Communication Theory II

Lecture 9: Error control coding: Linear Block Code

Error control coding

Error control coding, also known as channel coding, is a technique used in communication systems to detect and correct errors that may occur during data transmission over unreliable or noisy communication channels.

The purpose of error control coding is to enhance the reliability and robustness of data transmission, ensuring that the received data is as close as possible to the original transmitted data, even in the presence of errors

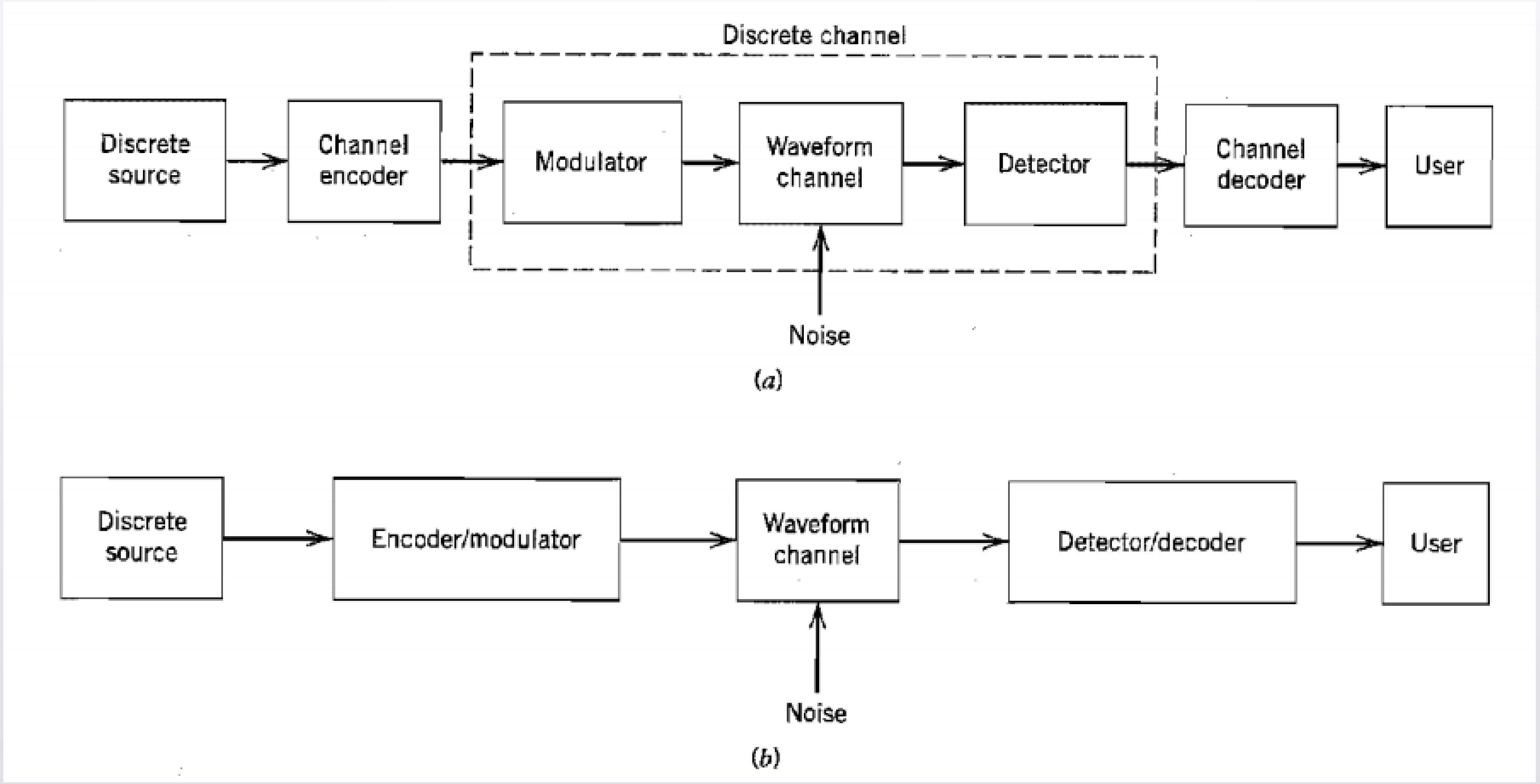
Error control coding

There are two main types of error control coding techniques:

1. Forward Error Correction (FEC): In FEC, the transmitter adds redundant information (error-correcting codes) to the original data before transmission. This redundancy allows the receiver to detect and correct errors without the need for retransmission or feedback from the transmitter.

2.Automatic Repeat Request (ARQ): In ARQ, the receiver detects errors and requests the sender to retransmit the corrupted data. The sender may have to retransmit the data several times until the receiver successfully receives error-free data. ARQ is widely used in many communication protocols, such as TCP (Transmission Control Protocol) in computer networks.

Communication system



Error control coding

Block codes and convolutional codes are two distinct types of error control coding techniques used in communication systems for error detection and correction

Block Codes: Block codes are a type of error control coding in which the input data stream is divided into fixed-size blocks, and redundancy (error-correcting codes) is added to each block before transmission. The encoder processes the data block by block, independently of the surrounding data. At the receiver's end, the received blocks are checked for errors, and if errors are detected, they are corrected using the added redundancy.

Convolutional Codes: Convolutional codes are a different type of error control coding where the encoding process involves a sliding window over the input data stream. Unlike block codes, which process fixed-size blocks independently, convolutional codes encode data by taking into account the history of previous bits. This history is stored in shift registers, and the encoder uses a combination of the current input bit and the contents of the shift registers to generate the encoded output.

Error control coding

• Non-recursive: Block codes do not involve feedback; each block's encoding is independent of the previous ones.
• Simple implementation: Block codes are generally easier to implement and decode, making them suitable for various applications.
• Burst error correction: They are effective at correcting burst errors, where multiple consecutive bits are affected by errors.

Block Codes: Block codes are a type of error control widely used in applications like data storage systems, CD/DVD error correction, and various digital communication systems.

Recursive: Convolutional codes use feedback and have memory, meaning the current encoding depends on past inputs.
Continuous output: The encoding process generates a continuous stream of encoded bits, which may result in better bandwidth efficiency compared to block codes.
More complex implementation: Decoding convolutional codes can be more complex than block codes, particularly for long constraint lengths.

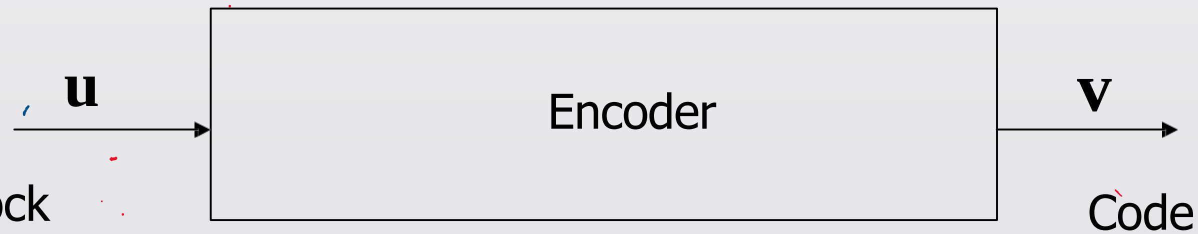
Convolutional codes are commonly used in applications like digital communication systems (e.g., wireless communication, satellite communication), especially when the channel has varying error characteristics. Viterbi algorithm is often used for decoding convolutional codes efficiently.

Introduction to Linear Block Codes

- We assume that the output of an information source is a sequence of binary digits “0” or “1”
- This binary information sequence is segmented into *message* block of fixed length, denoted by **u.**
  - Each message block consists of k information digits.
  - There are a total of 2^k distinct message.
- The encoder transforms each input message **u** into a binary n-tuple **v** with $n > k$
  - This n-tuple **v** is referred to as the *code word* (or *code vector*) of the message **u**.
  - There are distinct 2^n code words.

