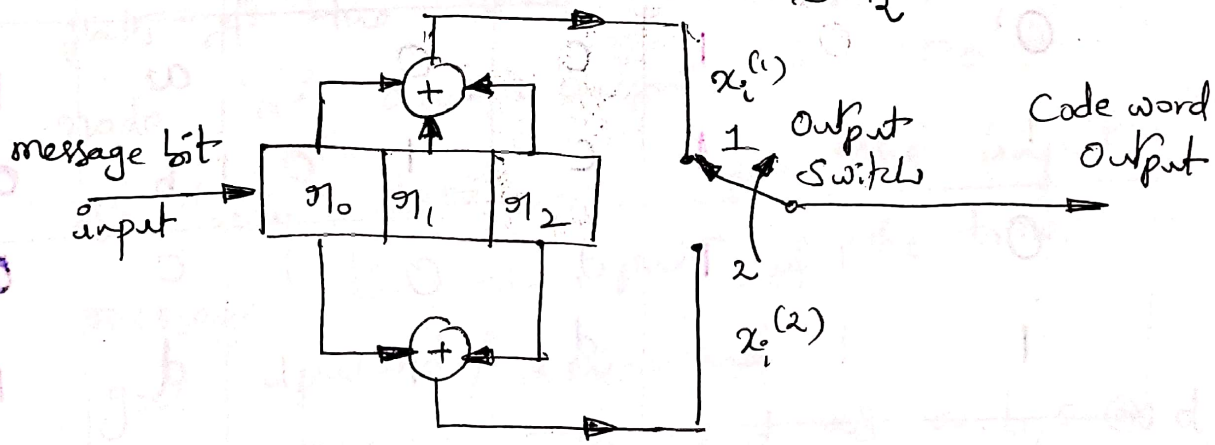


Fig: Convolutional Encoder

→ Let r_1 and r_2 be the state of the encoder

Binary Description		State
r_1	r_2	
0	0	a
1	0	b
0	1	c
1	1	d

State Transition Table

message bit input x_0	Present state (P.S) y_1	y_2		Next state (N.S) y_1	y_2		Code word output x_1	x_2
0	0	0	a	0	0	a	0	0
1	0	0	a	1	0	b	1	1
0	1	0	b	0	1	c	1	0
1	1	0	b	1	1	d	0	1
0	0	1	c	0	0	a	1	
1	0	1	c	1	0	b	0	
0	1	1	d	0	1	c	0	
1	1	1	d	1	1	d	1	

Code Tree: A full Binary Tree that represents a Coding Scheme that encoded bits are labeled on the branches and for a given message sequence input, output Code word can be directly read is called as Code Tree.

→ Code Tree for Convolutional Code is shown in figure below where code tree starts at node 'a'.

→ If input message bit is '1', then path of the tree goes downward towards node 'b' and output is '11'.

(or)

→ If input message bit is '0', then path of the tree goes upwards towards node 'a' and output is '00'.

→ Similarly depending upon the input message bit, the path of the tree goes upward / downward.

→ The nodes are marked as a, b, c & d and above the branches (ie paths between two nodes) outputs are shown.

→ It is observed that branch pattern abcd starts to repeat after 3rd bit. This is because any message bit is stored in shift register for three shifts

→ If message bit input M_2 [1 0 0 1 1] then corresponding encoder output from code tree is $x = [11\ 10\ 11\ 11\ 01]$.

→ In this way encoder output is obtained for any message bit sequence M.

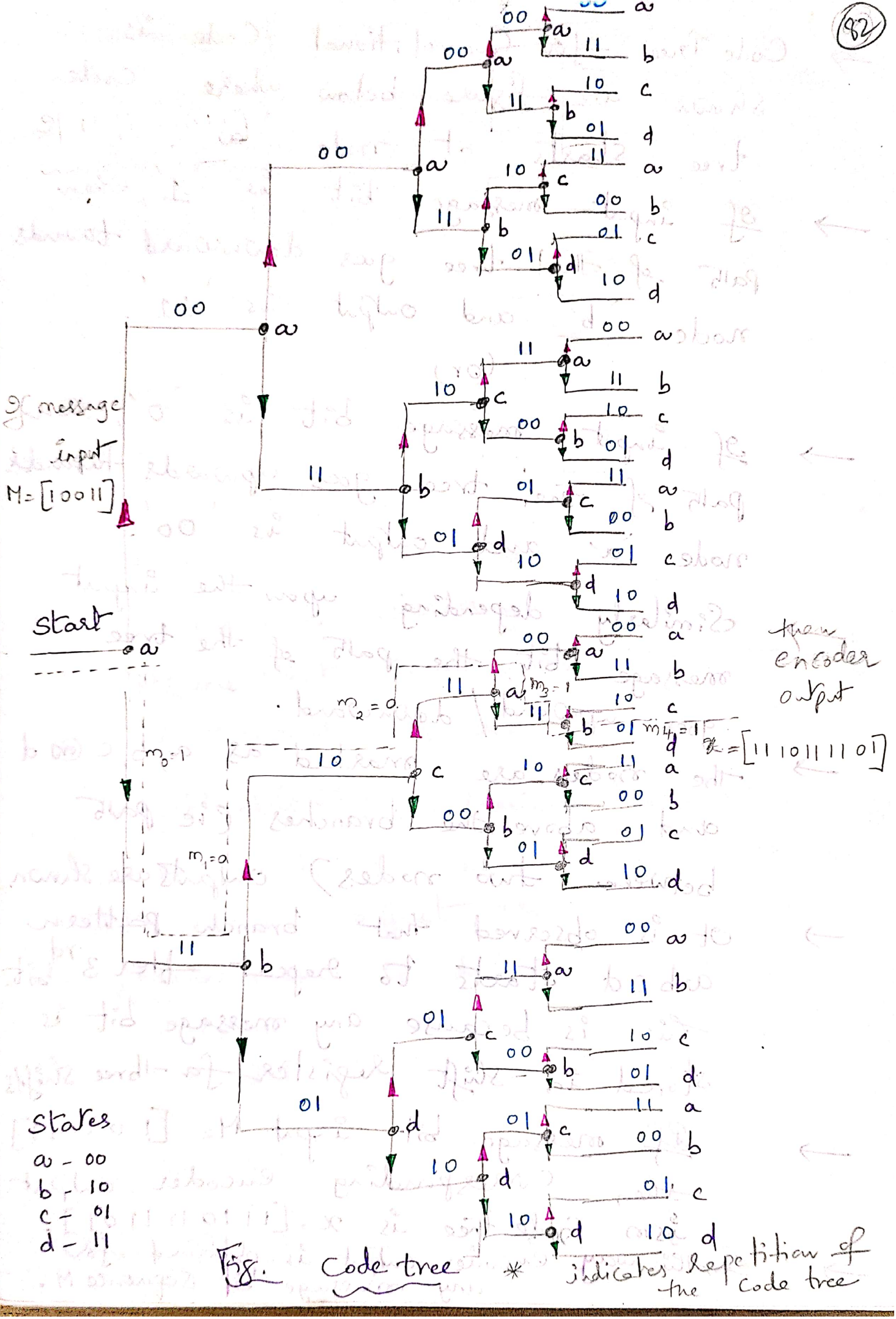


Fig. Code tree * indicates the repetition of code tree

Decoding of Convolutional Codes:

Let 'X' be the Transmitted Code Vector and then 'Y' be the Received Code Vector.

