Module-6
Channel Coding

Dr. Markkandan S

School of Electronics Engineering (SENSE)
Vellore Institute of Technology
Chennai

VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Introduction to Error Control Coding

Error Control Coding is essential in digital communication systems to ensure reliable transmission of information over noisy channels.

- The primary function of error control coding is to add redundancy to the transmitted message, allowing the detection and correction of errors at the receiver.
- Two main categories of error control codes:
  - **Block Codes**
  - **Convolutional Codes**

Block Codes

Block Codes Overview

Block Codes:

- A block code takes a message of k bits and encodes it into a codeword of n bits, where $n > k$.
- The ratio $R = \frac{k}{n}$ is known as the **code rate**.
- Block codes are commonly used in communication systems to detect and correct errors.

Example: **(7, 4) Linear Block Code (LBC)**

- $k = 4$ (message length)
- $n = 7$ (codeword length)

Introduction to Linear Block Codes

- Linear Block Codes (LBC) encode a k-bit message into an n-bit codeword where $n > k$.
- The codeword is generated using the generator matrix G.
- The Parity Check Matrix H is used for error detection and correction.
- Each codeword is a linear combination of the rows of G.

Encoding in Block Codes

For an (n, k) block code, the message vector $\mathbf{u}$ of length k is encoded into a codeword $\mathbf{v}$ of length n using the generator matrix G.

$$\mathbf{v} = \mathbf{u}G \tag{1}$$

Generator Matrix G for a (7, 4) LBC:

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

The codeword $\mathbf{v}$ is obtained by multiplying the message vector $\mathbf{u}$ by the generator matrix G.

Generator Matrix G

- The generator matrix G is a $k \times n$ matrix.
- It has the form:

$$G = [I_k \mid P]$$

where I_k is the $k \times k$ identity matrix, and P is the $k \times (n-k)$ parity submatrix.

Generator Matrix G

- The generator matrix G is a $k \times n$ matrix.
- It has the form:

$$G = [I_k \mid P]$$

where I_k is the $k \times k$ identity matrix, and P is the $k \times (n-k)$ parity submatrix.

Example: (7, 4) Code

For a (7, 4) code, the generator matrix G is:

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

Encoding a Message Using G

- To encode a k-bit message, multiply the message vector u by the generator matrix G:

$$v = u \cdot G$$

where v is the n-bit codeword.

Encoding a Message Using G

- To encode a k-bit message, multiply the message vector u by the generator matrix G:

$$v = u \cdot G$$

where v is the n-bit codeword.

Example: Encoding

Let the message vector be $u = [1, 0, 1, 1]$. The encoded codeword is:

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

After multiplication, the codeword is:

$$v = [1, 0, 1, 1, 0, 0, 1]$$

Parity Check Matrix H

- The parity check matrix H is used for error detection and correction.
- It must satisfy $G \cdot H^T = 0$.
- The structure of H for systematic codes is:

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

where P^T is the transpose of the parity submatrix from G and I_{n-k} is an identity matrix of size $(n-k) \times (n-k)$.

Constructing H for (7, 4) Code

Step-by-Step

- Parity Submatrix P from G:

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

- Transpose P:

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

- Identity matrix I_{n-k}:

$$I_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Parity Check Matrix

The **parity check matrix** H is used to detect errors in the received codeword.

For a (7, 4) LBC, the parity check matrix is:

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

The syndrome **s** is calculated as:

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

where **r** is the received vector, and H^T is the transpose of the parity check matrix.

Example of Error Detection

Consider the received vector $\mathbf{r} = [1\,0\,1\,0\,0\,0\,1]$.

Calculate the syndrome:

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

$$\mathbf{s} = \mathbf{r}H^T = [1\,0\,1]$$

The syndrome is non-zero, indicating that an error has occurred in the received vector.

Syndrome Calculation for Error Detection

- The syndrome is calculated using the received vector r and the parity check matrix H:

$$s = r \cdot H^T$$

- If $s = 0$, the codeword is valid. If $s \neq 0$, an error has occurred.

Example: Error Detection with H

Received Vector

Let the received vector be:

$$r = [1, 0, 1, 0, 0, 0, 1]$$

Example: Error Detection with H

Received Vector

Let the received vector be:

$$r = [1, 0, 1, 0, 0, 0, 1]$$

Syndrome Calculation

Multiply the received vector by H^T:

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

The non-zero syndrome indicates an error.

Example: (7,4) Linear Block Code

- Consider a $(7, 4)$ linear block code.
- We have 4 data bits u_1, u_2, u_3, u_4 and 3 parity bits p_1, p_2, p_3.
- The codeword is 7 bits: $[u_1, u_2, u_3, u_4, p_1, p_2, p_3]$.
- The parity check equations are:

$$p_1 = u_1 + u_2 + u_3$$
$$p_2 = u_1 + u_2 + u_4$$
$$p_3 = u_1 + u_3 + u_4$$

- These equations ensure that the parity bits are combinations of the data bits.