Introduction to Linear Block Codes

- This set of 2^n code words is called a *block* code.
- For a *block* code to be useful, there should be a one-to-one correspondence between a message **u** and its code word **v.**
- A desirable structure for a block code to possess is the linearity. With this structure, the encoding complexity will be greatly reduced.



Message block

(k info. digits)

(2^k distinct message)

Code word

(n tuple, $n > k$) (2^n

distinct code word)

Introduction to Linear Block Codes

- **Definition :** A block code of length n and 2^k code word is called a *linear* (n, k) code if its 2^k code words form a k-dimensional subspace of the vector space of all the n-tuple over the field GF(2).

- In fact, a binary block code is linear if the module-2 sum of two code word is also a code word
  - **0** must be code word.
  - The block code given in Table 3.1 is a (7, 4) linear code.

Introduction to Linear Block Codes

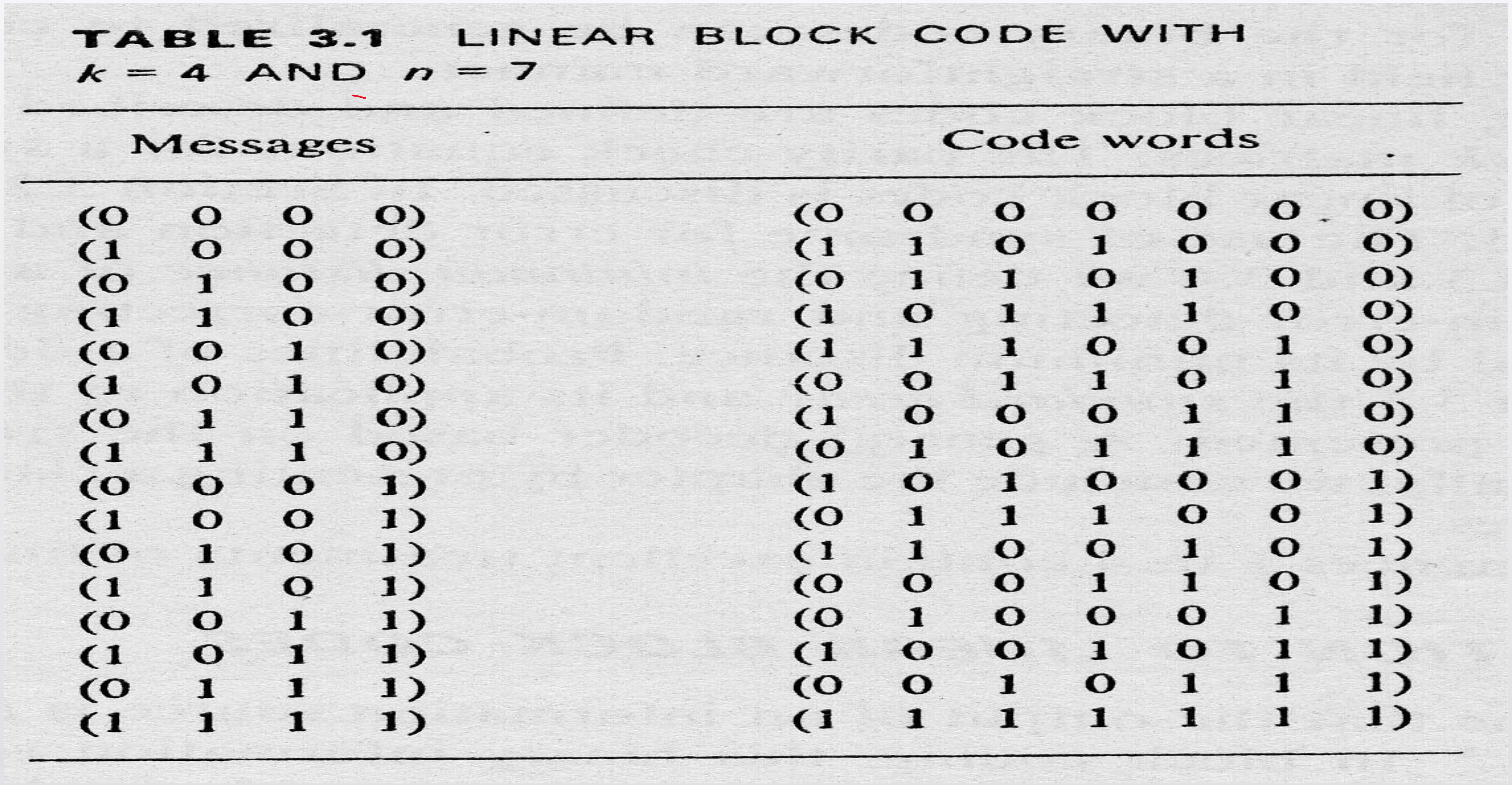


TABLE 3.1 LINEAR BLOCK CODE WITH $k = 4$ AND $n = 7$

Messages	Code words
(0 0 0 0)	(0 0 0 0 0 0 0)
(1 0 0 0)	(1 1 0 1 0 0 0)
(0 1 0 0)	(0 1 1 0 1 0 0)
(1 1 0 0)	(1 0 1 1 1 0 0)
(0 0 1 0)	(1 1 1 0 0 1 0)
(1 0 1 0)	(0 0 1 1 0 1 0)
(0 1 1 0)	(1 0 0 0 1 1 0)
(1 1 1 0)	(0 1 0 1 1 1 0)
(0 0 0 1)	(1 0 1 0 0 0 1)
(1 0 0 1)	(0 1 1 1 0 0 1)
(0 1 0 1)	(1 1 0 0 1 0 1)
(1 1 0 1)	(0 0 0 1 1 0 1)
(0 0 1 1)	(0 1 0 0 0 1 1)
(1 0 1 1)	(1 0 0 1 0 1 1)
(0 1 1 1)	(0 0 1 0 1 1 1)
(1 1 1 1)	(1 1 1 1 1 1 1)

For large *k*, it is virtually impossible to build up the loop up table.

Introduction to Linear Block Codes

- Since an (n, k) linear code C is a k-dimensional subspace of the vector space V_n of all the binary n-tuple, it is possible to find k linearly independent code word, $\mathbf{g}_0, \mathbf{g}_1, \ldots, \mathbf{g}_{k-1}$ in C

$$\mathbf{v} = u_0\mathbf{g}_0 + u_1\mathbf{g}_1 + \cdots + u_{k-1}\mathbf{g}_{k-1} \qquad (1)$$

where $u_i = 0$ or 1 for $0 \leq i < k$

Introduction to Linear Block Codes

- Let us arrange these *k* linearly independent code words as the rows of a *k* ×*n* matrix as follows:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \cdot \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{g}_{k-1} \end{bmatrix} = \begin{bmatrix} g_{00} & g_{01} & g_{02} & \cdot & \cdot & g_{0,n-1} \\ g_{10} & g_{11} & g_{12} & \cdot & \cdot & g_{1,n-1} \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot & \cdot & \cdot \\ g_{k-1,0} & g_{k-1,1} & g_{k-1,2} & \cdot & \cdot & g_{k-1,n-1} \end{bmatrix} \tag{2}$$

where $\mathbf{g}_i = (g_{i0}, g_{i1},\ldots,g_{i,n-1})$ for $0 \leq i < k$

Introduction to Linear Block Codes

- If $\mathbf{u} = (u_0, u_1, \ldots, u_{k\text{-}1})$ is the message to be encoded, the corresponding code word can be given as follows:

$$\mathbf{v} = \mathbf{u}.\ \mathbf{G}$$

$$= (u_0, u_1, \ldots, u_{k-1}) \bullet \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{g}_{k-1} \end{bmatrix} \qquad (3)$$

$$= u_0\mathbf{g}_0 + u_1\mathbf{g}_1 + \cdots + u_{k-1}\mathbf{g}_{k-1}$$

Introduction to Linear Block Codes

- Because the rows of **G** generate the (n, k) linear code C, the matrix **G** is called a *generator matrix* for C

