

Subject : **Digital Communication**
Code : **UEC 607**
Credit : **4**

Dr. Amit Mishra

Courtesy: Digital Communication-Simon Haykin

Error-Control Coding

Need of Error-Control Coding

- The three different channel-impairment scenarios for the data transmission over communication channels are –
 - ❖ AWGN channel – Satellite communication
 - ❖ Intersymbol interference - Telephone channel
 - ❖ multipath fading - Wireless channel
- These three scenarios are naturally different from each other, but they do share a common practical shortcoming: *reliability*. This is where the need for *error control coding* assumes paramount importance.
- The goal of design is to provide
 - a cost-effective facility for transmitting information from one end of the system *at a rate and level of reliability and quality* that are acceptable to a user at the other end.

Continued...

- The Probability of Error for any modulation scheme over a AWGN channel depends on
 - ❖ E_b/N_0 (*Signal energy per bit-to-noise power spectral density ratio*)
- This E_b/N_0 depends on
 - ❖ Transmitted signal power, channel bandwidth
 - ❖ Power spectral Density of Noise
- To ensure acceptable data quality for different application of communication system (i.e Audio, video, text, etc), there is some acceptable value of E_b/N_0 and Bit-Error Rate
- Hence, for a fixed E_b/N_0 , the only practical option available for changing data quality from problematic to acceptable is to use ***error-control coding***.

Continued...

Another practical motivation for the use of coding is

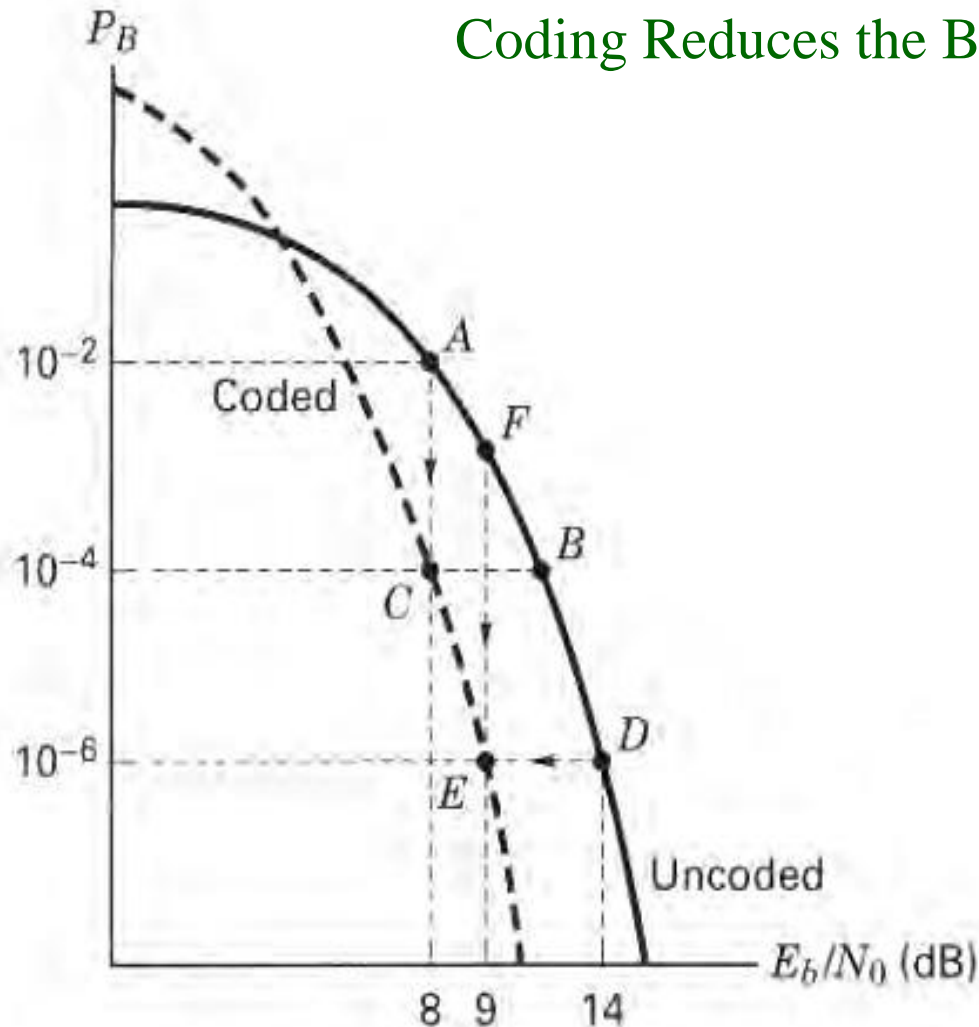
- ❖ To reduce the required E_b/N_0 for a fixed BER.
- ❖ This reduction in E_b/N_0 may, in turn, be exploited
 - to *reduce* the *required transmitted power*
 - or to reduce the *hardware costs* by requiring a smaller antenna size in the case of radio communications.

Comparison of Probability of Error for Coded and Un-coded

Coding Reduces the Bit error rate at high value of E_b/N_0

Prob. of Error ----- Decreases
Transmitted Power --- Decreases

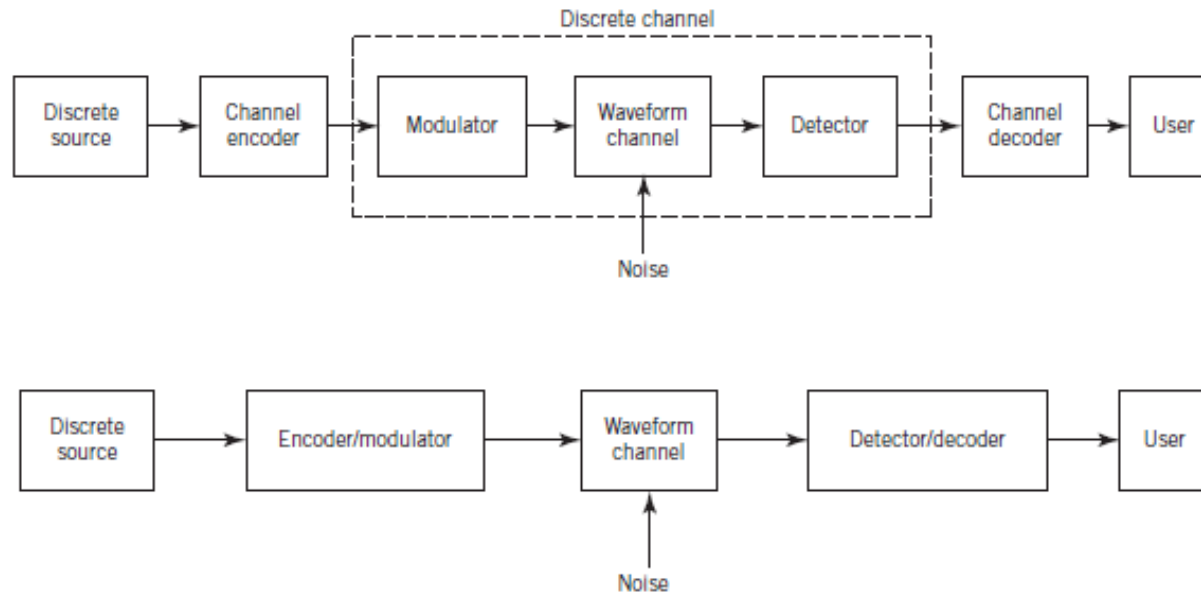
Cost of these two advantages
Signal Bandwidth ---- Increases
System complexity --- Increases
Data Rate ---- Increases



