Department of Electrical Engineering
University of Arkansas

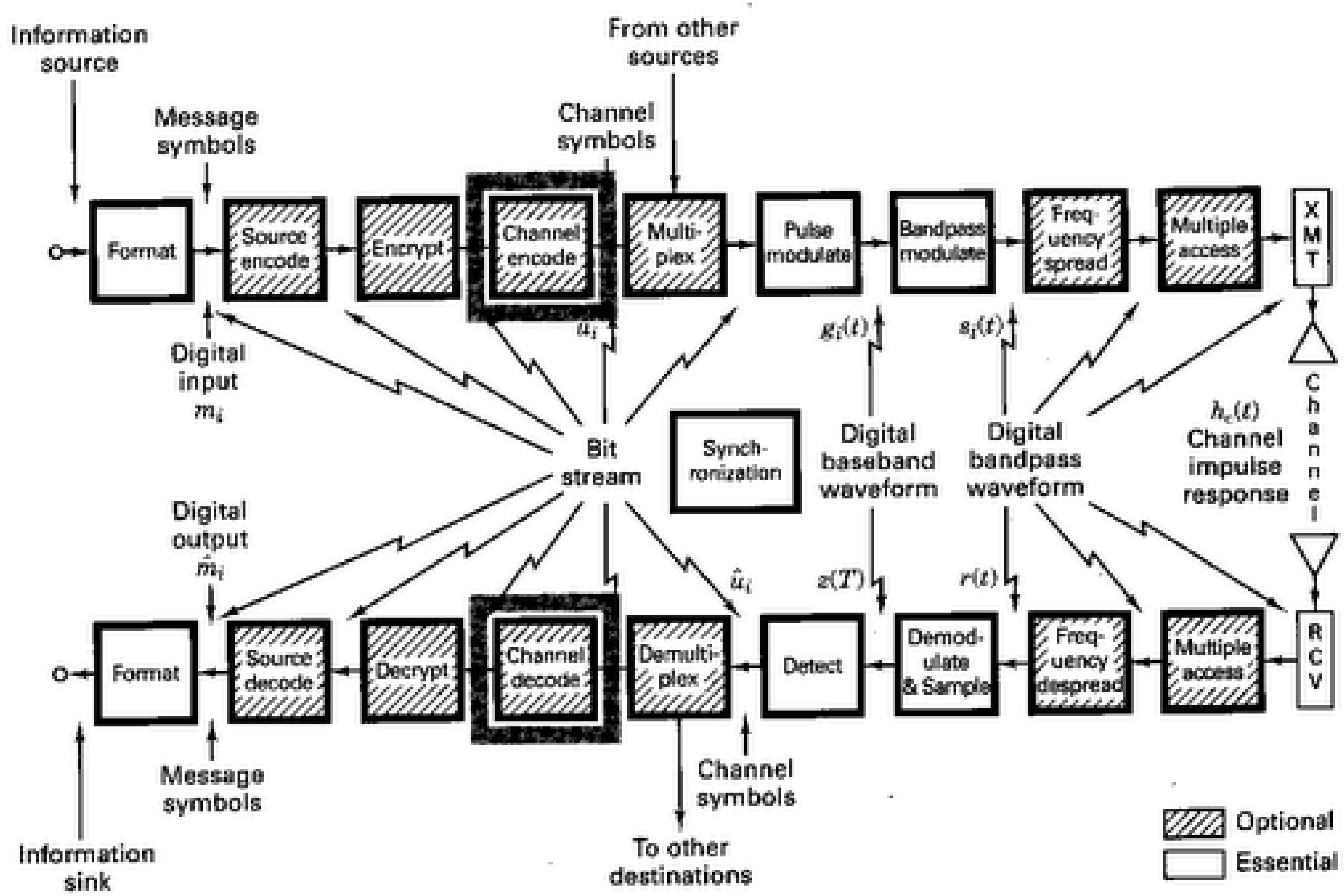
# ELEG 5663 Communication Theory
# Ch. 6 Coding: PART I

**Dr. Jingxian Wu**
**wuj@uark.edu**

# OUTLINE

- **Introduction**
- **Types of Error Control**
- **Structured Sequence**
- **Linear Block Code**
- **Error Detection and Correction Capability**

# INTRODUCTION



# INTRODUCTION

- **Channel Coding**
  - Channel coding with structured sequence
    - Protect the information from channel distortions by adding structured redundancy
    - Examples:
      - Cyclic Redundancy Check (CRC), Linear Block Code, Convolutional Code (CC), Turbo Code, Low Density Parity Check (LDPC), etc.