- Note that any k linearly independent code words of an (n, k) linear code can be used to form a generator matrix for the code

- It follows from (3) that an (n, k) linear code is completely specified by the k rows of a generator matrix **G**

Introduction to Linear Block Codes

Example 3.1

- the (7, 4) linear code given in Table 3.1 has the following matrix as a generator matrix :

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \mathbf{g}_2 \\ \mathbf{g}_3 \end{bmatrix} = \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

- If u = (1 1 0 1) is the message to be encoded, its corresponding code word, according to (3), would be

$$\begin{aligned} \mathbf{v} &= 1\cdot\mathbf{g}_0 + 1\cdot\mathbf{g}_1 + 0\cdot\mathbf{g}_2 + 1\cdot\mathbf{g}_3 \\ &= (1101000) + (0110100) + (1010001) \\ &= (0001101) \end{aligned}$$

Introduction to Linear Block Codes

- A desirable property for a linear block code is the *systematic structure* of the code words as shown in below.
  - where a code word is divided into two parts
    - The message part consists of k information digits
    - The redundant checking part consists of $n - k$ parity-check digits
- A linear block code with this structure is referred to as a *linear systematic block code*

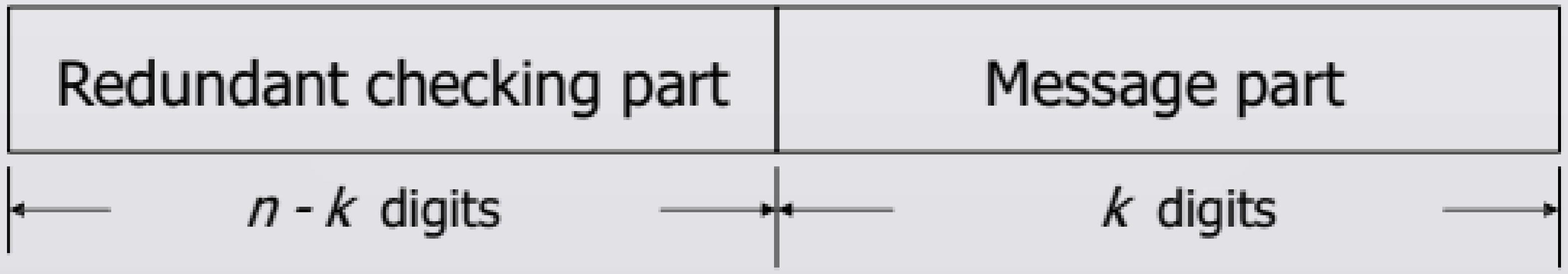


Fig. 1 Systematic format of a code word

Introduction to Linear Block Codes

$$\mathbf{m} = [m_0, m_1, \ldots, m_{k-1}]$$

Message bit

$$\mathbf{b} = [b_0, b_1, \ldots, b_{n-k-1}]$$

Parity

$$\mathbf{c} = [c_0, c_1, \ldots, c_{n-1}]$$

Code

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

$$\mathbf{b} = \mathbf{m}\mathbf{P}$$

$$\mathbf{b} = \mathbf{m}\mathbf{P}$$

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

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

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

$\mathbf{P}$ is the k-by-$(n-k)$ *coefficient matrix*

Introduction to Linear Block Codes

- A linear systematic (n, k) code is completely specified by a $k \times n$ matrix G of the following form :

$$\mathbf{G} = [\mathbf{P} \vdots \mathbf{I}_k]$$

$$
\begin{array}{ccc}
 & & \begin{array}{cccccc|ccccccc} \multicolumn{6}{c|}{\longleftarrow \textbf{P matrix} \longrightarrow} & \multicolumn{7}{c}{\longleftarrow \boldsymbol{k} \times \boldsymbol{k} \textbf{ identity matrix} \longrightarrow} \end{array} \\
\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \mathbf{g}_2 \\ \cdot \\ \cdot \\ \cdot \\ \mathbf{g}_{k-1} \end{bmatrix} & = &
\left[\begin{array}{cccccc|ccccccc}
p_{00} & p_{01} & \cdot & \cdot & \cdot & p_{0,n-k-1} & 1 & 0 & 0 & \cdot & \cdot & \cdot & 0 \\
p_{10} & p_{11} & \cdot & \cdot & \cdot & p_{1,n-k-1} & 0 & 1 & 0 & \cdot & \cdot & \cdot & 0 \\
p_{20} & p_{21} & \cdot & \cdot & \cdot & p_{2,n-k-1} & 0 & 0 & 1 & \cdot & \cdot & \cdot & 0 \\
 & & & & & & & & & & & & \\
 & & & & & & & & & & & & \\
 & & & & & & & & & & & & \\
p_{k-1,0} & p_{k-1,1} & \cdot & \cdot & \cdot & p_{k-1,\,n-k-1} & 0 & 0 & 0 & 0 & 0 & 0 & 1
\end{array}\right]
\end{array}
$$

where $p_{ij} = 0$ or 1

where $p_{ij} = 0$ or 1

Introduction to Linear Block Codes

- Let $\mathbf{u} = (u_0, u_1, \ldots, u_{k-1})$ be the message to be encoded
- The corresponding code word is

$$\begin{aligned} v &= (v_0, v_1, v_2, \ldots, v_{n-1}) \\ &= (u_0, u_1, \ldots, u_{k-1}) \cdot G \qquad (5) \end{aligned}$$

- It follows from (4) & (5) that the components of $\mathbf{v}$ are

$$\underbrace{v_{n-k+i}}_{\text{Message}} = u_i \qquad \text{for } 0 \leq i < k \qquad (6a)$$

and

$$\underbrace{v_j}_{\text{Parity check}} = u_0 p_{0j} + u_1 p_{1j} + \cdots + u_{k-1} p_{k-1,j} \qquad \text{for } 0 \leq j < n-k \qquad (6b)$$

Introduction to Linear Block Codes

- Equation (6a) shows that the rightmost k digits of a code word **v** are identical to the information digits $u_0, u_1, \dots u_{k-1}$ to be encoded

- Equation (6b) shown that the leftmost $n - k$ redundant digits are linear sums of the information digits

- The $n - k$ equations given by (6b) are called *parity-check equations* of the code

Introduction to Linear Block Codes

- **Example 3.2**
  - The matrix **G** given in example 3.1
  - Let **u** = (u_0, u_1, u_2, u_3) be the message to be encoded
  - Let **v** = (v_0, v_1, v_2, v_3, v_4, v_5,v_6) be the corresponding code word
  - Solution :

$$\mathbf{v} = \mathbf{u} \cdot \mathbf{G} = (u_0, u_1, u_2, u_3) \cdot \begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 0 & 0 & 1 \end{bmatrix}$$

Introduction to Linear Block Codes

- By matrix multiplication, we obtain the following digits of the code word **v**

$$v_6 = u_3$$
$$v_5 = u_2$$
$$v_4 = u_1$$
$$v_3 = u_0$$
$$v_2 = u_1 + u_2 + u_3$$
$$v_1 = u_0 + u_1 + u_2$$
$$v_0 = u_0 + u_2 + u_3$$

The code word corresponding to the message (1 0 1 1) is (1 0 0 1 0 1 1)

Introduction to Linear Block Codes

- For any $k \times n$ matrix **G** with k linearly independent rows, there exists an $(n\text{-}k) \times n$ matrix **H** with $n\text{-}k$ linearly independent rows such that any vector in the row space of **G** is orthogonal to the rows of **H** and any vector that is orthogonal to the rows of **H** is in the *row space* of **G**.

