EE4152/IM4152
Digital Communications

A/Prof Li Kwok Hung

EEE-S1-B1b-62

Tel: 6790-5028

Ekhli@ntu.edu.sg

EEE, NTU

Major Topics

- Introduction to Channel Coding
- Hamming Bound and Perfect Codes
- Linear Block Codes and Effects of Coding
- Convolutional Codes and Code Tree
- Viterbi Decoding and Trellis Diagram

Textbook & References

- **B P Lathi and Z Ding**, *Modern Digital and Analog Systems*, 4/Ed, Oxford University Press, 2010
- **S Haykin and M Moher**, *Communication Systems*, 5/Ed, John Wiley, 2010
- **J G Proakis and M Salehi**, *Communication Systems Engineering*, 2/Ed, Prentice-Hall, 2002

Introduction to Channel Coding

Ref:

Lathi and Ding, *Modern Digital and Analog Systems*

(pp. 802 - 803)

Digital Communications

Modulation	Bit-error rate (BER)
Coherent PSK	$Q\left(\sqrt{\frac{2E_b}{N_0}}\right)$
Coherent FSK / ASK	$Q\left(\sqrt{\frac{E_b}{N_0}}\right)$
DPSK	$\frac{1}{2}\exp\left(-\frac{E_b}{N_0}\right)$
Noncoherent FSK	$\frac{1}{2}\exp\left(-\frac{E_b}{2N_0}\right)$

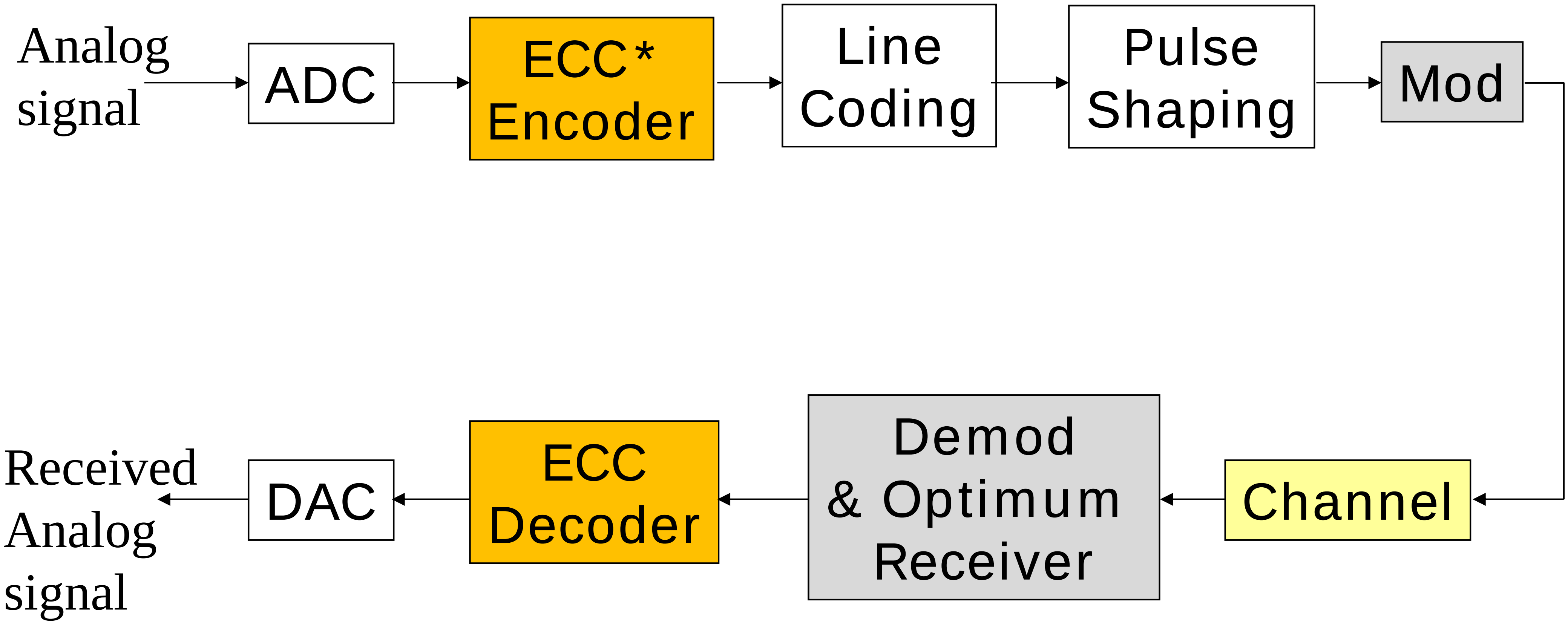
Why Channel Coding?

- The error probability P_e for a particular signaling scheme is a function of E_b/N_0.
- In practical systems, the energy per bit, E_b, is restricted to some fixed value due to hardware design or government regulations.
- The noise PSD N_0 is also fixed for a particular operating environment.
- Parameters of signaling schemes are chosen to minimize the complexity/cost.

Why Channel Coding (..)

- With all these constraints, it is often not possible to arrive at a signaling scheme that will achieve an acceptable P_e for a given application.
- Facing this problem, we can use channel coding to reduce P_e.

Digital Communication System



*Note: ECC – error-control coding

Fig. 1: Overview of a digital communication system

Channel Coding & Redundancy

- Channel coding is the calculated and careful insertion of redundancy.
- The channel encoder systematically adds redundant bits to the transmitted message digits. Note that these redundant bits do not carry any new information.
- The channel decoder uses these inserted bits to detect and/or correct errors, reducing the overall probability of error.

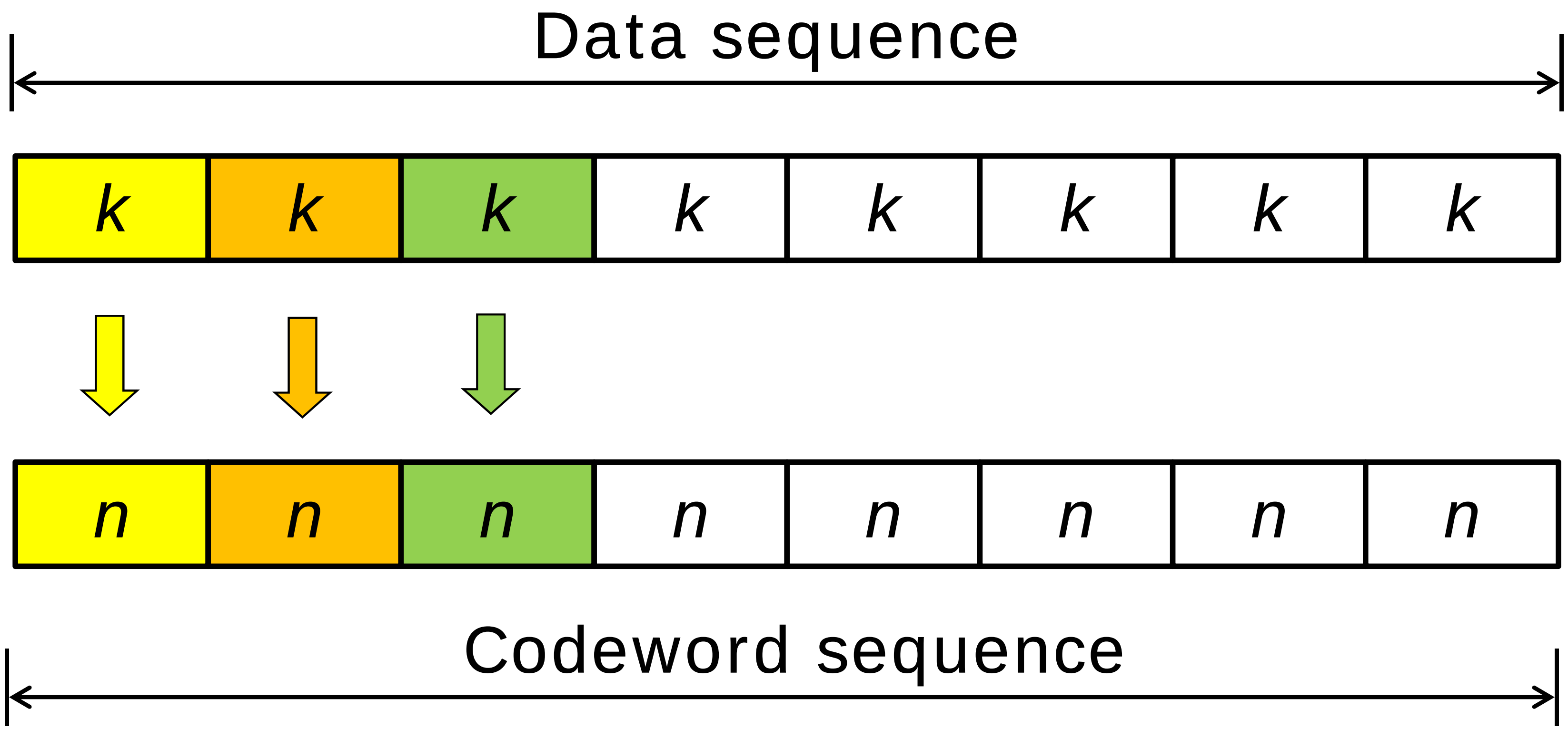
Source Coding & Channel Coding

- Source coding removes redundancy from the source output. For example, the Huffman code can achieve the shortest average code length for a given alphabet, obtaining the average codeword length close to the value of entropy.
- Channel coding inserts just enough controlled redundancy to protect the transmitted message bits such that the overall error probability is lowered.

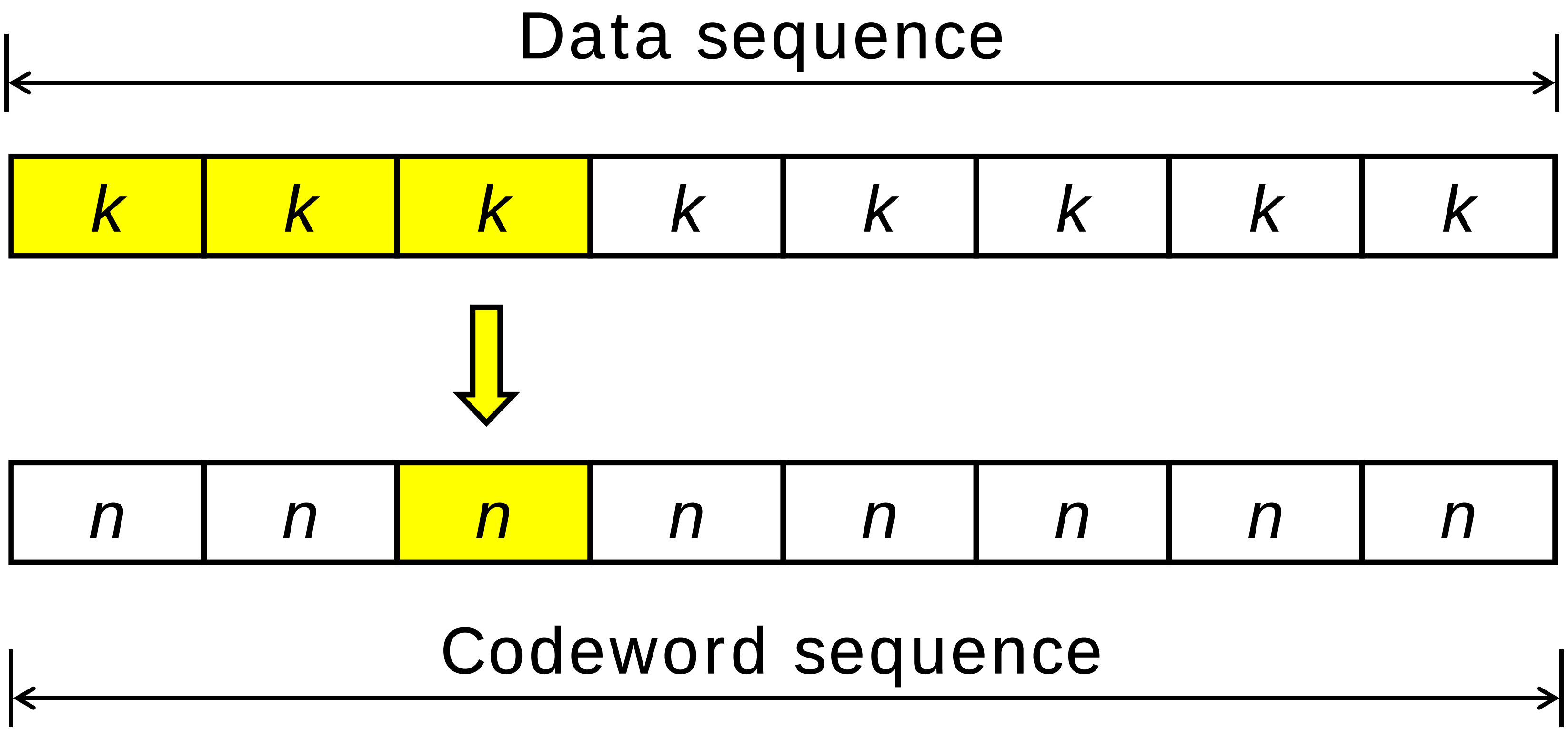
Types of Codes

- Block codes and convolutional codes
- In an (n, k) code, a block of k bits is encoded by a codeword of n bits, where $n > k$. For each block of k message bits, there is an unique n-bit codeword.
- In a convolutional code, the coded sequence of n bits depends not only on the k-bit data block but also on the previous $N - 1$ data blocks, where $N > 1$. The coding is done on a continuous basis.