$X = Y$ if there are no Transmission Errors

$X \neq Y$ if there are Transmission Errors.

i.e 0 as 1 and 1 as 0

There are three methods for decoding of Convolutional Codes. They are

* Viterbi Algorithm (i.e Maximum likelihood Decoding)
* Sequential Decoding
* feedback Decoding

Assume that transmission error probability of symbols '1's and '0's is same.

Viterbi Algorithm (i.e Maximum likelihood Decoding)

The branch metric, Path metric, Surviving paths and metric diversion effect are defined as follows:

→ Branch metric is defined as measure of discrepancy (or Hamming distance between transmitted signal and Received signal. It is indicated below that branch like ( )

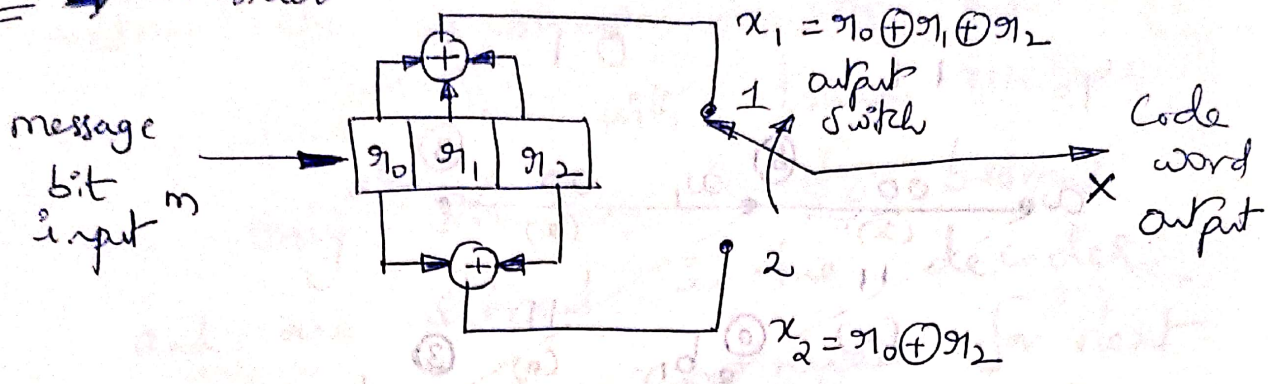
→ Path metric is defined as sum of all individual branch metrics along that path. It is indicated above/below the node like ○

→ Surviving Paths is defined as the paths of the decode signal with smallest (minimum) metric. The paths that are retained are called as survivors.

No. of surviving paths = $2^{(K-1)k}$ where K is constrained length and k is no. of message bits that can enter encoder at a time.

→ Metric diversion effect is defined as for two surviving paths originating from the same node, the running metric of less likely path tends to increase more rapidly than the metric of other paths $(K-1)k$ branches from common node. Hence because of not selecting less likely path having high metric & selecting only survival paths, memory storage is reduced.

Ex: → Consider the encoder shown in figure



→ If $M = [1\ 0\ 0\ 1\ 1]$ is message bit input sequence, then Transmitted Code word output is