  An n-tuple **v** is a code word in the code generated by **G** if and only if $\mathbf{v} \bullet H^T = 0$
- This matrix **H** is called a *parity-check matrix* of the code
- The 2^{n-k} linear combinations of the rows of matrix **H** form an $(n, n - k)$ linear code C_d
- This code is the *null space* of the (n, k) linear code C generated by matrix **G**
- C_d is called the *dual code* of C

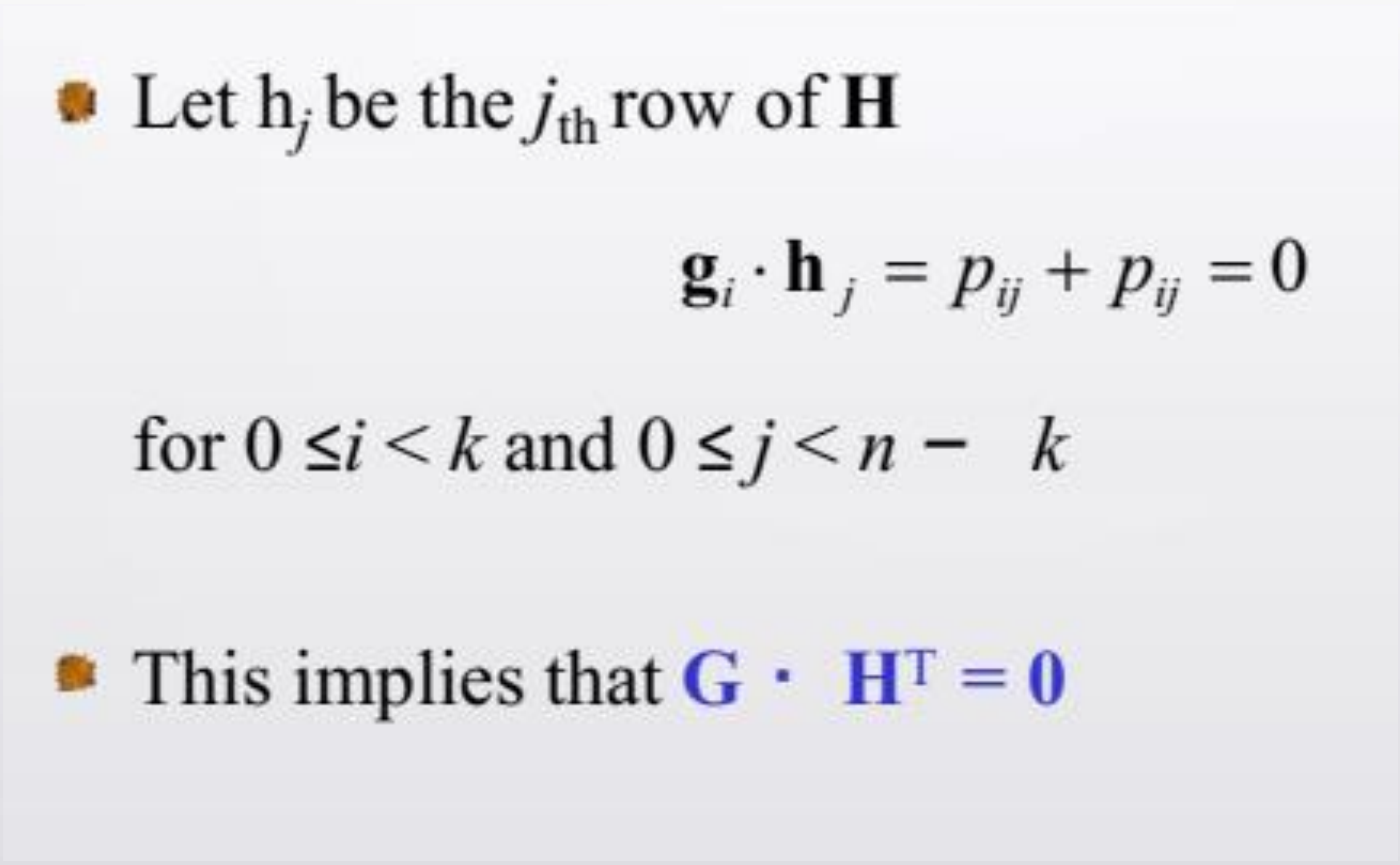
Introduction to Linear Block Codes

- If the generator matrix of an (n,k) linear code is in the systematic form of (4), the parity-check matrix may take the following form :

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

$$= \begin{bmatrix} 1 & 0 & 0 & . & . & . & 0 & p_{00} & p_{10} & . & . & . & p_{k-1,0} \\ 0 & 1 & 0 & . & . & . & 0 & p_{01} & p_{11} & . & . & . & p_{k-1,1} \\ 0 & 0 & 1 & . & . & . & 0 & p_{02} & p_{12} & . & . & . & p_{k-1,2} \\ . & & & & & & & & & & & & \\ . & & & & & & & & & & & & \\ . & & & & & & & & & & & & \\ 0 & 0 & 0 & . & . & . & 1 & p_{0,n-k-1} & p_{1,n-k-1} & . & . & . & p_{k-1,n-k-1} \end{bmatrix} \qquad (7)$$

Introduction to Linear Block Codes



- Let h_j be the j_{th} row of $\mathbf{H}$

$$\mathbf{g}_i \cdot \mathbf{h}_j = p_{ij} + p_{ij} = 0$$

for $0 \leq i < k$ and $0 \leq j < n - k$

- This implies that $\mathbf{G} \cdot \mathbf{H}^T = \mathbf{0}$

Introduction to Linear Block Codes

- Let **u** = (u_0, u_1, …, $u_{\text{k-1}}$) be the message to be encoded
- In systematic form the corresponding code word would be

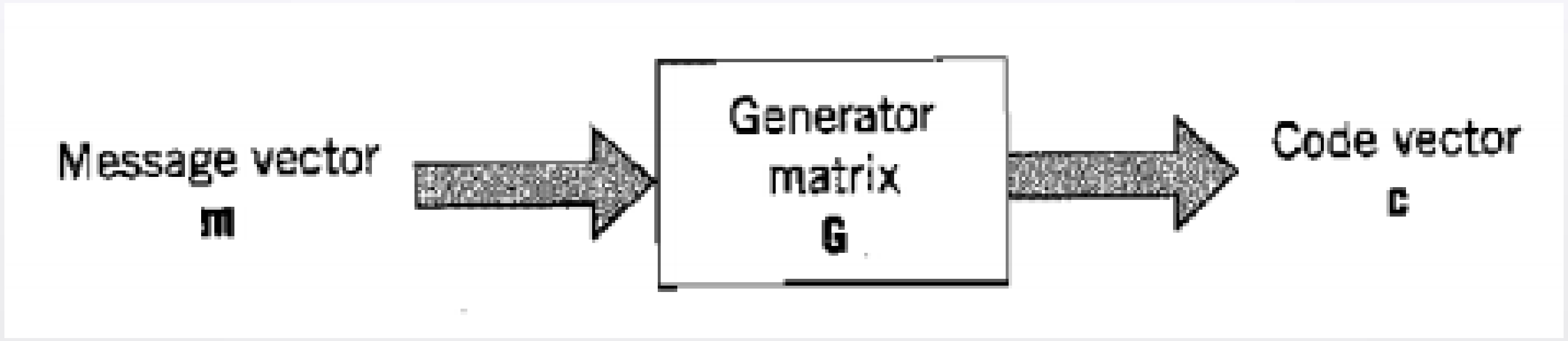
$$\mathrm{v} = (v_0, v_1, \ldots, v_{\text{n-k-1}}, u_0, u_1, \ldots, u_{\text{k-1}})$$

