Chapter 10
Information and Forward Error Correction

Wireless Information Transmission System Lab.
Institute of Communications Engineering
National Sun Yat-sen University

Outline

- Introduction
- Uncertainty, Information, and Entropy
- Source-Coding Theorem
- Lossless Data Compression
- Linear Block Code
- Convolutional Code

Chapter 10.1
Introduction

Wireless Information Transmission System Lab.
Institute of Communications Engineering
National Sun Yat-sen University

Chapter 10.1 Introduction

- In the context of communications, information theory deals with mathematical modeling and analysis of a communication system rather than with physical sources and physical channels.

- In particular, it provides answers to two fundamental questions:
- Q: What is the irreducible complexity below which a signal cannot be compressed? A: Entropy.

- Q: What is the ultimate transmission rate for reliable communication over a noisy channel? A: Capacity.

Chapter 10.2
Uncertainty, Information, and Entropy

Wireless Information Transmission System Lab.
Institute of Communications Engineering
National Sun Yat-sen University

- Suppose a source output is modeled as a discrete random variable, S, which takes on symbols from a fixed finite alphabet:

$$\mathcal{S} = \{s_0, s_1, \ldots, s_{K-1}\} \tag{10.1}$$

with probabilities

$$P(S = s_k) = p_k, \quad k = 0,1,\ldots,K-1 \tag{10.2}$$

- Of course, this set of probabilities must satisfy the condition:

$$\sum_{k=0}^{K-1} p_k = 1 \tag{10.3}$$

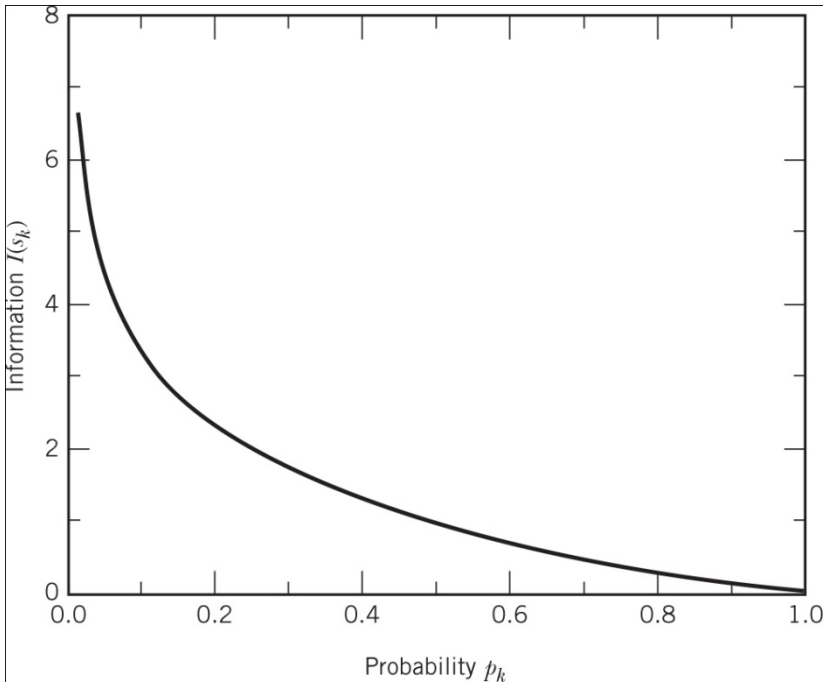
- A source having the properties that the symbols emitted by the source during successive signaling intervals are statistically independent is called a *discrete memoryless source*.

Uncertainty

- Can we find a measure of how much information is produced by such a source?
- The idea of information is closely related to that of uncertainty.
- Consider the event $S = s_k$, by Eq. (10.2), if the probability $p_k = 1$ and $p_i = 0$ for all $i \neq k$, then there is no information when symbol s_k is emitted (Because we know what the message from the source must be).
- On the other hand, if $P(s_k) \rightarrow p_k < p_i \leftarrow P(s_i)$, $i \neq k$ then there is more surprise and therefore information when s_k is emitted.
- The amount of information is related to the inverse of the probability of occurrence.

- We define the amount of information gained after observing the event $S = s_k$, which occurs with probability p_k, as the *logarithmic* function:

$$I(s_k) = \log\left(\frac{1}{p_k}\right) \tag{10.4}$$



1. $$I(s_k) = 0 \quad \text{for } p_k = 1 \tag{10.5}$$

Obviously, if we are absolutely *certain* of the outcome of an event, even before it occurs, there is *no* information gained.

2. $$I(s_k) \geq 0 \quad \text{for } 0 \leq p_k \leq 1 \tag{10.6}$$

The occurrence of an event $S = s_k$ either provides some or no information, but never brings about a *loss* of information.

3. $$I(s_k) > I(s_i) \quad \text{for } p_k < p_i \tag{10.7}$$

That is, the less probable an event is, the more information we gain when it occurs.

4. $I(s_k\, s_l) = I(s_k) + I(s_l)$ if s_k and s_l are statistically independent.

- It is the standard practice today to use a logarithm to base 2.
- The resulting unit of information is called the *bit* (*bi*nary digi*t*):

$$I(s_k) = \log_2\left(\frac{1}{p_k}\right) = -\log_2 p_k \quad \text{for } k = 0,1,\ldots,K-1 \qquad (10.8)$$

  - When $p_k = 1/2$, we have $I(s_k) = 1$ bit.
- Indeed, $I(s_k)$ is a discrete random variable that takes on the values $I(s_0)$, $I(s_1)$, …, $I(s_{K-1})$ with probabilities p_0, p_1, …, p_{K-1} respectively. The mean of $I(s_k)$ over the source alphabet $\mathcal{S}$ is given by:

$$H(\mathcal{S}) = \mathbf{E}\left[I(s_k)\right] = \sum_{k=0}^{K-1} p_k I(s_k) = \sum_{k=0}^{K-1} p_k \log_2\left(\frac{1}{p_k}\right) \qquad (10.9)$$

- The $H(\mathcal{S})$ is called the *entropy* of a discrete memoryless source with source alphabet $\mathcal{S}$. It is a measure of the *average information content per source symbol*.

Some Properties of Entropy

- Consider a discrete memoryless source whose mathematical model is defined by Eqs. (10.1) and (10.2). The entropy $H(\mathcal{S})$ of such a source is bounded as follows:

$$0 \le H(\mathcal{S}) \le \log_2 K \qquad (10.10)$$

where K is the number of symbols of the source alphabet $\mathcal{S}$.

- Furthermore, we may make two statements:
  1. $H(\mathcal{S}) = 0$, if and only if the probability $p_k = 1$ for some k, and the remaining probabilities in the set are all zero; this lower bound on entropy corresponds to *no uncertainty*.
  2. $H(\mathcal{S}) = \log_2 K$, if and only if the probability $p_k = 1/K$ for all k (i.e., all the symbols in the alphabet $\mathcal{S}$ are *equiprobable*); this upper bound on entropy corresponds to *maximum uncertainty*.

Example 10.1

- To illustrate the properties of $H(\mathcal{9})$, we consider a binary source for which symbol 0 occurs with probability p_0 and symbol 1 with probability $p_1 = 1 - p_0$. (We assume that the source is memoryless so that successive symbols emitted by the source are statistically independent.)

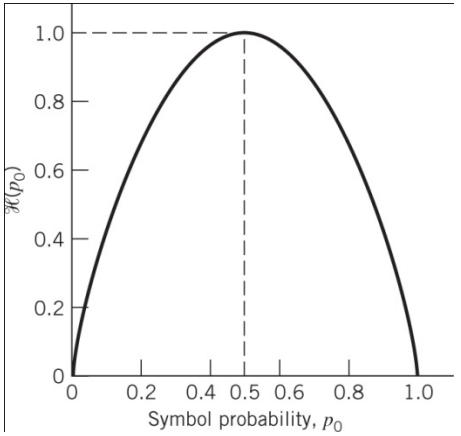
- The entropy of such a source equals

$$\begin{aligned} H(\mathcal{9}) &= -p_0 \log_2 p_0 - p_1 \log_2 p_1 \\ &= -p_0 \log_2 p_0 - (1-p_0)\log_2 (1-p_0) \text{ bits} \end{aligned} \tag{10.11}$$

- We note that :
  1. When $p_0 = 0$, the entropy $H(\mathcal{S}) = 0$; this follows the fact that $x \log x \to 0$ as $x \to 0$.
  2. When $p_0 = 1$, the entropy $H(\mathcal{S}) = 0$.
  3. The entropy $H(\mathcal{S})$ attains its maximum value, $H_{max} = 1$ bit, when $p_1 = p_0 = ½$, that is, symbols 1 and 0 are equally probable.
- Then we define the *entropy function* $\mathcal{H}(p_0)$:

$$\mathcal{H}(p_0) = -p_0 \log_2 p_0 - (1 - p_0) \log_2 (1 - p_0) \qquad (10.12)$$

- The $\mathcal{H}(p_0)$ of Eq. (10.12) is a function of the prior probability p_0 defined on the interval [0,1].