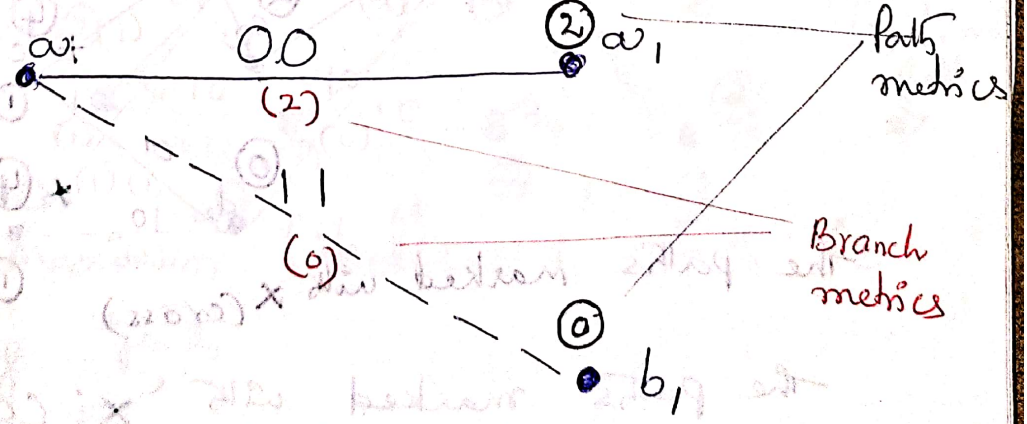
$$X = [1\ 1\ 1\ 0\ 1\ 1\ 1\ 1\ 0\ 1]$$

Let the Received Code word is

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

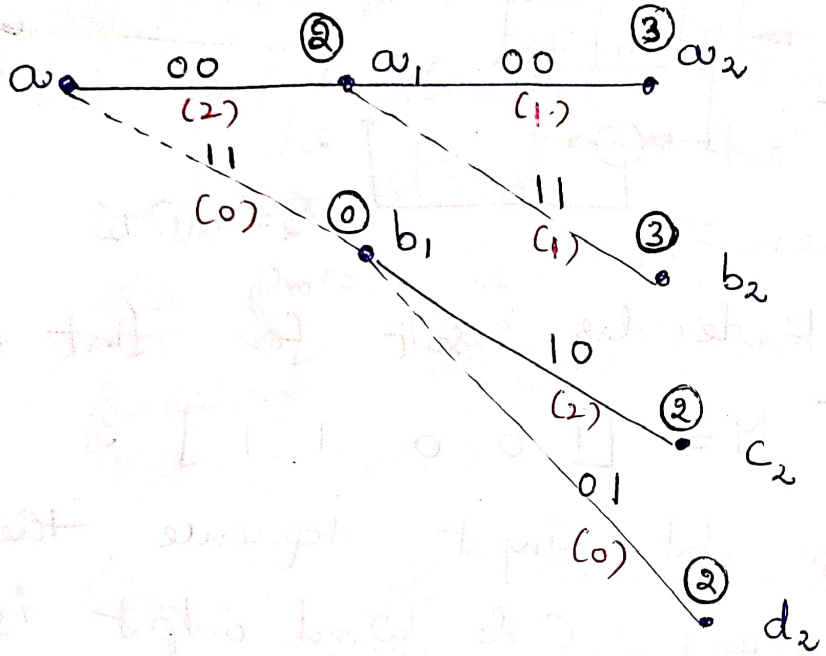
→ Viterbi decoder for first message bit

$Y = 11$



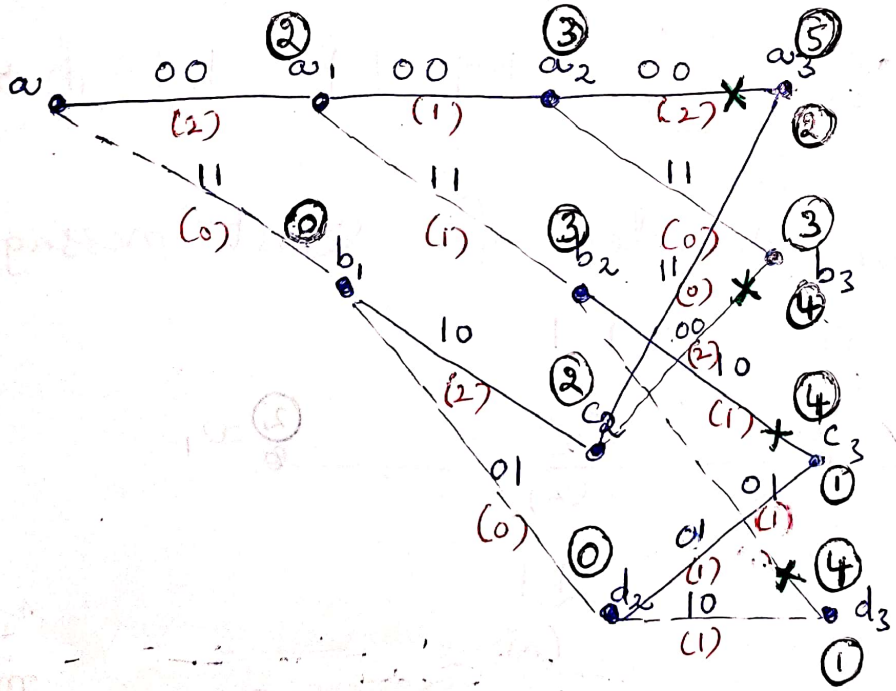
→ Viterbi decoder for second message bit

Y= 11 01



→ Viterbi decoder for third message bit

Y= 11 01 11

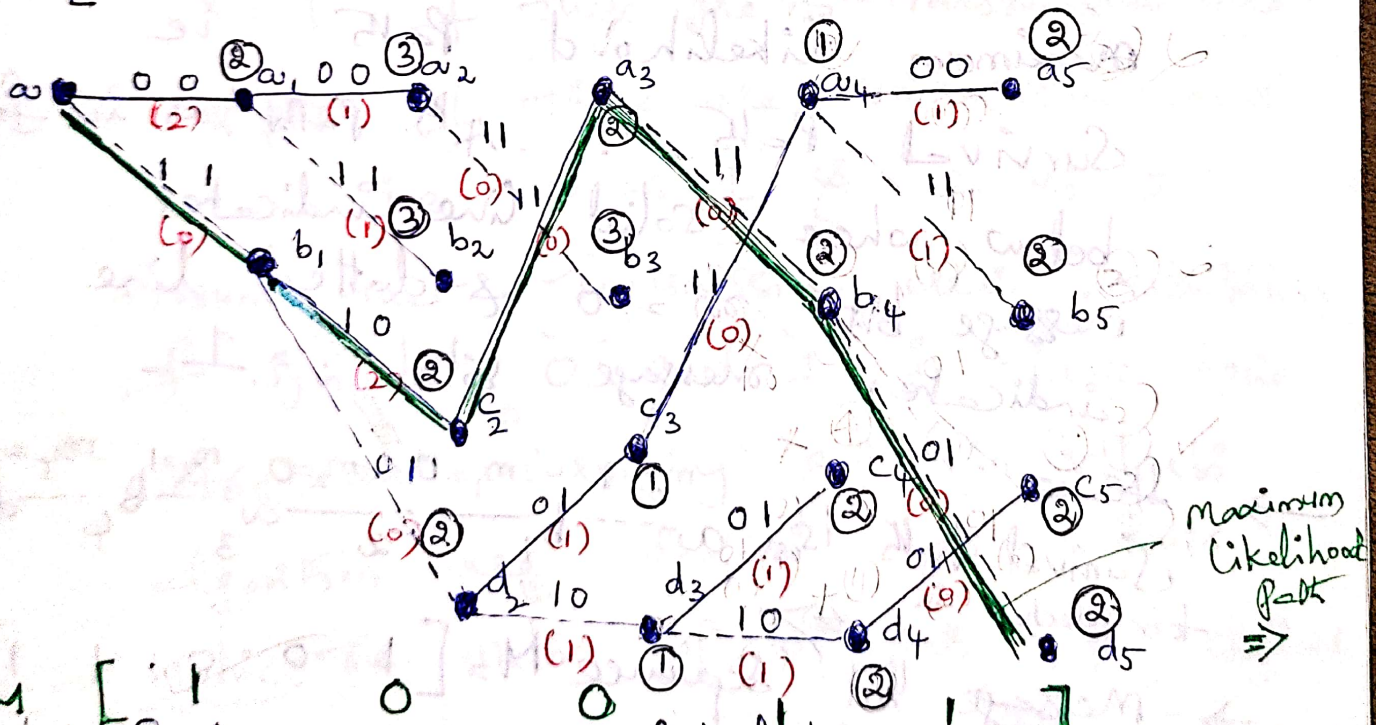


The paths marked with 'X' (cross) are cancelled because of having higher metrics ⑤, ④, ④, ④ when compared to the metrics of other paths coming to that

particular node ②, ③, ①, ① at a_3, b_3, c_3, d_3 respectively. as shown in fig. Now these four surviving paths with lower metrics are only shown in fig. below. and are stored in the decoder. This process is continued for next received bits (for fourth and fifth message bits)