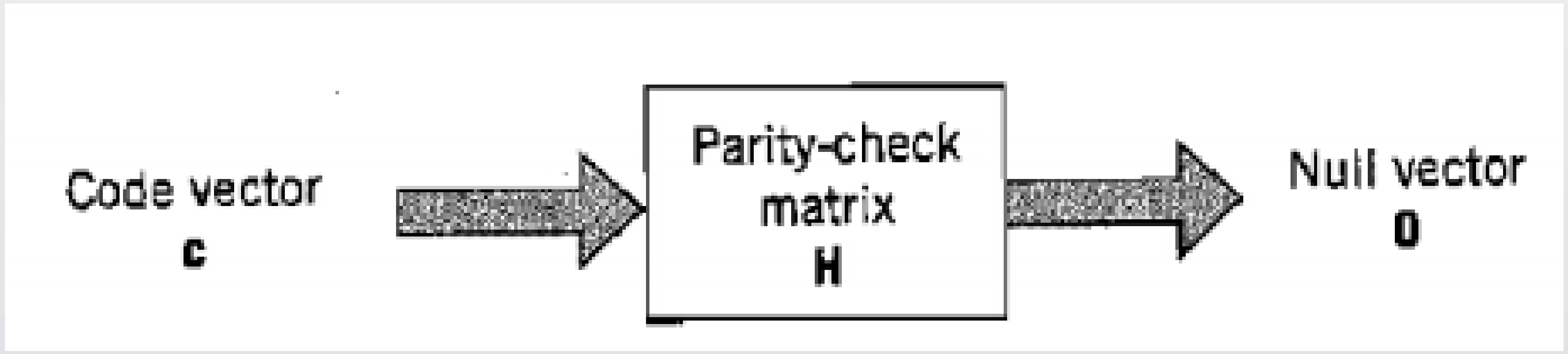
- Using the fact that $\mathbf{v} \bullet \mathbf{H}^T = \mathbf{0}$, we obtain

$$v_j + u_0 p_{0j} + u_1 p_{1j} + \cdots + u_{k\text{-}1} p_{k\text{-}1,j} = 0 \qquad (8)$$

- Rearranging the equation of (8), we obtain the same parity-check equations of (6b)

- An (n, k) linear code is completely specified by its parity-check matrix


Message vector
m
Generator
matrix
G
Code vector
c


Code vector
c
Parity-check
matrix
H
Null vector
0

Introduction to Linear Block Codes

- **Example 3.3**
  - Consider the generator matrix of a (7,4) linear code given in example 3.1
  - The corresponding parity-check matrix is

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

Introduction to Linear Block Codes

- Summaries
  - For any (n, k) linear block code $\boldsymbol{C}$, there exists a $k \times n$ matrix $\mathbf{G}$ whose row space given $\boldsymbol{C}$
  - There exist an $(n - k) \times n$ matrix $\mathbf{H}$ such that an n-tuple $\mathbf{v}$ is a code word in $\boldsymbol{C}$ if and only if $\mathbf{v} \bullet \mathbf{H}^T = 0$
  - If $\mathbf{G}$ is of the form given by (4), then $\mathbf{H}$ may take form given by (7), and vice versa

Introduction to Linear Block Codes

- Based on the equation of (6a) and (6b), the encoding circuit for an (n, k) linear systematic code can be implemented easily
- The encoding circuit is shown in Fig. 3.2
  - where
    - $\longrightarrow\boxed{}\longrightarrow$ denotes a shift-register stage (flip-flop)
    - $\longrightarrow\textcircled{\scriptsize P_{ij}}\longrightarrow$ denotes a connection if $p_{ij} = 1$ and no connection if $p_{ij} = 0$
    - $\oplus$ denotes a modulo-2 adder
    - As soon as the entire message has entered the message register, the n-k parity-check digits are formed at the outputs of the n-k module-2 adders
  - The complexity of the encoding circuit is linear proportional to the block length
  - The encoding circuit for the (7,4) code given in Table 3.1 is shown in Fig 3.3

Introduction to Linear Block Codes

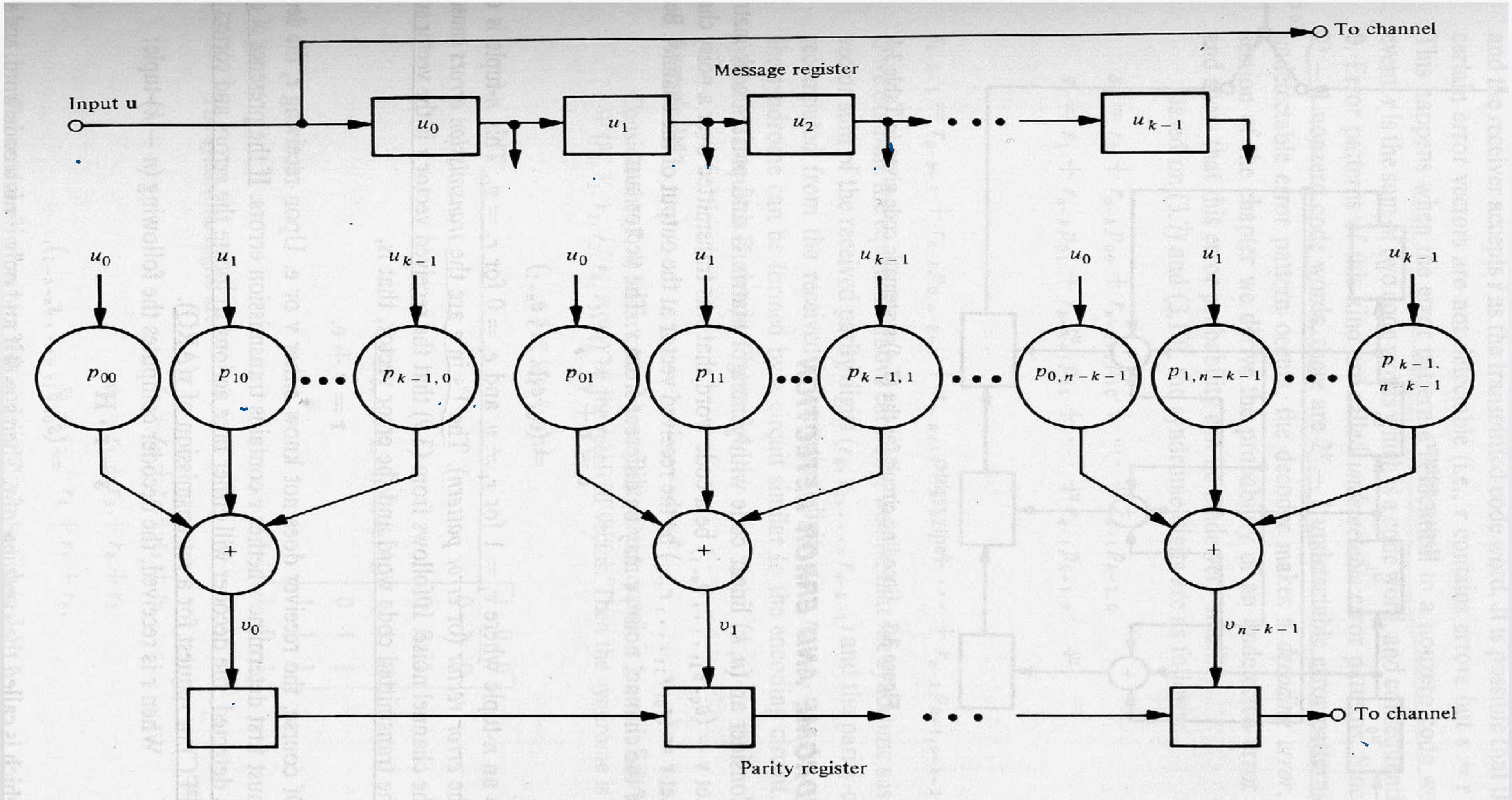


Figure 3.2 Encoding circuit for a linear systematic (n, k) code.