Step-by-Step Construction of Parity Matrix P

- The parity check equations define how the parity bits are computed from the data bits.Let's express these equations in matrix form:

$$p_1 = u_1 + u_2 + u_3 \quad \Rightarrow \quad p_1 = [1,1,1,0] \cdot \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

$$p_2 = u_1 + u_2 + u_4 \quad \Rightarrow \quad p_2 = [1,1,0,1] \cdot \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix} ; p_3 \quad = u_1 + u_3 + u_4 \Rightarrow \qquad p_3 = [1,0,1,1] \cdot \begin{bmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \end{bmatrix}$$

- From these equations, we can extract the parity matrix P.

Constructing the Generator Matrix G

- The generator matrix G is constructed by combining the identity matrix I_k and the parity matrix P.The generator matrix G will be used to encode the message.
- Here, I_k is a 4×4 identity matrix, and P is the matrix derived from the parity check equations.

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

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

Encoding a Message Using G

- Let's encode the message $\mathbf{u} = [1, 0, 1, 1]$.
- The encoded codeword $\mathbf{v}$ is obtained by multiplying the message vector $\mathbf{u}$ with the generator matrix G:

$$\mathbf{v} = \mathbf{u} \cdot G = [1, 0, 1, 1] \cdot \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

$$\mathbf{v} = [1, 0, 1, 1, 0, 1, 0]$$

- This is the encoded 7-bit codeword that includes both data and parity bits.

Constructing the Parity Check Matrix H

- The parity check matrix H is constructed from the parity matrix P.
- H has dimensions $(n-k) \times n$, i.e., 3×7, and is formed as follows:

$$H = \begin{bmatrix} P^T & I_{n-k} \end{bmatrix}$$

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

- H is used for detecting errors in the received codeword.

Syndrome Calculation

- Suppose the received vector is $\mathbf{r} = [1, 0, 1, 1, 0, 0, 1]$.
- To check for errors, we calculate the syndrome:

$$\mathbf{s} = \mathbf{r} \cdot H^T = [1, 0, 1, 1, 0, 0, 1] \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- The result is:

$$\mathbf{s} = [1, 0, 1]$$

- A non-zero syndrome indicates that there is an error in the received codeword.

- The syndrome $\mathbf{s} = [1, 0, 1]$ corresponds to the third row of H, indicating an error in the third bit.
- The corrected codeword is:

$$\mathbf{v}' = [1, 0, 0, 1, 0, 0, 1]$$

- The error in the third bit has been corrected, and the valid codeword is restored.

Summary of Encoding and Decoding

- The generator matrix G is used to encode a message into a codeword.
- The parity check matrix H helps detect errors by calculating the syndrome.
- Errors can be corrected based on the syndrome.

Summary

- Block codes are essential for error detection and correction in digital communication systems.
- The generator matrix G is used for encoding messages, while the parity check matrix H helps detect errors.
- A (7, 4) LBC was used as an example to demonstrate encoding and error detection.
- Encoding is done by multiplying the message vector with the generator matrix G.
- Decoding involves calculating the syndrome using the parity check matrix H to detect errors.
- The generator matrix and parity check matrix provide a systematic way to detect and correct errors.

LBC Encoding Circuit

- The encoding process involves taking the data bits $u = [u_1, u_2, \ldots, u_k]$ and computing the codeword using the generator matrix G.
- The encoding circuit generates the codeword by using XOR gates to combine the data bits and produce the parity bits.

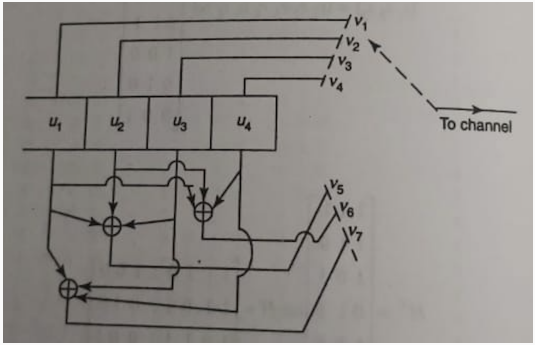


Figure 1: LBC Encoding Circuit

Example: Encoding Circuit in (7,4) LBC

- Consider the parity check equations for a (7,4) LBC:

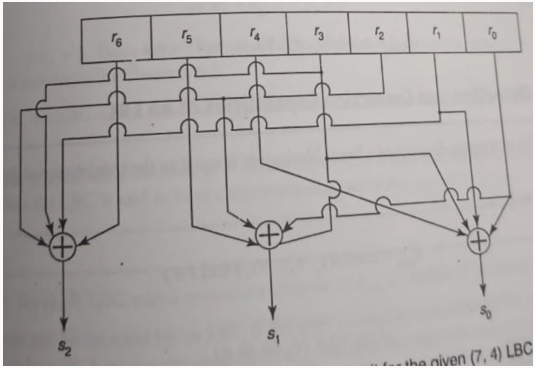
$$p_1 = u_1 + u_2 + u_3$$
$$p_2 = u_1 + u_2 + u_4$$
$$p_3 = u_1 + u_3 + u_4$$

- These equations are implemented using XOR gates in the encoding circuit, which computes the parity bits from the data bits.
- The generator matrix G is used to encode the message:

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

Syndrome Calculation Circuit

- The syndrome calculation circuit is used to detect errors in the received codeword.
- The received codeword $r = [r_1, r_2, \ldots, r_n]$ is multiplied by the transpose of the parity-check matrix H^T to calculate the syndrome s.
- If the syndrome is non-zero, it indicates the presence of errors.