→ Decoding of Convolutional Code by using Viterbi Algorithm

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



M [1 0 0 1 1]

→ finding maximum likelihood path ie surviving path

As four nodes a_5, b_5, c_5, d_5 are having same path metric ie ②, consider their branch metrics

1st path

$a \cdot (0) \cdot b_1 \cdot (0)\ d_2\ (1) \cdot c_3\ (0)\ a_4\ (1)\ a_5$

2nd path

$a\ (0)\ b_1\ (0)\ d_2\ (1)\ c_3\ (0)\ a_4\ (1)\ b_5$

3rd path

$a\ (0)\ b_1\ (0)\ d_2\ (1)\ d_3\ (1)\ d_4\ (0)\ c_5$

4th path

$a\ (0)\ b_1\ (2)\ c_2\ (0)\ a_3\ (0)\ b_4\ (0)\ d_5$

→ In above four paths, as 4th path is having more no. of branch metrics as '0's when compared to the other paths (1st, 2nd, 3rd) maximum likelihood path ie survived path is 4th path as in fig. below where solid line indicates message bit $m = 0$ & dotted line indicates message bit $m = 1$.

→ Hence survived path is $a \xrightarrow{m_0=1} b_1 \xrightarrow{m_1=0} c_2 \xrightarrow{m_2=0} a_3 \xrightarrow{m_3=1} b_4 \xrightarrow{m_4=1} d_5$

to be selected is

⇒ Message bit sequence $M = [1\ 0\ 0\ 1\ 1]$

Decoding of Convolutional Codes:

The methods used for decoding of Convolutional Codes are:

- Viterbi Algorithm
- Sequential Decoding
- Feedback Decoding

Viterbi Algorithm ie Maximum Likelihood Decoding

Let 'X' be the Transmitted Code Vector and 'Y' be the Received Code Vector

$X = Y$ if there are no Transmission errors

$X \neq Y$ if there are transmission errors i.e '0' as '1' & '1' as '0'

Assume that Transmission error probability of symbols '0' and '1' is same. The metric and surviving path in viterbi algorithm are defined as follows:

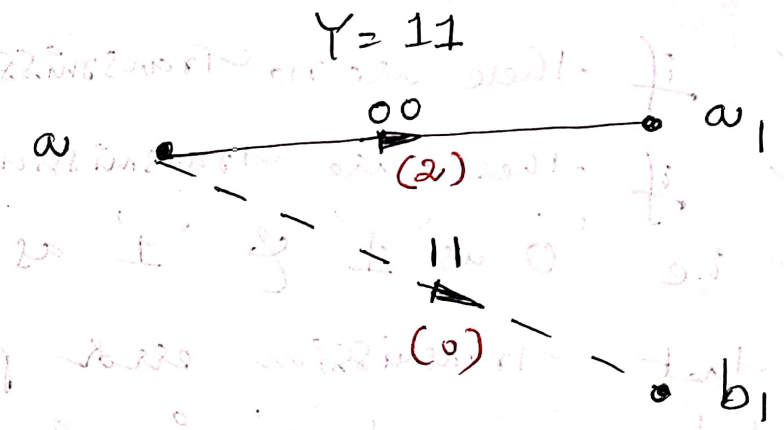
Metric: The metric of a path is defined as the discrepancy between the received signal and decoded output signal.

i.e Hamming Distance

for example

Received Signal Y	Decoded Signal X	Metric ie discrepancy (or Hamming distance)
01	00	1
01	01	0
01	10	2
01	11	1
000	101	2
000	111	3
000	000	0

It is indicated as follows in below that P=5.



Operation of Convolutional encoder by using State Transition Table, Code Tree, Trellis Diagram and State Diagram.

Consider the convolutional Encoder shown in fig. below.

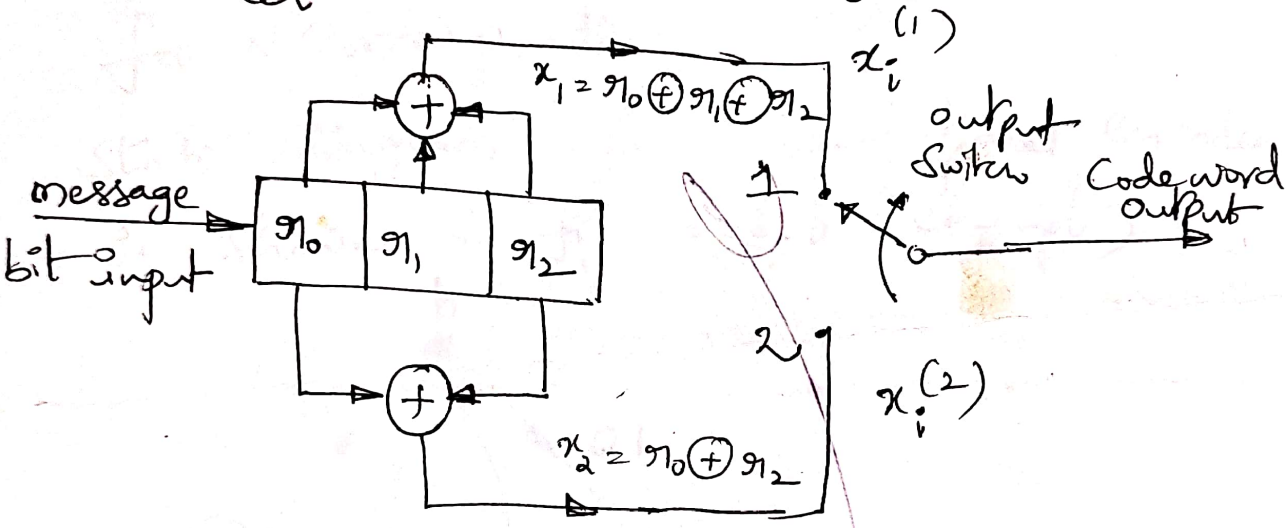


fig: Convolutional Encoder.

→ let m_1 & m_2 be the state of the encoder

Binary Description m_1	m_2	State
0	0	a
1	0	b
0	1	c
1	1	d

State diagram

State diagram is the pictorial representation of the behaviour of a system in which transition of states from Present state (PS) to the Next state (NS) and output for a corresponding input.