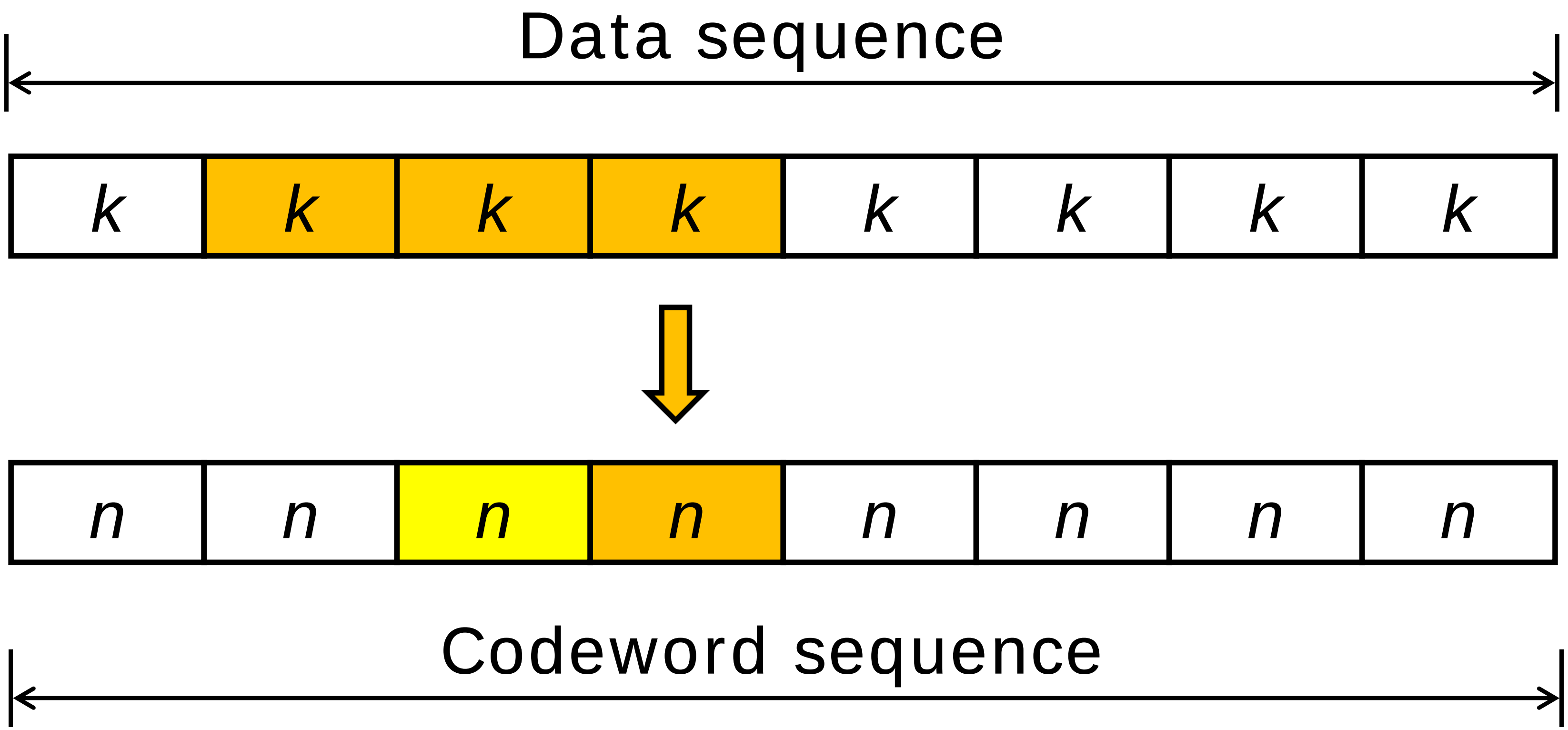
Encoding for Block Codes



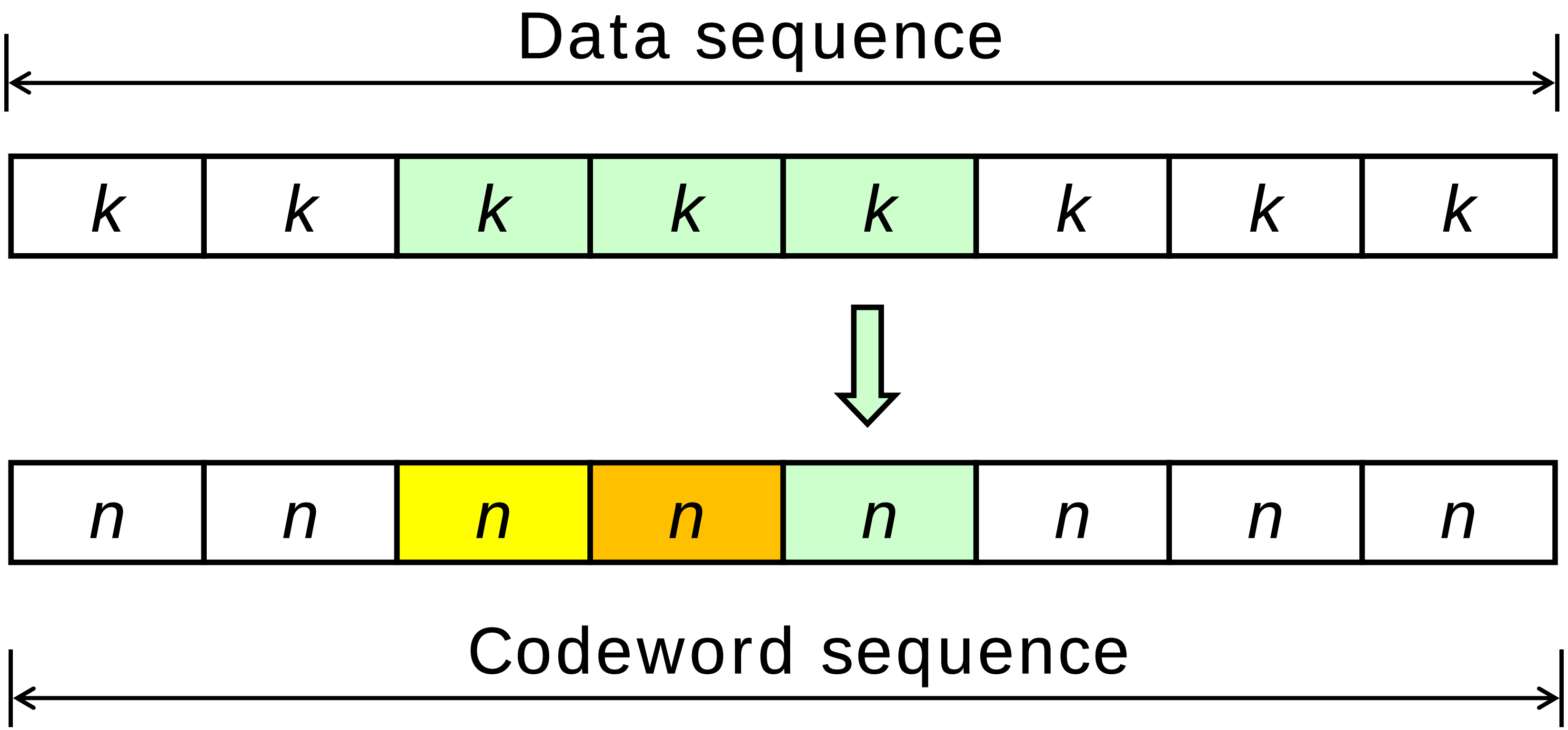
Encoding for Convolutonal Codes (1)



Encoding for Convolutonal Codes (2)



Encoding for Convolutonal Codes (3)



Check Digits & Code Rate

- If k data bits are transmitted by a codeword of n bits, the number of check digits is $m = n - k$.
- The code rate is k/n.
- For an (n, k) code, $\mathbf{d} = (d_1, d_2, \ldots, d_k)$ is a k-dim data vector (data word) and $\mathbf{c} = (c_1, c_2, \ldots, c_n)$ is an n-dim code vector (codeword).
- For a binary code, there are 2^k data words and 2^n codewords in the vector space.

Modulo-2 Addition

- The rules of modulo-2 addition are

$$0 \oplus 0 = 0$$

$$0 \oplus 1 = 1$$

$$1 \oplus 0 = 1$$

$$1 \oplus 1 = 0$$

- The modulo-2 symbol $\oplus$ is equivalent to the logical exclusive-or operator.

Hamming Weight/Distance

- The Hamming weight, $w(\mathbf{c})$, of a code vector $\mathbf{c}$ is defined to be the number of nonzero elements in $\mathbf{c}$.
- **Example:** if $\mathbf{c} = (1, 0, 0, 1, 0, 1, 1)$, then $w(\mathbf{c}) = 4$.
- The Hamming distance, $d(\mathbf{u}, \mathbf{v})$, between two code vectors $\mathbf{u}$ and $\mathbf{v}$ is defined to be the number of positions in which they are different.
- **Example:** if $\mathbf{u} = (1, 0, 1, 0, 1)$ and $\mathbf{v} = (1, 1, 1, 0, 0)$, then $d(\mathbf{u}, \mathbf{v}) = w(\mathrm{u} \oplus \mathbf{v}) = 2$.

Exercise 1

- A block code consists of 8 codewords $\{\mathbf{c}_1 = [0001011], \mathbf{c}_2 = [1110000], \mathbf{c}_3 = [1000110], \mathbf{c}_4 = [1111011], \mathbf{c}_5 = [0110110], \mathbf{c}_6 = [1001101], \mathbf{c}_7 = [0111101], \mathbf{c}_8 = [0000000]\}$. Suppose the received codeword is $\mathbf{r} = [1101011]$. What is the decoded codeword if the minimum-Hamming-distance criterion is adopted?

Ans: The decoded codeword is $\mathbf{c}_4 = [1111011]$ because $w(\mathbf{r} \oplus \mathbf{c}_4) = 1$ is the minimum.

Hamming Bound and Perfect Codes

Ref:

Lathi and Ding, *Modern Digital and Analog Systems*

(pp. 803 - 805)

Example 1

- For a vector $\mathbf{c} = (1, 0, 0, 1)$, there are $\binom{4}{1} = 4$ vectors with Hamming distance of 1 from $\mathbf{c}$. They are

$$\mathbf{c} \oplus [1000] = [0001]$$

$$\mathbf{c} \oplus [0100] = [1101]$$

$$\mathbf{c} \oplus [0010] = [1011]$$

$$\mathbf{c} \oplus [0001] = [1000]$$

- Similarly, there are $\binom{4}{2} = 6$ vectors with Hamming distance of 2 from $\mathbf{c}$.

Problem Formulation

- There are 2^n codewords in the vector space, and 2^k of them are assigned to data words. Suppose we want to find a channel code that will correct up to t errors. What is the relationship among the parameters n, k and t?
- The answer to this question is called the Hamming bound:

$$2^{n-k} \geq \sum_{j=0}^{t} \binom{n}{j} \tag{1}$$

Hamming Bound

- In the vector space, there are 2^n vertices and 2^k of them are assigned to data-mapped codewords.
- To be specific, if a data word $\mathbf{d}_j$ is mapped to a codeword $\mathbf{c}_j$, then we need to form a Hamming sphere of radius t centered at $\mathbf{c}_j$. All the vertices within the Hamming sphere are closer to $\mathbf{c}_j$ than other data-mapped codewords.
- All Hamming spheres of radius t centered at the data-mapped codewords are nonoverlapping.

Hamming Bound (..)

- Hence, the total number of vertices occupied by 2^k codewords and the associated Hamming spheres is

$$2^k + 2^k \sum_{j=1}^{t} \binom{n}{j} = 2^k \sum_{j=0}^{t} \binom{n}{j}$$

- Note that the total number of vertices is bounded by 2^n. Hence,

$$2^n \geq 2^k \sum_{j=0}^{t} \binom{n}{j} \quad \text{or} \quad 2^{n-k} \geq \sum_{j=0}^{t} \binom{n}{j} \qquad (2)$$

Hamming Bound (...)

- Note that the Hamming bound is a necessary condition but not a sufficient condition in general. That is, if some values of n, k and t satisfy (2), it does not mean that a t-error correcting code of n digits can be constructed.
- However, for single-error correcting codes, it is a necessary and sufficient condition.

Example 2

- The figure is a visualization of eight codewords with $n = 6$.
- There are 6 / 15 vertices on the surface of the inner / outer layer, representing Hamming distances of 1 and 2 from the codeword.
- Note that $\binom{6}{1} = 6$ and $\binom{6}{2} = 15$.