- **Coding can only be used in digital communication systems.**

# OUTLINE

- Introduction
- **Types of Error Control**
- Structured Sequence
- Linear Block Code
- Error Detection and Correction Capability

# ERROR CONTROL

- **Error control**
  - Error detection and retransmission:
    - detect the presence of errors at the Rx by using the structured sequence
    - Rx notifies Tx about the error
    - Tx retransmit the message
    - Requires a two-way link between Tx and Rx
  - Forward error correction (FEC)
    - The Rx can correct the errors by using the structured sequence
    - Only a one-way link is required.

# ERROR CONTROL: ARQ

- **Automatic Retransmission Query (ARQ)**
  – When the Rx detects an error, it will require the Tx to retransmit the information.
  – Such an error control procedure is called ARQ.
  – 3 types of ARQ:
    - Stop-and-wait
    - Continuous ARQ with pull back
    - Continuous ARQ with selective repeat
- **Stop-and-wait ARQ**
  – The Rx waits for an acknowledgement (ACK) of each transmission before transmitting the next packet.

# ERROR CONTROL: ARQ

- **Continuous ARQ with pullback**
  - The Tx transmits continuously without waiting for a ACK
  - In case of NAK (negative ACK), the Tx will retransmit the error packet and all the subsequent packets

- **Continuous ARQ with selective repeat**
  - In case of NAK, the Tx will only retransmit the error packet
  - More complicated operation at the Rx (re-order packets, memory)

# OUTLINE

- Introduction
- Types of Error Control
- **Structured Sequence**
- Linear Block Code
- Error Detection and Correction Capability

# STRUCTURED SEQUENCE

- **Channel coding**
  - Protect the transmitted information by adding redundancy.
  - E.g. repetition code:
    - '0': '000'
    - '1': '111'
- **Error detection**
  - Include only enough redundant information such that the Rx can detect an error by looking at the Rx data.
    - E.g. repeat '1' 2 times. Tx (1 1), Rx (0 1) ➔ Receiver knows there is an error, but couldn't guess what is transmitted
    - Send back Negative Acknowledgement (automatic-repeat query: ARQ)
- **Error correction**
  - Include enough redundant information such that the Rx can recover the original information by looking at the Rx data.
    - E.g. repeat '1' 3 times. Tx (1 1 1), Rx (0 1 1) ➔ Receiver will guess that (1 1 1) is transmitted ➔ detect '1'
    - Majority decision rule ➔ minimize the probability of error.

# STRUCTURED SEQUENCE

- **Linear block code (LBC)**
  - Every $k$ bits of information corresponds to a codeword of length $n$ bits
    - E.g. repetitionon code 1-bit of information, 3-bit codeword
  - $n > k$: there are ($n$-$k$) bits of redundancy
  - The code is called: ($n$, $k$) linear block code
  - Definition:
    - code rate: $r = k/n$
    - Measures the efficiency of the code (1-$r$: the percentage of redundancy)
    - E.g.: (3, 1) repetition code: r = 1/3. (2, 1) repetition code: r = 1/2.


UNIVERSITY OF ARKANSAS

# STRUCTURED SEQUENCE: SINGLE PARITY CHECK

- **Single parity check**
  - Adding a single parity bit to a block of data bits
  - Code rate: $r = k/(k+1)$
  - Odd parity: add a parity bit such that the summation of all the bits yields an odd result (odd number of ones)

    $\underline{1}\ 0\ 1\ 0\ 1\ 0 \qquad \underline{0}\ 0\ 0\ 1\ 1\ 1$

  - Even parity: add a parity bit such that the summation of all the bits yields an even result (even number of ones)

    $\underline{0}\ 0\ 1\ 0\ 1\ 0 \qquad \underline{1}\ 0\ 0\ 1\ 1\ 1$

  - Error detection only
  - Will the code be able to detect even number of errors?

# STRUCTURED SEQUENCE: RECTANGULAR CODE

- **Rectangular code (product code)**
  - Form a rectangle of message bits of *M* rows and *N* columns.
  - A horizontal parity check is appended to each row
  - A vertical parity check is appended to each column
  - Code rate:

$$r = \frac{MN}{(M+1)(N+1)}$$

  - Any single bit error will cause:
    - A parity check error in a row
    - A parity check error in a column
    - The single bit error can be detected and corrected!
  - Product code can detect any 1 bit error