State diagram for Convolutional encoder is shown in fig below (example)

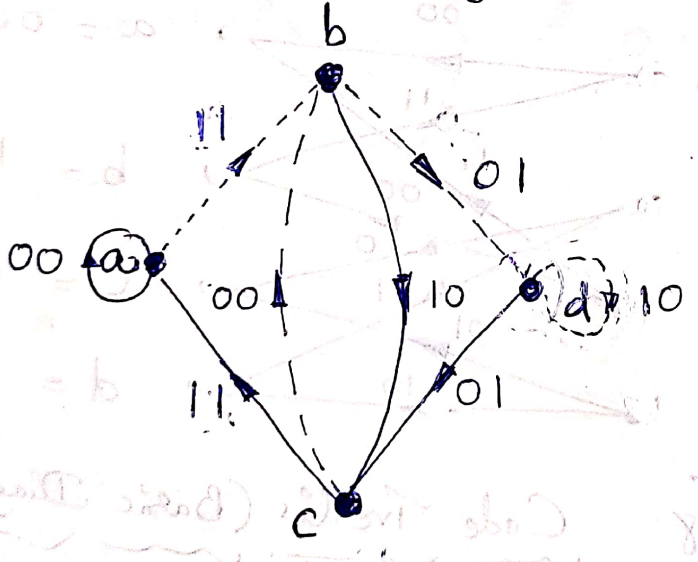
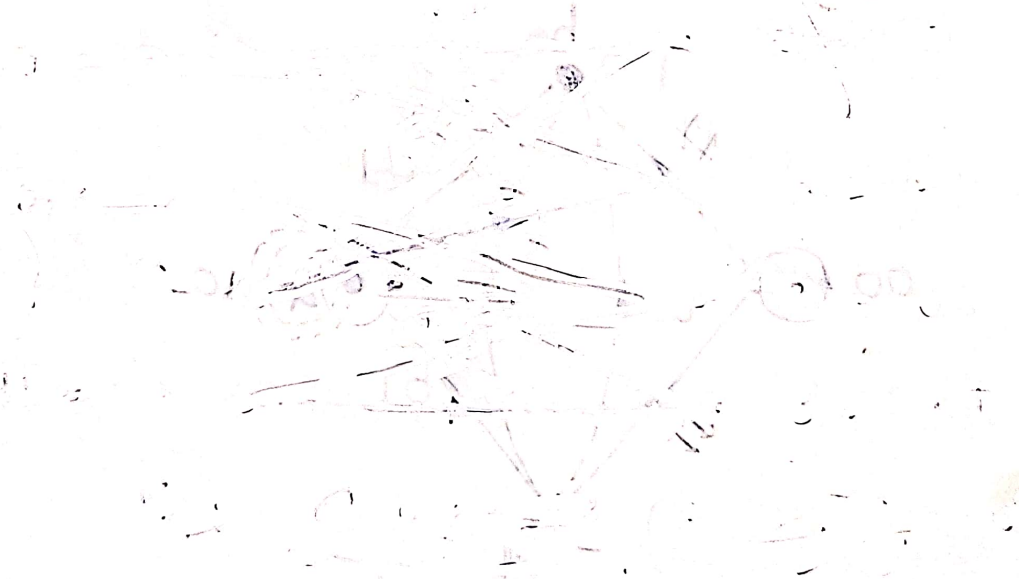


Fig: state diagram

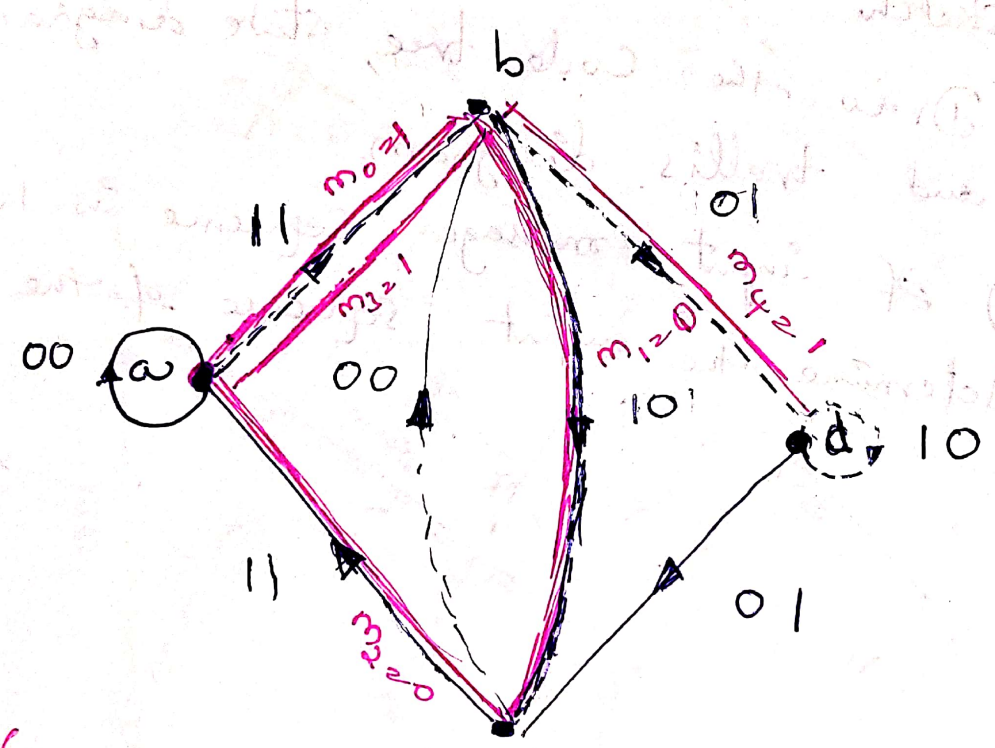
four nodes denote four possible current (or Present state (PS) and Path between them represents transition to next state (NS)
The solid line ——— represents input message m=0 and broken line '----' represents input message m=1.
Along with each transition line the output x_1x_2 is represented above the arrow mark like $\xrightarrow{x_1x_2}$

Code Trellis (or) Trellis structure

Code trellis is the more compact representation of the code tree in which every state goes to some other state depending upon the input message bit where as in Trellis, it represents single and unique diagram for such transitions. Trellis structure for convolutional encoder is shown in fig below.



The nodes on left side denote four possible current (or) Present state (PS) and those on right side represent next state (N.S). The solid line ——— represents input message $m = 0$ and broken line - - - - - represent input message $m = 1$. Along with each transition line the output $x_1 x_2$ is represented above the arrow mark like this $\xrightarrow{x_1 x_2}$.