Chapter 10.3
Source-Coding Theorem

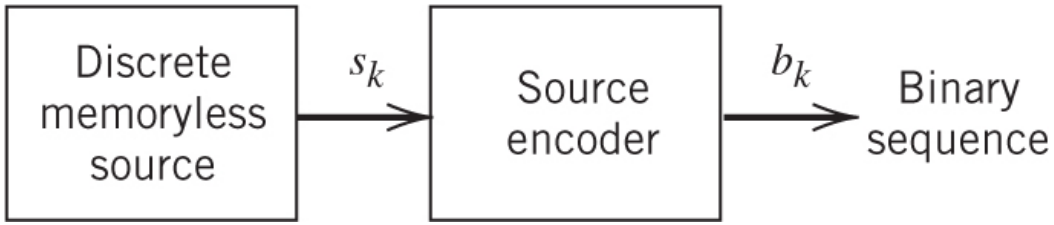
Wireless Information Transmission System Lab.
Institute of Communications Engineering
National Sun Yat-sen University

Chapter 10.3 Source-Coding Theorem

- An important problem in communications is the *efficient representation* of data generated by a discrete source.
- The process by which this representation is accomplished is called *source encoding*. The device that performs the representation is called a *source encoder*.

- Our primary interest is in the development of an efficient source encoder that satisfies two functional requirements:
  1. The codewords produced by the encoder are in *binary* form.
  2. The source code is *uniquely decodable*, so that the original source sequence can be reconstructed perfectly from the encoded binary sequence.

Chapter 10.3 Source-Coding Theorem



- We assume that the source has an alphabet with K different symbols, and that kth symbol s_k occurs with probability p_k, $k = 0, 1, \ldots, K - 1$. Let the binary codeword assigned to symbol s_k by the encoder have length l_k, measured in bits. We define the average codeword length, $\overline{L}$, of the source encoder as

$$\overline{L} = \sum_{k=0}^{K-1} p_k l_k \qquad (10.14)$$

- $\overline{L}$ represents the *average number of bits per source symbol* used in the source encoding process.

Chapter 10.3 Source-Coding Theorem

- Let $L_{\min}$ denote the *minimum* possible value of $\overline{L}$. We then define the *coding efficiency* of the source encoder as

$$\eta = \frac{L_{\min}}{\overline{L}} \tag{10.15}$$

- With $\overline{L} \geqq L_{\min}$, we clearly have $\eta \leqq 1$. The source encoder is said to be *efficient* when approaches unity.

- Q: How is the minimum value $L_{\min}$ determined?
- A: The answer is embodied in Shannon's first theorem: the source-coding theorem.

Chapter 10.3 Source-Coding Theorem

Shannon First Theorem—The source-coding theorem

Given a discrete memoryless source of entropy $H(\mathcal{S})$, the average codeword length $\overline{L}$ for any distortionless source encoding is bounded as

$$\overline{L} \geq H(\mathcal{S}) \tag{10.16}$$

- Accordingly, the entropy $H(\mathcal{S})$ represents a *fundamental limit* on the average number of bits per source symbol necessary to represent a discrete memoryless source in that it can be made as small as, but no smaller than the entropy $H(\mathcal{S})$. Thus with $L_{\min} = H(\mathcal{S})$, we may rewrite the efficiency of a source encoder in terms of the entropy $H(\mathcal{S})$ as

$$\eta = \frac{H(\mathcal{S})}{\overline{L}} \tag{10.17}$$

Chapter 10.4
Lossless Data Compression

Wireless Information Transmission System Lab.
Institute of Communications Engineering
National Sun Yat-sen University

Chapter 10.4 Lossless Data Compression

- A common characteristic of signals generated by physical source is that, in their natural form, they contain a significant amount of information that is *redundant*, the transmission of which is therefore wasteful of primary communication resources.

- For *efficient* signal transmission, the *redundant information should be removed from the signal prior to transmission*. This operation is ordinary performed on a signal in digital form, in which case we refer to it as *lossless data compression*.

- In this section, we discuss some source-coding schemes for data compression.

Prefix Coding

- A *prefix code* is defined as a code in which no codeword is the prefix of any other codeword.

TABLE 10.2 Illustrating the definition of a prefix code

Source Symbol	Probability of Occurrence	Code I	Code II	Code III
s_0	0.5	0	0	0
s_1	0.25	1	10	01
s_2	0.125	00	110	011
s_3	0.125	11	111	0111

Example in Code II :
1011111000…
↓
$s_1 s_3 s_2 s_0 s_0 \ldots$

- A prefix code has the important property that it is always *uniquely decodable*.
- Prefix codes are *instantaneously decodable*.

Huffman Coding

- The basic idea behind Huffman coding is to assign to each symbol of an alphabet a sequence of bits roughly equal in length to the amount of information conveyed by the symbol in question.

- The essence of the algorithm used to synthesize the Huffman code is to replace the prescribed set of source statistics of a discrete memoryless source with a simpler one.

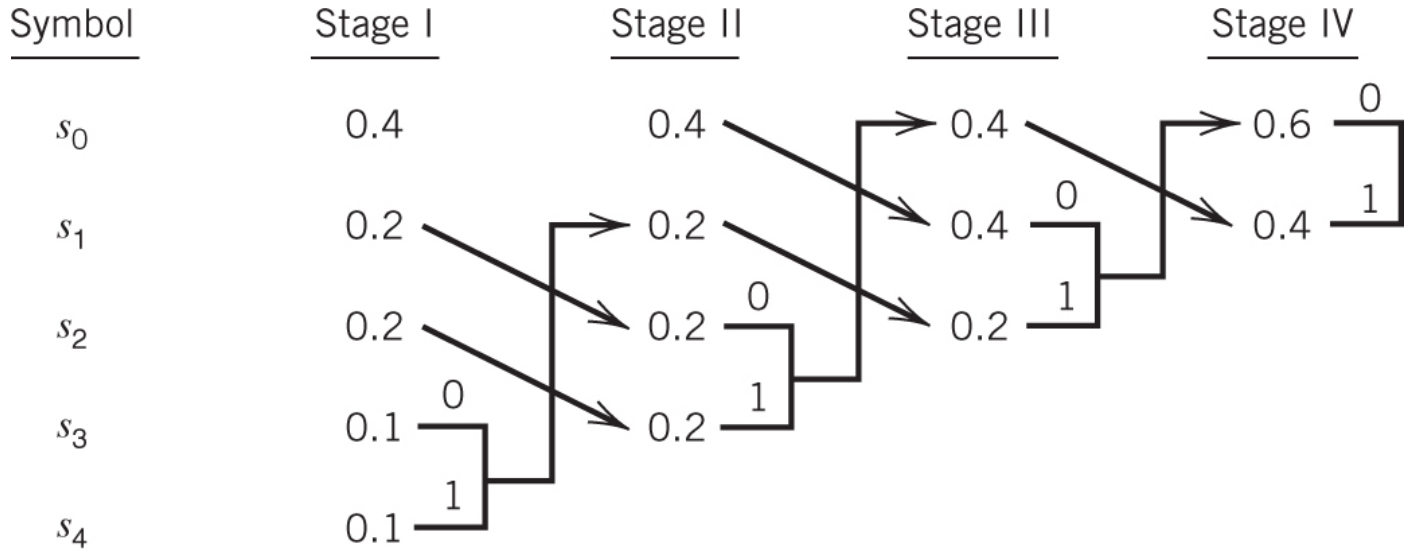
- This reduction process is continued in a step-by-step manner until we are left with a final set of only two source statistics (symbols), for which (0,1) is an optimal code.

Chapter 10.4 Lossless Data Compression

- Specifically, the Huffman encoding algorithm proceeds as follows:
  1. The source symbols are listed in order of decreasing probability. The two source symbols of lowest probability are assigned a 0 and a 1. This part of the step is referred to as a *splitting* stage.
  2. These two source symbols are regarded as being *combined* into a new source symbol with probability equal to the sum of the two original probabilities. (The list of source symbols, and therefore source statistics, is thereby *reduced* in size by one.) The probability of the new symbol is placed in the list in accordance with its value.
  3. The procedure is repeated until we are left with a final list of source statistics (symbols) of only two, for which a 0 and a 1 are assigned.

- The code for each (original) source symbol is found by working backward and tracing the sequence of 0s and 1s assigned to that symbol as well as its successors.