Example: Syndrome Calculation in (7,4) LBC

- The parity-check matrix H for a (7,4) LBC is:

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

- The syndrome s is calculated as:

$$s = r \cdot H^T$$

- If $s = [0, 0, 0]$, no errors are detected. If $s \neq [0, 0, 0]$, it points to the location of the error.

Properties of Linear Block Codes

- **Systematic Codes:** LBCs are often systematic, meaning the data bits appear unchanged in the codeword, followed by parity bits.
- **Hamming Distance:** The minimum Hamming distance d_{min} of an LBC determines its error detection and correction capability.
- **Error Detection:** An LBC can detect up to $t_d = d_{min} - 1$ errors.
- **Error Correction:** An LBC can correct up to $t_c = \left\lfloor \frac{d_{min}-1}{2} \right\rfloor$ errors.
- **Linearity:** The sum of any two valid codewords in an LBC is also a valid codeword.

Introduction to Hamming Distance and Hamming Weight

- **Hamming Distance:** The number of positions at which the corresponding symbols of two codewords differ.
- **Hamming Weight:** The number of non-zero bits (1s) in a codeword.
- **Minimum Hamming Distance** $d_{\mathbf{min}}$: The smallest Hamming distance between any two distinct codewords in a linear block code.
- These properties determine the error detection and correction capabilities of the code.

Example Codewords for a (7,4) Linear Block Code

- Consider the following generator matrix G for a (7,4) LBC:

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

- Codewords are generated by multiplying message vectors u with G. For example:

$$\text{Message: } [1\,0\,1\,1] \quad \Rightarrow \quad \text{Codeword: } [1\,0\,1\,1\,0\,1\,0]$$
$$\text{Message: } [0\,1\,0\,1] \quad \Rightarrow \quad \text{Codeword: } [0\,1\,0\,1\,1\,0\,0]$$

Calculating Hamming Weight of Codewords

- The Hamming weight of a codeword is the number of 1s in the codeword.
- Example: For the codeword [1 0 1 1 0 1 0]:

$$\text{Hamming weight} = 1 + 1 + 1 + 1 = 4$$

- Example: For the codeword [0 1 0 1 1 0 0]:

$$\text{Hamming weight} = 1 + 1 + 1 = 3$$

- Hamming weight is used to calculate the minimum Hamming distance of the code.

Calculating Minimum Hamming Distance d_{min}

- The minimum Hamming distance d_{min} is the smallest Hamming distance between any pair of distinct codewords.
- Hamming distance between two codewords is the number of positions where their bits differ.
- Example: Consider two codewords [1 0 1 1 0 1 0] and [0 1 0 1 1 0 0].

$$\text{Difference: } 1\ \underline{0}\ 1\ \underline{1}\ 0\ \underline{1}\ 0 \quad \text{and} \quad 0\ \underline{1}\ 0\ \underline{1}\ 1\ \underline{0}\ 0$$

- Hamming distance $= 4$ (the positions of underlined bits differ).
- Minimum Hamming distance $d_{\text{min}} = 3$ for this code, as the smallest Hamming distance between any two codewords is 3.

Example: Calculating d_{min} for (7,4) LBC

- Let's compute d_{min} for a (7,4) linear block code with these codewords:

$$\text{Codeword 1: } [1\,0\,1\,1\,0\,1\,0]$$
$$\text{Codeword 2: } [0\,1\,0\,1\,1\,0\,0]$$
$$\text{Codeword 3: } [1\,1\,0\,0\,1\,1\,0]$$
$$\text{Codeword 4: } [0\,0\,0\,0\,0\,0\,0]$$

- Calculate the Hamming distance between each pair of codewords:

$$d(C_1, C_2) = 4, \quad d(C_1, C_3) = 3, \quad d(C_2, C_3) = 4$$

- Minimum Hamming distance $d_{\text{min}} = 3$, which means this code can detect up to 2 errors and correct 1 error.

Hamming Codes

Introduction to Hamming Codes

- Hamming Codes are linear error-correcting codes that can detect and correct single-bit errors.
- Invented by Richard Hamming in 1950.
- Hamming codes are used for reliable data transmission over noisy channels.
- For a message length of 4, we need to design a Hamming code that can detect and correct a single error.

Designing a Hamming Code (Single Error-Correcting)

Step-by-Step Design:

- Message length: 4 bits.
- Number of parity bits p: We need to determine p such that:

$$2^p \geq n + p + 1$$

  where n is the number of data bits.
- For $n = 4$, solve $2^p \geq 4 + p + 1$.
- $p = 3$, so we need 3 parity bits.
- The total codeword length is $n + p = 7$ bits.