If $M_2 = [1\ 0\ 0\ 1\ 1]$

then $x_i = [11\ 10\ 11\ 11\ 01]$

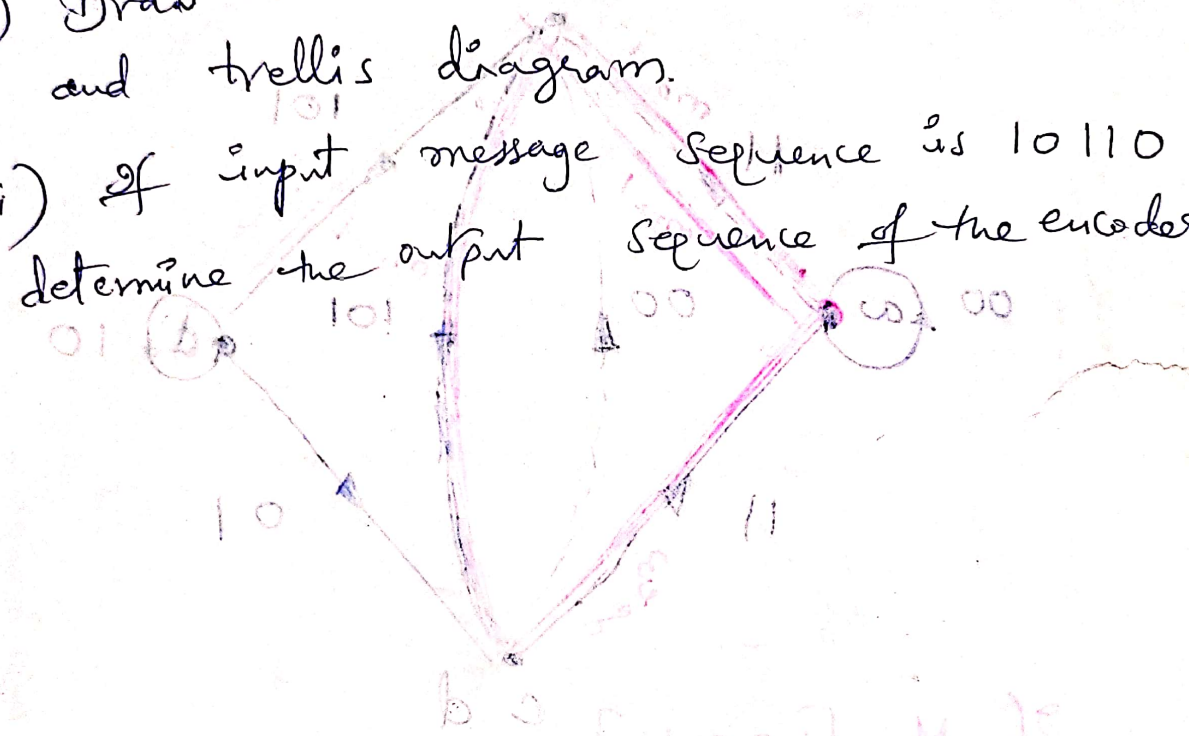
Ex: A rate 1/3 convolutional encoder has generating vectors as $g_1 = [1\ 0\ 0]$, $g_2 = [1\ 1\ 1]$ and $g_3 = [1\ 0\ 1]$

i) Sketch the encoder configuration.

ii) Draw the code tree, state diagram and trellis diagram.

iii) If input message sequence is 10110 determine the output sequence of the encoder

Sol.