Design Trade-off

- ❖ The design trade-offs in the use of error-control coding is
 - **Acceptable error performance Vs bandwidth .**
(Performance Improved at the cost of Increase in signal bandwidth)
 - **Acceptable error performance Vs system complexity.**
(Performance Improved at the cost of Increase in system complexity)
 - **Power Vs Bandwidth.**
(Performance Improved at less E_b/N_0 but at the cost of Increase in signal bandwidth and system complexity)
 - **Acceptable error performance Vs Data Rate.**
(Performance Improved at less E_b/N_0 but at the cost of Increase in data rate)

Model of digital communication system



- ❖ The **discrete source generates information** in the form of binary symbols.
- ❖ The **channel encoder** accepts message bits and **adds redundancy** according to a prescribed rule, thereby producing an encoded data stream at a higher bit rate.
- ❖ The **channel decoder** in the **receiver exploits the redundancy** to decide which message bits in the original data stream, given a noisy version of the encoded data stream, were actually transmitted.
- ❖ The combined goal of the channel encoder and decoder is to minimize the effect of channel noise.

Discrete Memoryless Channels

- ❖ The waveform channel is said to be *memoryless* if in a given interval the detector **output depends only on the signal transmitted in that interval** and not on previous transmission.
- ❖ Under this condition, we may model the combination of the **modulator**, **the waveform channel**, and **the demodulator (detector)** as a *discrete memoryless channel*.

discrete memoryless channel = **Modulator**
+
waveform channel
+
the demodulator (detector).

Discrete Memoryless Channels

Such a channel is completely described

❖ by the *set of transition probabilities* denoted by $p(j|i)$,

where

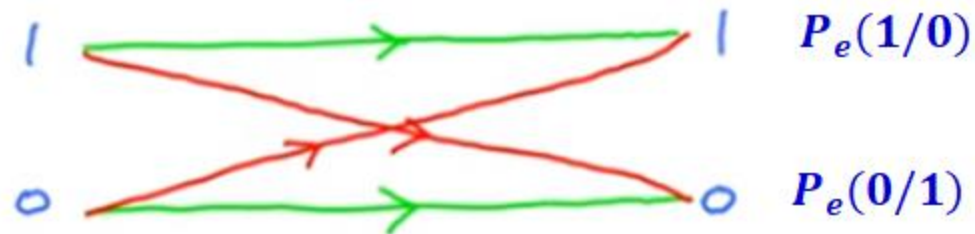
i denotes a modulator input symbol,

j denotes a demodulator output symbol, and

$p(j|i)$ is the probability of *receiving symbol “ j ”* given that symbol “ i ” was sent.

Example:

Binary symmetric channel (BSC)



Channel coding theorem

If a discrete memoryless channel has **capacity C** and a source generates **information at a rate less than C** , then there exists a coding technique such that the output of the source may be transmitted over the channel with an arbitrarily low probability of symbol error.

Types of Coding

- There are many different error-correcting codes that we can use –
 - *block codes*
 - *convolutional codes*.
- The distinguishing feature for this particular classification is the **presence or absence of *memory* in the encoders** for the two codes.

Block Code

- ❖ To generate an (n, k) block code,
 - the channel encoder accepts information in successive **k -bit message blocks**;
 - for each block, it **adds $n - k$ redundant bits** that are algebraically related (**even/odd parity** etc.) to the k message bits,
 - thereby producing an **overall encoded block of n bits**, where $n > k$.
- ❖ The **n -bit block** is called a **codeword**, and **n** is called the **block length** of the code.
- ❖ The channel encoder produces bits at the rate **$R_0 = (n/k)R_s$** , where **R_s** is the **bit rate of the information source**.
- ❖ The dimensionless ratio **$r = k/n$** is called the **code rate**, where $0 < r < 1$.
- ❖ The bit rate **R_0** , coming out of the encoder, is called the **channel data rate**.

Convolutional code

- In a convolutional code, the encoding operation may be viewed as the *discrete-time convolution* of
 - *the input sequence with the impulse response of the encoder*
- The duration of the impulse response equals the memory of the encoder.
- Accordingly, the encoder for a convolutional code operates on the incoming message sequence, using a “sliding window” equal in duration to its own memory.
- This, in turn, means that in a convolutional code, unlike in a block code, the channel encoder accepts message bits as a continuous sequence and thereby generates a continuous sequence of encoded bits at a higher rate.

Linear Block Codes

❖ A code is said to be **linear**

“if any two codewords in the code can be added in modulo-2 arithmetic to produce a third codeword in the code”

❖ Consider, an (n, k) linear block code, in which k bits of the n code bits are always identical to the **message sequence** to be transmitted. The $(n - k)$ bits in the remaining portion are computed from the message bits in accordance with a prescribed **encoding rule** that determines the mathematical structure of the code.

❖ Accordingly, these $(n - k)$ bits are referred to as **parity-check bits**.

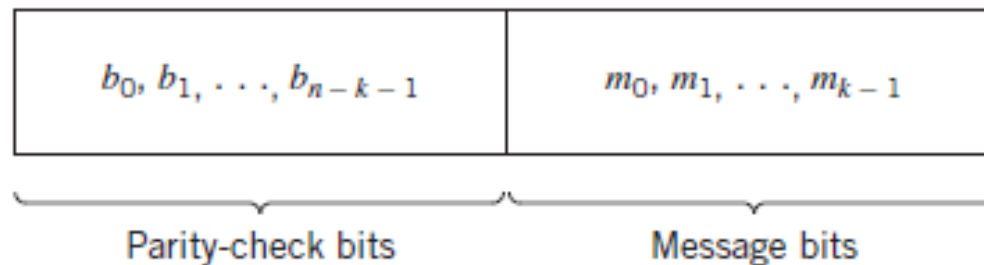
❖ **Block codes** in which the message bits are transmitted in unaltered form are called **systematic codes**.

❖ For applications requiring *both* **error detection** and **error correction**, the use of **systematic block codes** simplifies implementation of the decoder.