Placement of Parity Bits

- The 7-bit codeword will contain 4 data bits and 3 parity bits.
- Parity bits are placed at positions that are powers of 2:
  - Parity bit P_1 at position 1.
  - Parity bit P_2 at position 2.
  - Parity bit P_4 at position 4.
- Data bits are placed at the remaining positions:

$$[P_1, P_2, D_1, P_4, D_2, D_3, D_4]$$

- Example: For message $[D_1\, D_2\, D_3\, D_4] = [1\,0\,1\,1]$, the initial codeword is $[P_1, P_2, 1, P_4, 0, 1, 1]$.

Example: Hamming Code Encoding and Parity Bit Calculation

Problem: Encode the message [1 0 1 1] using a (7,4) Hamming code.

Solution:

1. Arrange the data bits [1 0 1 1] as $[P_1, P_2, 1, P_4, 0, 1, 1]$.
2. Calculate the parity bits using XOR:
   - $P_1 = D_1 \oplus D_2 \oplus D_4 = 1 \oplus 0 \oplus 1 = 0$
   - $P_2 = D_1 \oplus D_3 \oplus D_4 = 1 \oplus 1 \oplus 1 = 1$
   - $P_4 = D_2 \oplus D_3 \oplus D_4 = 0 \oplus 1 \oplus 1 = 0$
3. The final encoded codeword is:

$$[P_1\, P_2\, D_1\, P_4\, D_2\, D_3\, D_4] = [0\,1\,1\,0\,0\,1\,1]$$

Example: Hamming Code Error Detection and Correction

Problem: A received codeword is $[0\,1\,0\,0\,0\,1\,1]$. Detect and correct the error.

Solution:

- The received codeword differs from the transmitted codeword. Perform syndrome calculation:
  - $S_1 = P_1 \oplus D_1 \oplus D_2 \oplus D_4 = 0 \oplus 0 \oplus 0 \oplus 1 = 1$
  - $S_2 = P_2 \oplus D_1 \oplus D_3 \oplus D_4 = 1 \oplus 0 \oplus 1 \oplus 1 = 1$
  - $S_4 = P_4 \oplus D_2 \oplus D_3 \oplus D_4 = 0 \oplus 0 \oplus 1 \oplus 1 = 0$
- Syndrome $[S_4\, S_2\, S_1] = [0\,1\,1] = 3$ (binary).
- Error detected in position 3. Flip the third bit of the received codeword:

$$[0\,1\,\underline{0}\,0\,0\,1\,1] \rightarrow [0\,1\,1\,0\,0\,1\,1]$$

- The corrected codeword is $[0\,1\,1\,0\,0\,1\,1]$, matching the original transmitted codeword.

Cyclic Codes

Introduction to Cyclic Codes

- Cyclic codes are a subclass of linear block codes.
- Any cyclic shift of a valid codeword results in another valid codeword.
- Widely used for error detection and correction due to efficient encoding and decoding.
- Defined by a generator polynomial $g(x)$.

Properties of Cyclic Codes

- Cyclic codes are linear codes.
- Codewords can be represented as polynomials over a binary field.
- A valid cyclic codeword can be generated by multiplying the message polynomial by the generator polynomial $g(x)$.
- Cyclic shift property: If $c(x)$ is a codeword, then $x \cdot c(x) \mod (x^n - 1)$ is also a codeword.

Generator Polynomial $g(x)$

- The generator polynomial $g(x)$ defines the cyclic code.
- $g(x)$ divides $x^n - 1$, where n is the length of the codeword.
- The degree of $g(x)$ determines the number of parity bits, $n - k$, where k is the number of data bits.
- Example: For a (7,4) cyclic code, a valid generator polynomial is:

$$g(x) = 1 + x + x^3$$

Designing the Generator Polynomial in Cyclic Codes

- Generator Polynomial $g(x)$: A key component in designing cyclic codes. It defines the cyclic code and is used for encoding and decoding.
- The generator polynomial must divide $x^n - 1$, where n is the length of the codeword.
- Steps to Design:
  1. Factorize $x^n - 1$ into irreducible polynomials over the finite field $GF(2)$.
  2. Choose one of the factors as the generator polynomial $g(x)$.
- The degree of $g(x)$ determines the number of parity bits $n - k$, where k is the number of data bits.

Example: Designing the Generator Polynomial

Problem: Design the generator polynomial for a (7,4) cyclic code.

Solution:

- Factor $x^7 - 1$ over $GF(2)$:

$$x^7 - 1 = (x - 1)(x^3 + x^2 + 1)(x^3 + x + 1)$$

- Choose one of the irreducible factors as the generator polynomial. For example:

$$g(x) = x^3 + x + 1$$

- The degree of $g(x)$ is 3, so the number of parity bits is $7 - 4 = 3$.
- The cyclic code generated by $g(x)$ will have a minimum Hamming distance of $d_{\min} \geq 3$, enabling single-error correction.