Chapter 10.4 Lossless Data Compression



(*a*)

Symbol	Probability	Codeword
s_0	0.4	00
s_1	0.2	10
s_2	0.2	11
s_3	0.1	010
s_4	0.1	011

(*b*)

Example 10.3

- According to figure (a) and (b), the average codeword length is therefore :

$$\begin{aligned}\bar{L} &= 0.4(2)+0.2(2)+0.2(2)+0.1(3)+0.1(3)\\ &= 2.2\end{aligned}$$

- The entropy of the specified discrete memoryless source is calculated as follows [see Eq. (10.9)]:

$$\begin{aligned}H(L) &= 0.4\log_2\left(\frac{1}{0.4}\right)+0.2\log_2\left(\frac{1}{0.2}\right)+0.2\log_2\left(\frac{1}{0.2}\right)\\ &\quad +0.1\log_2\left(\frac{1}{0.1}\right)+0.1\log_2\left(\frac{1}{0.1}\right)\\ &= 0.52877+0.46439+0.46439+0.33219+0.33219\\ &= 2.12193\end{aligned}$$

Chapter 10.4 Lossless Data Compression

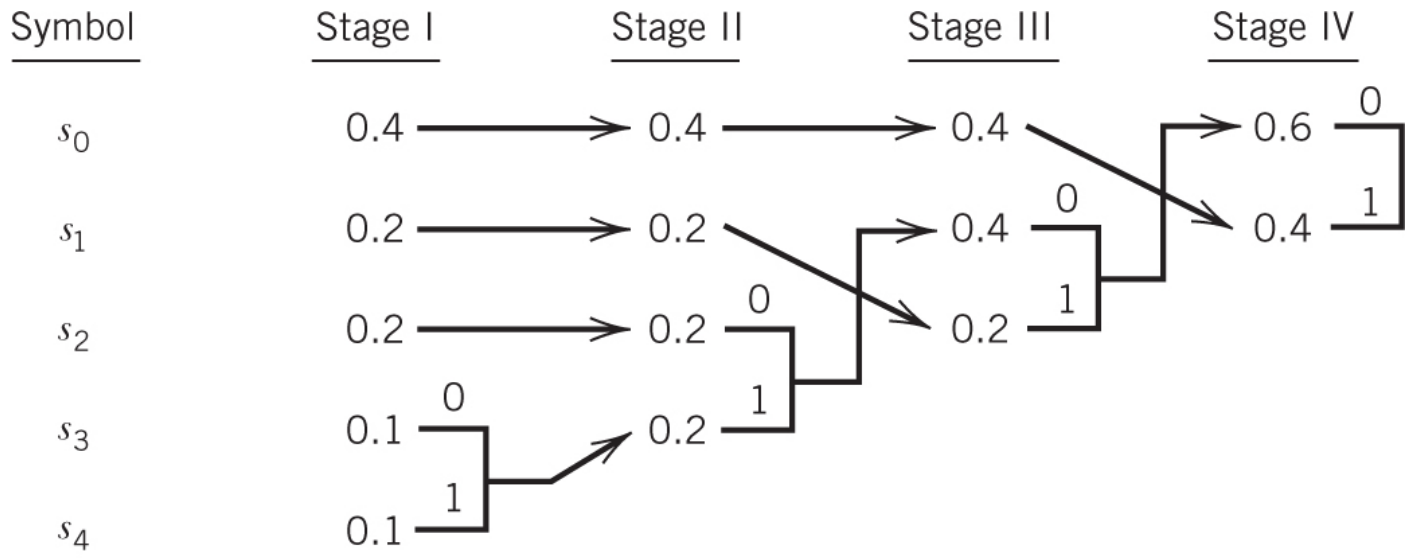
- As a measure of the variability in codeword lengths of a source code, we define the *variance* of the average codeword length $\bar{L}$ over the ensemble of source symbols as

$$\sigma^2 = \sum_{k=0}^{K-1} p_k \left(l_k - \bar{L}\right)^2 \qquad (10.23)$$

where $p_0, p_1, \ldots, p_{K-1}$ are the source statistics, and l_k is the length of the codeword assigned to source symbol s_k.

- It is usually found that when a combined symbol is moved *as high as possible*, the resulting Huffman code has a significantly smaller variance σ^2 than when it is moved as low as possible. On this basis, it is reasonable to choose the former Huffman code over the latter.

Chapter 10.4 Lossless Data Compression



(*a*)

Symbol	Probability	Codeword
s_0	0.4	1
s_1	0.2	01
s_2	0.2	000
s_3	0.1	0010
s_4	0.1	0011

(*b*)

Example 10.4

- Consider again the same discrete memoryless source in Example 10.3. This time, however, we move the probability of a combined symbol as low as possible.

- The average codeword length for the second Huffman code is therefore

$$\begin{aligned}\overline{L} &= 0.4(1)+0.2(2)+0.2(3)+0.1(4)+0.1(4) \\ &= 2.2\end{aligned}$$

which is exactly the same as that for the first Huffman code for Example 10.3.

- However, to use of Eq. (10.23) yields the variance of the first Huffman code obtained in Example 10.3 as

$$\sigma_1^2 = 0.4(2-2.2)^2 + 0.2(2-2.2)^2 + 0.2(2-2.2)^2$$
$$+0.1(3-2.2)^2 + 0.1(3-2.2)^2 = 0.16$$

- On the other hand, for the second Huffman code obtained in this example, we have from Eq. (10.23):

$$\sigma_2^2 = 0.4(1-2.2)^2 + 0.2(2-2.2)^2 + 0.2(3-2.2)^2$$
$$+0.1(4-2.2)^2 + 0.1(4-2.2)^2 = 1.36$$

- This results confirm that the minimum variance Huffman code is obtained by moving the probability of a combined symbol as high as possible.

Linear Block Code

Wireless Information Transmission System Lab.
Institute of Communications Engineering
National Sun Yat-sen University

Chapter 10.11 Linear Block Code

- Consider then an (n, k) linear block code
  - k : bits of message sequence.
  - n : code bits.
  - n-k bits are referred to as *parity check bits* of the code.

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

- In fact, a binary block code is linear iff the module-2 sum of two code word is also a code word
  - **0** must be code word.

Chapter 10.11 Linear Block Code

- Generator Matrix
  - 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, $\mathbf{g}_0$, $\mathbf{g}_1$,…, $\mathbf{g}_{k-1}$ in C

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

  where $u_i = 0$ or 1 for $0 \le i < k$.
  - Let us arrange these k linearly independent code words as the rows of a $k \times n$ matrix as follows:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = \begin{bmatrix} g_{00} & g_{01} & \cdots & g_{0,n-1} \\ g_{10} & g_{11} & \cdots & g_{1,n-1} \\ \vdots & \vdots & \ddots & \vdots \\ g_{k-1,0} & g_{k-1,1} & \cdots & g_{k-1,n-1} \end{bmatrix}$$

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

Chapter 10.11 Linear Block Code

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

$$\mathbf{v} = \mathbf{u} \cdot \mathbf{G} = (u_0, u_1, \ldots, u_{k-1}) \cdot \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = u_0 \mathbf{g}_0 + u_1 \mathbf{g}_1 + \cdots + u_{k-1} \mathbf{g}_{k-1}$$

- 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, i.e. generator matrix is not unique.

Chapter 10.11 Linear Block Code

- A desirable property for a linear block code is the *systematic structure* of the code words as shown in Fig. 3.1
  - 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*

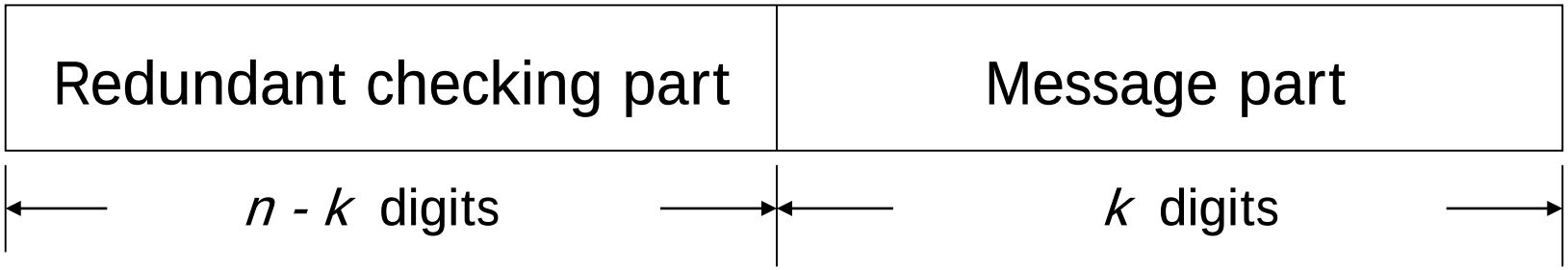


Fig. 3.1 Systematic format of a code word

Chapter 10.11 Linear Block Code

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

P matrix $\longleftarrow$ $\longrightarrow$ $k \times k$ identity matrix

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \mathbf{g}_2 \\ . \\ . \\ . \\ \mathbf{g}_{k-1} \end{bmatrix} = \left[\begin{array}{cccc|ccccccc} p_{00} & p_{01} & . & . & . & p_{0,n-k-1} & 1 & 0 & 0 & . & . & . & 0 \\ p_{10} & p_{11} & . & . & . & p_{1,n-k-1} & 0 & 1 & 0 & . & . & . & 0 \\ p_{20} & p_{21} & . & . & . & p_{2,n-k-1} & 0 & 0 & 1 & . & . & . & 0 \\ & & & & & & & & & & & & \\ & & & & & & & & & & & & \\ & & & & & & & & & & & & \\ p_{k-1,0} & p_{k-1,1} & . & . & . & p_{k-1,n-k-1} & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{array}\right] \quad (3.4)$$

where $p_{ij} = 0$ or 1

Chapter 10.11 Linear Block Code

- 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 $\mathbf{v} \cdot \mathrm{H}^{\mathrm{T}} = 0$
- 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