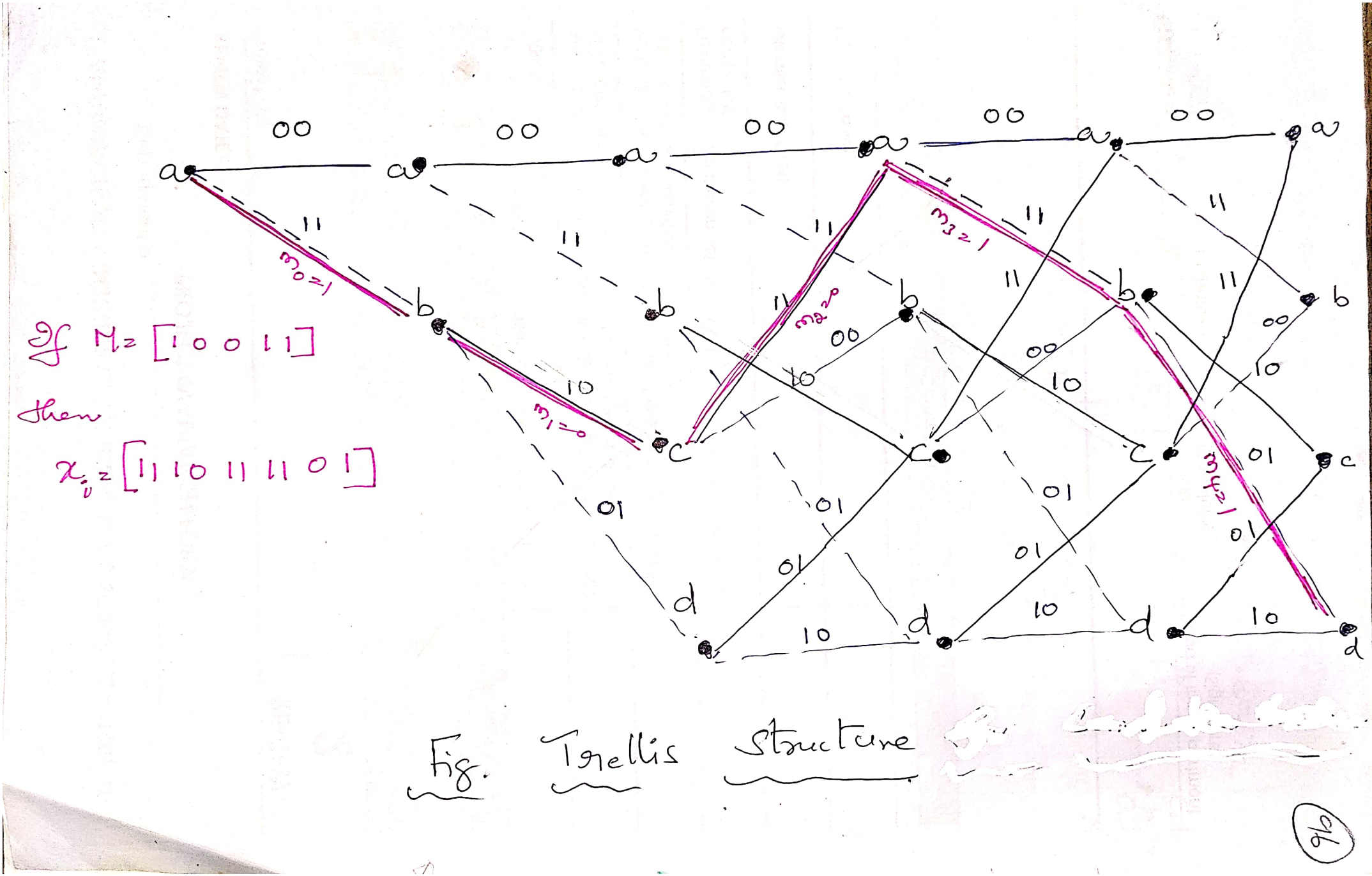
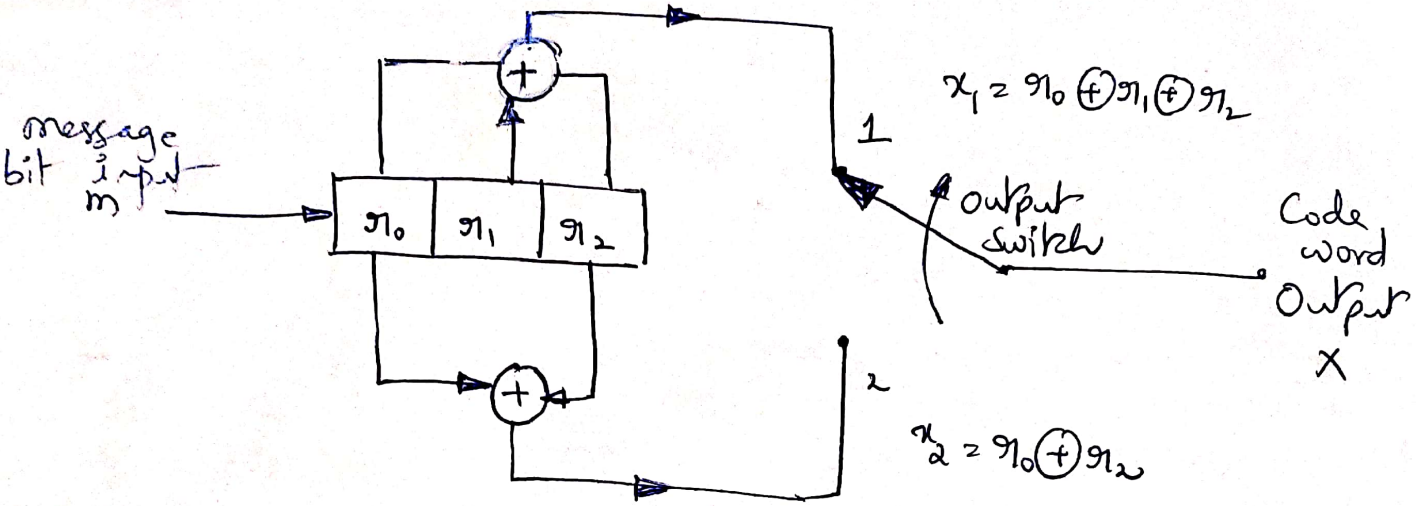


Fig. Trellis Structure

Ex:- Consider the encoder shown in figure



Given that Transmitted code word output for the message $M = [1\ 1\ 0\ 1]$ is given by $X = [11\ 01\ 01\ 00\ 01]$ and received code word is $Y = [11\ 01\ 11\ 00\ 01]$. Find Survival path for decoding Y to get M by using Viterbi Algorithm.

1. Define Error Control Coding. Give examples.

Ans. Error Control Coding refers to the technique of adding redundancy to the transmitted data so that errors introduced during transmission can be detected and/or corrected. It improves the reliability of digital communication.

Examples: Parity codes, Hamming codes, Cyclic codes, BCH codes, Reed–Solomon codes.

2. Explain Hamming Weight and Hamming Distance with example.

Ans**.**

- **Hamming Weight (w):** Number of 1's in a binary code word. Example: w(101100) = 3.
- **Hamming Distance (d):** Number of bit positions in which two codewords differ. Example: d(101100, 100110) = 2.

  The minimum distance of a code determines its error detection & correction capability.

3. What is Syndrome? Write expression for Syndrome.

Ans. A **Syndrome** is a vector used to detect and locate the presence of errors in received codewords with respect to transmitted code words X.

It is obtained by multiplying the received vector R *i.e Y* with the Transpose of the Parity check matrix *H*. $S = R.H^T$ or $S = Y.H^T$. If $\mathbf{S = 0}$, no error; if $\mathbf{S \neq 0}$, error exists.

4. Differentiate Hard Decision and Soft Decision Coding

Ans.

Hard Decision Coding	Soft Decision Coding
The received signal values are converted directly into binary form (0 or 1) for decoding.	The actual received signal values are used for decoding.
Only the sign or threshold is considered, so amplitude information is lost.	Confidence levels and reliability information are used, leading to better error correction.
Less computational complexity but lower decoding performance and higher Bit Error Rate (BER).	More computational complexity but better decoding performance and lower Bit Error Rate (BER).

5. Example for 1D and 2D Parity Check.

Ans.

- **1D Parity:** Add one parity bit to data word to the end.

  Example: 1011 (even parity) → 10111 whereas 1011 (odd parity) → 10110

For even parity, if received code is 10101, parity becomes odd → Error detected.
For odd parity, if received code is 11101, parity becomes even → Error detected.

- **2D Parity:** Arrange data in a matrix and add parity bits to last row and last column.

  Example data block with row & column parities.

Transmitted Code Block

Transmitted Data Bits with Even Parity			
1	1	0	0
1	0	0	1
1	1	1	1
1	0	1	0

Received Code Block

Received Data Bits with Parity Bits (Without Error)				No. of 1's ↓
1	1	0	0	even
1	0	0	1	even
1	1	1	1	even
1	0	1	0	even
even	even	even	even	←No. of 1's

Received Code Block

Received Data Bits with Parity Bits (With Error)				No. of 1's ↓
1	1	0	0	even
1	1	0	1	odd
1	1	1	1	even
1	0	1	0	even
even	odd	even	even	←No. of 1's

7. Relation between Generator Matrix and Parity Check Matrix.

Ans. For a linear (n, k) block code: If the generator matrix G is in **systematic form**:

$$G_{k\times n} = [I_{k\times k} : P_{k\times q}]$$

Then the corresponding **Parity Check Matrix H** is:

$$H_{q\times n} = [P^T_{q\times k} : I_{q\times q}]$$

Where I is Identity Matrix and P is Submatrix.