Structure of systematic codeword

- ❖ m_0, m_1, \dots, m_{k-1} block of k arbitrary message bits.
- ❖ $b_0, b_1, \dots, b_{n-k-1}$ $(n - k)$ *parity-check bits* in the codeword
- ❖ c_0, c_1, \dots, c_{n-1} n -bit code-word
- ❖ The $(n - k)$ leftmost bits of a code-word are identical to the corresponding parity-check bits and the k rightmost bits of the codeword are identical to the corresponding message bits. So the codeword is

$$c_i = \begin{cases} b_i, & i = 0, 1, \dots, n - k - 1 \\ m_{i+k-n}, & i = n - k, n - k + 1, \dots, n - 1 \end{cases}$$



Generator Matrix

- ❖ The $(n - k)$ **parity-check** bits are *linear sums* of the k **message bits**, as shown by the generalized relation

$$b_i = p_{0i}m_0 + p_{1i}m_1 + \dots + p_{k-1,i}m_{k-1} \quad \mathbf{b} = \mathbf{mP}$$

- ❖ where the coefficients are defined as

$$p_{ij} = \begin{cases} 1 & \text{if } b_i \text{ depends on } m_j \\ 0 & \text{otherwise} \end{cases}$$

- ❖ The coefficients p_{ij} are chosen in such a way that the rows of the **generator matrix** are linearly **independent** and the **parity-check equations** are *unique*.
- ❖ The \mathbf{P} is the **k -by- $(n - k)$ coefficient matrix** defined by

$$\mathbf{P} = \begin{bmatrix} p_{00} & p_{01} & \dots & p_{0,n-k-1} \\ p_{10} & p_{11} & \dots & p_{1,n-k-1} \\ \vdots & \vdots & & \vdots \\ p_{k-1,0} & p_{k-1,1} & \dots & p_{k-1,n-k-1} \end{bmatrix}$$

This system of equations may be rewritten in a compact form using matrix notation .

$$\mathbf{m} = [m_0, m_1, \dots, m_{k-1}] \quad \text{1-by-}k \text{ message vector 'm'}$$

$$\mathbf{b} = [b_0, b_1, \dots, b_{n-k-1}] \quad \text{1-by-}(n-k) \text{ parity check vector 'b'}$$

$$\mathbf{c} = [c_0, c_1, \dots, c_{n-1}] \quad \text{1-by-}n \text{ code vector 'c'}$$

Thus \mathbf{c} may be expressed as a partitioned row vector in terms of the vectors \mathbf{m} and \mathbf{b} as:

$$\mathbf{c} = [\mathbf{b} \mid \mathbf{m}]$$

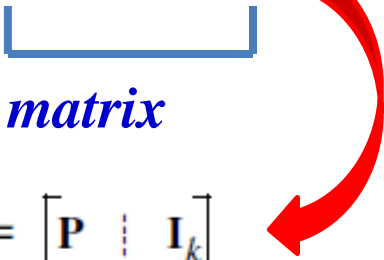
Using relation between \mathbf{b} , \mathbf{m} and \mathbf{P} as $\mathbf{b} = \mathbf{mP}$

$$\mathbf{c} = \mathbf{m} [\mathbf{P} \mid \mathbf{I}_k]$$

where \mathbf{I}_k is the k -by- k identity matrix

$$\mathbf{I}_k = \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}$$

The code vector is given as:

$$\mathbf{c} = \mathbf{m} \begin{bmatrix} \mathbf{P} & \vdots & \mathbf{I}_k \end{bmatrix}$$


Now, defining the ***k -by- n generator matrix***

$$\mathbf{G} = \begin{bmatrix} \mathbf{P} & \vdots & \mathbf{I}_k \end{bmatrix}$$

The **generator** matrix \mathbf{G} is said to be in the ***canonical form***, in that

- its ***k rows*** are linearly independent; that is,
- it is not possible to express any row of the matrix \mathbf{G} as a linear combination of the remaining rows.
- Using the definition of the **generator** matrix \mathbf{G} , the codeword can be written as—

$$\mathbf{c} = \mathbf{m}\mathbf{G}$$

Closure Property:

The sum of any two code words in the code is another codeword.

Proof:-

consider a pair of code vectors \mathbf{c}_i and \mathbf{c}_j corresponding to a pair of message vectors \mathbf{m}_i and \mathbf{m}_j , respectively.

Sum of \mathbf{c}_i and \mathbf{c}_j *can be expressed as –*

$$\begin{aligned}\mathbf{c}_i + \mathbf{c}_j &= \mathbf{m}_i \mathbf{G} + \mathbf{m}_j \mathbf{G} \\ &= (\mathbf{m}_i + \mathbf{m}_j) \mathbf{G}\end{aligned}$$

The modulo-2 sum (ex-or) of \mathbf{m}_i and \mathbf{m}_j represents a new message vector.

Correspondingly, the modulo-2 sum (ex-or) of \mathbf{c}_i and \mathbf{c}_j represents a new code vector.


Relation between Generator Matrix and Parity-check Matrix

- Let \mathbf{H} denote an $(n - k)$ -by- n matrix, defined as

$$\mathbf{H} = \begin{bmatrix} \mathbf{I}_{n-k} & \vdots & \mathbf{P}^T \end{bmatrix} \quad \mathbf{H} = \text{Parity check Matrix}$$

Where \mathbf{P}^T is an $(n - k)$ -by- k matrix, representing the transpose of the coefficient matrix \mathbf{P} , and \mathbf{I}_{n-k} is the $(n - k)$ -by- $(n - k)$ identity matrix. Therefore

$$\mathbf{G} = \begin{bmatrix} \mathbf{P} & \vdots & \mathbf{I}_k \end{bmatrix} \quad \mathbf{H}\mathbf{G}^T = \begin{bmatrix} \mathbf{I}_{n-k} & \vdots & \mathbf{P}^T \end{bmatrix} \begin{bmatrix} \mathbf{P}^T \\ \vdots \\ \mathbf{I}_k \end{bmatrix} = \mathbf{P}^T + \mathbf{P}^T \quad \mathbf{H}\mathbf{G}^T = \mathbf{0}$$

 Modulo-2 addition (ex-or operation)

In modulo-2 (ex-or) arithmetic, the matrix sum $\mathbf{P}^T + \mathbf{P}^T$ is $\mathbf{0}$, thus

$$\mathbf{c} = \mathbf{m}\mathbf{G}$$

Multiplying Code vector “ \mathbf{c} ” by \mathbf{H}^T ,

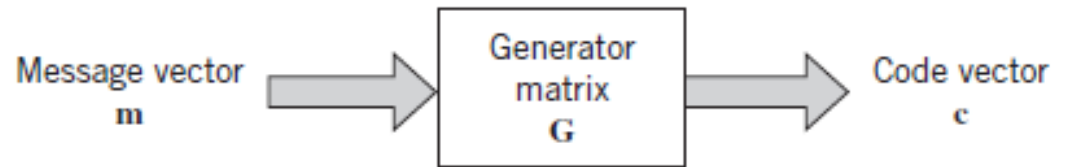
$$\mathbf{c}\mathbf{H}^T = \mathbf{m}\mathbf{G}\mathbf{H}^T = \mathbf{0}$$

Continued...