Chapter 10.11 Linear Block Code

- If the generator matrix of an (n,k) linear code is in the systematic form of (3.4), the parity-check matrix may take the following form :

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

$$= \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & p_{00} & p_{10} & \cdots & p_{k-1,0} \\ 0 & 1 & 0 & \cdots & 0 & p_{01} & p_{11} & \cdots & p_{k-1,1} \\ 0 & 0 & 1 & \cdots & 0 & p_{02} & p_{12} & \cdots & p_{k-1,2} \\ \vdots & & & & & & & & \\ 0 & 0 & 0 & \cdots & 1 & p_{0,n-k-1} & p_{1,n-k-1} & \cdots & p_{k-1,n-k-1} \end{bmatrix} \quad (3.7)$$

Chapter 10.11 Linear Block Code

- Let h_j be the j_{th} row of **H**

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

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

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

Syndrome decoding - I

- The generator matrix **G** is uses in the encoding operation at the transmitter. On other hand, the parity-check matrix **H** is used in the decoding operation at the receiver.
- Let ,

$$\mathbf{r} = \mathbf{c} + \mathbf{e} \qquad (10.78)$$

- **r** denote the 1-by-*n* *received vector* that results from sending the code vector **c** over a noisy channel.
- **e** is the *error vector* or *error pattern.*

$$\mathbf{e} = \mathbf{r} + \mathbf{c} = (e_0, e_1, \ldots, e_{n-1})$$

$$e_i = \begin{cases} 1 & \text{if an error has occurred in the } i\text{th location} \quad (r_i \neq c_i) \\ 0 & \text{otherwise} \quad (r_i = c_i) \end{cases} \qquad (10.79)$$

$$i = 0, 1, \ldots, n-1$$

Syndrome decoding - I

- The receiver has to decode the code vector **c** from the received **r**.
- The algorithm commonly used to perform this decoding operation stars with the computation of a 1-by-(*n*-*k*) vector the *error-syndrome vector* or simply the *syndrome*.
- Given a 1-by-*n* received vector **r**, the syndrome is defined as

$$\mathbf{s} = \mathbf{r}\mathbf{H}^T = (s_0, s_1, \ldots, s_{n-k-1}) \qquad (10.80)$$

- **s**=**0** if and only if **r** is a code word and receiver accepts **r** as the transmitted code word.
- **s**≠**0** if and only if **r** is not a code word and 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*.
  - There are 2^k-1 undetectable error patterns.

Syndrome decoding - I

- The syndrome has the following important properties.
- Property 1 :
  - *The syndrome depends only on the error pattern, and not on the transmitted codeword.*

Proof : Using Eqs.(10.78) and (10.80) and then Eq.(10.77) to obtain

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

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

$$= \mathbf{e}\mathbf{H}^T \quad (\because \mathbf{c}\mathbf{H}^T = \mathbf{0})$$

$$= (s_0, s_1, \ldots, s_{n-k-1}) \qquad (10.81)$$

- Hence, the parity-check matrix **H** of a code permits us to compute the syndrome **s**, which depends only upon the error pattern **e**.

Syndrome decoding - I

- Property 2 :
  - *All error patterns that differ by a code word have the same syndrome.*

- Proof:
  - For k message bits, there are 2^k distinct code vectors denoted as c_i, i=1, 2, …, 2^k. For any error pattern $\mathbf{e}$, we define the 2^k distinct vectors $\mathbf{e}_i$ as

$$\mathbf{e}_i = \mathbf{e} + \mathbf{c}_i, \quad i = 1, \ldots, 2^k \tag{10.82}$$

  - In any event, multiplying both sides of Eq. (10.82) by matrix $\mathbf{H}^T$, we get

$$\begin{aligned} \mathbf{e}_i \mathbf{H}^T &= \mathbf{e}\mathbf{H}^T + \mathbf{c}_i \mathbf{H}^T \\ &= \mathbf{e}\mathbf{H}^T \end{aligned} \tag{10.83}$$

    which is independent of index i.

Syndrome decoding - I

- The set of vectors $\{e_i, i=1, 2, \ldots, 2^k\}$ so defined is called a *coset* of the code. In other words, a code set has exactly 2^k elements that differ at most by a code vector.
- Because there are 2^n possible received vectors, and a coset has 2^k different elements, so an (n,k) linear block code has 2^{n-k} possible cosets.
- From Eq. (10.83), we may state that each coset of the code is characterized by a unique syndrome.
- Based on Eqs. (10.69), (10.75), and (10.81) we obtain the $(n-k)$ elements of the syndrome **s** as:

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

$$
\begin{aligned}
s_1 &= e_0 + e_{n-k}p_{00} + e_{n-k+1}p_{10} + \cdots + e_{n-1}p_{k-1,1} \\
s_2 &= e_1 + e_{n-k}p_{01} + e_{n-k+1}p_{11} + \cdots + e_{n-1}p_{k-1,2} \\
&\vdots \\
s_{n-k} &= e_{n-k-1} + e_{n-k}p_{0,n-k+1} + \cdots e_{n-1}p_{k-1,n-k}
\end{aligned}
\qquad (10.84)
$$

Syndrome decoding - I

- The syndrome digits are linear combinations of the error digits.
- The syndrome digits can be used for error detection.
- Because the $n - k$ linear equations of (10.84) 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.
- Knowledge of the syndrome **s** reduce the search for the true error pattern **e** from 2^n to 2^{n-k}.
- In particular, the decoder has the task of making the best selection from the coset corresponding to **s.**

Minimum Distance Consideration

- Let code vectors $\mathbf{c}_1$ and $\mathbf{c}_2$ have the same number of elements.
- The *Hamming distance* between $\mathbf{c}_1$ and $\mathbf{c}_2$, denoted $d(\mathbf{c}_1,\mathbf{c}2)$, is defined as the number of places where they differ.
  - For example, the *Hamming distance* between $\mathbf{c}_1$=(1001011) and $\mathbf{c}_2$=(0100011) is 3.
- The *Hamming weight* (or simply weight) of a code vector $\mathbf{c}$, denote by $w(\mathbf{c})$, is defined as the number of nonzero elements of $\mathbf{c}$.
  - For example, the *Hamming weight* of $\mathbf{c}$=(1001011) is 3.
- From the definition of hamming distance and definition of module-2 addition that the Hamming weight between two n-tuple, $\mathbf{c}_1$ and $\mathbf{c}_2$, is equal to the Hamming weight of the sum of $\mathbf{c}_1$ and $\mathbf{c}_2$, that is

$$d(\mathbf{c}_1,\mathbf{c}_2) = w(\mathbf{c}_1+\mathbf{c}_2)$$

Minimum Distance Consideration

- The *minimum distance* $d_{\min}$ of a linear block code is defined as the smallest Hamming distance between any pair of code vectors in the code.
- Given a block code C ,the minimum distance of C, denoted $d_{\min}$, is defined as $d_{\min} = \min\left\{d\left(\mathbf{c}_1, \mathbf{c}_2\right) : \mathbf{c}_1, \mathbf{c}_2 \in C, \mathbf{c}_1 \neq \mathbf{c}_2\right\}$
- If C is a linear block, the sum of two vectors is also a code vector.
- From $d\,(\mathbf{c}_1,\mathbf{c}_2)=w\,(\mathbf{c}_1+\mathbf{c}_2)$, the Hamming distance between two code vectors in C is equal to the Hamming weight of third code vector in C

$$
\begin{aligned}
d_{\min} &= \min\left\{w\left(\mathbf{c}_1 + \mathbf{c}_2\right) : \mathbf{c}_1, \mathbf{c}_2 \in C, \mathbf{c}_1 \neq \mathbf{c}_2\right\} \\
&= \min\left\{w\left(\mathbf{x}\right) : \mathbf{x} \in C, \mathbf{x} \neq \mathbf{0}\right\} \\
&= w_{\min}
\end{aligned}
$$

$w_{\min}$ is called the minimum weight of the linear code C

Minimum Distance Consideration

- The minimum distance $d_{\min}$ is related to the parity-check matrix **H** of the code.
- Let the matrix **H** be expressed in terms of its columns as follows:

$$\mathbf{H} = \left[\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_n\right] \tag{10.85}$$

- From Eq.(10.77) we get

$$\mathbf{c}\mathbf{H}^T = \left[c_1 \; c_2 \; \ldots c_n\right] \begin{bmatrix} h_1^T \\ h_2^T \\ \vdots \\ h_n^T \end{bmatrix} = c_1 h_1^T + c_2 h_2^T + \cdots + c_n h_n^T = 0$$

The vector **c** must have 1s in such positions that the correspond rows of $\mathbf{H}^T$ sum to the zero vector **0**.