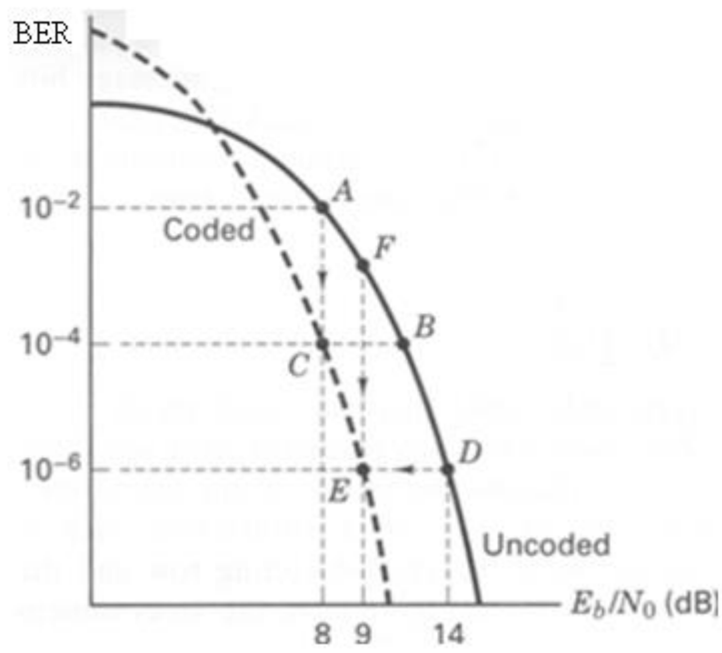
$$\begin{array}{c|ccccc}
\underline{0} & 0 & 1 & 1 & 0 & 0 \\
\underline{1} & 1 & 0 & 1 & 1 & 0 \\
\underline{0} & 0 & 1 & 0 & 1 & 0 \\
\hline
\underline{1} & \underline{1} & \underline{0} & \underline{0} & \underline{0} & \underline{0}
\end{array}$$

*M*=3, *N*=5

# STRUCTURED SEQUENCE: WHY ERROR CORRECTION?

- **Trade-off 1: Error performance v.s. bandwidth**
  - fixing SNR and data rate
  - Error correction code ➔ smaller BER
  - Error correction code ➔ More bandwidth is needed to transmit (information bits + redundancy bits)

$$\frac{P_r}{N} = \frac{E_b}{N_0}\frac{R}{W}$$



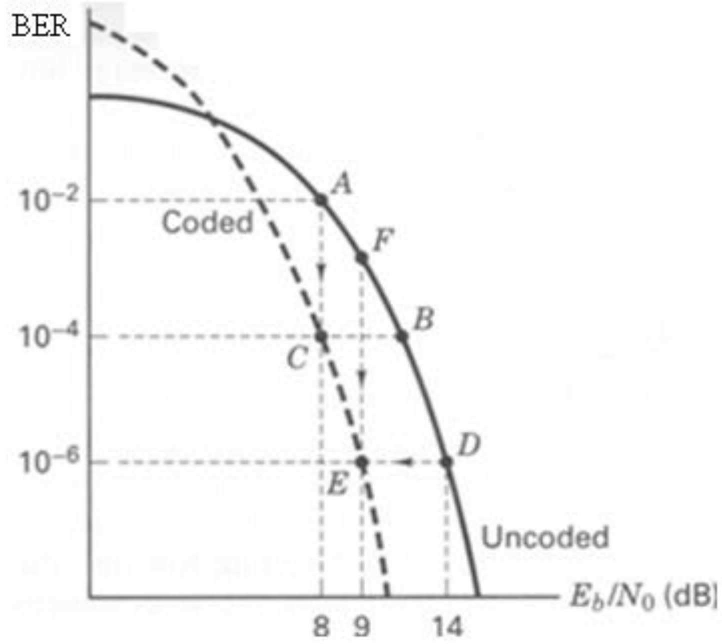

UNIVERSITY OF ARKANSAS

## STRUCTURED SEQUENCE: WHY ERROR CORRECTION?

- **Trade-off 2: power v.s. bandwidth**
  - Fixing BER and data rate
  - Error correction code ➔ smaller $E_b / N_0$ ➔ smaller $P_r$
  - Error correction code ➔ More bandwidth is needed to transmit (information bits + redundancy bits)
  - Coding gain: at a given BER, coding gain is the difference in Eb/N0 between coded and uncoded system

$$\frac{E_b}{N_0} = \frac{P_r}{N_0}\frac{1}{R}$$

$$G(dB) = \left(\frac{E_b}{N_0}\right)_u dB - \left(\frac{E_b}{N_0}\right)_c dB$$