Introduction to Linear Block Codes

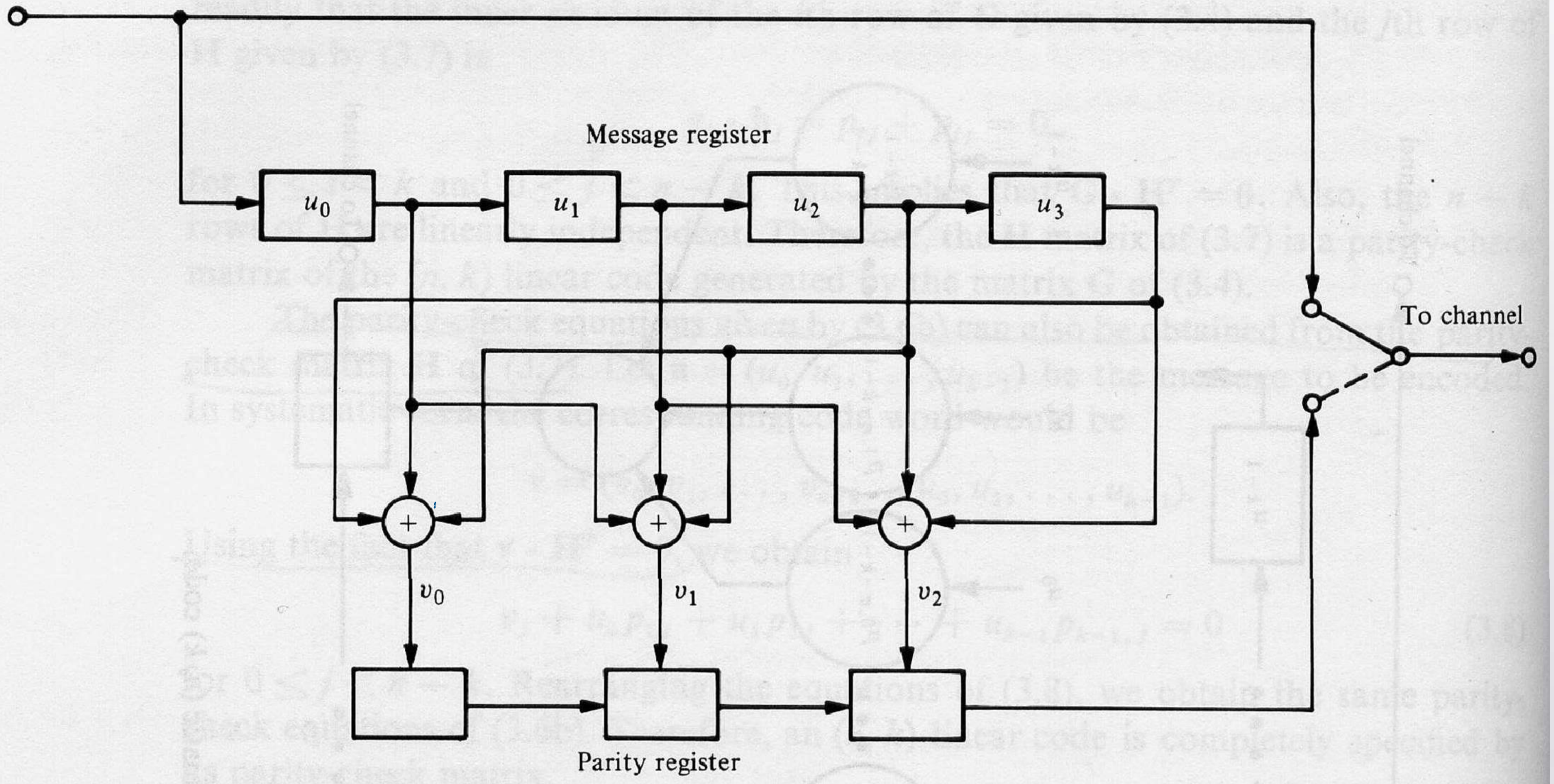
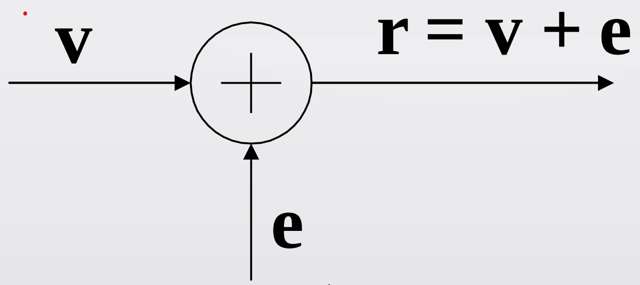


Figure 3.3 Encoding circuit for the (7, 4) systematic code given in Table 3.1.

Syndrome and Error Detection

- Let $\mathbf{v} = (v_0, v_1, \ldots, v_{n-1})$ be a code word that was transmitted over a noisy channel
- Let $\mathbf{r} = (r_0, r_1, \ldots, r_{n-1})$ be the received vector at the output of the channel



- $\mathbf{e} = \mathbf{r} - \mathbf{v} = (e_0, e_1, \ldots, e_{n-1})$ is an n-tuple
  - $e_i = 1$ for $r_i \neq v_i$
  - $e_i = 0$ for $r_i = v_i$
  - The n-tuple $\mathbf{e}$ is called the *error vector* (or *error pattern*)

Syndrome and Error Detection

- Upon receiving **r**, the decoder must first determine whether **r** contains transmission errors
- If the presence of errors is detected, the decoder will take actions to locate the errors
  - Correct errors (FEC)
  - Request for a retransmission of **v** (ARQ)
- When **r** is received, the decoder computes the following $(n - k)$-tuple :

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

$$= (s_0, s_1, \ldots, s_{n-k-1}) \qquad (10)$$

which is called the *syndrome* of **r**

Syndrome and Error Detection

- $\mathbf{s} = \mathbf{0}$ if and only if **r** is a code word and receiver accepts **r** as the transmitted code word
- $\mathbf{s} \neq \mathbf{0}$ if and only if **r** is not a code word and the presence of errors has been detected
- When the error pattern **e** is identical to a nonzero code word (i.e., **r** contain errors but $\mathbf{s} = \mathbf{r} \cdot \mathbf{H}^T = \mathbf{0}$), error patterns of this kind are called *undetectable* error patterns
  - Since there are $2^k - 1$ nonzero code words, there are $2^k - 1$ undetectable error patterns

Syndrome and Error Detection

- Based on (7) and (10), the syndrome digits are as follows : $\mathbf{H} = [\mathbf{I}_{n-k} \quad \mathbf{P}^T]$ $\mathbf{s} = \mathbf{r} \bullet \mathbf{H}^T$

$$s_0 = r_0 + r_{n-k}p_{00} + r_{n-k+1}p_{10} + \cdots + r_{n-1}p_{k-1,0}$$
$$s_1 = r_1 + r_{n-k}p_{01} + r_{n-k+1}p_{11} + \cdots + r_{n-1}p_{k-1,1}$$
$$\cdot \qquad (11)$$
$$s_{n-k-1} = r_{n-k-1} + r_{n-k}p_{0,n-k-1} + r_{n-k+1}p_{1,n-k-1} + \cdots + r_{n-1}p_{k-1,n-k-1}$$