Derivation of Generator Polynomial $g(x)$

- The generator polynomial $g(x)$ must divide $x^n - 1$.
- Example for a (7,4) cyclic code:

$$x^7 - 1 = (x - 1)(x^3 + x^2 + 1)(x^3 + x + 1)$$

- Choose one of the irreducible factors as the generator polynomial. In this case:

$$g(x) = 1 + x + x^3$$

- The codewords are generated by multiplying the message polynomial $m(x)$ by $g(x)$.

Factorization of $x^n - 1$ and Possible $g(x)$

n	Factorization of $x^n - 1$	Possible Generator Polynomials $g(x)$
2	$x^2 - 1 = (x-1)(x+1)$	$g(x) = x-1, x+1$
3	$x^3 - 1 = (x-1)(x^2+x+1)$	$g(x) = x^2+x+1$
4	$x^4 - 1 = (x-1)(x+1)(x^2+1)$	$g(x) = x^2+1, x^2-1$
5	$x^5 - 1 = (x-1)(x^4+x^3+x^2+x+1)$	$g(x) = x^4+x^3+x^2+x+1$
6	$x^6 - 1 = (x-1)(x+1)(x^2+x+1)(x^2-x+1)$	$g(x) = x^2+x+1, x^2-x+1$
7	$x^7 - 1 = (x-1)(x^6+x^5+x^4+x^3+x^2+x+1)$	$g(x) = x^3+x+1, x^6+\cdots+1$
8	$x^8 - 1 = (x-1)(x+1)(x^2+1)(x^4+1)$	$g(x) = x^4+1, x^2+1$
9	$x^9 - 1 = (x-1)(x^3-1)(x^3+1)$ $= (x-1)(x-1)(x^2+x+1)(x+1)(x^2-x+1)$	$g(x) = x^3+1$, $g(x) = x^2+x+1$

Table 1: Factorization of $x^n - 1$ and Possible $g(x)$ for Various n

Example of Message Encoding

- For a (7,4) cyclic code, let the message polynomial $m(x) = 1 + x$ (corresponding to the message bits [1 0 1 0]).
- The generator polynomial is $g(x) = 1 + x + x^3$.
- Multiply the message polynomial by the generator polynomial:

$$c(x) = m(x) \cdot g(x) = (1 + x) \cdot (1 + x + x^3) = 1 + 2x + 2x^2 + x^4 + x^5$$

- In binary (mod 2), the codeword becomes:

$$c(x) = 1 + x^2 + x^4 + x^5$$

- Codeword: [1 0 1 0 1 1 0]

Cyclic Code Encoder Circuit

- The encoding process can be implemented using shift registers and XOR gates.
- The generator polynomial determines the connections of the XOR gates.

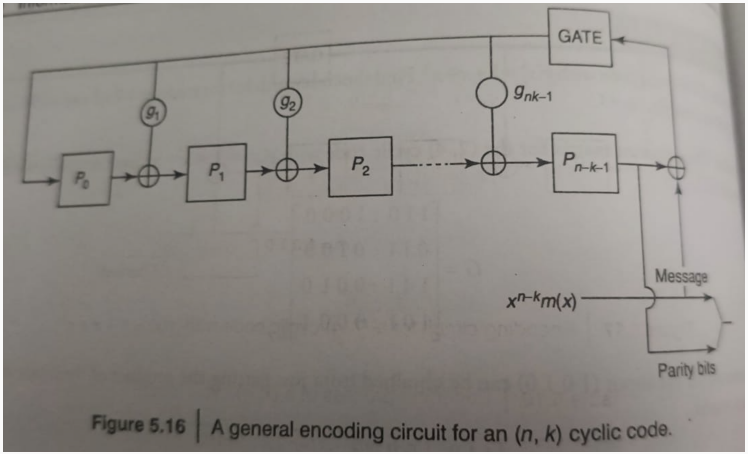


Figure 5.16 | A general encoding circuit for an (n, k) cyclic code.

Encoding Circuit for (7,4) Cyclic Code

- The shift register implementation of the (7,4) cyclic code encoder is based on the generator polynomial $g(x) = 1 + x + x^3$.
- Shift the data bits through the shift register, and the parity bits are generated using XOR gates.
- The final codeword consists of the original data bits followed by the parity bits.

Syndrome Calculation

- The syndrome is calculated by dividing the received polynomial $r(x)$ by the generator polynomial $g(x)$.
- If the remainder (syndrome) is zero, no errors are detected. Otherwise, errors have occurred.
- Syndrome $S(x) = r(x) \mod g(x)$

Example: Syndrome Calculation

- Suppose the transmitted codeword is $c(x) = 1 + x^2 + x^4 + x^5$.
- The received polynomial is $r(x) = 1 + x + x^2 + x^4 + x^5$.
- Perform the division $r(x) \div g(x)$, where $g(x) = 1 + x + x^3$, to calculate the syndrome.
- The syndrome is non-zero, indicating that an error has occurred.

Cyclic Code Decoder Circuit

- The decoder circuit uses the syndrome to detect errors.
- The error can be located using lookup tables or other error-correcting algorithms.
- The syndrome points to the error pattern, which can then be corrected.