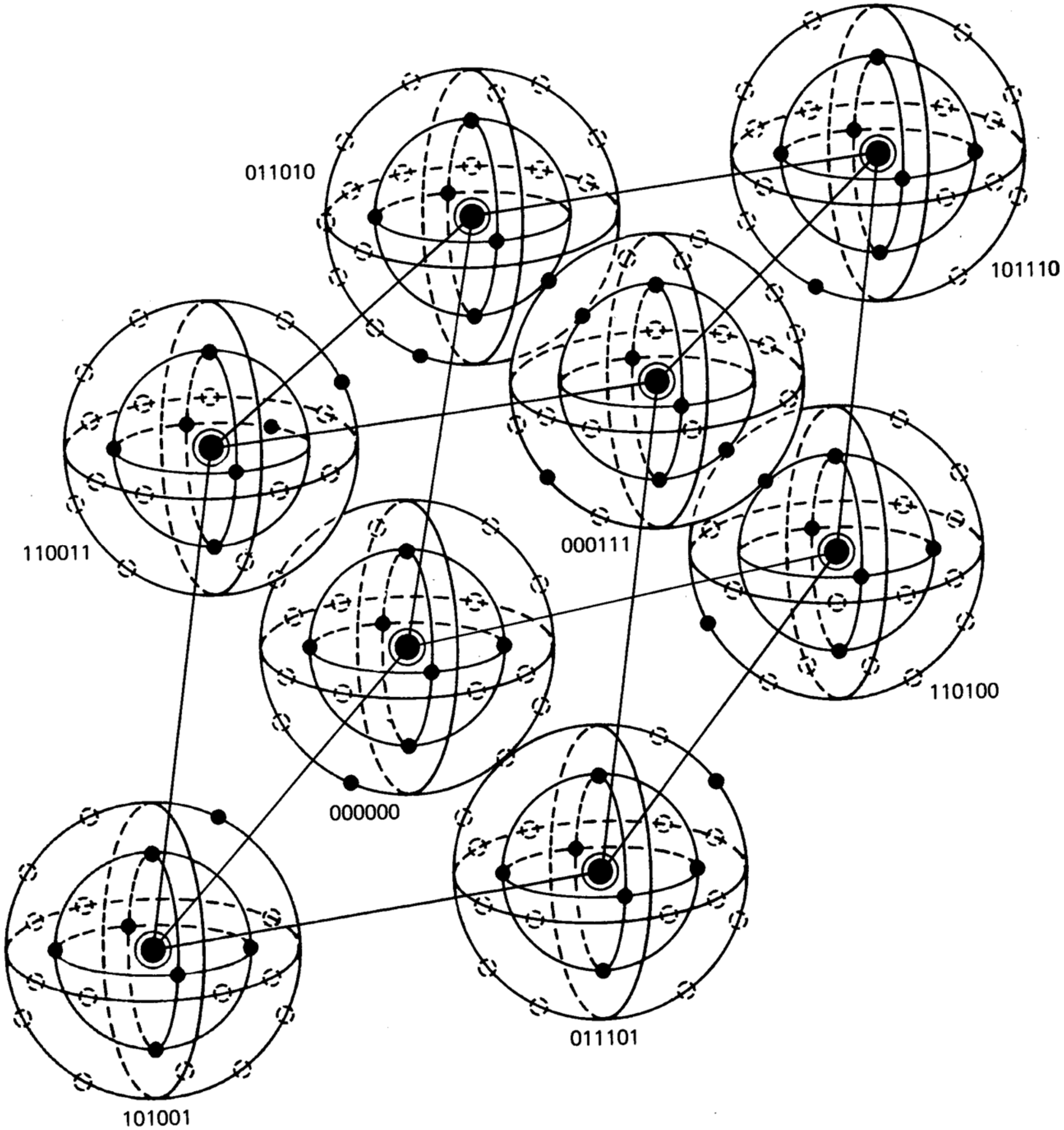


Fig. 2: Hamming bound

Perfect Codes

- If (2) is satisfied with equality, then the code is called a perfect code.
- In such a code, the Hamming spheres are nonoverlapping and exhaust all the 2^n vertices, leaving no vertex outside any sphere.
- There are not many perfect codes. Single-error correcting perfect codes are called Hamming codes.

$$2^{n-k} = \sum_{j=0}^{1} \binom{n}{j} = 1 + n \qquad (3)$$

Exercise 2

- From the table on the right-hand side, identify perfect codes and Hamming codes.

Ans: Perfect codes include (3, 1), (7, 4), (15, 11), (31, 26), (23, 12).

Hamming codes include (3, 1), (7, 4), (15, 11), (31, 26).

Some examples of error correcting codes

n	*k*	*t*	Code rate
3	1	1	1/3
4	1	1	1/4
5	2	1	2/5
6	3	1	1/2
7	4	1	4/7
15	11	1	11/15
31	26	1	26/31
10	4	2	2/5
15	8	2	8/15
10	2	3	1/5
15	5	3	1/3
23	12	3	12/23

Error Correction/Detection

- A t-error-correcting code satisfies

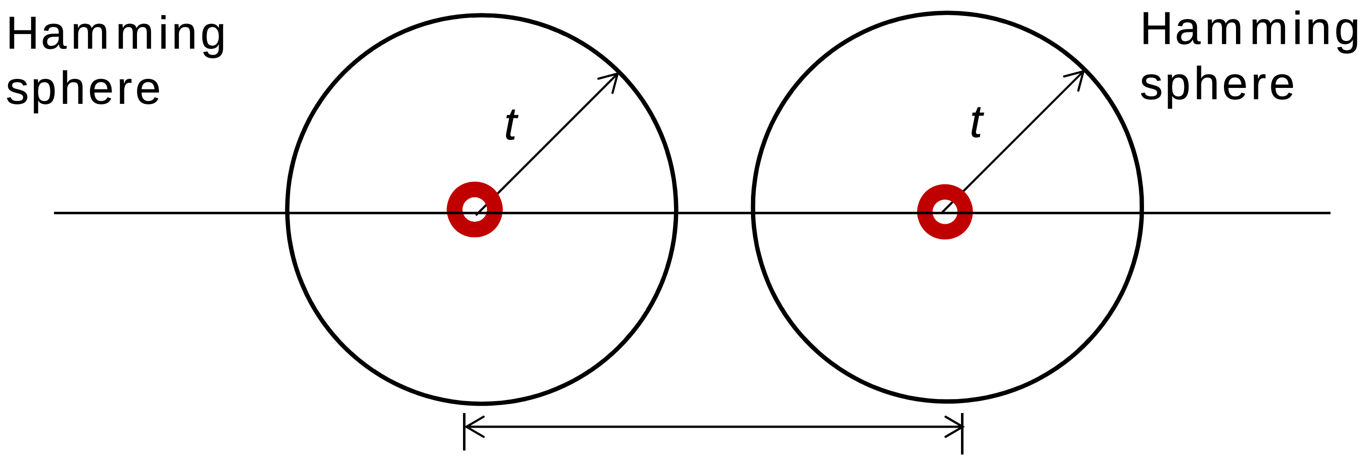
  $$d_{\min} = 2t + 1 \qquad (4)$$

  where $d_{\min}$ is the minimum distance between any two data-mapped codewords.

- Suppose we want to design a code to detect (not to correct) up to t errors. When the receiver detects an error, it requests retransmission. Because error detection requires fewer check digits, these codes operate at a higher efficiency and satisfy

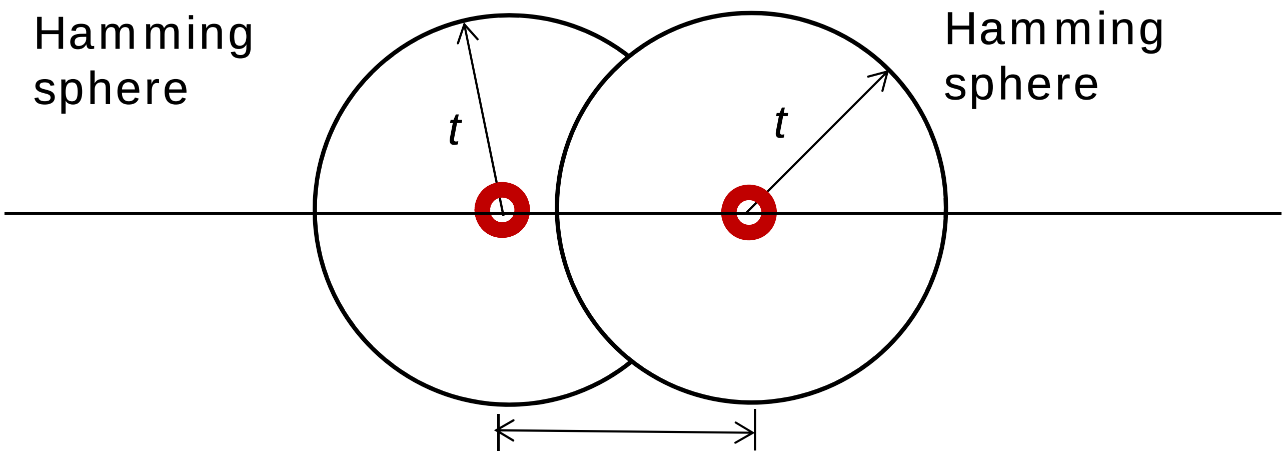
  $$d_{\min} = t + 1. \qquad (5)$$

Error Correction



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

Error Detection



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

Linear Block Codes

Ref:

Lathi and Ding, *Modern Digital and Analog Systems*

(pp. 802 - 826)

Linear Block Codes

- An (n, k) block code contains 2^k binary sequences of length n called codewords.
- A code $\mathcal{C}$ consists of 2^k binary codewords:

$$\mathcal{C} = \{\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{2^k}\} \tag{6}$$

- A block code is linear if any linear combination of two codewords is also a codeword. That is, if $\mathbf{c}_i$ and $\mathbf{c}_j$ are codewords, then $\mathbf{c}_i \oplus \mathbf{c}_j$ is also a codeword.

Linear Block Codes (..)

- A linear block code is a k-dim subspace of an n-dim space.
- The all-zero sequence $\mathbf{0}$ is a codeword because it can be written as $\mathbf{c}_i \oplus \mathbf{c}_i = \mathbf{0}$ for any codeword $\mathbf{c}_i$.

Example 3

- A $(5, 2)$ code is defined by
  $$\mathcal{C} = \{[00000], [10100], [01011], [11111]\}$$
- It is easy to verify that the code is linear. If any two codewords are added using modulo-2 addition, then we obtain another codeword in $\mathcal{C}$.
- For example,
  $$[11111] \oplus [01011] = [10100] \in \mathcal{C}$$
  $$[10100] \oplus [01011] = [11111] \in \mathcal{C}$$

Systematic Code

- Data word **d** = $[d_1, d_2, ..., d_k]$
- Codeword **c** = $[c_1, c_2, ..., c_n]$
- If $c_i = d_i$ for $i = 1, 2, \ldots, k$ and the remaining bits are linear combinations of data bits, then **c** is called a systematic code.
- In general, data bits can be placed in any positions of the codewords. For example, they may be placed at the end of the codewords.

Systematic Code (..)

- For block codes, systematic codes are preferred as data bits can be sent while encoding is being done, and the performance is the same for systematic and non-systematic codes.
- For convolutional codes, non-systematic codes have better performance than systematic codes, but are less stable.

Generator Matrix

- For the general case of linear block codes, all the n digits of $\mathbf{c}$ are the linear transformations (modulo-2 additions) of k data bits.
- Hence, $\mathbf{c} = \mathbf{dG}$, where the $k \times n$ matrix $\mathbf{G}$ is the *generator matrix*.
- For systematic codes, $\mathbf{G}$ can be partitioned into two parts as $\mathbf{G} = [\mathbf{I}_k \mid \mathbf{P}]$, where $\mathbf{I}_k$ is a $k \times k$ identity matrix and $\mathbf{P}$ is a $k \times (n - k)$ matrix.

Generator Matrix (..)

$$\mathbf{G} = \begin{bmatrix} \underbrace{\begin{matrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{matrix}}_{\mathbf{I}_k} & \underbrace{\begin{matrix} h_{11} & h_{21} & \cdots & h_{m1} \\ h_{12} & h_{22} & \cdots & h_{m2} \\ \vdots & \vdots & \ddots & \vdots \\ h_{1k} & h_{2k} & \cdots & h_{mk} \end{matrix}}_{\mathbf{P}(k\times m)} \end{bmatrix} \quad (7)$$

- The size of the coefficient matrix $\mathbf{P}$ is $k \times m$, where $m = n - k$. $\mathbf{P}$ will be used to define the parity-check matrix later.