- Because $d_{\min} = w_{\min}$, the smallest Hamming weight equals the minimum distance of the code. Hence, *the minimum distance of a linear block code is defined by the minimum number of rows of the matrix* $\mathbf{H}^T$ whose sum is equal to the zero vector.

Minimum Distance Consideration

- The minimum distance, $d_{\min}$, determines the error-correcting capability of the code.
- If a code vector $\mathbf{c}_i$ is transmitted and the received vector is $\mathbf{r}=\mathbf{c}_i+\mathbf{e}$, we require that the decoder output $\hat{\mathbf{c}}=\mathbf{c}_i$, whenever $w(\mathbf{e})\leq t$.

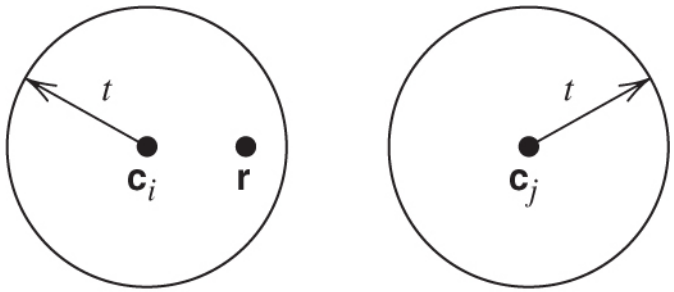
error bits $\leq t$

- The best strategy for decoder then is to pick the code vector $d(\mathbf{c}_i,\mathbf{r})$ closest to the received vector $\mathbf{r}$, that is, the one for which is smallest.

- With such a strategy, the decoder will be able to detect and correct all error patterns of $w(\mathbf{e})\leq t$, we will show that $d_{\min}\geq 2t+1$.

Minimum Distance Consideration

- We construct two spheres, each of radius t, around the points that represent $\mathbf{c}_i$ and $\mathbf{c}_j$.
- Let these two spheres are disjoint, $d(\mathbf{c}_i,\mathbf{c}_j) \geq 2t+1$, as depicted in Figure 10.23a.
  - If the code vector $\mathbf{c}_i$ is transmitted, and $d(\mathbf{c}_i,\mathbf{r}) \leq t$, it is clear that the decoder will pick $\mathbf{c}_i$ as it is the code vector closest to the received vector $\mathbf{r}$.

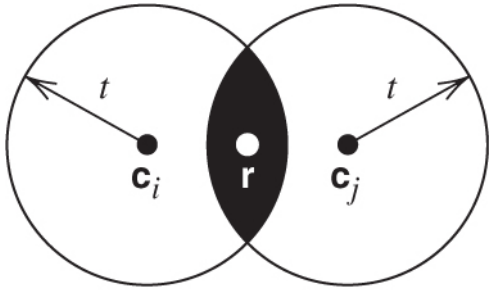


(a)

Figure 10.23

Minimum Distance Consideration

- Let these two spheres are intersect, $d(\mathbf{c}_i,\mathbf{r}) \leq 2t$, as depicted in Figure 10.23b.
  - If then the code vector $\mathbf{c}_i$ is transmitted, there exists a received vector $\mathbf{r}$, and $d(\mathbf{c}_i,\mathbf{r}) \leq 2t$. But now, $\mathbf{r}$ is as close to $\mathbf{c}_i$ as it is to $\mathbf{c}_j$, so there is now the possibility of the decoder picking the vector $\mathbf{c}_j$, which is wrong.



(*b*)

Figure 10.23

Minimum Distance Consideration

- An (n,k) linear block code has the power to correct all error patterns of weight t or less if and only if

$$d\left(\mathbf{c}_i, \mathbf{c}_j\right) \geq 2t+1 \text{ for all } \mathbf{c}_i \text{ and } \mathbf{c}_j$$

- By definition, the smallest distance between any pair of code vectors is the minimum distance of the code, $d_{\min}$.

- So, an (n,k) *linear block code of minimum distance* $d_{\min}$ *can correct up to t errors if, and only if,*

$$t \leq \left\lfloor \frac{1}{2}\left(d_{\min} - 1\right) \right\rfloor \qquad (10.86)$$

Syndrome Decoding - II

We are now ready to describe a syndrome-based decoding scheme for linear block code.

- Let $\mathbf{c}_1, \mathbf{c}_2, \ldots, \mathbf{c}_{2^k}$ denote the 2^k code vectors of an (n,k) linear block code.
- $\mathbf{r}$ denote the receiver vector, which may have one of 2^n possible values.
- The receiver has the task of partitioning the 2^n possible received vector into 2^k disjoint subset $D_1, D_2, \ldots, D_{2^k}$, D_i is the ith subset correspond to code vector $\mathbf{c}_i$ for $1 \leq i \leq 2^k$.
- For decoding to be correct, $\mathbf{r}$ must be in the subset that belongs to $\mathbf{c}_i$.

Syndrome Decoding - II

- The 2^k subsets described herein constitute a *standard array* of the linear block code.
- To construct it, we may exploit the linear structure of the code by proceeding as follows:

1. The 2^k code vectors are placed in a row with the all-zero code vector $\mathbf{c}_1$ as the left-most element.

2. An error pattern $\mathbf{e}_2$ is picked and placed under $\mathbf{c}_1$, and a second row is formed by adding $\mathbf{e}_2$ to each of the remaining code vectors in the first row; it is important that the error pattern chosen as the first element in a row not have previously appeared in the stand array. (Note that $\mathbf{e}_1$=0).

3. Step 2 is repeated until all the possible error pattern have been accounted for.

Syndrome Decoding - II

- Figure 10.24 illustrates the structure of the stand array.
- The 2^{n-k} rows of the array represent the cosets of the code, and their first elements $\mathbf{e}_2, \ldots, \mathbf{e}_{2^{n-k}}$ are called *coset leaders*.

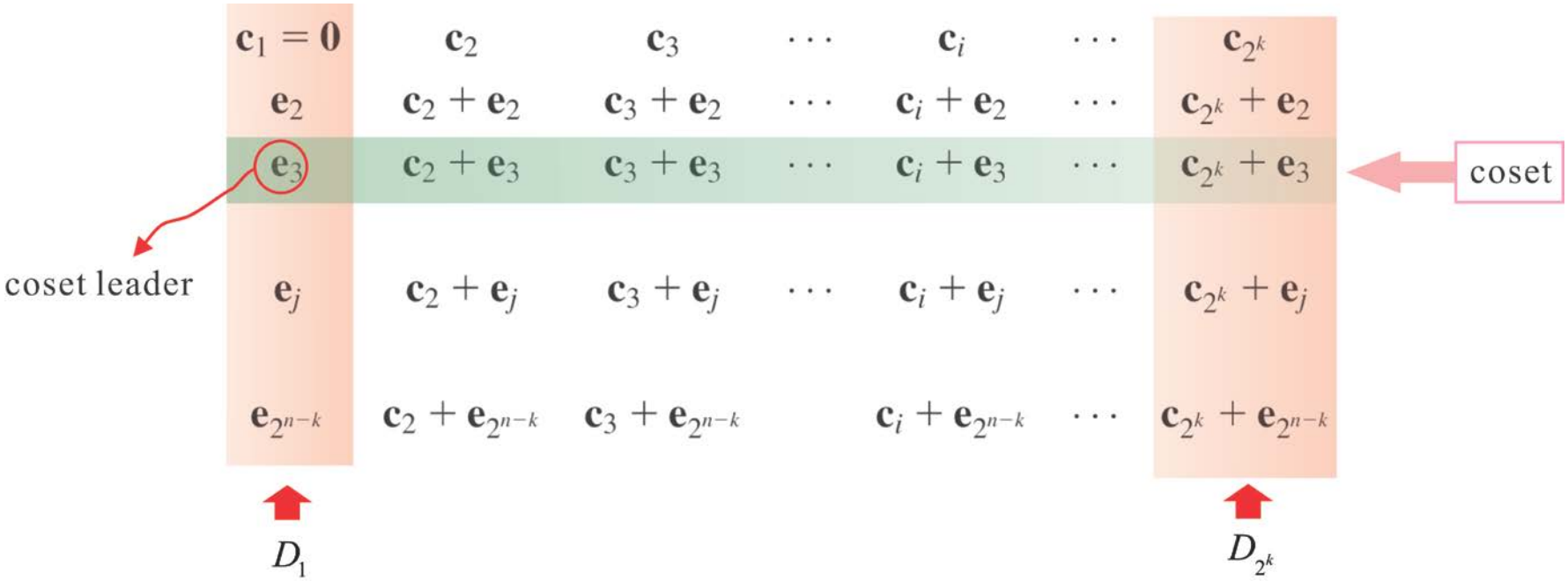


Figure 10.24 Stand array for an (n, k) block code

Syndrome Decoding - II

- For a given channel, the probability of decoding error is minimized when the most likely error are chosen as the coset leader.
- In the case of a binary symmetric channel, the smaller the Hamming weight of an error pattern the more likely it is to occur.

  ➔The standard array should be constructed with each coset leader having the minimum Hamming weight in its coset.