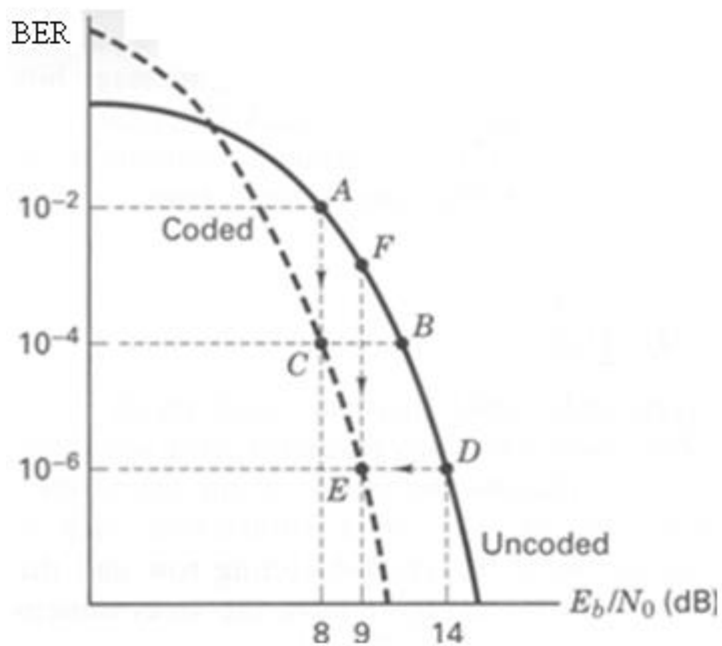



UNIVERSITY OF ARKANSAS

# STRUCTURED SEQUENCE: WHY ERROR CORRECTION?

- **Trade-off 3: data rate v.s. bandwidth**
  - Fix BER and Tx power
  - Error correction code ➔ smaller $E_b / N_0$ ➔ bigger $R$
  - Error correction code ➔ More bandwidth is needed to transmit (information bits + redundancy bits)

$$\frac{E_b}{N_0} = \frac{P_r}{N_0}\frac{1}{R}$$



# OUTLINE

- Introduction
- Types of Error Control
- Structured Sequence
- **Linear Block Code**
- Error Detection and Correction Capability

# LBC: PREPARATIONS

- **Binary operations**
  - 0 + 0 = 0, 0 + 1 = 1, 1 + 0 = 1, 1 + 1 = 0
  - 0 x 0 = 0, 0 x 1 = 0, 1 x 0 = 0, 1 x 1 = 1
- **Matrix operations**
  - The product of an *m* x *k* matrix and *k* x *n* matrix
    - Has a size of *m x n*

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \qquad \mathbf{B} = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \\ b_{31} & b_{32} \end{bmatrix} \qquad \mathbf{C} = \mathbf{A} \times \mathbf{B} = \begin{bmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \\ c_{31} & c_{32} \end{bmatrix}$$

$$c_{mn} = a_{m1}b_{1n} + a_{m2}b_{2n} + a_{m3}b_{3n}$$

$$c_{21} = a_{21}b_{11} + a_{22}b_{21} + a_{23}b_{31}$$

UNIVERSITY OF ARKANSAS

# LBC: AN EXAMPLE

- **(7, 4) code**
  - 4-bit information, 7-bit codeword (3-bit redundancy)
  - Information vectors
    - [0 0 0 0], [0 0 0 1], [0 0 1 0], [0 0 1 1], … …, [1 1 1 0], [1 1 1 1]
    - 2^4 = 16 possible information vectors ➔ 16 codewords
  - Choose 16 codewords out of 2^7 = 128 possible 7-bit combinations.
  - Codeword vector **c** is a linear function of information vector **x**

$$\mathbf{c}_{(1\times 7)} = \mathbf{x}_{(1\times 4)} \cdot \mathbf{G}_{(4\times 7)} = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 \end{bmatrix} \times \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

  - The codeword for [0 0 0 0]:
  - The codeword for [0 0 1 1]:
  - The codeword for [0 1 0 1]:

UNIVERSITY OF ARKANSAS

# LBC: GENERATION MATRIX

- **(n, k) linear block code**
  - k-bit of information, n-bit of codeword
  - Information vector (1 x *k* row vector) $\mathbf{x} = [x_1, x_2, \cdots, x_k]$
  - Codeword vector (1 x *n* row vector) $\mathbf{c} = [c_1, c_2, \cdots, c_n]$
  - Linear relationship between information vector and codeword vector
- **Generation matrix (size *k x n* matrix)**

$$\mathbf{c} = \mathbf{x} \cdot \mathbf{G}$$

$$\mathbf{G} = \left[\mathbf{I}_{k\times k} \middle| \mathbf{P}_{k\times(n-k)}\right] = \left[\begin{array}{c|c} \underbrace{\begin{matrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{matrix}}_{k\times k} & \underbrace{\begin{matrix} p_{11} & p_{12} & \cdots & p_{1(n-k)} \\ p_{21} & p_{22} & \cdots & p_{2(n-k)} \\ \vdots & \vdots & \ddots & \vdots \\ p_{n1} & p_{n2} & \cdots & p_{n(n-k)} \end{matrix}}_{k\times(n-k)} \end{array}\right]$$