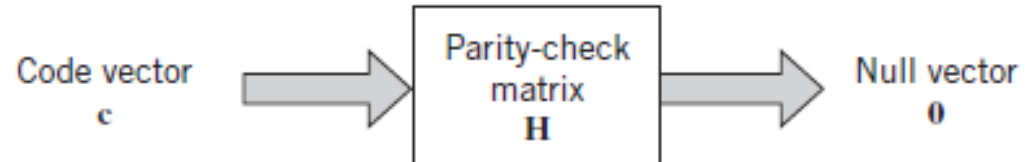
The matrix **H** is called the *parity-check matrix* of the code and the following equation is called *parity-check equation*.

$$\mathbf{cH}^T = \mathbf{mGH}^T = \mathbf{0}$$

$$\mathbf{c} = \mathbf{mG}$$




$$\mathbf{cH}^T = \mathbf{mGH}^T = \mathbf{0}$$



Example-1: Prepare the code word (c) corresponding to each message (m) using even parity bit.

- Based on rule of **‘Even Parity Code’**

Message	Parity	Codeword	$c = \begin{bmatrix} \mathbf{b} & \vdots & \mathbf{m} \end{bmatrix}$
000	0	0	000
100	1	1	100
010	1	1	010
110	0	0	110
001	1	1	001
101	0	0	101
011	0	0	011
111	1	1	111



Example-2: Consider a (6,3) Linear block code. Prepare code word (c) corresponding to each message vector (m) [from 000 to 111] based on rule of given k -by- n (6, 3) generator matrix $\mathbf{G} = [\mathbf{P} \mid \mathbf{I}_k]$

$$\mathbf{G} = \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

Solution: For $m=[1 \ 1 \ 0]$

$$\begin{aligned} \mathbf{c} = \mathbf{mG} \quad [1 \ 1 \ 0] \begin{bmatrix} \mathbf{V}_1 \\ \mathbf{V}_2 \\ \mathbf{V}_3 \end{bmatrix} &= 1 \cdot \mathbf{V}_1 + 1 \cdot \mathbf{V}_2 + 0 \cdot \mathbf{V}_3 \\ &= 1 \ 1 \ 0 \ 1 \ 0 \ 0 + 0 \ 1 \ 1 \ 0 \ 1 \ 0 + 0 \ 0 \ 0 \ 0 \ 0 \ 0 \\ &= 1 \ 0 \ 1 \ 1 \ 1 \ 0 \quad (\text{codeword for the message vector } 1 \ 1 \ 0) \end{aligned}$$

Message vector	Codeword
000	000000
100	110100
010	011010
110	101110
001	101001
101	011101
011	110011
111	000111

Syndrome (syndrome vector (**s**) depends on **error vector** (**e**) and **parity check matrix** (**H**))

- ❖ Let **r** is the received noisy code vector of size 1-by-n
- ❖ Therefore, the **vector r** can be expressed as the sum of the **original code vector c** and a **new vector e**, as shown by

$$\mathbf{r} = \mathbf{c} + \mathbf{e}$$

- ❖ The **vector e** is called the *error vector or error pattern*.
- ❖ The i_{th} element of **e** equals 0 if the corresponding element of **r** is the same as that of **c**.
- ❖ On the other hand, the i_{th} element of **e** equals 1 if the corresponding element of **r** is different from that of **c**,
- ❖ In this case an *error* is said to have occurred in the i_{th} location.

Continued...

- That is, for $i = 1, 2, \dots, n$,

$$e_i = \begin{cases} 1 & \text{if an error has occurred in the } i\text{th location} \\ 0 & \text{otherwise} \end{cases}$$

- The receiver has the task of decoding the code vector \mathbf{c} from the received vector \mathbf{r} .
- The algorithm commonly used to perform this decoding operation starts with the computation of a $1\text{-by-}(n - k)$ vector called the ***error-syndrome vector*** or simply the ***syndrome***.
- The importance of the syndrome lies in the fact that it depends only upon the error pattern.

Continued...

- Given a **1-by- n received vector \mathbf{r}** , the corresponding syndrome is formally defined as

$$\mathbf{s} = \mathbf{r}\mathbf{H}^T$$

- The syndrome depends only on the error pattern and not on the transmitted code word*

$$\mathbf{s} = \mathbf{r}\mathbf{H}^T$$

Since,

$$\mathbf{r} = \mathbf{c} + \mathbf{e}$$

Then,

$$\mathbf{s} = (\mathbf{c} + \mathbf{e})\mathbf{H}^T = \mathbf{c}\mathbf{H}^T + \mathbf{e}\mathbf{H}^T = \mathbf{e}\mathbf{H}^T$$

 **= $\mathbf{0}$ (already proved)**

Property-1 of Syndrome

- The parity check matrix **H** of a code permits us to compute the syndrome **s**, which depends only upon the error pattern **e**.
- Suppose that the **error pattern e** contains a pair of errors in locations i and j caused by the additive channel noise, as shown by

$$\mathbf{e} = [0 \dots 0 \mathbf{1}_i 0 \dots 0 \mathbf{1}_j 0 \dots 0]$$

- Then, substituting this error pattern yields the syndrome
- where \mathbf{h}_i and \mathbf{h}_j are respectively the i th and j th rows of the matrix \mathbf{H}^T .
Thus,

$$\mathbf{s} = \mathbf{h}_i + \mathbf{h}_j$$

- **For a linear block code, the syndrome **s** is equal to the sum of those rows of the transposed parity-check matrix \mathbf{H}^T where errors have occurred due to channel noise.**

Property-2 of Syndrome

All error patterns that differ by a codeword have the same syndrome

- For k message bits, there are 2^k distinct **code vectors** denoted as \mathbf{c}_i , where $i = 0, 1, \dots, 2^k - 1$. Correspondingly, for any error pattern \mathbf{e} we define the 2^k distinct vectors \mathbf{e}_i as follows

$$\mathbf{e}_i = \mathbf{e} + \mathbf{c}_i \quad \text{for } i = 0, 1, \dots, 2^k - 1$$

- The set of vectors $(\mathbf{e}_i, i = 0, 1, \dots, 2^k - 1)$ is called a ***coset*** of the code. In other words, a coset has exactly 2^k elements that differ at most by a code vector.
- Thus, an (n, k) linear block code has 2^{n-k} possible cosets.

Property-2 of Syndrome

All error patterns that differ by a codeword have the same syndrome

- In any event, multiplying both sides of $\mathbf{e}_i = \mathbf{e} + \mathbf{c}_i$ by the matrix \mathbf{H}^T and again using

$$\mathbf{c}\mathbf{H}^T = 0,$$

$$\mathbf{e}_i\mathbf{H}^T = \mathbf{e}\mathbf{H}^T + \mathbf{c}_i\mathbf{H}^T = \mathbf{e}\mathbf{H}^T$$

- Thus , each coset of the code is characterized by a unique syndrome which is independent of the index i .