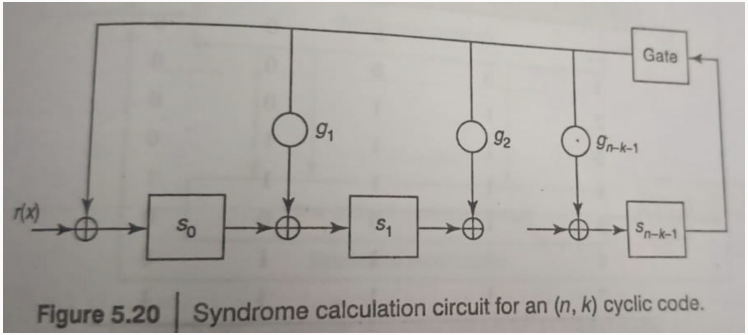


Figure 4: Cyclic Code Decoder Circuit

Error Detection and Correction Capability

- The error detection and correction capability of a cyclic code depends on its minimum Hamming distance d_{min}.
- A cyclic code can detect up to $d_{\text{min}} - 1$ errors.
- It can correct up to $\left\lfloor \frac{d_{\text{min}}-1}{2} \right\rfloor$ errors.
- For a (7,4) cyclic code with $d_{\text{min}} = 3$, the code can detect up to 2 errors and correct 1 error.

Example of Error Detection

- Consider the received polynomial $r(x) = 1 + x + x^2 + x^4 + x^5$.
- Perform polynomial division by $g(x) = 1 + x + x^3$.
- If the remainder is non-zero, an error has occurred.
- The error can be corrected by identifying the error pattern using the syndrome.

Example of Error Correction

- Using the syndrome, the location of the error can be identified.
- Flip the corresponding bit in the received codeword to correct the error.
- Example: Suppose the error occurs at the x^1 term. The received polynomial $r(x) = 1 + x + x^2 + x^4 + x^5$ is corrected to $1 + x^2 + x^4 + x^5$.

Systematic Cyclic Codes

- In a systematic cyclic code, the message bits are transmitted unchanged, followed by the parity bits.
- The encoding process appends the parity bits to the message.
- The generator polynomial $g(x)$ ensures that the parity bits are generated correctly.

Example of Systematic Encoding

- For a message $m(x) = 1 + x$, the codeword in a systematic cyclic code is formed by appending the parity bits.
- The generator polynomial is used to compute the parity bits, and the final codeword is $[1\,0\,1\,0\,p_1\,p_2\,p_3]$.

Algebraic Properties of Cyclic Codes

- Cyclic codes are closed under addition: The sum of two codewords is also a valid codeword.
- Cyclic codes are also closed under scalar multiplication.
- These properties make cyclic codes useful for error correction as the cyclic structure simplifies decoding.

Applications of Cyclic Codes

- Cyclic codes are widely used in error detection and correction.
- Applications include:
  - Cyclic Redundancy Check (CRC) in network communications.
  - Data storage systems.
  - Satellite communications.

Cyclic Redundancy Check (CRC)

- CRC is a widely used error-detecting code based on cyclic codes.
- A specific generator polynomial is used to generate the CRC bits.
- CRC is highly efficient for detecting burst errors in data transmission.

Performance of Cyclic Codes

- Cyclic codes are highly efficient in terms of error detection and correction.
- They have low computational complexity due to their polynomial structure.
- Cyclic codes provide robust error correction with minimal overhead in many communication systems.

Summary

- Cyclic codes are a powerful class of linear block codes used for error detection and correction.
- They are defined by their generator polynomial $g(x)$, which encodes and decodes messages efficiently.
- Cyclic codes are widely used in communication systems, especially in applications like CRC.

Example Problem 1: Cyclic Encoding

- Consider a (7, 4) cyclic code with generator polynomial $g(x) = 1 + x + x^3$.
- Encode the message $m(x) = 1 + x^2$ (corresponding to the message bits [1 0 1 0]).

Solution:

- Multiply the message polynomial $m(x)$ by the generator polynomial $g(x)$:

$$c(x) = m(x) \cdot g(x) = (1 + x^2) \cdot (1 + x + x^3)$$

- Perform polynomial multiplication:

$$c(x) = 1 + x + x^3 + x^2 + x^4 + x^5$$

- Since we're working in $\mathbb{F}_2$, reduce the powers modulo 2:

$$c(x) = 1 + x + x^2 + x^4 + x^5$$

- The final encoded codeword is:

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

Example Problem 2: Cyclic Decoding

- A received codeword for the (7, 4) cyclic code is $r(x) = 1 + x + x^2 + x^4$ (received bits [1 1 1 0 1 0 0]).
- The generator polynomial is $g(x) = 1 + x + x^3$.
- Determine if the received codeword contains any errors using syndrome calculation.