The first k bits of codeword are the same as information bits: systematic code

# LBC: GENERATION MATRIX

- **Generation matrix (cont'd)**

The k-th row of **G**

$$\mathbf{c} = \mathbf{x} \times \mathbf{G} = [x_1, x_2, \cdots, x_k] \cdot \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_k \end{bmatrix} = x_1 \cdot \mathbf{g}_1 + x_2 \cdot \mathbf{g}_2 + \cdots + x_k \cdot \mathbf{g}_k$$

- **Example**

$$\mathbf{c}_{(1\times 7)} = \mathbf{x}_{(1\times 4)} \cdot \mathbf{G}_{(4\times 7)} = [x_1 \quad x_2 \quad x_3 \quad x_4] \times \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{bmatrix}$$

– The codeword for [ 0 1 0 0]

– The codeword for [ 0 1 0 1]

# LBC: PARITY CHECK MATRIX

- **Parity check matrix (an *n* x (*n*-*k*) matrix)**

$$\mathbf{H}_{n\times(n-k)} = \begin{bmatrix} \mathbf{P}_{k\times(n-k)} \\ \mathbf{I}_{(n-k)\times(n-k)} \end{bmatrix}$$

- **The product between G and H is an all zero matrix**

$$\mathbf{G}_{k\times n} \times \mathbf{H}_{n\times(n-k)} = \left[\mathbf{I}_{k\times k} \middle| \mathbf{P}_{k\times(n-k)}\right] \times \begin{bmatrix} \mathbf{P}_{k\times(n-k)} \\ \mathbf{I}_{(n-k)\times(n-k)} \end{bmatrix} = \mathbf{P}_{k\times(n-k)} + \mathbf{P}_{k\times(n-k)} = \mathbf{0}_{k\times(n-k)}$$

- **For any valid codeword** $\mathbf{c} = \mathbf{x} \times \mathbf{G}$
  - Thus $\mathbf{c}\cdot\mathbf{H} = \mathbf{x}\cdot\mathbf{G}\cdot\mathbf{H} = \mathbf{x}\cdot(\mathbf{G}\cdot\mathbf{H}) = \mathbf{c}\cdot\mathbf{0} = \mathbf{0}$
  - Any valid codeword multiplied by **H** is 0!
    - The receiver can use this to detect if there is an error during transmission.

# LBC: PARITY CHECK MATRIX

- **Parity check matrix**

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

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

– y = c + 0 = [0 0 1 1 1 0 1]

y H =

– y = c + e = [0 0 1 1 1 0 1] + [1 0 0 0 0 0 0] = [1 0 1 1 1 0 1]

y H =

# LBC: SYNDROME DECODING

- **Received vector**

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

codeword ($\mathbf{c}$) error vector ($\mathbf{e}$)

- **Syndrome**

$$\mathbf{s}_{1\times(n-k)} = \mathbf{y}_{1\times n}\mathbf{H}_{n\times(n-k)} = (\mathbf{c}+\mathbf{e})\cdot\mathbf{H} = \mathbf{c}\cdot\mathbf{H} + \mathbf{e}\cdot\mathbf{H} = \mathbf{e}\cdot\mathbf{H}$$

$$\boxed{\mathbf{s} = \mathbf{e}\cdot\mathbf{H}}$$

  - Syndrome: for each error pattern, there is a syndrome **s**
    - We can guess the error pattern **e** by looking at the syndrome!
  - Error vector: n-bit ➔ $2^n$ error vectors
  - Syndrome: (n-k)-bit ➔ $2^{n-k}$ syndromes
  - There are more error vectors than syndromes!
    - Some error vectors might have the same syndrome
    - Given syndrome, there might be more error vectors ➔ just randomly choose one.

# LBC: SYNDROME DECODING EXAMPLE

- **Example (5, 2) systematic code**

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

  – codewords

| Information: **x** | Codeword: **c** |
|---|---|
| 00 | 00000 |
| 01 | 01011 |
| 10 | 10101 |
| 11 | 11110 |

  – # of error vectors:

  – # of syndromes

# LBC: SYNDROME DECODING EXAMPLE

- **Example (5, 2) systematic code**
  - Syndromes

    1. e = [0 0 0 0 0] ➔ s =
    2. e = [0 0 0 0 1] ➔ s =
    3. e = [0 0 0 1 0] ➔ s =
    4. e = [0 0 1 0 0] ➔ s =
    5. e = [0 1 0 0 0] ➔ s =
    6. e = [1 0 0 0 0] ➔ s =
    7. e = [1 0 0 0 1] ➔ s =
    8. e = [1 0 0 1 0] ➔ s =
    9. e = [1 0 1 0 0] ➔ s =
    10. e = [1 1 0 0 0] ➔ s =
    11. ……