- *Syndrome Decoding*
  1. For the received vector **r**, compute the syndrome $\mathbf{s}=\mathbf{r}\mathbf{H}^T$.
  2. Within the coset characterized by the syndrome **s**, identify the coset leader (i.e., the error pattern with the largest probability of occurrence); call it $\mathbf{e}_0$.
  3. Compute the code vector

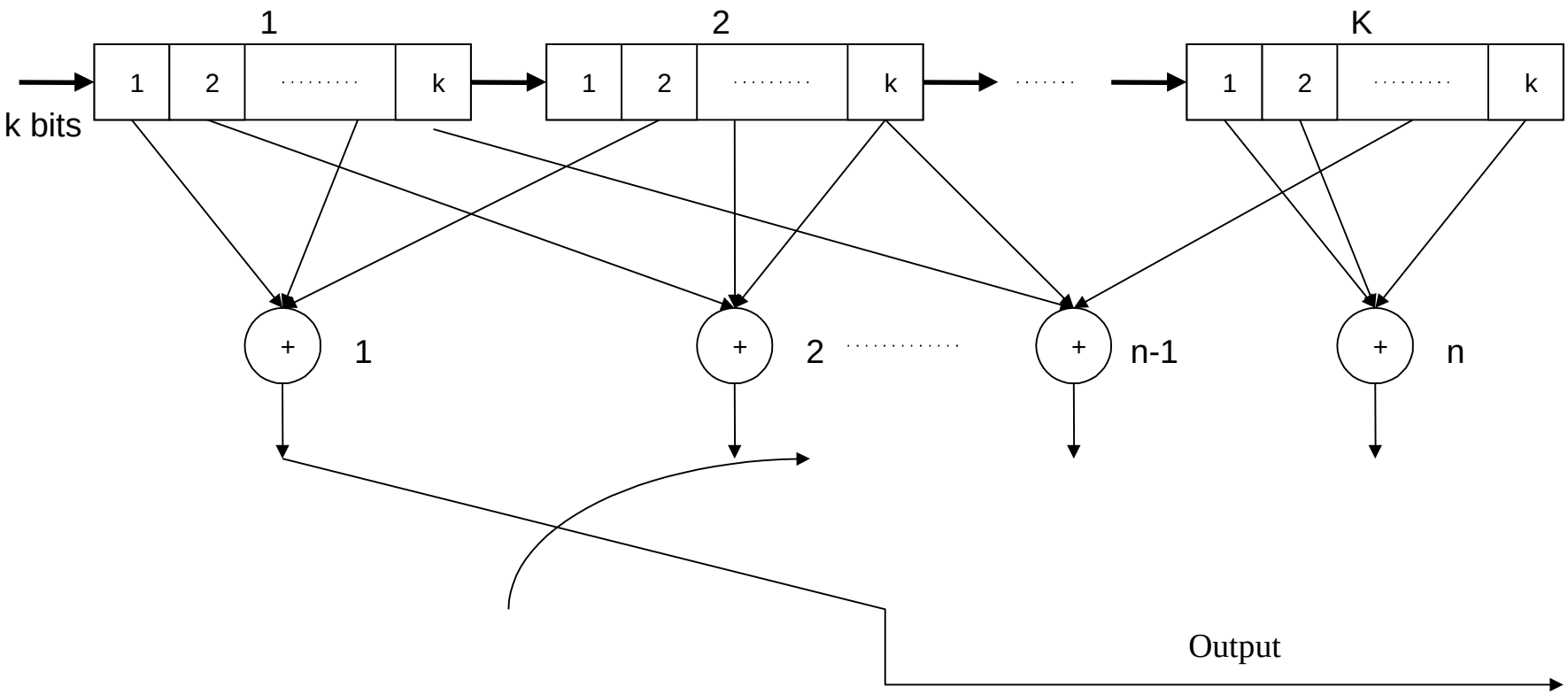
$$\mathbf{c} = \mathbf{r} + \mathbf{e}_0 \qquad (10.87)$$

  as the decoded version of the received vector **r.**

Convolutional Code

Wireless Information Transmission System Lab.
Institute of Communications Engineering
National Sun Yat-sen University

Structure of Convolutional Encoder

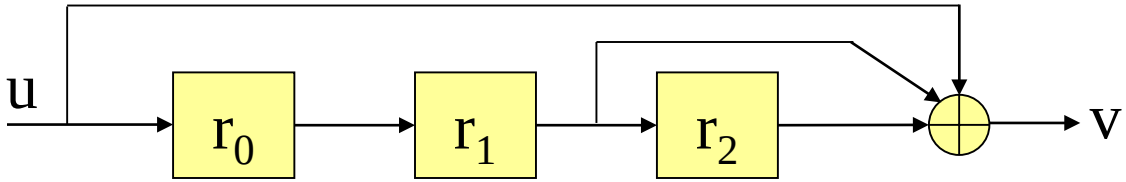


Convoltuional Code

- Convolutional codes
  - k = number of bits shifted into the encoder at one time
    - k=1 is usually used!!
  - n = number of encoder output bits corresponding to the k information bits
  - $r = k/n$ = code rate
  - K = constraint length, encoder memory

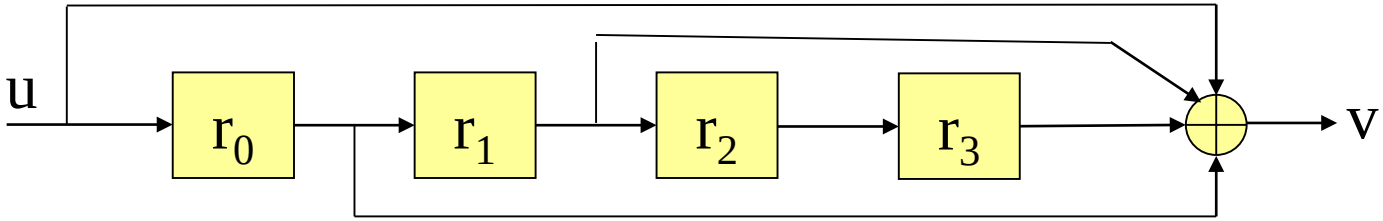
- Each encoded bit is a function of the present input bits and their past ones.

Generator Sequence



$$g_0^{(1)} = 1,\ g_1^{(1)} = 0,\ g_2^{(1)} = 1,\ \text{and}\ g_3^{(1)} = 1.$$

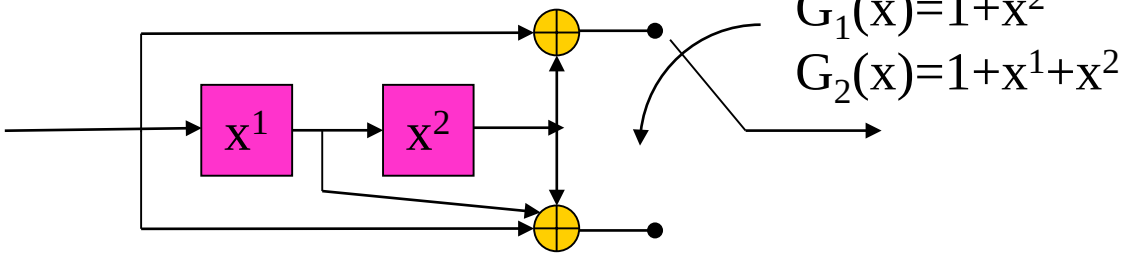
Generator Sequence: $g^{(1)}=(1\ 0\ 1\ 1)$



$$g_0^{(2)} = 1,\ g_1^{(2)} = 1,\ g_2^{(2)} = 1,\ g_3^{(2)} = 0,\ \text{and}\ g_4^{(2)} = 1.$$

Generator Sequence: $g^{(2)}=(1\ 1\ 1\ 0\ 1)$

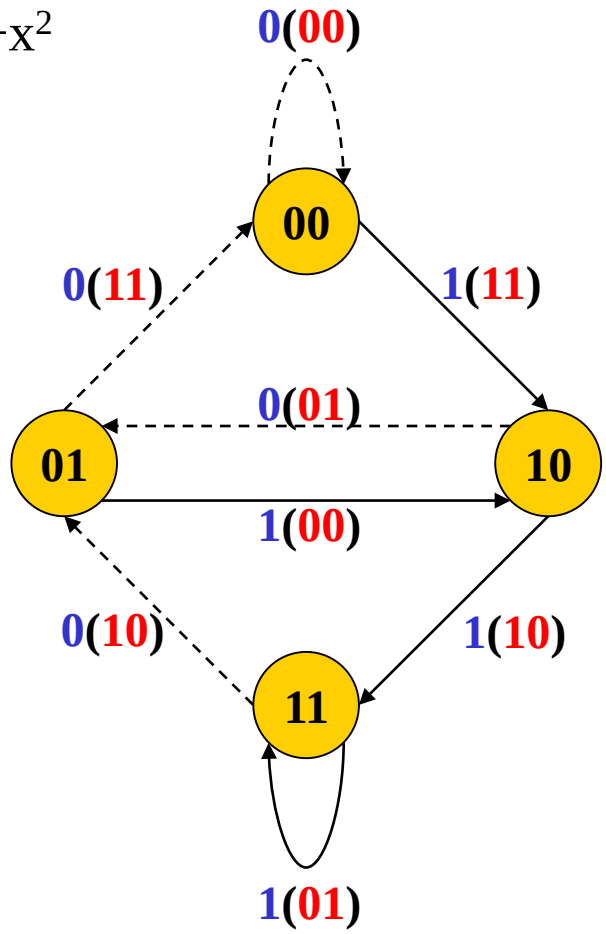
Convolutional Codes
An Example – (rate=1/2 with K=2)