Relation between G and H is: $G.H^T = 0$

8. Linearity and Cyclic Shift Properties of Cyclic Code.

Ans. If $X = (x_0, x_1, x_2, \ldots, x_{n-1})$ is a valid cyclic codeword,

➤ **Linearity Property:**

Modulo sum (XOR) of any two codewords is also another valid codeword. If

$x_1, x_2 \in X \Rightarrow x_1 \oplus x_2 \in X$

➤ **Cyclic Shift Property:**

Any cyclic shift (left or right) of a codeword is another valid codeword. If

$x \in X \Rightarrow \text{CyclicShift}(x) \in X$

9. Compare Systematic and Non-Systematic Cyclic Codes.

Ans.

Systematic Cyclic Code	Non-Systematic Cyclic Code
Message bits followed by check bits in the codeword and hence directly visible.	Message bits are mixed with code bits and hence are not directly visible.
X = [m0 m1 m2 … mk-1: c0 c1 c2 … cq-1] 1×n	X = [x0 x1 x2 x3 x4 x5 x6 … xn-1] 1×n
Easy decoding.	Complex decoding.

10. Define BCH Codes.

Ans. BCH (Bose–Chaudhuri–Hocquenghem) codes are cyclic error-correcting codes used for correcting multiple random bit errors as well as burst errors. They are constructed using Galois Field (GF) arithmetic, which provides a systematic and guaranteed error-correction capability " t ". i.e

$$t = \left\lfloor \frac{d_{min} - 1}{2} \right\rfloor$$

Where:

- t= number of errors the code can correct
- d_{min}= minimum Hamming distance of the code

BCH codes are widely used in communication and storage systems due to their **high reliability and efficient decoding algorithms**.

11. What is Reed–Solomon Codes?

Ans. Reed–Solomon (RS) codes are non-binary block codes used for burst error correction. They are operated over Galois Field: GF (q) with error correction capability " t ".

$$t = \frac{n - k}{2}$$

Where:

- n= codeword length
- k= number of message symbols
- t= number of **symbol errors** the code can correct.

The RS Codes are are widely used in CDs, DVDs, QR codes, Satellite communication.

12. Define Hadamard Code and write H_2 and H_4.

Ans. Hadamard codes are **linear block codes** constructed from **Hadamard matrices**, which contain entries of **+1 and –1** (or equivalently **0 and 1** in binary form).

They have **large minimum Hamming distance**, which makes them useful for **error detection and correction**, especially in communication systems.

The **Hadamard matrix** is constructed recursively as:

$$H_{2n} = \begin{bmatrix} H_n & H_n \\ H_n & -H_n \end{bmatrix}$$

Hadamard Matrices are:

$$H_2 = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

$$H_4 = \begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & -1 & 1 & -1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}$$

13. How to get G from G(p) in Non-systematic cyclic codes?

Ans.

- Write the generator polynomial

$G(p) = g_0 + g_1 p + g_2 p^2 + \cdots + g_{n-k}\, p^{n-k}$,

where n= code length and k= message length.

- Form the first row of Gusing the coefficients of $G(p)$and pad with zeros to length n:
- $[g_0 \;\; g_1 \;\; g_2 \;\; \cdots \;\; g_{n-k} \;\; 0 \;\; 0 \;\; \cdots \;\; 0]$.
  Generate the remaining $k-1$ **rows** by shifting the previous row **one position to the right** (inserting 0 at the left) until you have k rows so that Generator matrix [G]kxn is obtained.

Example: Let

$G(p) = 1 + p + p^3, n = 7, k = 4.$

Coefficient row:

$$[1 \;\; 1 \;\; 0 \;\; 1 \;\; 0 \;\; 0 \;\; 0].$$

Shift to form the generator matrix (4×7):

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

14. Define State Transition Table and Diagram in Convolutional Coding.

Ans.

- A State Transition Table is a Tabular representation of a convolutional encoder, where each row shows the Input applied, the Present State (PS), the Next State (NS), and the corresponding Outputs
- A State Transition Diagram is a Graphical Representation of state behaviour, where States are shown as Nodes, and Directed branches between them indicate state transitions. Each branch is labelled with input/outputs**.**

16. Define Code Tree and Trellis Diagram.

Ans**.**

- A Code Tree Diagram is a Tree-like Representation of all possible encoder output sequences for input bit sequences over time, where each level corresponds to a time step and each branch represents a possible output for a given input.

- A Code Trellis Diagram is a Grid-like Representation of expanded states, where states are repeated at each time unit forming vertical columns, and the branches between states represent state transitions with their corresponding input/output labels. The trellis is mainly used for Viterbi decoding.

17. Define Code Dimension, Code Rate and Constraint Length.

Ans. Let "k" be the number of message (input) bits taken at a time and "n" be the number of output bits produced by the encoder.

- Code Dimension or Code Parameters for a Block or Convolutional encoder is (n, k)
- Code Rate (R) is defined as the ratio of input bits to the output bits. i.e. R=k/n
- Constraint Length (K) is defined as the number of memory elements i.e. flip-flops in the encoder that determine how many past bits can influence the current output.

18. Give Steps in Viterbi Algorithm for maximum likelihood decoding?

Ans.

- Draw the Trellis Diagram representing state transitions over time.
- Compute Branch Metrics for each branch based on the received sequence.
- Compute Path Metrics by accumulating branch metrics along each path.
- Select Survivor Paths at each state by choosing paths with minimum path metric.
- Choose the Maximum Likelihood Path from survivor paths for decoding.

19. Define Turbo Codes.

Ans. Turbo Codes are forward error correction codes constructed by parallel concatenation of two convolutional encoders separated by an interleave.
They are decoded using iterative (soft-input soft-output) decoding and achieve performance close to the Shannon limit.

Code Rate R=k/n

Where k = Number of Input message bits, n = Number of Output encoded bits

20. Define LDPC Codes.

Low Density Parity Check (LDPC) Codes are Linear Block Codes defined by a Sparse Parity-Check matrix "H", where most entries are zero. They use iterative belief propagation decoding and achieve performance close to the Shannon limit.

Parity-Check Condition**:** $H.X^{T}=0$. Where "X" is a valid LDPC codeword

Code Rate R=k/n

Where k = Number of Input message bits, n = Number of Output encoded bits