- The syndrome **s** is the vector sum of the received parity digits ($r_0, r_1, \ldots, r_{n-k-1}$) and the parity-check digits recomputed from the received information digits ($r_{n-k}, r_{n-k+1}, \ldots, r_{n-1}$).
- A general syndrome circuit is shown in Fig. 3.4

Syndrome and Error

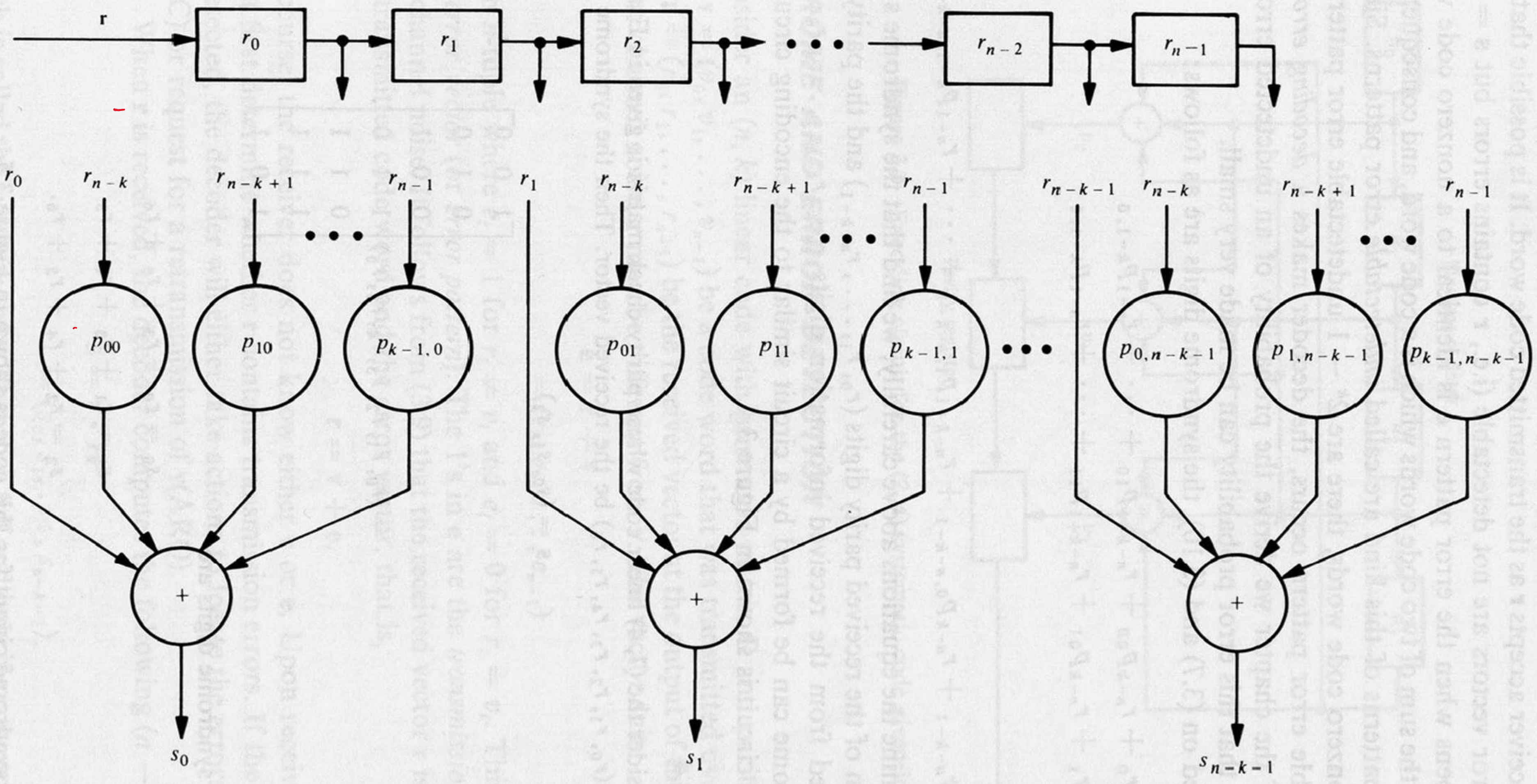


Figure 3.4 Syndrome circuit for a linear systematic (n, k) code.

Syndrome and Error Detection

- **Example 3.4**
  - The parity-check matrix is given in example 3.3
  - Let $\mathbf{r} = (r_0, r_1, r_2, r_3, r_4, r_5, r_6, r_7)$ be the received vector
  - The syndrome is given by

$$\mathbf{s} = (s_0, s_1, s_2) = (r_0, r_1, r_2, r_3, r_4, r_5, r_6) \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 1 & 0 \\ 0 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix}$$

Syndrome and Error Detection

- **Example 3.4**
  - The syndrome digits are

$$s_0 = r_0 + r_3 + r_5 + r_6$$
$$s_1 = r_1 + r_3 + r_4 + r_5$$
$$s_2 = r_2 + r_4 + r_5 + r_6$$

  - The syndrome circuit for this code is shown below

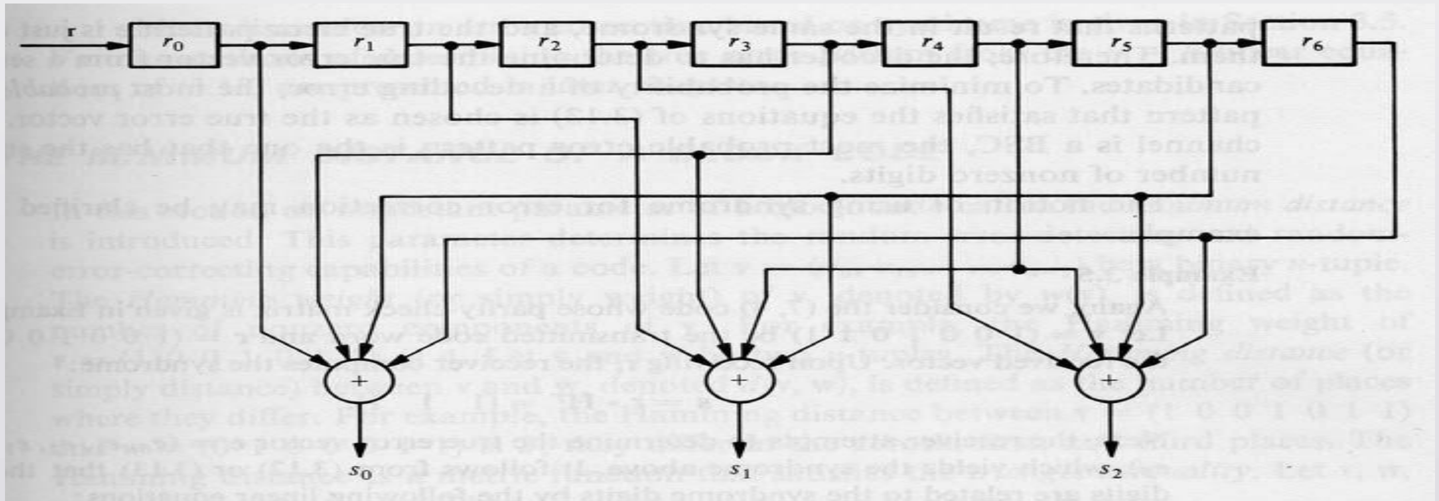


Figure 3.5 Syndrome circuit for the (7, 4) code given in Table 3.1.