Parity-Check Bits

- For systematic codes, the codeword can be expressed as

$$\mathbf{c} = \mathbf{d}\mathbf{G} = \mathbf{d}\begin{bmatrix} \mathbf{I}_k & \mathbf{P} \end{bmatrix}$$
$$= \begin{bmatrix} \mathbf{d} & \mathbf{d}\mathbf{P} \end{bmatrix} = \begin{bmatrix} \mathbf{d} & \mathbf{c}_p \end{bmatrix} \quad (8)$$

- The parity-check bits or check bits $\mathbf{c}_p$ are added as redundancy to protect the data word $\mathbf{d}$.

Example 4

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

$$\mathbf{G} = \begin{bmatrix} \underbrace{\begin{matrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix}}_{\mathbf{I}_3} & \underbrace{\begin{matrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{matrix}}_{\mathbf{P}} \end{bmatrix}$$

Note that

$$\mathbf{c}_p = \begin{bmatrix} d_1 & d_2 & d_3 \end{bmatrix} \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix} = \begin{bmatrix} d_1 \oplus d_3 & d_2 \oplus d_3 & d_1 \oplus d_2 \end{bmatrix}$$

Example (..)

- The minimum distance between any two codewords is at least 3.
- This code can correct up to $t = 1$ error.
- Note that the minimum distance is also equal to the minimum weight of the nonzero codewords in the code.

Code word $\mathbf{d}$	Codeword $\mathbf{c} = \mathbf{dG}$
$\mathbf{d}_1 = 000$	$\mathbf{c}_1 = 000000$
$\mathbf{d}_2 = 001$	$\mathbf{c}_2 = 001110$
$\mathbf{d}_3 = 010$	$\mathbf{c}_3 = 010011$
$\mathbf{d}_4 = 011$	$\mathbf{c}_4 = 011101$
$\mathbf{d}_5 = 100$	$\mathbf{c}_5 = 100101$
$\mathbf{d}_6 = 101$	$\mathbf{c}_6 = 101011$
$\mathbf{d}_7 = 110$	$\mathbf{c}_7 = 110110$
$\mathbf{d}_8 = 111$	$\mathbf{c}_8 = 111000$

Example (…)

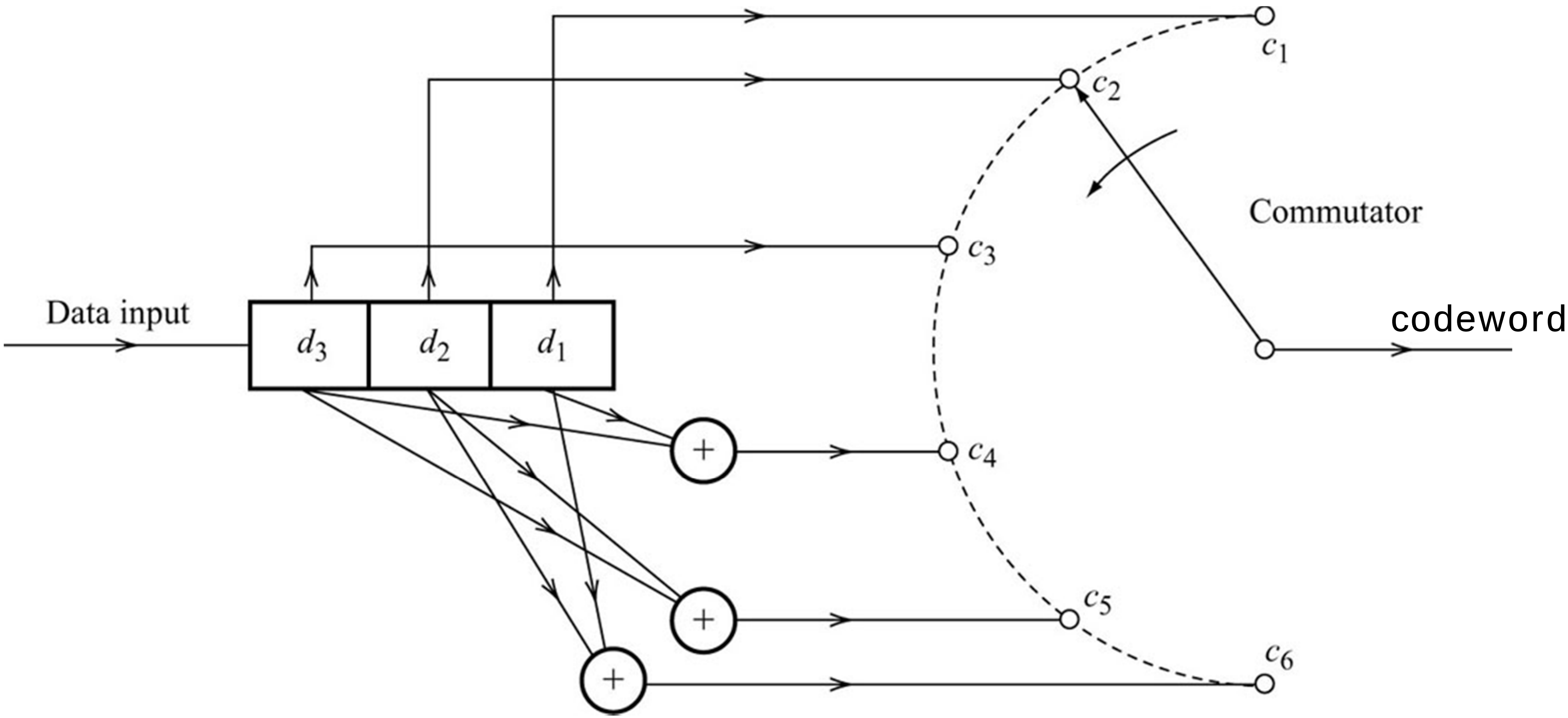


Fig. 2: Encoder for linear block code

Elements of G

- It is difficult to find a generator matrix **G** that can produce a good code, especially for large values of n and k.
- In general, we have to ensure that every row/column of **G** is unique.

Decoding of Block Codes

- From (8), we know that

$$\mathbf{dP} \oplus \mathbf{c}_p = \begin{bmatrix} \underbrace{\mathbf{d} \quad \mathbf{c}_p}_{\mathbf{c}} \end{bmatrix} \begin{bmatrix} \mathbf{P} \\ \mathbf{I}_m \end{bmatrix} = \mathbf{0}$$

Defining

$$\mathbf{H}^T = \begin{bmatrix} \mathbf{P} \\ \mathbf{I}_m \end{bmatrix} \quad \text{or} \quad \mathbf{H} = \begin{bmatrix} \mathbf{P}^T & \mathbf{I}_m \end{bmatrix} \tag{9}$$

we have

$$\mathbf{cH}^T = \mathbf{0} \tag{10}$$

where $\mathbf{H}$ is called the parity-check matrix and $\mathbf{H}^T$ is its transpose. We can use it for decoding.

Encoding and Decoding for Block Codes

$$\mathbf{d} \longrightarrow \mathbf{G} \longrightarrow \mathbf{c} \longrightarrow \mathbf{H}^{T} \longrightarrow \mathbf{0}$$

Generator matrix for encoding

Parity-check matrix for decoding

Example 5

- Repetition codes represent the simplest type of linear block codes.
- A data bit is encoded into a block of identical n coded bits.
- For a (3, 1) repetition code,

$$\mathbf{G} = \begin{bmatrix} 1 & \underbrace{1 \quad 1}_{\mathbf{P}} \end{bmatrix} \quad \text{and} \quad \mathbf{H} = \begin{bmatrix} \mathbf{P}^T & \mathbf{I}_m \end{bmatrix} = \begin{bmatrix} 1 & 10 \\ 1 & 01 \end{bmatrix}$$

Exercise 3

- Show that **G** and **H** are orthogonal. That is,

$$\mathbf{G}\mathbf{H}^T = \mathbf{0} \tag{11}$$

- Determine parity-check matrix **H** for the generator matrix **G** in **Example 4**.

Ans:

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

Syndrome

- Consider a received word $\mathbf{r}$ corrupted by channel noise

$$\mathbf{r} = \mathbf{c} \oplus \mathbf{e} \tag{12}$$

- For example,

$$\mathbf{c} = [100110] \quad \mathbf{e} = [000011] \quad \mathbf{r} = [100101]$$

- The above error word $\mathbf{e}$ causes two errors in $\mathbf{r}$. Due to the possible channel errors,

$$\mathbf{s} = \mathbf{r}\mathbf{H}^T \tag{13}$$

is in general a nonzero row vector called a syndrome.

Syndrome Decoding

- Upon receiving $\mathbf{r}$, we can compute the syndrome

$$\begin{aligned} \mathbf{s} = \mathbf{r}\mathbf{H}^T &= (\mathbf{c}_i \oplus \mathbf{e}_i)\mathbf{H}^T \\ &= \mathbf{c}_i\mathbf{H}^T \oplus \mathbf{e}_i\mathbf{H}^T = \mathbf{e}_i\mathbf{H}^T \end{aligned} \tag{14}$$

- The syndrome $\mathbf{s}$ depends only on the error word $\mathbf{e}_i$, not on the transmitted codeword $\mathbf{c}_i$.
- Unfortunately, knowledge of $\mathbf{s}$ does not allow us to solve $\mathbf{e}_i$ uniquely. It is possible that, for $j \neq i$,

$$\mathbf{r} = \mathbf{c}_i \oplus \mathbf{e}_i = \mathbf{c}_j \oplus \mathbf{e}_j$$

Example 6

- Different error words can have the same syndrome. In **Example 4**, if $\mathbf{e}_1$ = [010000] and $\mathbf{e}_2$ = [101000] , then

$$\left(\mathbf{c}\oplus\mathbf{e}_j\right)\mathbf{H}^T=\mathbf{e}_j\mathbf{H}^T=\mathbf{e}_j\begin{bmatrix}1&0&1\\0&1&1\\1&1&0\\1&0&0\\0&1&0\\0&0&1\end{bmatrix}=[011]\quad j=1,2$$

Example 7

- Referring to the (6, 3) code in **Example 4**, the received codeword $\mathbf{r} = [100011]$ can be obtained by many different combinations of $\mathbf{c}_j$ and $\mathbf{e}$:

$$\mathbf{r} = [\underbrace{101011}_{\mathbf{c}_6}] \oplus [001000]$$

$$\boxed{\mathbf{s} = [1 \quad 1 \quad 0]}$$

$$= [\underbrace{100101}_{\mathbf{c}_5}] \oplus [000110]$$

$$= [\underbrace{001110}_{\mathbf{c}_2}] \oplus [101101]$$

- Which error word should we choose with the same syndrome?

Maximum Likelihood Rule

- One reasonable criterion is the maximum likelihood (ML) rule. We decide $\mathbf{c}_i$ if

$$\Pr(\mathbf{r} \mid \mathbf{c}_i) > \Pr(\mathbf{r} \mid \mathbf{c}_j) \quad \text{all } j \neq i \tag{15}$$

- For a binary symmetric channel (BSC) with error probability less than 0.5, the ML rule is to choose that $\mathbf{c}_i$ which is closest in Hamming distance to $\mathbf{r}$. That is, the corresponding error word $\mathbf{e}$ has a minimum weight.

$$\min_i d(\mathbf{c}_i, \mathbf{r}) \Longleftrightarrow \min_i w(\mathbf{e}_i)$$

Example 8

- In **Example 6**, we have many possible error patterns. Using the ML rule, we should choose the error word with minimum weight. That is,

$$\mathbf{e}_{\min} = [001000]$$

$$\boxed{\begin{aligned} \mathbf{e}_1 &= [001000] \\ \mathbf{e}_2 &= [000110] \\ \mathbf{e}_3 &= [101101] \end{aligned}}$$

- The estimate of the codeword is thus

$$\begin{aligned} \hat{\mathbf{c}} &= \mathbf{r} \oplus \mathbf{e}_{\min} \\ &= [100011] \oplus [001000] \\ &= [101011] \end{aligned}$$

Syndrome Table for Decoding

- The syndrome $\mathbf{s} = \mathbf{r}\mathbf{H}^T$ is an m-tuple vector. Hence, it can identify 2^m error patterns.
- Corresponding to each syndrome, we should find the error pattern with the minimum weight.
- A simple way is to first identify the single-error patterns and their corresponding syndromes. Then we move on to find the double-error patterns and their syndromes. The procedure is continued until we use up all 2^m choices.

Example 9

- Referring to **Example 4**, we have $2^{n-k} = 2^3 = 8$ error patterns. Note that all-zero error pattern results in $\mathbf{s}$ = [000].
- We still have $8 - 1 = 7$ error patterns to choose. We use 6 of them to correct single-error patterns.
- This still leave one error pattern to choose. For double-error patterns, there are many choices such as [100010] and [010100].

Example 9 (..)

- The left column of the table lists all possible error patterns that the code can correct.
- The syndrome of [100000] is the first row of $\mathbf{H}^T$. The syndrome of [010000] is the second row of $\mathbf{H}^T$, and so on.