$G_1(x)=1+x^2$

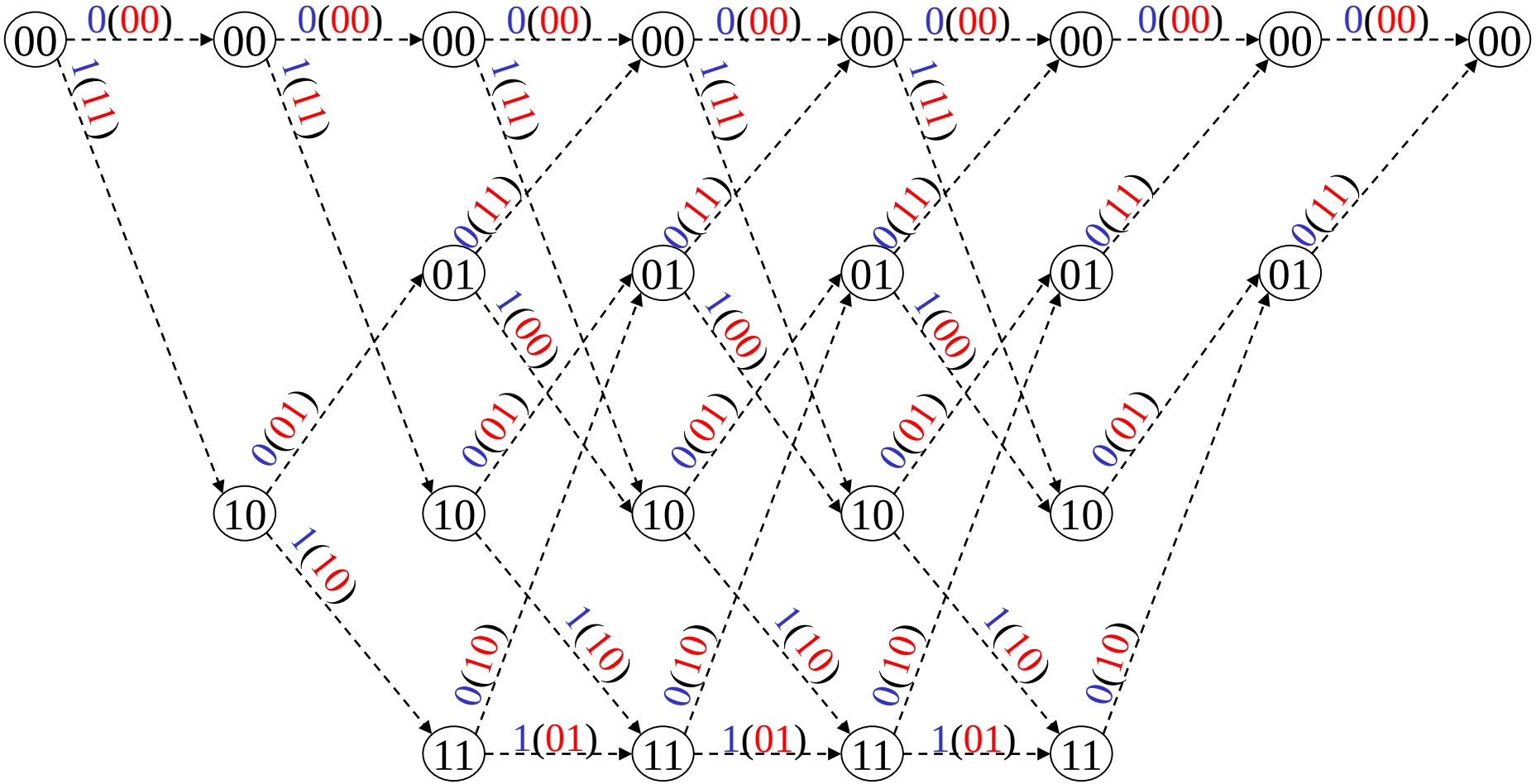
$G_2(x)=1+x^1+x^2$

	Present	Next	Output
0	00	00	00
1	00	10	11
0	01	00	11
1	01	10	00
0	10	01	01
1	10	11	10
0	11	01	10
1	11	11	01



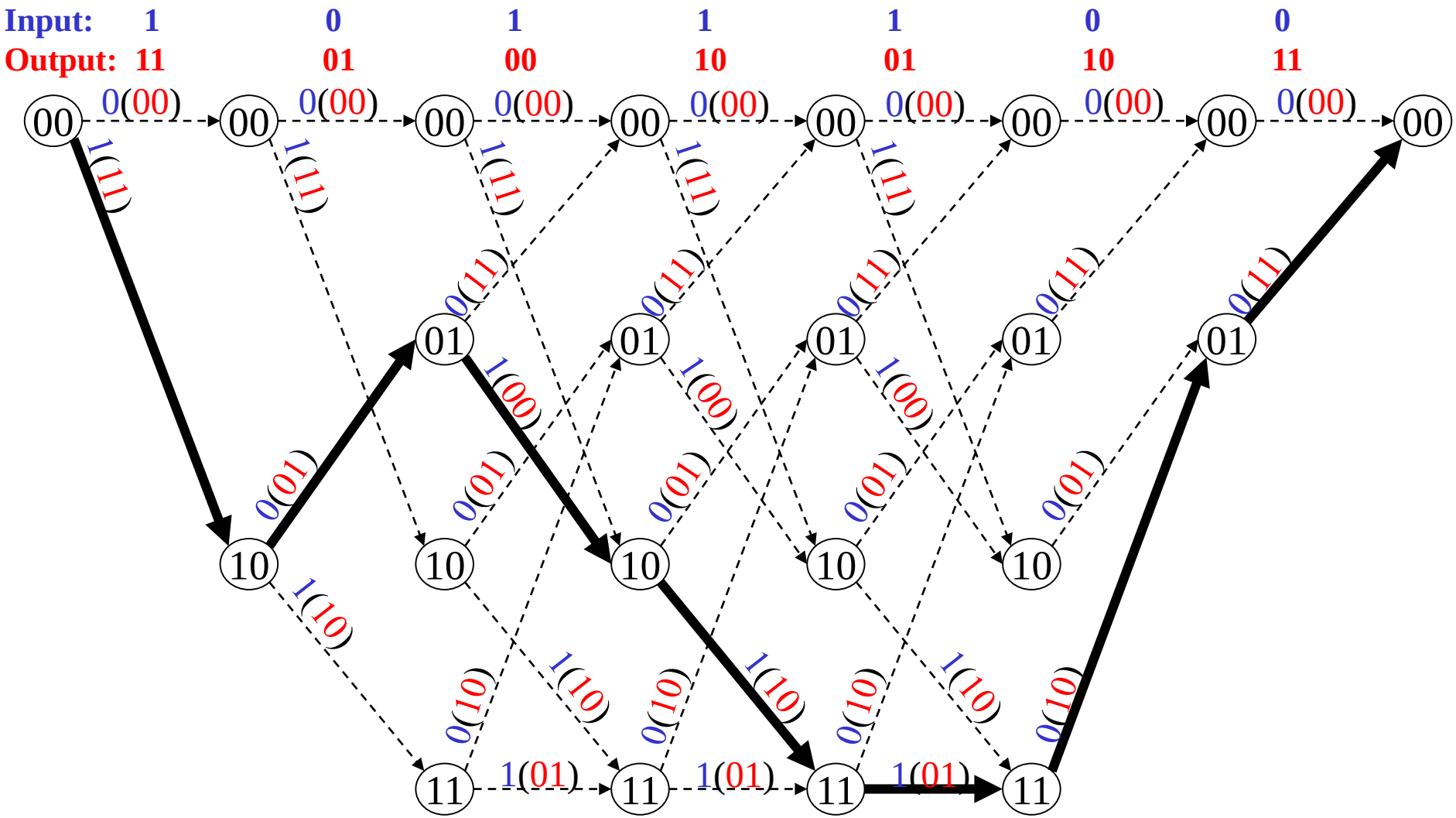
State Diagram

Trellis Diagram Representation



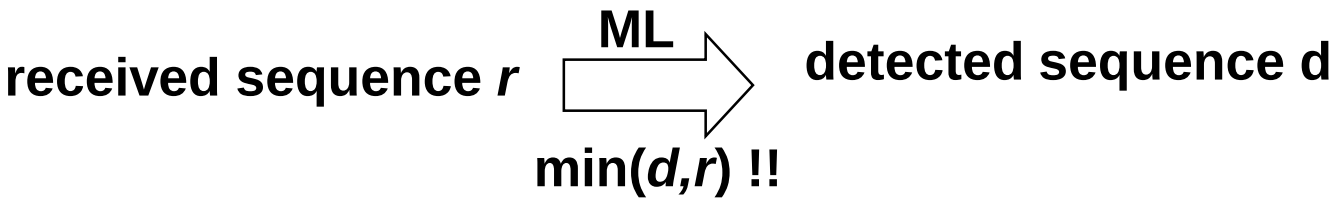
Trellis termination: K tail bits with value 0 are usually added to the end of the code.