Syndrome and Error Detection

- Since **r** is the vector sum of **v** and **e**, it follows from (10) that

$$\mathbf{s} = \mathbf{r} \bullet \mathbf{H}^T = (\mathbf{v} + \mathbf{e}) \bullet \mathbf{H}^T = \mathbf{v} \bullet \mathbf{H}^T + \mathbf{e} \bullet \mathbf{H}^T$$

- however,

$$\mathbf{v} \bullet \mathbf{H}^T = \mathbf{0}$$

- consequently, we obtain the following relation between the syndrome and the error pattern :

$$\mathbf{s} = \mathbf{e} \bullet \mathbf{H}^T \qquad (12)$$

Syndrome and Error Detection

- If the parity-check matrix **H** is expressed in the systematic form as given by (7), multiplying out $\mathbf{e} \bullet \mathbf{H}^T$ yield the following linear relationship between the syndrome digits and the error digits :

n-*k* equations and *n* variables

$$s_0 = e_0 + e_{n-k} p_{00} + e_{n-k+1} p_{10} + \cdots + e_{n-1} p_{k-1,0}$$

$$s_1 = e_1 + e_{n-k} p_{01} + e_{n-k+1} p_{11} + \cdots + e_{n-1} p_{k-1,1}$$

$$\vdots \qquad (13)$$

$$s_{n-k-1} = e_{n-k-1} + e_{n-k} p_{0,n-k-1} + \cdots + e_{n-1} p_{k-1,n-k-1}$$

Syndrome and Error Detection

- The syndrome digits are linear combinations of the error digits
- The syndrome digits can be used for error correction
- Because the $n - k$ linear equations of (13) do not have a unique solution but have 2^k solutions
- There are 2^k error pattern that result in the same syndrome, and the true error pattern **e** is one of them
- The decoder has to determine the true error vector from a set of 2^k candidates
- To minimize the probability of a decoding error, the most *probable* error pattern that satisfies the equations of (13) is chosen as the true error vector

Syndrome and Error Detection

- **Example 3.5**
  - We consider the (7,4) code whose parity-check matrix is given in example 3.3
  - Let **v** = (1 0 0 1 0 1 1) be the transmitted code word
  - Let **r** = (1 0 0 1 0 0 1) be the received vector
  - The receiver computes the syndrome

$$\mathbf{s} = \mathbf{r} \bullet \mathbf{H}^T = (1\ 1\ 1)$$

  - The receiver attempts to determine the true error vector **e** = (e_0, e_1, e_2, e_3, e_4, e_5, e_6), which yields the syndrome above

$$1 = e_0 + e_3 + e_5 + e_6$$

$$1 = e_1 + e_3 + e_4 + e_5$$

$$1 = e_2 + e_4 + e_5 + e_6$$

  - There are $2^4 = 16$ error patterns that satisfy the equations above

Syndrome and Error Detection

- **Example 3.5**
  - The error vector **e** = (0 0 0 0 0 1 0) has the smallest number of nonzero components
  - If the channel is a **BSC**, **e** = (0 0 0 0 0 1 0) is the most probable error vector that satisfies the equation above
  - Taking **e** = (0 0 0 0 0 1 0) as the true error vector, the receiver decodes the received vector **r** = (1 0 0 1 0 0 1) into the following code word

$$\begin{aligned}\mathbf{v}^* = \mathbf{r} + \mathbf{e} &= (1\ 0\ 0\ 1\ 0\ 0\ 1) + (0\ 0\ 0\ 0\ 0\ 1\ 0) \\ &= (1\ 0\ 0\ 1\ 0\ 1\ 1)\end{aligned}$$

  - where **v*** is the actual transmitted code word

The Minimum Distance of a Block Code

- Let **v** = (v_0, v_1, … , v_{n-1}) be a binary *n*-tuple, the *Hamming weight* (or simply weight) of **v**, denote by *w*(**v**), is defined as the number of nonzero components of **v**
  - For example, the Hamming weight of **v** = (1 0 0 0 1 1 0) is 3

Let **v** and **w** be two *n*-tuple, the *Hamming distance* between **v** and **w**, denoted *d*(**v**,**w**), is defined as the number of places where they differ

For example, the Hamming distance between **v** = (1 0 0 1 0 1 1) and **w** = (0 1 0 0 0 1 1) is 3

The Minimum Distance of a Block Code

- The Hamming distance is a metric function that satisfied the triangle inequality

$$d(\mathbf{v}, \mathbf{w}) + d(\mathbf{w}, \mathbf{x}) \geq d(\mathbf{v}, \mathbf{x}) \qquad (14)$$

  - the proof of this inequality is left as a problem

- From the definition of Hamming distance and the definition of module-2 addition that the Hamming distance between two n-tuple, **v** and **w**, is equal to the Hamming weight of the sum of **v** and **w**, that is

$$d(\mathbf{v}, \mathbf{w}) = w(\mathbf{v} + \mathbf{w}) \qquad (15)$$

  - For example, the Hamming distance between **v** = (1 0 0 1 0 1 1) and **w** = (1 1 1 0 0 1 0) is 4 and the weight of **v** + **w** = (0 1 1 1 0 0 1) is also 4

The Minimum Distance of a Block Code

- Given, a block code C, the minimum distance of C, denoted $d_{\min}$, is defined as

$$d_{\min} = \min\{d(\mathbf{v}, \mathbf{w}) : \mathbf{v}, \mathbf{w} \in C, \mathbf{v} \neq \mathbf{w}\} \qquad (16)$$

- If C is a linear block, the sum of two vectors is also a code vector
- From (3.15) that the Hamming distance between two code vectors in C is equal to the Hamming weight of a third code vector in C

$$\begin{aligned} d_{\min} &= \min\{w(\mathbf{v} + \mathbf{w}) : \mathbf{v}, \mathbf{w} \in C, \mathbf{v} \neq \mathbf{w}\} \\ &= \min\{w(\mathbf{x}) : \mathbf{x} \in C, \mathbf{x} \neq \mathbf{0}\} \qquad (17) \\ &\equiv w_{\min} \end{aligned}$$

The Minimum Distance of a Block Code

- The parameter $w_{\min} \equiv \{w(\mathbf{x}): \mathbf{x} \in C, \mathbf{x} \neq \mathbf{0}\}$ is called the *minimum weight* of the linear code C
- **Theorem 3.1** The minimum distance of a linear block code is equal to the minimum weight of its nonzero code words
- **Theorem 3.2** Let C be an (n, k) linear code with parity-check matrix $\mathbf{H}$.
  - For each code vector of Hamming weight l, there exist l columns of $\mathbf{H}$ such that the vector sum of these l columns is equal to the zero vector.
  - Conversely, if there exist l columns of $\mathbf{H}$ whose vector sum is the zeros vector, there exists a code vector of Hamming weight l in C.

Example

In a *single-parity-check code*, a single parity bit is appended to a block of k message bits $(m_1, m_2, \ldots, m_k)$. The single parity bit b_1 is chosen so that the code word satisfies the *even parity rule*:

$$m_1 + m_2 + \cdots + m_k + b_1 = 0, \qquad \text{mod } 2$$

For $k = 3$, set up the 2^k possible code words in the code defined by this rule.

Message Sequence	Single-parity-check code
0 0 0	0 0 0 0
0 0 1	0 0 1 1
0 1 0	0 1 0 1
0 1 1	0 1 1 0
1 0 0	1 0 0 1
1 0 1	1 0 1 0
1 1 0	1 1 0 0
1 1 1	1 1 1 1

Example

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

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

Show that $HG^T = 0$

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

$$= \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}, \quad \text{mod-2}$$

Example

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

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

$\mathbf{m} = (011)$ then $\mathbf{c} = \mathbf{m}G$

$\mathbf{c} = \mathbf{m}G = (011100).$

$\mathbf{r} = (011001)$. Compute $\mathbf{s}^T = H\mathbf{r}^T$

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

$H \cdot e_i,$