Syndrome Table

e	$\mathbf{s} = \mathbf{e}\mathbf{H}^T$
000000	000
100000	101
010000	011
001000	110
000100	100
000010	010
000001	001
100010	111

Decoding Procedure

- Upon receiving $\mathbf{r}$, compute the syndrome $\mathbf{s} = \mathbf{r}\mathbf{H}^T$.
- Find the error pattern $\mathbf{e}$ corresponding to $\mathbf{s}$ by using the syndrome table.
- The estimate of the codeword is

$$\hat{\mathbf{c}} = \mathbf{r} \oplus \mathbf{e}$$

Single-Error Correcting Codes

- It is still not clear how to choose the elements of **G** or **H**. Unfortunately, there is no systematic way to do this, except for the case of single-error correcting codes (Hamming codes).
- We shall focus on the (7, 4) Hamming code, which satisfies the Hamming bound exactly.

(7, 4) Hamming Code

- $m = 7 - 4 = 3$
- It contains $2^3 = 8$ error patterns in the syndrome table.
- One of them is the all-zero error pattern.
- It uses $8 - 1 = 7$ of them for single-error patterns.
- To be specific, $[1000000]\mathbf{H}^T$ results in the 1st row of $\mathbf{H}^T$; $[0100000]\mathbf{H}^T$ results in the 2nd row of $\mathbf{H}^T$; $[0010000]\mathbf{H}^T$ results in the 3rd row of $\mathbf{H}^T$; …

(7, 4) Hamming Code (..)

- Hence, all 7 rows of $\mathbf{H}^T$ must be distinct and nonzero.
- Because there exist 7 nonzero patterns of 3 digits, it is possible to arrange them as the 7 rows of $\mathbf{H}^T$.
- There are many ways in which these rows can be ordered.

Example 10

- One possible form is

$$\mathbf{H}^T = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \\ \hline 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{G} = \left[\begin{array}{cccc:ccc} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 1 & 1 \end{array}\right]$$

Example 10 (..)

- Another possible form is listed on the RHS. If the first bit of **c** is wrong, the syndrome will be [001]. If the second bit of **c** is wrong, the syndrome will be [010]. … If the last bit of **c** is wrong, the syndrome will be [111].

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

Cyclic Codes

- Cyclic Codes are a subclass of linear block codes.
- As seen before, a procedure for selecting a generator matrix is relatively easy for single-error correcting codes. This procedure cannot be extended to multiple-error correcting codes.
- Cyclic codes have a fair amount of mathematical structure that permits the design of multiple-error correcting codes. Moreover, encoding and decoding can be implemented through simple shift registers and modulo-2 adders.

Effects of Channel Coding

- We will compare the relative performance of coded and uncoded systems for block codes, with the assumption that the information rate and power are the same.
- For an (n, k) block code, the bit rate will be higher for the coded system. The energy per bit is also smaller under the fixed power assumption, resulting in a higher probability of bit error.
- We must justify the use of coding can overcome the increase in bit error and obtain a significant improvement.

Some Symbols for Comparison

- q_u, q_c – probability of channel bit error for uncoded and coded systems
- P_{ew} – probability of codeword error for the coded system
- P_{eu}, P_{ec} – probability of bit error for uncoded and coded systems
- t – the (n, k) block code can correct up to t errors in every n-bit block

Probability of Codeword Error

$$P_{ew} = \sum_{j=t+1}^{n} \binom{n}{j} (1-q_c)^{n-j} q_c^j \qquad (15)$$

where

$$\binom{n}{j} = \frac{n!}{j!(n-j)!} \qquad (16)$$

is the number of combinations of n items taken j at a time. Note that j bit errors occur in a block of n bits.

BER for Coded Systems

$$P_{ec} = \sum_{j=t+1}^{n} \frac{j}{n} \binom{n}{j} (1-q_c)^{n-j} q_c^j \qquad (17)$$

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

The first term dominates all other terms.

$$\cong \binom{n-1}{t} (1-q_c)^{n-t-1} q_c^{t+1}$$

$$\cong \binom{n-1}{t} q_c^{t+1} \qquad \text{for } q_c \ll 1 \qquad (18)$$

BER for Polar Signaling

$$q_u = Q\left(\sqrt{\frac{2E_p}{N_0}}\right), \qquad q_c = Q\left(\sqrt{\frac{2kE_p}{nN_0}}\right) \tag{19}$$

Hence,

$$P_{eu} = q_u = Q\left(\sqrt{\frac{2E_p}{N_0}}\right), \tag{20}$$

$$P_{ec} = \binom{n-1}{t}\left[Q\left(\sqrt{\frac{2kE_p}{nN_0}}\right)\right]^{t+1} \tag{21}$$

BER using (7, 4) Block Code

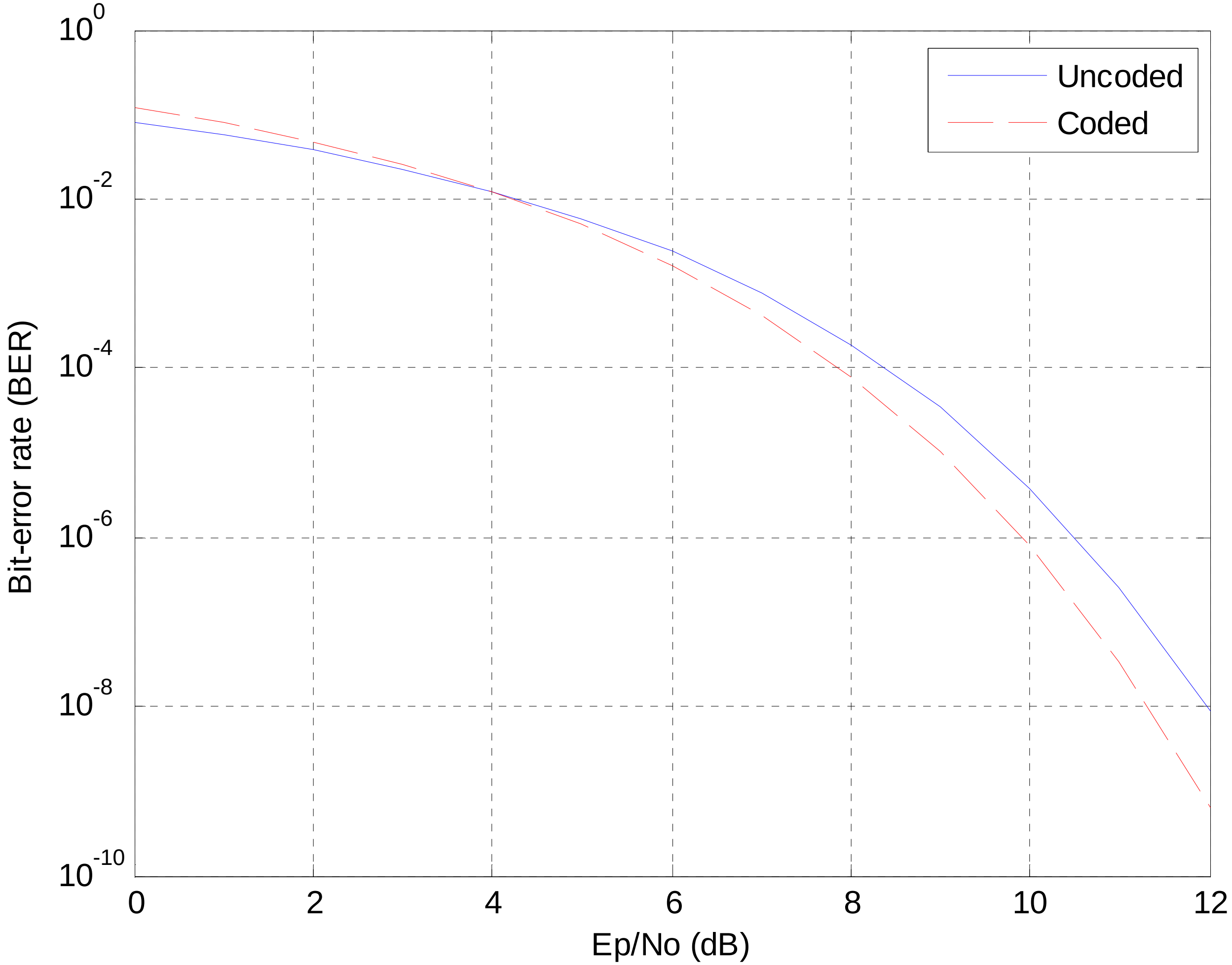


Fig. 3: BER comparison with (7, 4) Hamming code

BER using (15,11) Block Code

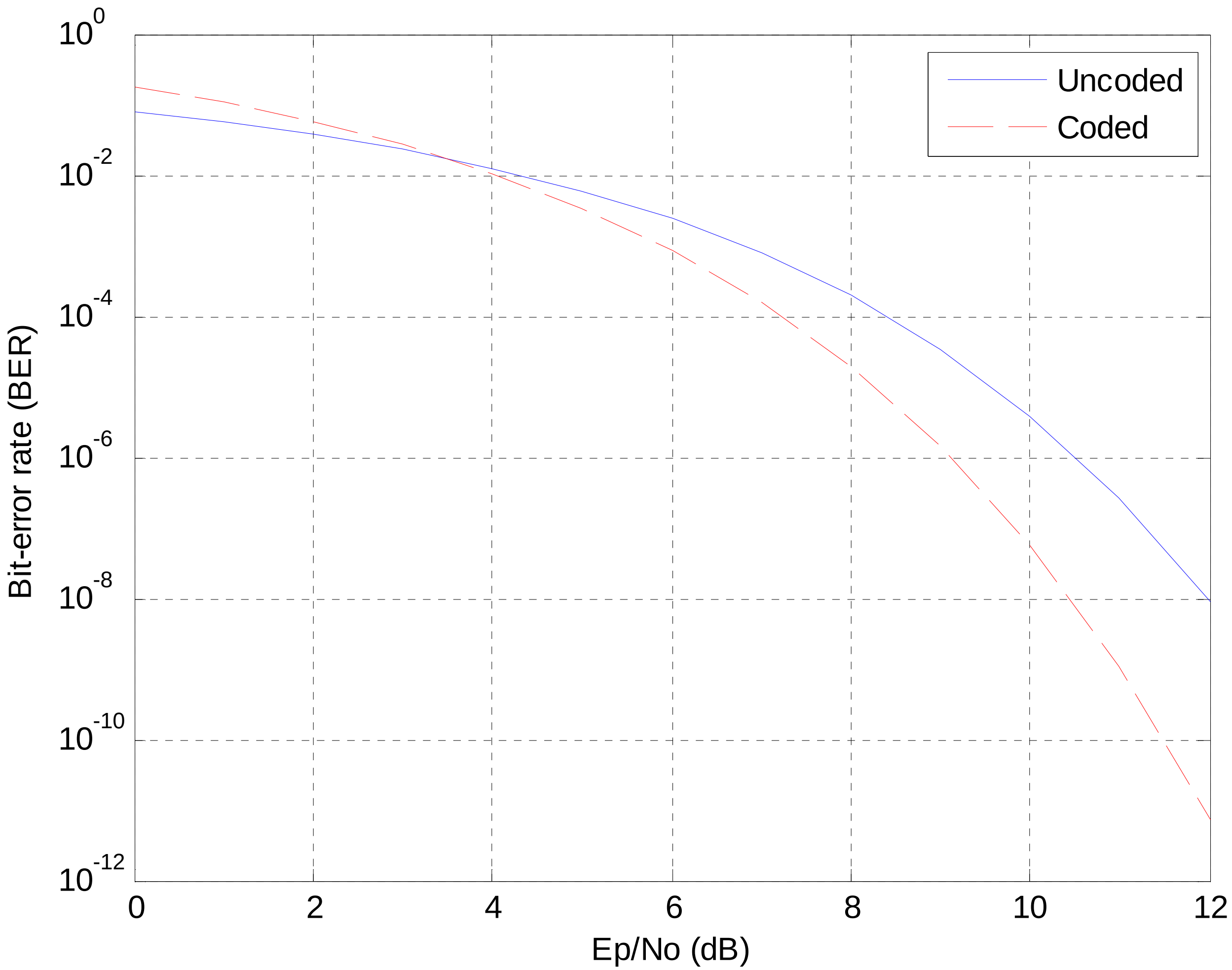


Fig. 4: BER comparison with (15, 11) Hamming code

Observations

- The (15, 11) block code is more powerful than the (7, 4) block code.
- Coding is useful in the region of $E_p/N_o > 4$ dB.
- The improvement is quite modest unless E_p/N_o is very high. Note that we are using single-error correcting codes. Their error-correcting capability is limited.

Convolutional Codes and Code Tree

Ref:

Lathi and Ding, *Modern Digital and Analog Systems*

(pp. 827 - 838)

Convolutional Codes (CCs)

- For block codes, the block of n coded bits generated by the encoder depends only on the corresponding block of k input data bits.
- For CCs, the block of n coded bits generated by the encoder depends not only on the corresponding block of k message bits but also on previous $N - 1$ blocks. The encoding is done in a continuous manner.
- For CCs, k, n and N are usually small.

Parameters of Conv Codes

- CCs can be described by 3 integer parameters.
- At each time instant k data bits are shifted into the register, generating n coded output bits
- N is called the constraint length, which describes the memory of the encoder.