Minimum Distance Considerations

- Hamming Distance
- Hamming Weight

Hamming Distance

- Consider a pair of code vectors \mathbf{c}_1 and \mathbf{c}_2 that have the same number of elements.
- The Hamming distance, denoted by $d(\mathbf{c}_1, \mathbf{c}_2)$, between such a pair of code vectors is defined as the number of locations in which their respective elements differ.

$$\mathbf{U} = 1 \ 0 \ 0 \ 1 \ 0 \ 1 \ 1 \ 0 \ 1$$

$$\mathbf{V} = 0 \ 1 \ 1 \ 1 \ 1 \ 0 \ 1 \ 0 \ 0$$

$$d(\mathbf{U}, \mathbf{V}) = 6$$

- In a corresponding way, a new parameter called the minimum distance d_{\min} , can be introduced for which :
- The minimum distance d_{\min} of a linear block code is the smallest Hamming distance between any pair of codewords.

Hamming Weight

- Consider a pair of code vectors \mathbf{c}_1 and \mathbf{c}_2 that have the same number of elements.
- The **Hamming weight** $w(\mathbf{c})$ of a code vector \mathbf{c} is defined as the **number of nonzero elements in the code vector**.
- Equivalently, it may be stated that the **Hamming weight** of a code vector is the **distance between the code vector** and the *all-zero code vector*.

Error correcting and Detecting Capability

- The **minimum distance** of a linear **block code** is defined by the **minimum number of rows** of the matrix \mathbf{H}^T whose sum is equal to the **zero vector**.
- Therefore, the **minimum distance** d_{\min} of a **linear block code** determines the ***error-correcting and detecting capability of the code***.
- An (n,k) linear block code can **detect** and **correct** all error patterns whose **Hamming weight** is **less than or equal** to t .

$$w(\mathbf{e}) \leq t$$

provided that the **minimum distance** of the code is **equal to or greater** than $2t + 1$.

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

$$t \leq \left\lfloor \frac{d_{\min} - 1}{2} \right\rfloor$$

$\lfloor x \rfloor$ --- Largest Integer not to increase x

Error correcting and Detecting Capability

Error detecting capability --- $(d_{min} - 1)$

Error correcting capability is ----- $t \leq \left\lfloor \frac{d_{min}-1}{2} \right\rfloor$

$\lfloor x \rfloor$ --- Largest Integer not to increase x

Hamming Codes

For any positive integer $m \geq 3$, there exists a **linear block code** with the following parameters:

code length	$n = 2m - 1$
number of message bits	$k = 2m - m - 1$
number of parity-check bits	$n - k = m$

Such a linear block code for which the error-correcting capability $t = 1$ is called a Hamming code.

Example-3

For a (6,3) code, the generator matrix G is

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

- Find
- All corresponding code vectors
 - Minimum Hamming distance
 - Verify that this code is a single-error correcting code.
 - Parity Check Matrix
 - Determine Transmitted codeword if received codeword is 100011.

codevector = messagebits + codebits

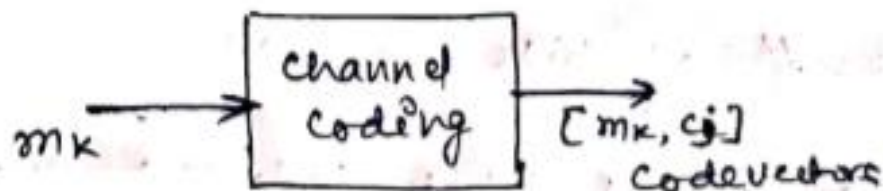
$$G = \begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

(n, k)

$(6, 3)$ code

$n = \text{no. of bits (codeword vector)}$

$k = \text{no. of bits (message vector)}$



$$G = [I : P]$$

I : Identity matrix

P : Parity bit matrix

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

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

$$\#(C_j) = n - k \\ = 6 - 3 = 3$$

$$C_j = m_k \oplus P$$

m_k = message bits vector

P = Parity bit Matrix

C_j = code bits

\oplus = modulo sum (XOR)

$$[C_0 \ C_1 \ C_2] = [m_0 \ m_1 \ m_2] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$$

$$C_0 = m_0 \oplus m_2$$


$$C_1 = m_1 \oplus m_2$$

$$C_2 = m_0 \oplus m_1$$


a) code vector corresponding to n
100 : 100101

b) minimum Hamming distance
 $d_{min} = 3$

8
code
vectors



m_0	m_1	m_2	C_0	C_1	C_2	Weight
0	0	0	0	0	0	1
0	0	1	1	1	0	3
0	1	0	0	1	1	3
0	1	1	1	0	1	4
1	0	0	1	0	1	3
1	0	1	0	1	1	4
1	1	0	1	1	0	4
1	1	1	0	0	0	3



c) ① $d_{\min} \geq s+1$

$$3 \geq s+1$$

$$\therefore s \leq 2$$

$s = \#$ of bits for error detection

or means among 6 bit code vector we can detect 2 bit's where error occurred.

② $d_{\min} \geq 2t+1$

$$3 \geq 2t+1$$

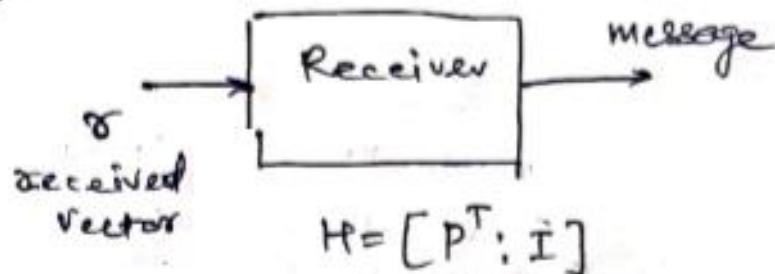
$$2t \leq 2$$

$$t \leq 1$$

$t = \#$ of bits for error correction

or means among 6 bit code vector we can detect 1 bit for error correction.

d)



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

$$s = r[H^T]$$

$s =$ error vector

Given $y = [100011]$

\therefore error $S = y[H^T] = [100011] \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} = [110]$

Bit correction

	1	2	3	4	5	6
$y =$	1	0	0	0	1	1

↑
3

Corrected y

$y = 101011$

This vector is not zero. There are two bits where error occurred.
We can correct 1 bit among

Example-4

Hamming Codes ($m = 3$)

$m = 3$, yielding the (7, 4) Hamming code with $n = 7$ and $k = 4$.

The generator of this code is defined by

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

$\underbrace{\hspace{10em}}_{\mathbf{P}} \quad \underbrace{\hspace{10em}}_{\mathbf{I}_k}$

The corresponding parity-check matrix is given by

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

$\underbrace{\hspace{10em}}_{\mathbf{I}_{n-k}} \quad \underbrace{\hspace{10em}}_{\mathbf{P}^T}$

Continued...

With $k = 4$, there are $2^k = 16$ distinct message words. For a given message word, the corresponding codeword is obtained by using $\mathbf{c} = \mathbf{mG}$

Message word	Codeword	Weight of codeword	Message word	Codeword	Weight of codeword
0000	0000000	0	1000	1101000	3
0001	1010001	3	1001	0111001	4
0010	1110010	4	1010	0011010	3
0011	0100011	3	1011	1001011	3
0100	0110100	3	1100	1011100	4
0101	1100101	4	1101	0001101	3
0110	1000110	3	1110	0101110	4
0111	0010111	4	1111	1111111	7

Since the smallest of the Hamming weights for the nonzero codewords is 3, it follows that the minimum distance of the code is 3.