Solution:

- Perform polynomial division of $r(x)$ by $g(x)$ to find the remainder (syndrome):

$$r(x) = 1 + x + x^2 + x^4$$

- Divide $r(x)$ by $g(x) = 1 + x + x^3$ using long division or synthetic division:

$$r(x) \div g(x) = \text{quotient with remainder} \Rightarrow \text{remainder} = 1$$

- Since the remainder is non-zero, the syndrome is $S(x) = 1$, indicating an error in the received codeword.

Example Problem 3: Correcting Errors with Cyclic Codes

Problem:

- The received codeword $r(x) = 1 + x + x^2 + x^4$ was found to contain an error.
- Use the syndrome to identify the location of the error and correct the codeword.

Solution:

- The syndrome calculation yielded $S(x) = 1$, which corresponds to a single-bit error in the first bit of the codeword.
- Flip the first bit of the received codeword $[1\,1\,1\,0\,1\,0\,0]$ to correct the error.
- The corrected codeword is $[0\,1\,1\,0\,1\,0\,0]$, which corresponds to the polynomial:

$$c(x) = x + x^2 + x^4$$

- The error has been successfully corrected.

Example Problem 4: Systematic Encoding in Cyclic Codes

Problem:

- Use systematic encoding to encode the message $m(x) = 1 + x^2$ with the generator polynomial $g(x) = 1 + x + x^3$.
- The codeword must have the data bits followed by the parity bits.

Solution:

- Perform systematic encoding by first multiplying the message by x^{n-k} (shifting it):

$$x^3 \cdot m(x) = x^3 \cdot (1 + x^2) = x^3 + x^5$$

- Perform polynomial division of $x^3 \cdot m(x)$ by $g(x)$:

$$(x^3 + x^5) \div (1 + x + x^3) = \text{quotient with remainder} = x + x^2$$

- Add the remainder (parity bits) to the shifted message:

$$\text{Codeword: } c(x) = x^3 + x^5 + x + x^2 = 1 + x + x^2 + x^3 + x^5$$

Example Problem 5: Syndrome Calculation for Error Detection

Problem:

- A cyclic code with generator polynomial $g(x) = 1 + x^2 + x^3$ is used to transmit a message.
- The received polynomial is $r(x) = 1 + x^2 + x^4 + x^5$.
- Perform syndrome calculation to detect if any errors occurred.

Solution:

- Perform polynomial division of $r(x)$ by $g(x)$:

$$r(x) = 1 + x^2 + x^4 + x^5$$

- Divide $r(x)$ by $g(x) = 1 + x^2 + x^3$:

$$r(x) \div g(x) = \text{quotient with remainder} \Rightarrow \text{remainder} = x$$

- Since the remainder is non-zero, the syndrome $S(x) = x$, indicating that an error occurred.

Example Problem 6: Correcting Multiple Errors in Cyclic Codes

Problem:

- A received codeword is $r(x) = 1 + x + x^3 + x^4$, and the generator polynomial is $g(x) = 1 + x + x^3$.Perform syndrome calculation and correct the errors.

Solution:

- Calculate the syndrome by dividing $r(x)$ by $g(x)$:

$$r(x) = 1 + x + x^3 + x^4$$

- Perform polynomial division:

$$r(x) \div g(x) = \text{quotient with remainder} \Rightarrow \text{remainder} = x$$

- The syndrome indicates an error at the second bit position.
- Correct the error by flipping the second bit of the received codeword [1 1 0 1 1].
- The corrected codeword is [1 0 0 1 1].

Summary of Example Problems

- These examples demonstrate encoding, decoding, and syndrome calculation in cyclic codes.
- The generator polynomial $g(x)$ is central to all operations in cyclic codes.
- Encoding involves multiplying the message by $g(x)$, while decoding and error correction use the syndrome.
- Systematic encoding ensures that the message bits are transmitted directly, with parity bits appended.
- Cyclic codes are highly efficient for error detection and correction in communication systems.

Convolution Coding

Overview of Convolutional Codes

- **Definition**: Convolutional codes are error-correcting codes used in digital communication to detect and correct errors during transmission.
- **Key Features**:
  - Continuous encoding of data.
  - Use of memory to generate output based on current and previous input bits.
  - Introduction of redundancy for error detection and correction.

Properties of Convolutional Codes

- **Constraint Length (K)**: The number of input bits that affect the output.
- **Code Rate (k/n)**: Ratio of input bits to output bits.
- **Free Distance (d_free)**: Minimum Hamming distance between any two distinct codewords.
- **Error-Correcting Capability**: Related to the free distance.

Convolutional Encoder - Basics

- **Concept**: Convolutional encoders take k input bits and generate n output bits, with output influenced by both current and previous input bits.
- **Example**: A 1/2 rate convolutional encoder with two shift registers and XOR gates.