Convolutional Encoder

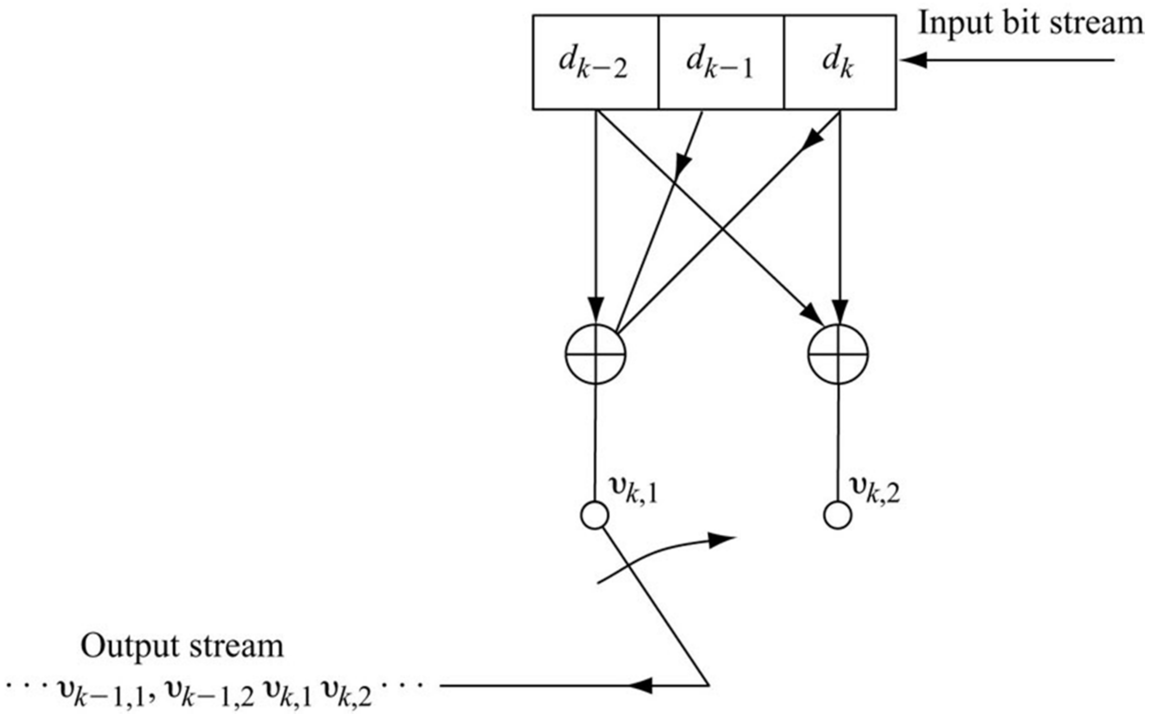


Fig. 5: Convolutional encoder with $(n, k, N) = (2, 1, 3)$

Operation of CC Encoder

$$v_{k,1} = d_{k-2} \oplus d_{k-1} \oplus d_k$$
$$v_{k,2} = d_{k-2} \oplus d_k$$

- Refer to the encoder in Fig. 5 as an example
- All the contents of the register are initially zero
- The input digits are assumed to be 11010
- We add $N - 1 = 2$ zeros to the input stream to make sure that all input digits pass through the shift register (see shifts 6 & 7).

Shift	d_{k-2}	d_{k-1}	d_k	$v_{k,1}$	$v_{k,2}$
0	0	0	0	--	--
1	0	0	1	1	1
2	0	1	1	0	1
3	1	1	0	0	1
4	1	0	1	0	0
5	0	1	0	1	0
6	1	0	0	1	1
7	0	0	0	0	0

Observations

- We have added $N - 1$ zeros, called augmented data, to the data stream.
- We actually apply the input stream 1101000.
- The encoder output is 11010100101100.
- It can be seen that unlike the block encoder, the CC encoder operates on a continuous basis, and each data digit influences N groups in the output.

Code Rate of CCs

- For L data bits (with $k = 1$), there are $nL + n(N - 1)$ coded bits.
- The code rate is thus

$$r = \frac{L}{n(L+N-1)} \cong \frac{1}{n} \quad \text{for } L >> N \tag{22}$$

- CCs have no particular block size.

Representation of CCs

- Structure properties of CCs can be represented in graphic forms through 3 equivalent diagrams, namely, code tree, state transition diagram and trellis diagram.
- They all show a repetitive structure due to the memory feature of CCs.
- These diagrams facilitate the encoding and decoding processes of CCs.

Code Tree

- Refer to the CC encoder in Fig. 5.
- Each branch of the code tree represents an input data bit. (input: 11010)
- At each node, the upper/lower branch represents 0/1
- Corresponding output bits are indicated on the branch.

$$v_{k,1} = d_{k-2} \oplus d_{k-1} \oplus d_k$$

$$v_{k,2} = d_{k-2} \oplus d_k$$

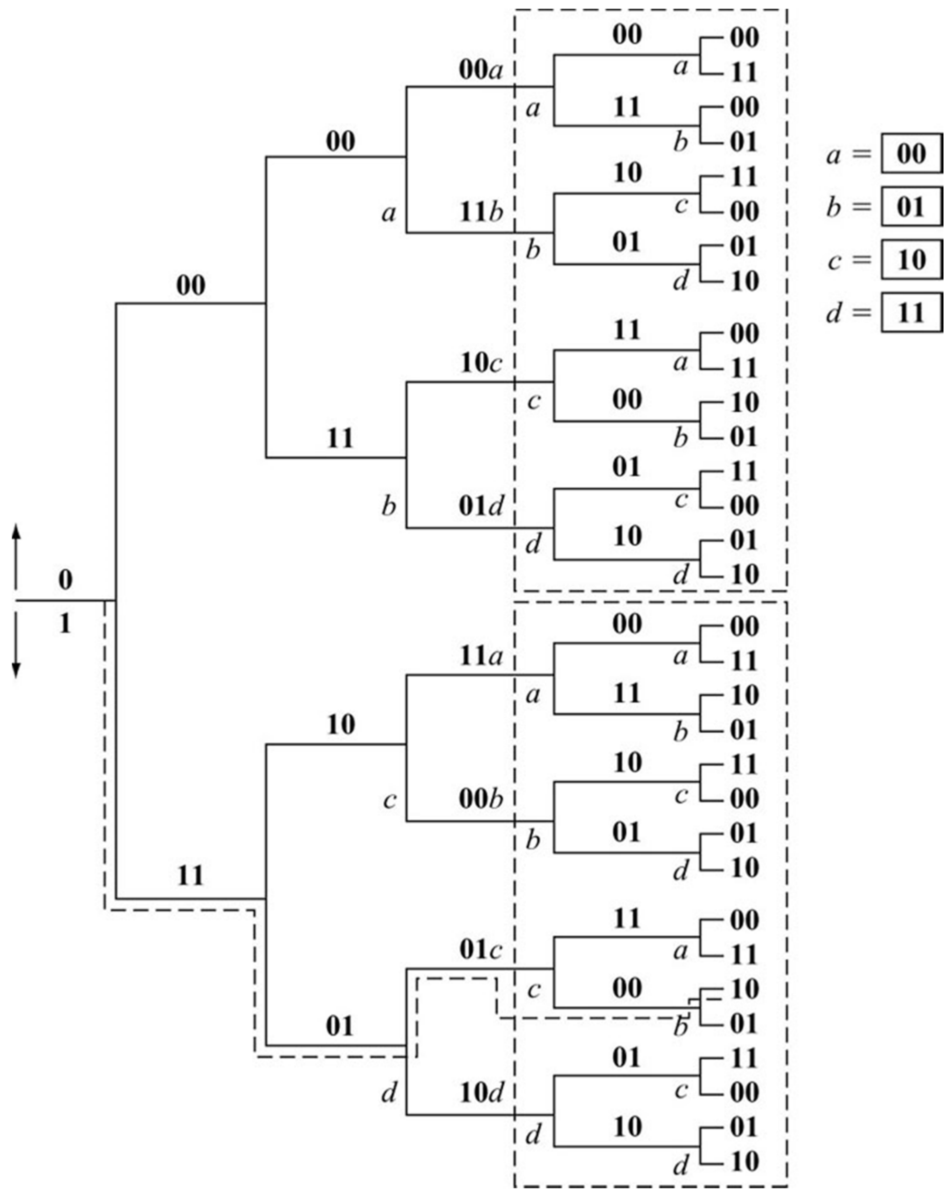


Fig. 6: Code tree for encoding

Code Tree (..)

- The code tree becomes repetitive after $N = 3$ branches because of the memory of the encoder.
- The states of (d_{k-2}, d_{k-1}) are indicated by a, b, c and d.
- If we actually apply the input stream 1101000, then the encoder output is 11 01 01 00 10 11 00.
- The encoding process is complicated for long input data sequences.

State Transition Diagram

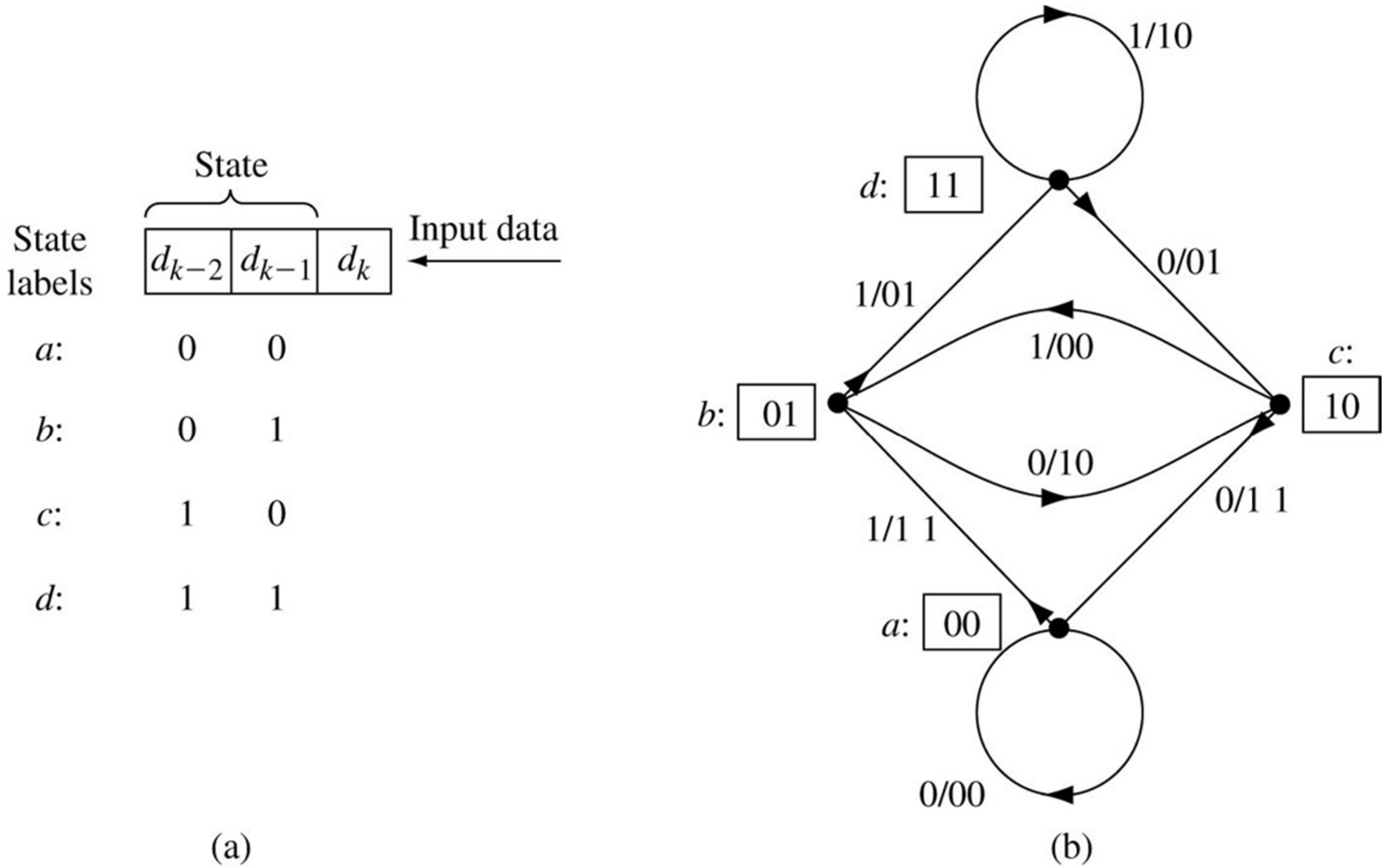


Fig. 7: (a) Encoder states and (b) State transition diagram

State Transition Diagram (..)