Continued...

Suppose, the code vector [1110010] is sent and the received vector is [1100010] with an error in the third bit. So syndrome will be

$$s = rH^T$$

$$s = [1100010] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} = [0 \ 0 \ 1]$$

$$s = [0 \ 0 \ 1] = eH^T =$$

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

Continued...

As for modulo-2 operation, subtraction is same as addition, therefore, the correct transmitted codeword is given as

$$\mathbf{c} = \mathbf{r} + \mathbf{e}_0 \mathbf{c} = 1100010 \oplus 0010000 = 1110010$$

Similarly, the **error pattern** for all **eight syndrome** will be given as

Syndrome	Error pattern
000	0000000
100	1000000
010	0100000
001	0010000
110	0001000
011	0000100
111	0000010
101	0000001

Example-5

Consider a (6, 3) block code whose generator matrix is

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

1. **Find code words of the message - 110 , 101 and 111**
2. Find H, the parity check matrix of the code
3. Compute the syndrome for the received vector $\mathbf{r} = 001110$, Is this a valid code vector?

Find code words of the message - 110 , 101 and 111

The code word (U) can be generated from the generation matrix (G) by using the following expression:

$$U = m G$$

Where, U is the code word, m is the message word, and G is the generation matrix.

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} = \begin{matrix} V_1 \\ V_2 \\ V_3 \end{matrix}$$

$$U_4 = [1 \ 1 \ 0] \begin{bmatrix} V_1 \\ V_2 \\ V_3 \end{bmatrix} = 1 \cdot V_1 + 1 \cdot V_2 + 0 \cdot V_3$$

$$= 1 \ 1 \ 0 \ 1 \ 0 \ 0 + 0 \ 1 \ 1 \ 0 \ 1 \ 0 + 0 \ 0 \ 0 \ 0 \ 0 \ 0$$

$$= 1 \ 0 \ 1 \ 1 \ 1 \ 0 \text{ (codeword for the message vector } 1 \ 1 \ 0)$$

$$+ \begin{matrix} 1 \ 1 \ 0 \ 1 \ 0 \ 0 \\ 0 \ 1 \ 1 \ 0 \ 1 \ 0 \end{matrix}$$

$$1 \ 0 \ 1 \ 1 \ 1 \ 0 \text{ (codeword for the message vector } 1 \ 1 \ 0)$$

Similarly the code words for message 101 and 111 can be determine as

Message word	Code word
110	101110
101	011101
111	000111

(b) Find H, the parity check matrix of the code

To fulfill the orthogonality requirements for a systematic code, the H- matrix can be written as

$$\mathbf{H} = \left[\mathbf{I}_{n-k} \mid \mathbf{P}^T \right]$$

$$\mathbf{H}^T = \left[\begin{array}{c} \mathbf{I}_{n-k} \\ \mathbf{P} \end{array} \right]$$

$$G = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix} = \begin{matrix} V_1 \\ V_2 \\ V_3 \end{matrix}$$

P



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

(c) Compute the syndrome for the received vector $r = 0\ 0\ 1\ 1\ 1\ 0$, Is this a valid code vector?

$$S = r H^T$$

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

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

Since, Syndrome $-S$ is not equal to zero, hence this is not valid code vector.

Convolutional Codes

- In block coding, codewords are produced on a **block-by-block** basis.

(K-bit Message Block produces n-bit code block)

- Therefore, encoder require **a buffer** to store an entire message block before generating the associated codeword.
- There are applications, however, where the message bits come in *serially* rather than in large blocks, in which case the use of a buffer may be undesirable.
- In such situations, the use of *convolutional coding* may be the preferred method.
- A convolutional coder generates redundant bits by using *modulo-2 convolutions*; hence the name *convolutional codes*.

Continued...

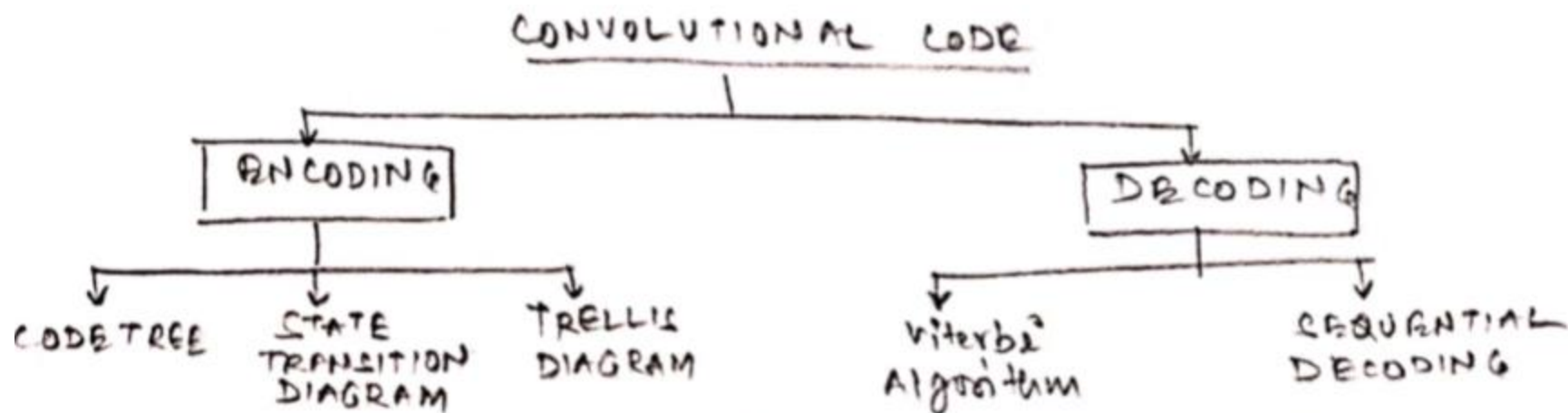
- Convolution Encoder may be viewed as a *finite-state machine* that consists of an *M-stage shift register* with prescribed connections to *n modulo-2 adders* and a *multiplexer* that serializes the outputs of the adders.
- Length of message sequence = L and Memory of Encoder = M
- Length of Coded output sequence = $n(L + M)$ bits,
- The *code rate* is therefore given by

$$r = \frac{L}{n(L + M)} = \frac{1}{n(1 + M/L)} \quad \text{bits/symbol}$$

$$r \approx \frac{1}{n} \quad \text{bits/symbol}$$

Since, $L \gg M$,

Convolutional Codes



Convolutional Codes

Non linear code: (n, k, l)
(represented by three parameters)

n = code word length
 k = message bit sequence length
 l = Number of output

Advantages

Highly secure:- Due to ^{large} code word length

convolutional code : $(12, 4, 2)$ $\left\{ \begin{array}{l} 2^{12} = 4096 \text{ bits} \\ 2^4 = 16 \text{ bits} \end{array} \right.$

Linear code (Block) : $(7, 4)$ $\left\{ \begin{array}{l} 2^7 = 128 \text{ bits} \\ 2^4 = 16 \text{ bits} \end{array} \right.$

message bits

- ① The probability of loss of message bit is less in convolutional code.
- ② No. of error correction bits are more in convolutional code

Continued...

$$\text{code rate} = \text{code efficiency} = \frac{k}{n}$$

Code word length

$$\text{code efficiency} = \frac{k}{k \cdot l} = \frac{1}{l}$$

If no. of bits entered in the encoder.