# LBC: SYNDROME DECODING EXAMPLE

- **Example (5, 2) systematic code**
  - Syndrome table

| syndromes | error vector |
|---|---|
| 0 0 0 | 00000 |
| 0 0 1 | 00001 |
| 0 1 0 | 00010 |
| 1 0 0 | 00100 |
| 0 1 1 | 01000 |
| 1 0 1 | 10000 |
| 1 1 1 | 10010 |
| 1 1 0 | 11000 |

  - It can correct all 1-bit error
  - It can detect 2-bit error
  - More than 2-bit error is beyond the capability of this (5, 2)-code

# LBC: STANDARD ARRAY

- **Standard array:**
  - Columns: all $2^k$ codewords
  - Rows: all $2^{n-k}$ correctable error patterns
  - Example: (5, 2) systematic code with

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

  - Each row is called a coset
  - The first element in a row is called a coset leader
  - All elements in the same coset have the same syndrome

# LBC: SYNDROME DECODING

- **Example (5, 2) systematic code**

  – Tx

  x = [0 1] ➔ c =

  y = c + e = c + [0 0 1 0 0] =

  – Rx

  s = yH =

  e =

  c = y + e

  x =

  – Tx

  x = [1 1] ➔ c =

  y = c + e = c + [1 0 0 1 0] =

  – Rx

  s = yH =

  e =

  c = y + e

  x =

# LBC: SYNDROME DECODING

- **Example (5, 2) systematic code**
  - Tx

    $x = [0\ 1] \rightarrow c =$

    $y = c + e = c + [0\ 0\ 1\ 1\ 0] =$
  - Rx

    $s = yH =$

    $e =$

    $c = y + e$

    $x =$

  - Tx

    $x = [1\ 1] \rightarrow c =$

    $y = c + e = c + [0\ 0\ 0\ 0\ 1] =$
  - Rx

    $s = yH =$

    $e =$

    $c = y + e$

    $x =$

UNIVERSITY OF ARKANSAS

# OUTLINE

- Introduction
- Types of Error Control
- Structured Sequence
- Linear Block Code
- **Error Detection and Correction Capability**

# PROPERTIES

- **Design of LBC**
  - k bits of information: $2^k$ possible combinations
  - n bits of codeword: $2^n$ possible combinations
  - one-to-one relationship between (k-bit information, n-bit codeword)
    - Out of the $2^n$ possible combinations, we only need $2^k$ codewords
      - How to choose the $2^k$ codeword?
  - E.g. (3, 1) LBC: choose 2 codewords out of 8 possible combinations.
    - ‘0’: (0 0 0), ‘1’ (1 1 1) ➔ repetition code
    - ‘0’: (0 0 1), ‘1’ (1 1 0) ➔ another LBC
    - ‘0’: ( 0 1 1), ‘1’ (0 1 0) ➔ another LBC
    - ……
    - Different LBC has different error performance, we want choose the one minimizing the error probability.

# PROPERTIES: HAMMING DISTANCE

- **Design of LBC (Cont'd)**
  - Rule of thumb: we want the codewords to be as "different" as possible.
    - '0': (0 0 0), '1' (1 1 1) ➔ repetition code: 3 bits difference
    - '0': ( 0 1 1), '1' (0 1 0) ➔ another LBC: 1 bit difference
  - Definition: Hamming distance
    - The Hamming distance $d_{ij}$ between two binary codewords $\mathbf{c}_i = \left[c_{i1}, c_{i2}, \cdots, c_{in}\right]$ $\mathbf{c}_j = \left[c_{j1}, c_{j2}, \cdots, c_{jn}\right]$ is defined as the number of bits in which they differ.
    - Larger Hamming distance ➔ two codewords are further apart ➔ the probability of choosing the wrong codeword is smaller ➔ better power efficiency
  - Minimum Hamming distance:
    - If there are more than two codewords, one Hamming distance for each pair of codewords

$$d_{\min} = \min_{i \neq j} d_{ij}$$

    - E.g. the minimum Hamming distance of the (5, 2) code is:

# PROPERTIES: ERROR DETECTION CAPABILITY

- **Error detection capability**
  - For LBC code with minimum Hamming distance $d_{\min}$
    - All error patterns with error bits less than or equal to

      $$t = d_{\min} - 1$$

      can be detected (non-zero syndrome) (why?)
    - All error patterns with error bits less than or equal to

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

      can be corrected (why?)