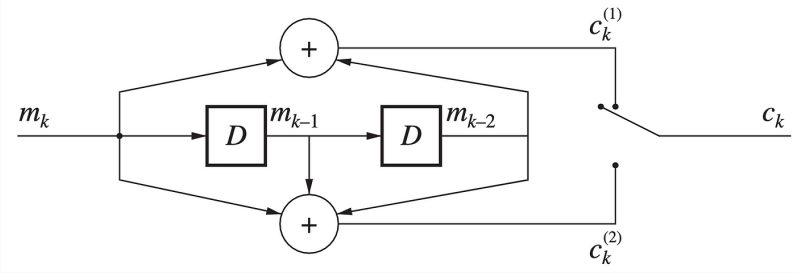


Figure 5: Convolutional Encoder Example

State Diagram of a Convolutional Encoder

- **State Diagram**: Represents the possible states of the encoder based on memory contents.

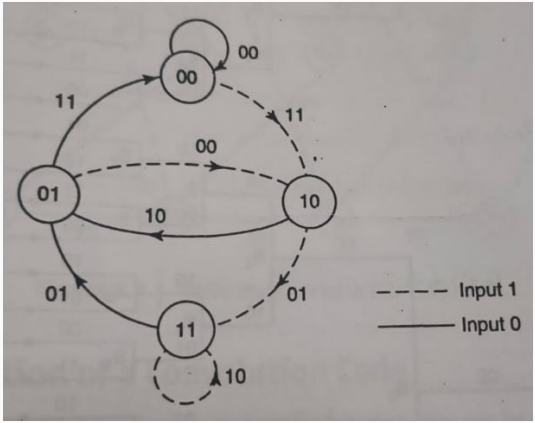
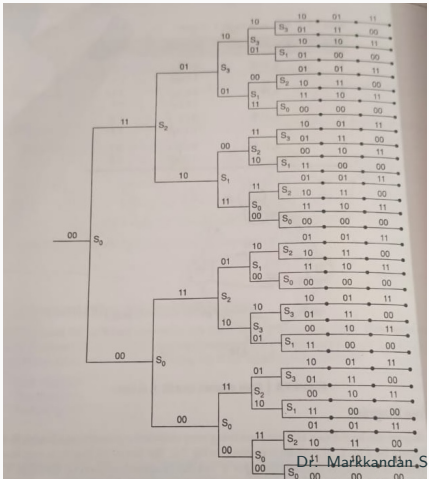


Figure 6: State Diagram for a 2-State Encoder

Tree Diagram of a Convolutional Encoder

- **Tree Diagram**: Shows all possible output sequences as a function of input bits.
- **Example**: Start with initial state and show branches representing each input bit.



Trellis Diagram for Convolutional Codes

- **Trellis Diagram**: A time-based representation of state transitions in a convolutional code. For a 2-state encoder, show transitions over time with possible output values.

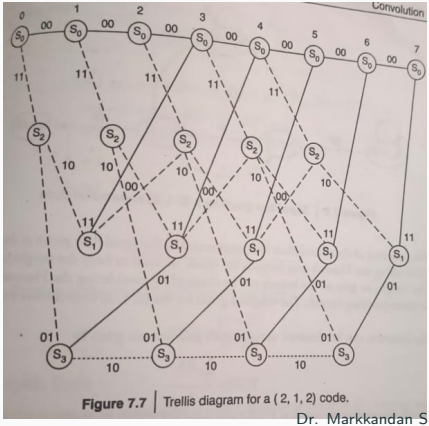


Figure 7.7 | Trellis diagram for a (2, 1, 2) code.

Transfer Function of Convolutional Codes

- **Definition**: The transfer function describes the input-output relationship of the convolutional code.
- **Formula**:

$$T(D) = \sum_{i=0}^{\infty} c_i D^i$$

where D is the delay and c_i are the coefficients representing output weights.

- **Example**: Calculation of the transfer function for a $1/2$ rate encoder.

Introduction to Viterbi Decoding

- **Definition**: Viterbi decoding is a maximum likelihood algorithm used to decode convolutional codes by finding the most likely sequence of transmitted data.
- **Key Concepts**:
  - Path Metric: Measure of the likelihood of a given path.
  - Branch Metric: Computed at each stage based on the Hamming distance.
  - Survivor Path: The most likely sequence at each stage.

Viterbi Decoding Process

- **Steps of Viterbi Decoding**:
  1. Initialize the state metrics.
  2. Recursively compute the branch and path metrics.
  3. Trace back the most likely path from the trellis diagram.

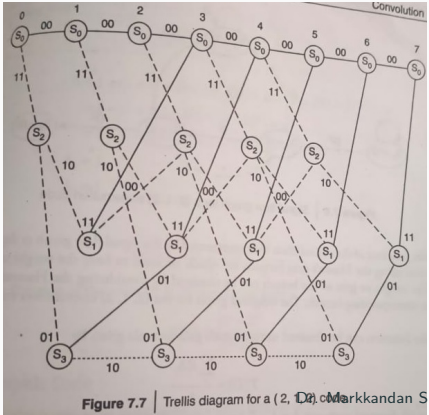


Figure 7.7 | Trellis diagram for a (2, 1, 2) convolution

Conclusion

- **Summary**: Convolutional codes are a powerful tool for error correction in communication systems.
- **Viterbi Decoding**: Optimal decoding method for convolutional codes.
- **Applications**: Used in a wide range of communication systems from deep space to mobile networks.