- When the previous two data bits are 01 ($d_{k\text{-}2} = 0$, $d_{k\text{-}1} = 1$), the state of the encoder is *b*, and so on.
- The number of states is equal to $2^{N-1} = 4$.
- When the encoder is in state *a*, and we input 1, the encoder output is 11 (labeled with 1/11). The encoder now goes to state *b* for the next data bit ($d_{k\text{-}2} = 0$, $d_{k\text{-}1} = 1$).
- When the encoder is in state *a*, we have 0/00. The encoder remains in state *a* as ($d_{k\text{-}2} = 0$, $d_{k\text{-}1} = 0$).

State Transition Diagram (...)

- Note that the encoder cannot switch from state *a* to state *c* or *d* directly. From a given state, the encoder is only allowed to switch to two states directly by entering a single data bit.
- Hence, the state transition path is not arbitrary. Only certain transition paths are allowed. The prohibited paths can be exploited for the decoding process

Trellis Diagram

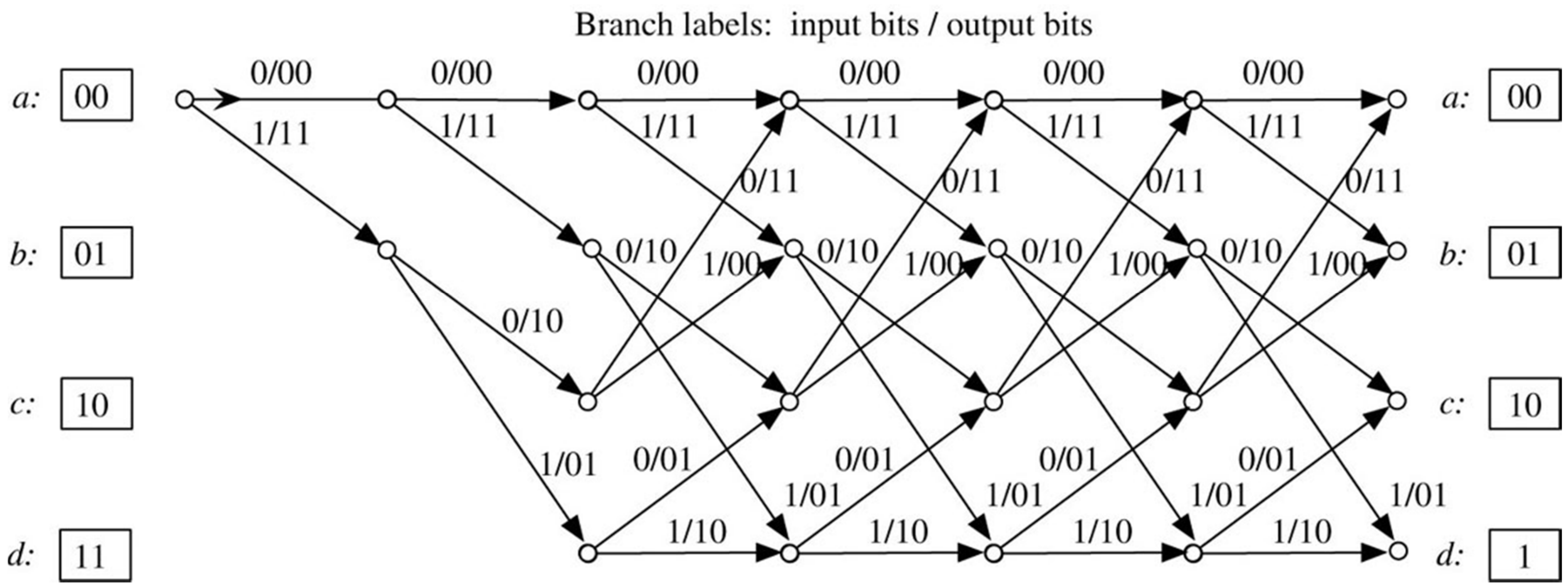


Fig. 8: Trellis diagram for convolutional encoding

Trellis Diagram (..)

- The diagram starts from scratch (all-zero state or state a) and makes transition corresponding to each input data bit.
- For example, when the first input bit is 0, the encoder output is 00. The trellis branch is labeled as 0/00. Another branch is labeled as 1/11.
- The structure of the trellis diagram is completely repetitive after $N = 3$ branches.

Encoding using Trellis diagram

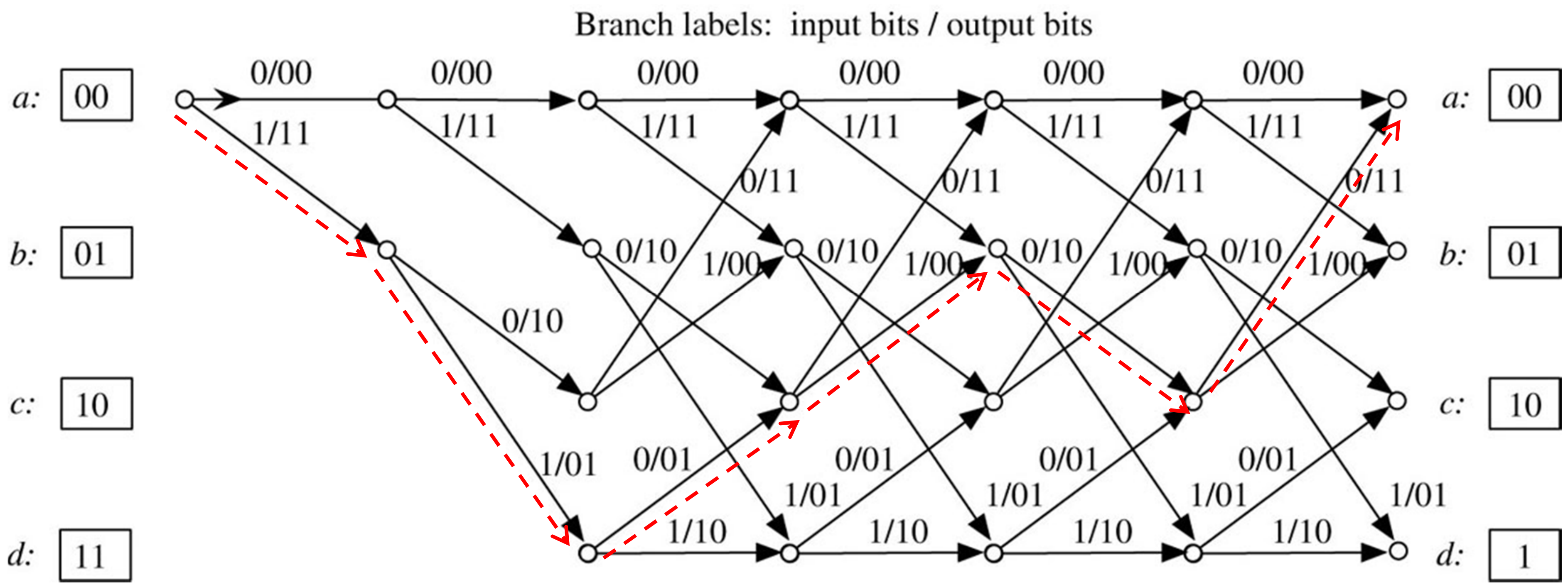


Fig. 9: Input 110100 and output 11 01 01 00 10 11

Decoding: The Viterbi Algorithm

- Unlike block codes, we cannot use syndrome decoding for decoding CCs.
- In AWGN channels, the maximum-likelihood (ML) decoding requires the selection of a codeword closed to the received path.
- The Viterbi algorithm (VA) simplifies the ML decoding process by eliminating the unlikely paths and keeping the surviving paths at each node.

Example 11

- We now study a decoding example of the VA for ML decoding of CCs.
- We again use the CC encoder in Fig. 5.
- The first 12 received digits are **01 10 11 00 00 00**.
- After two stages, there exists one path to each state (see Fig. 10a). Their Hamming distances from the received path are 2, 2, 1, and 3, respectively.

Example 11: VA (a)

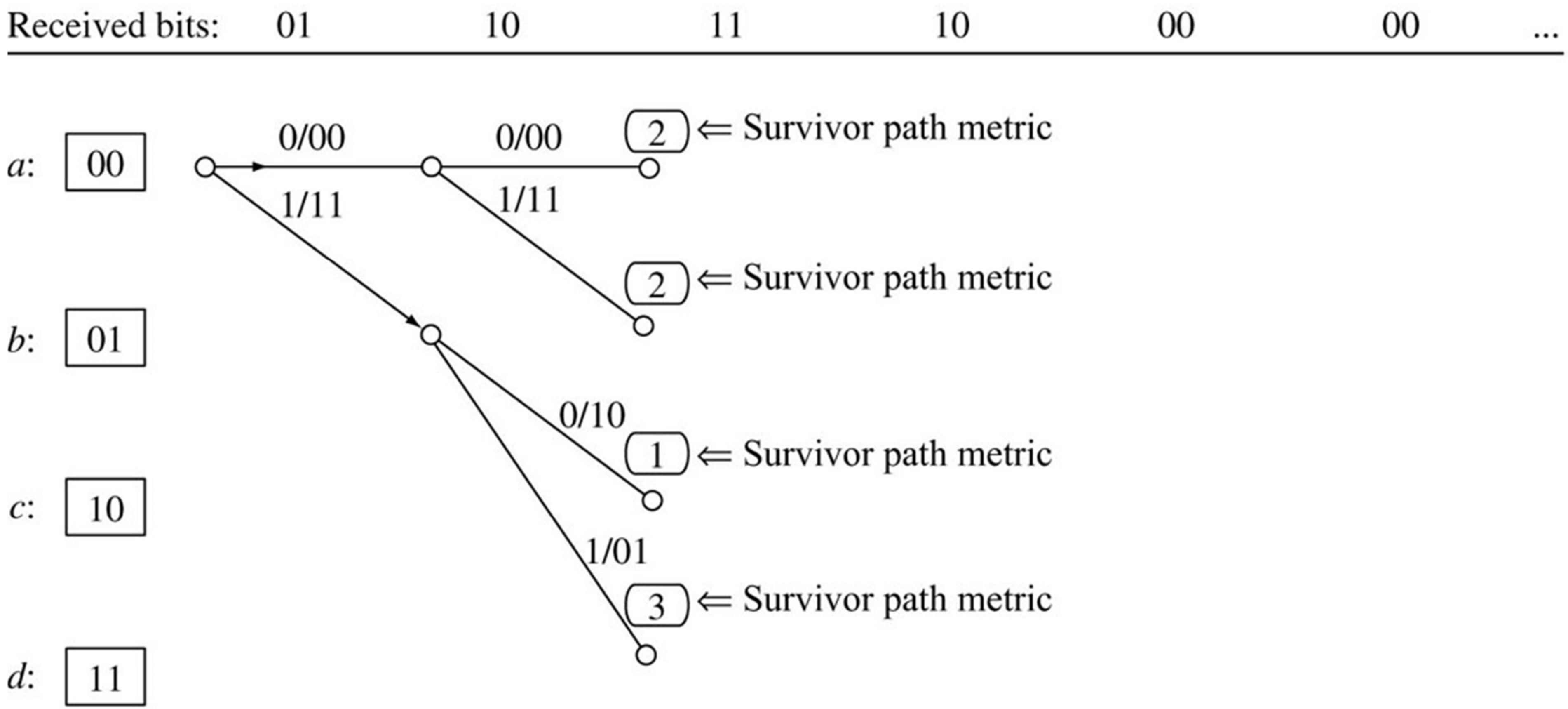


Figure 10 (a)

Example 11 (..)

- At stage 3, there are two paths leading to state *a*; one from *a* to *a* and the other from *c* to *a*. Their Hamming distances are $2 + 2 = 4$ and $1 + 0 = 1$. Hence, the former is discarded and the latter is kept.
- Repeat the same step for states *b*, *c* and *d*. The surviving path is kept for each state.