- E.g. $d_{\min} = 3$
  - All 1-bit error can be corrected
  - All 2-bit error can be detected

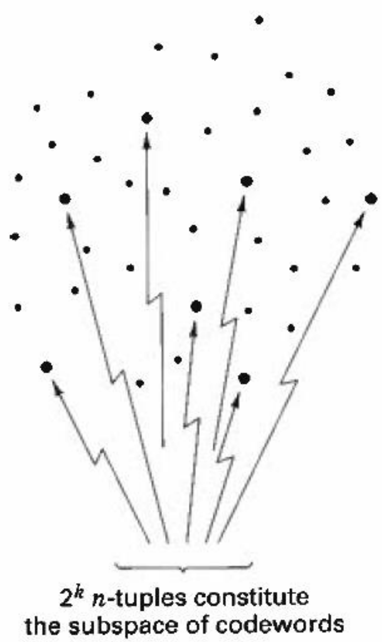
University of Arkansas

# PROPERTIES: VECTOR SUBSPACE

- **Vector subspace**
  - Vector space $V_n$ : the set of all length-*n* vectors
  - Vector subspace: a subset $S$ of vector space $V_n$ is called a subspace if the following two conditions are met:
    - 1. The all-zero vector is in $S$
    - 2. the sum of any two vectors in $S$ is also in $S$ (closure property)
  - Example
    - Vector space $V_3$
      - 000, 001, 010, 011, 100, 101, 110, 111
    - Are the following vector subspaces?
      - {000, 001}
      - {001, 010, 011}
      - {000, 001, 010, 011}
      - {000, 100, 010, 001}

# PROPERTIES: VECTOR SUBSPACE

- **($n$, $k$) Linear block code**
  - The $2^k$ length-$n$ codewords form a vector subspace of $V_n$
    - 1. the all-zero length-n vector is a valid codeword
    - 2. the sum of any two codewords is still a codeword.
  - Geometric representation




- We want the codewords to be as far apart as possible (maximize $d_{\min}$ ).

UNIVERSITY OF ARKANSAS

# PROPERTIES: OPTIMUM DECODER

- **Optimum decoder**
  - If all the codewords are equiprobable, then the optimum receiver is the maximum likelihood receiver
  - E.g. AWGN channel with BPSK
    - Message vector: $\mathbf{s}_m = [s_{m1}, c_{m2}, \cdots, s_{mk}] \in \{0,1\}^k$
    - Encoding: $\mathbf{c}_m = [c_{m1}, c_{m2}, \cdots, c_{mn}] \in \{0,1\}^n$
    - BPSK modulation: $\mathbf{x}_m = [x_{m1}, x_{m2}, \cdots, x_{mn}] \in \{-1,1\}^n$
    - Received signal: $\mathbf{r} = \mathbf{x}_m + \mathbf{z}$ ← AWGN
    - BPSK demodulation: $\mathbf{y} = \text{demod}(\mathbf{r}) = [y_1, y_2, \cdots, y_n] \in \{-1,1\}^n$
    - Decode: $\hat{\mathbf{s}}_m = \text{decode}(\mathbf{y})$
    - Maximum likelihood decoder: choose $\mathbf{s}_m$ that can maximize the likelihood function

$$p(\mathbf{y} \mid \mathbf{s}_u)$$

# PROPERTIES: OPTIMUM DECODER

- **Optimum Decoder (Cont'd)**
  - Likelihood function

$$p(\mathbf{y} \mid \mathbf{s}_u) = p(y_1 \mid s_{u1}) \times p(y_2 \mid s_{u2}) \times \cdots \times p(y_n \mid s_{un})$$