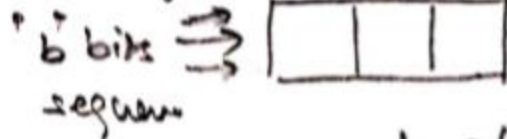
encoder shift register

$$\therefore n = (N + k - 1) \cdot l$$

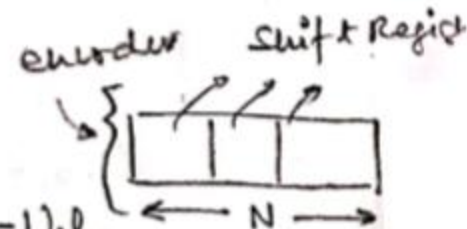
$\therefore k \gg N$ where $N = \text{No. of shift register in encoder}$
 $k = \text{message bits}$

$$\therefore n \approx k \cdot l$$

or - if



$$\text{code efficiency} = \frac{b}{l}$$

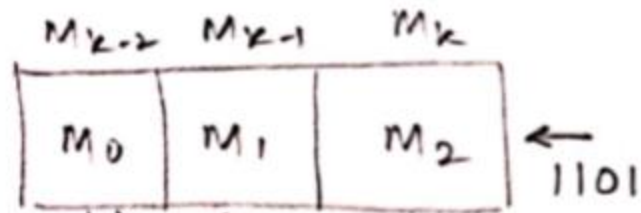


Example-1

$$l=2$$

$$N=3, K=4$$

$$n = (N+K-1) \cdot l$$



modulo
sum

outputs $\leftarrow x_1$

x_2

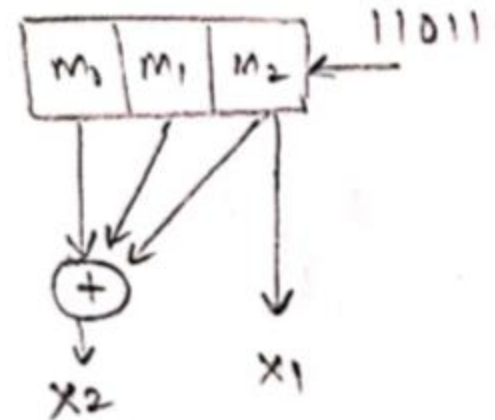
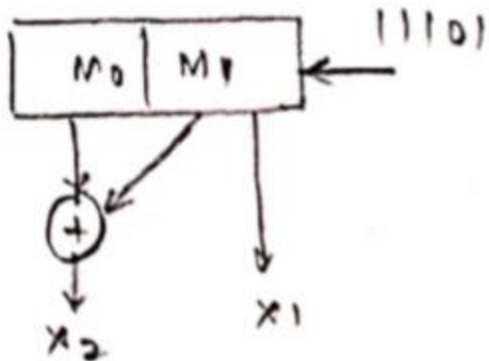
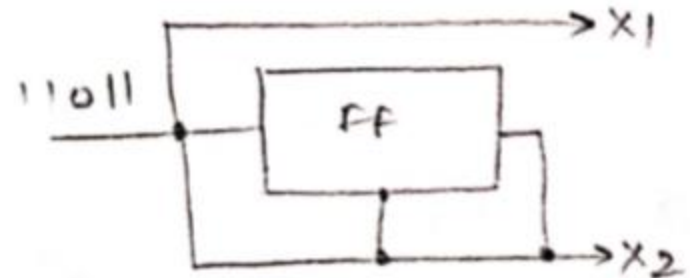
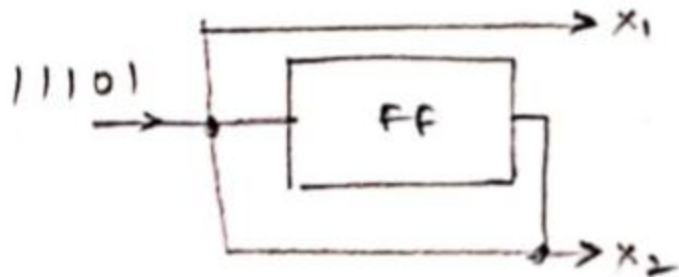
$$x_1 = m_0 \oplus m_1 \oplus m_2$$

$$x_2 = m_0 \oplus m_2$$

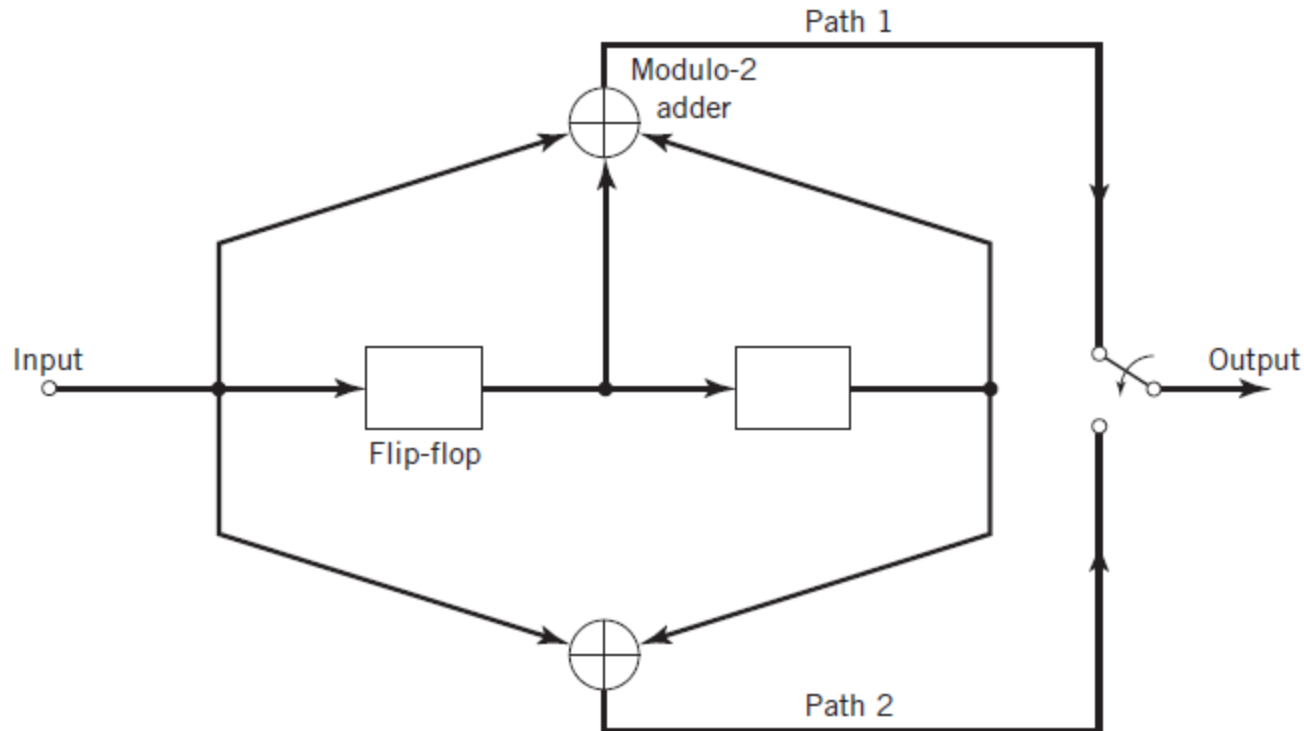
$X = \underline{11} \underline{01} \underline{01} \underline{00} \underline{10} \underline{11}$