Example 11: VA (b)

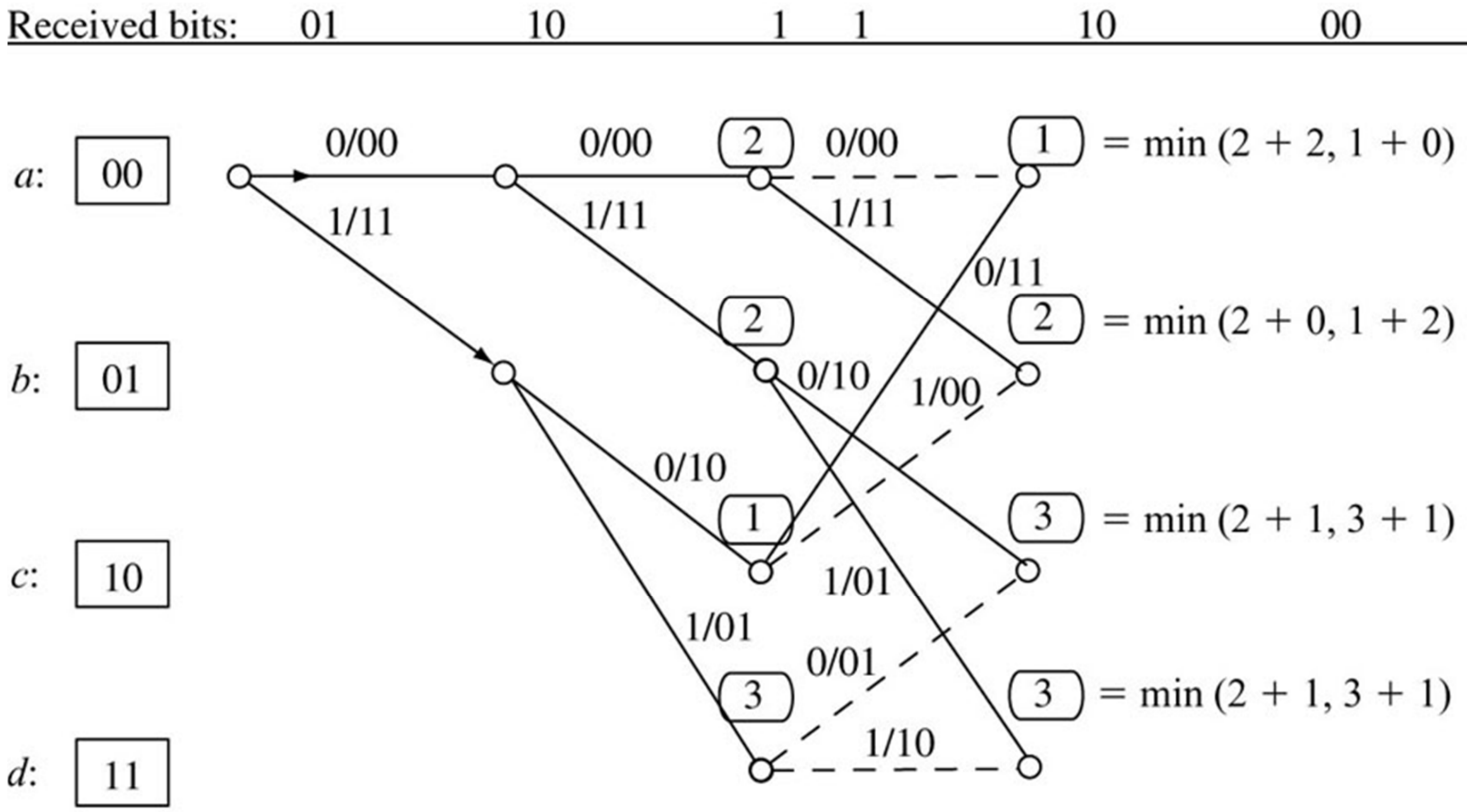


Figure 10 (b)

Example 11: VA (c)

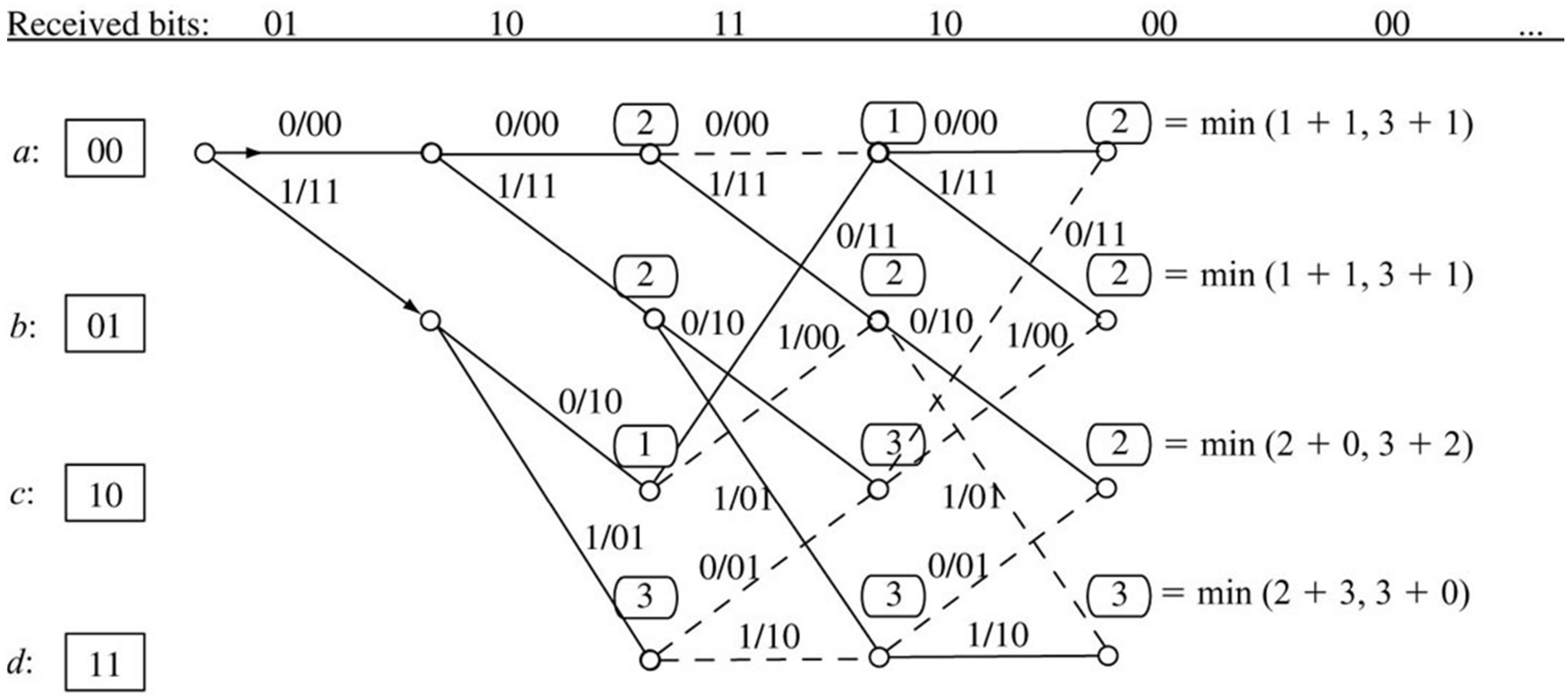


Figure 10 (c)

Example 11: VA (d)

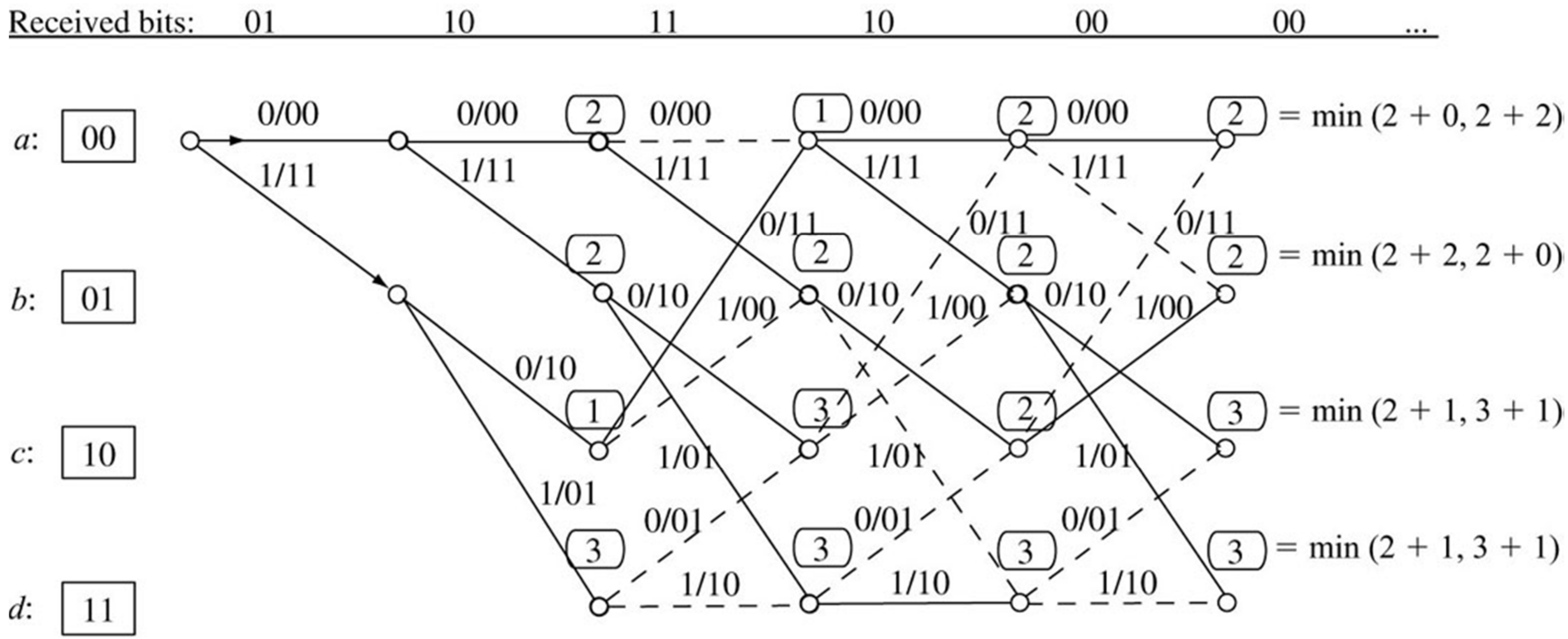


Figure 10 (d)

Example 10 (...)

- The final optimum path after stage 6 is identified as the shaded solid path with minimum distance of 2 ending in state *a*.
- The decoded codeword is 11 10 11 00 00 00, and the corresponding information bits are 100000.
- The earlier stages do not exhibit a merged path. We make an ML decision based on the metrics (i.e., Hamming distances accumulated at states *a*, *b*, *c* and *d*) up to that stage.

Example 11: VA (e)

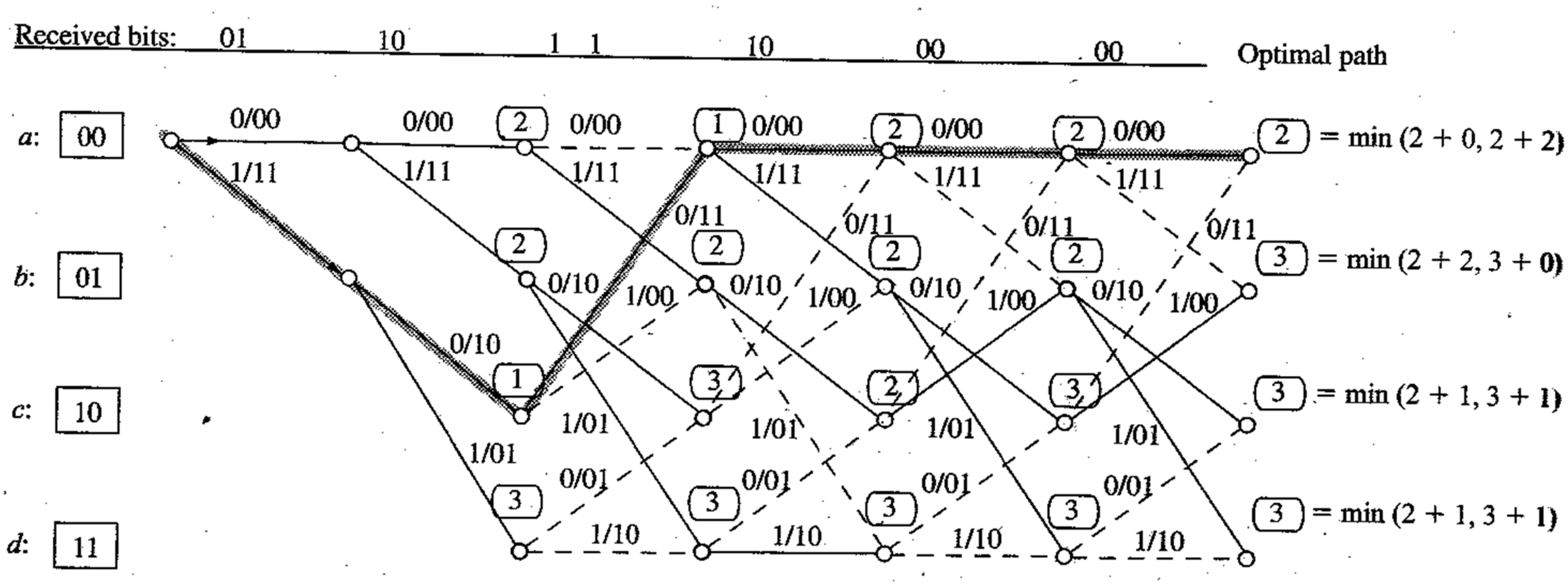


Figure 10 (e)

Truncation

- Sometimes it is possible to see that all four contending paths have a common tail for their earlier stages. This means that the earlier-stage branches are the most reliable outputs.
- Without a common tail, we may do the truncation at some stage, which is designed to a force a decision on one path among all the survivors without leading to a long decoding delay.

Other Decoding Methods

- The storage and computational complexity of the VA are proportional to 2^{N-1} and are very attractive for constraint length $N < 10$.
- To achieve very low error probabilities, longer constraint lengths are required.
- We may use *sequential decoding* whose complexity increases linearly with N.
- We can replace hard-decision decoding with *soft-decision decoding* to improve performance.