  - $p(y_1 \mid s_{u1}) =$

$$p(\mathbf{y} \mid \mathbf{s}_u) = p^j (1 - p)^{n-j}$$

# PROPERTIES: OPTIMUM DECODER

- **Optimum decoder (Cont'd)**
  - Likelihood function

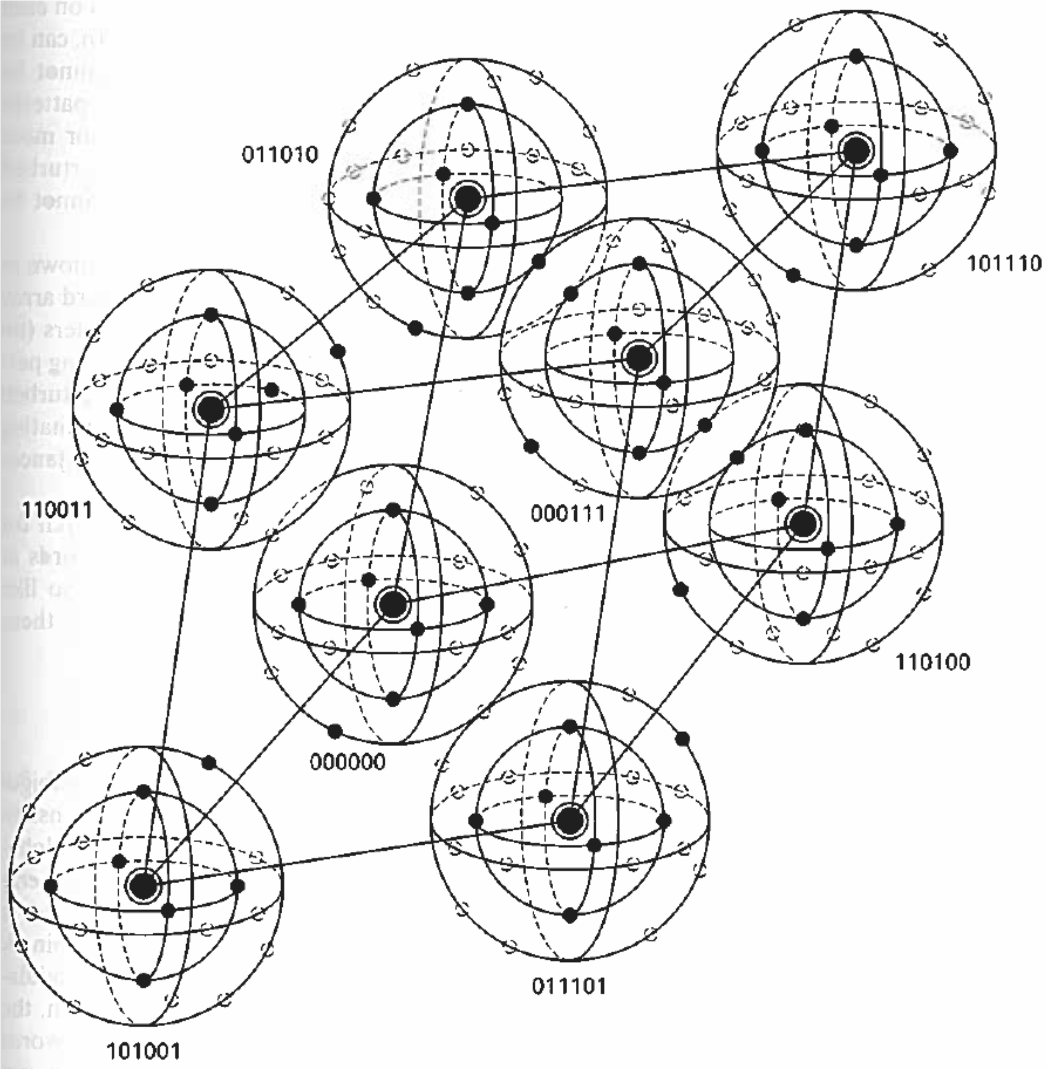
$$p(\mathbf{y} \mid \mathbf{s}_u) = p^j (1-p)^{n-j}$$

    - $j$ is the Hamming distance between $\mathbf{y}$ and $\mathbf{s}_u$
  - Since $p < 1-p$, maximize $p(\mathbf{y} \mid \mathbf{s}_u)$ ➔ minimize $j$
    - Choose $\mathbf{s}_u$ that has the smallest Hamming distance with $\mathbf{y}$

$$\hat{\mathbf{s}}_m = \arg\min_{\mathbf{s}_u} \| \mathbf{y} - \mathbf{s}_u \|_H$$

# PROPERTIES: OPTIMUM DECODER

- **Visualization of a 6-Tuple space**




UNIVERSITY OF ARKANSAS

# PROPERTIES: ERROR PERFORMANCE

- **Message error probability**
  - The probability that the decoder commits an erroneous decoding (the probabilities that there are more than *t*-bit error)

$$P_M \le \sum_{j=t+1}^{n} \binom{n}{j} p^j (1-p)^{n-j}$$

    - *t*: the correction capability of the code
    - The equality holds if all t bits errors can be corrected, but no t+1 bits errors can be corrected.

- **Bit error probability**

$$P_B \approx \frac{1}{n} \sum_{j=t+1}^{n} j \binom{n}{j} p^j (1-p)^{n-j}$$

# PROPERTIES: ERROR PERFORMANCE

- **Example:**
  - Consider a (16, 8) linear block code capable of double-error corrections. Assume that BPSK is used with the received $E_b / N_0$ is 10 dB. (1) what is the message error probability? (2) what is the bit error probability?

# PROPERTIES: SOME SPECIFIC LBC

- **Some specific LBC**
  - Hamming code

$$(n,k)=(2^m-1,2^m-1-m)$$

$$d_{\min}=3$$

  - Cyclic code
    - cyclic redundancy check
  - Reed-Solomon code
    - Compact Disc (CD)