m_0	m_1	m_2	x_1	x_2
0	0	1	1	1
0	1	1	0	1
1	1	0	0	1
1	0	1	0	0
0	1	0	1	0
1	0	0	1	1

Example-2



Constraint length-3, rate -1/2 convolutional encoder



Each path connecting the output to the input of a convolutional encoder may be characterized in terms of its *impulse response*.

Equivalently, each path can be characterized in terms of a generator polynomial, defined as the *unit-delay transform* of the impulse response.

convolutional encoder

To be specific, let the *generator sequence* $(g_0^{(i)}, g_1^{(i)}, g_2^{(i)}, \dots, g_M^{(i)})$ denote the impulse response of the i^{th} path, where the coefficients $g_0^{(i)}, g_1^{(i)}, g_2^{(i)}, \dots, g_M^{(i)}$ equal symbol 0 or 1. Correspondingly, the *generator polynomial* of the i^{th} path is defined by

$$g^{(i)}(D) = g_0^{(i)} + g_1^{(i)}D + g_2^{(i)}D^2 + \dots + g_M^{(i)}D^M$$

where D denotes the *unit-delay variable*. The complete convolutional encoder is described by the set of generator polynomials

$$\{g^{(i)}(D)\}_{i=1}^M$$

Constraint length-3, rate -1/2 convolutional encoder

The impulse response of path--1
(i.e., upper path) is (1, 1, 1). Hence,
the generator polynomial of this path is

$$g^{(1)}(D) = 1 + D + D^2$$

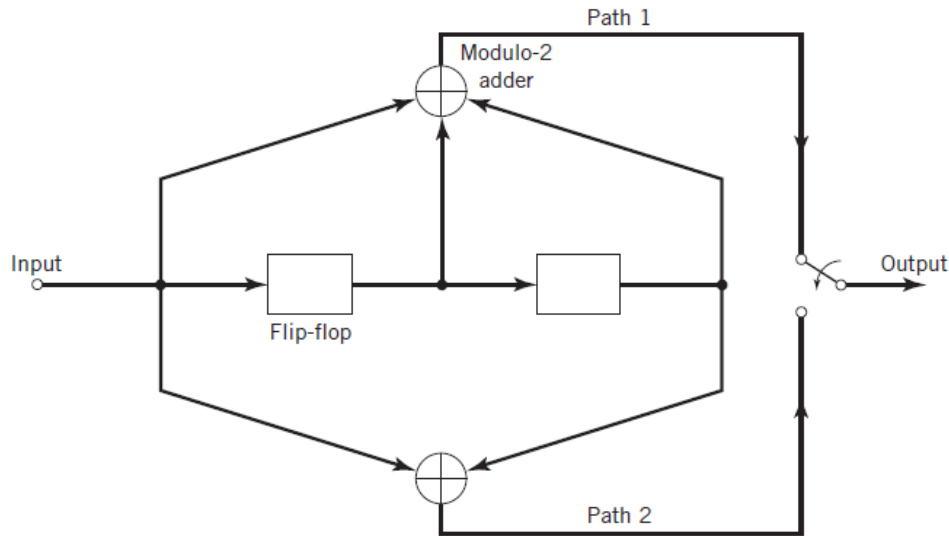
The impulse response of path 2
(i.e., lower path) is (1, 0, 1).

The generator polynomial of this second path is

$$g^{(2)}(D) = 1 + D^2$$

For an incoming message sequence given by (10011), the polynomial representation will be

$$m(D) = 1 + D^3 + D^4$$



Constraint length-3, rate -1/2 convolutional encoder

- As with Fourier transformation, convolution in the time domain is transformed into multiplication in the D -domain. Hence, the output polynomial of path-1 is given by

$$\begin{aligned} c^{(1)}(D) &= g^{(1)}(D)m(D) \\ &= (1 + D + D^2)(1 + D^3 + D^4) \\ &= 1 + D + D^2 + D^3 + D^6 \end{aligned}$$

$$1 + D^3 + \underbrace{D^4}_{\text{blue}} + \quad D \underbrace{+ D^4}_{\text{blue}} + \underbrace{D^5 + D^5}_{\text{red}} + \quad D^2 + \underbrace{D^5 + D^5}_{\text{red}} + D^6$$

The sums $D^4 + D^4$ and $D^5 + D^5$ are both zero in accordance with the rules of binary arithmetic

$$1 + D^3 + D + D^2 + D^6$$

The output sequence of path-1 is

D^0	D^1	D^2	D^3	D^4	D^5	D^6
1	1	1	1	0	0	1

Constraint length-3, rate -1/2 convolutional encoder

Similarly, the output polynomial of path 2 is given by

$$\begin{aligned}c^{(2)}(D) &= g^{(2)}(D)m(D) \\&= (1 + D^2)(1 + D^3 + D^4) \\&= 1 + D^2 + D^3 + D^4 + D^5 + D^6\end{aligned}$$

The output sequence of path 2 is therefore (1011111).

Constraint length-3, rate -1/2 convolutional encoder

Output sequence of

path-1 is $[\boxed{1} \ 1 \ \boxed{1} \ 1 \ \boxed{1} \ 0 \ \boxed{0} \ 1]$

path-2 is $[\boxed{1} \ 0 \ \boxed{1} \ 1 \ \boxed{1} \ \boxed{1} \ 1]$.

$\mathbf{c} = (11, 10, 11, 11, 01, 01, 11)$

Finally, *multiplexing* the two output sequences of paths 1 and 2, we get the encoded sequence

Constraint length-3, rate -1/2 convolutional encoder

Notes:

- The message sequence of length $L = 5$ bits produces an encoded sequence of length $n(L + - 1) = 14$ bits.
- For the shift register to be restored to its initial all-zero state, a terminating sequence of $- 1 = 2$ zeros is appended to the last input bit of the message sequence.
- The terminating sequence of $- 1$ zeros is called the *tail of the message*.

Thanks !