Encoding Process



Viterbi Decoding Algorithm

- Maximum Likelihood (ML) decoding rule

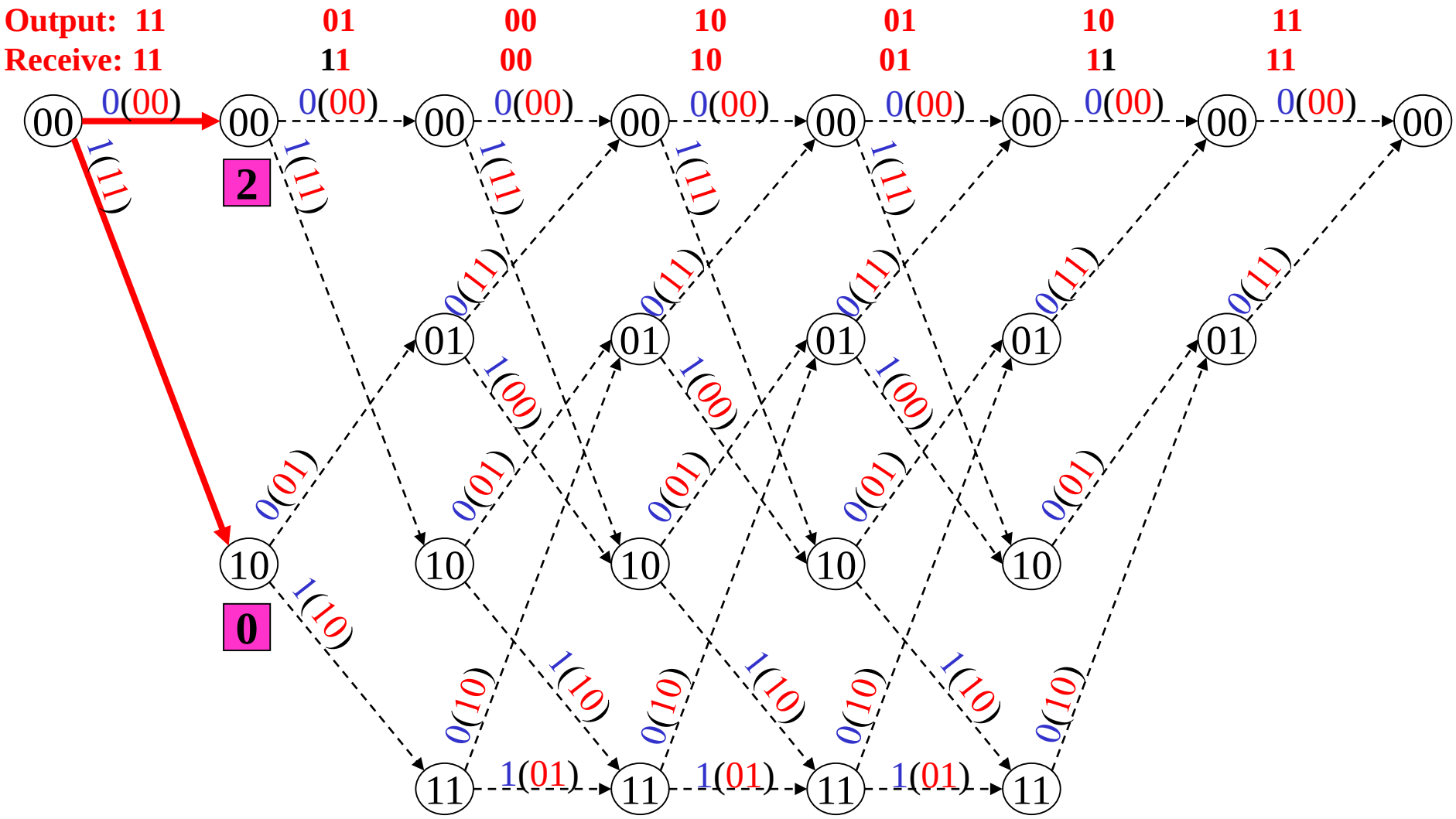


- Viterbi Decoding Algorithm
  - An efficient search algorithm
    - Performing ML decoding rule.
    - Reducing the computational complexity.

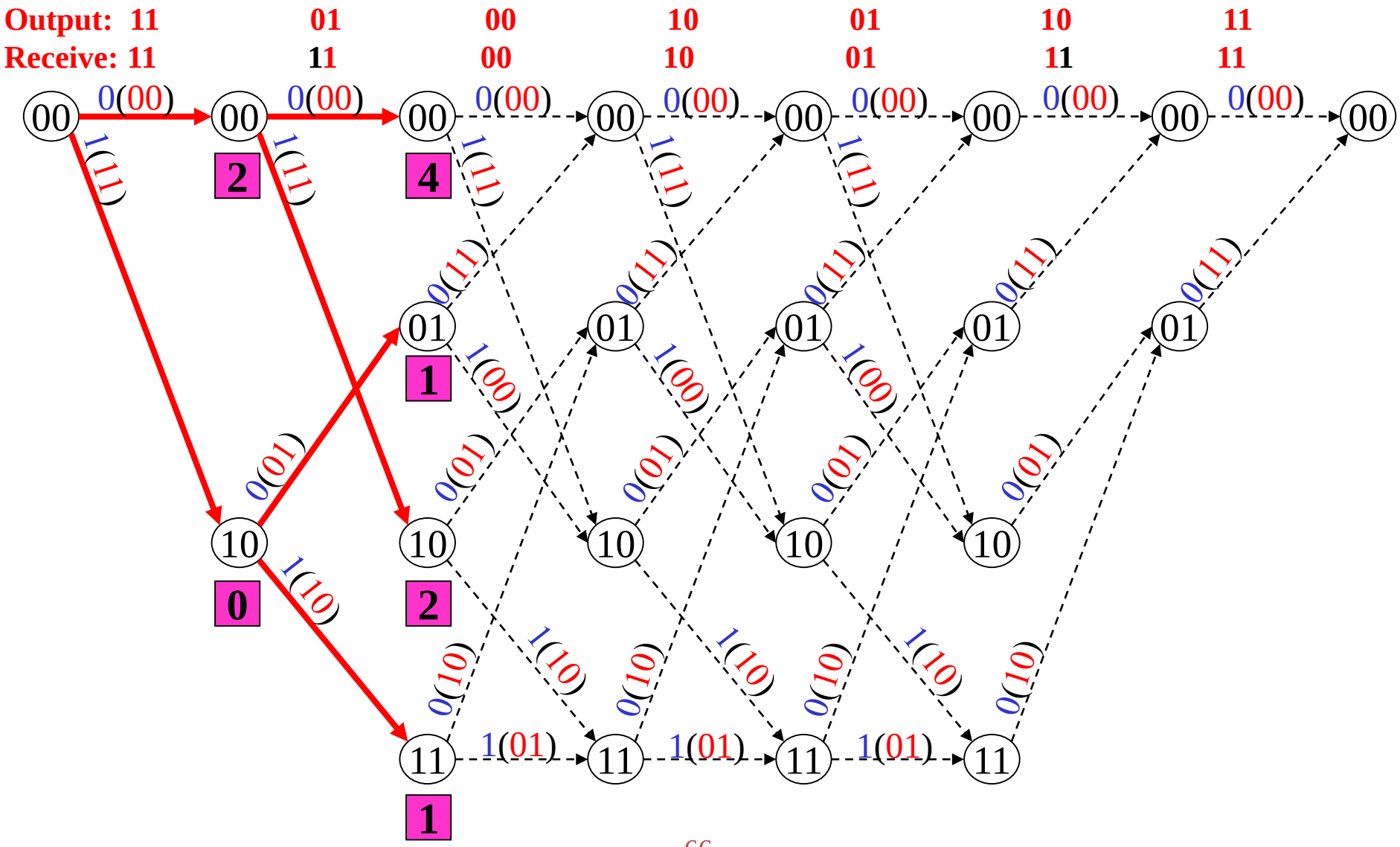
Viterbi Decoding Algorithm

- Basic concept
  - Generate the code trellis at the decoder
  - The decoder penetrates through the code trellis *level by level* in search for the transmitted code sequence
  - At each level of the trellis, the decoder computes and compares the *metrics* of all the partial paths entering a node
  - The decoder *stores* the partial path with the larger metric and *eliminates* all the other partial paths. The stored partial path is called the *survivor*.

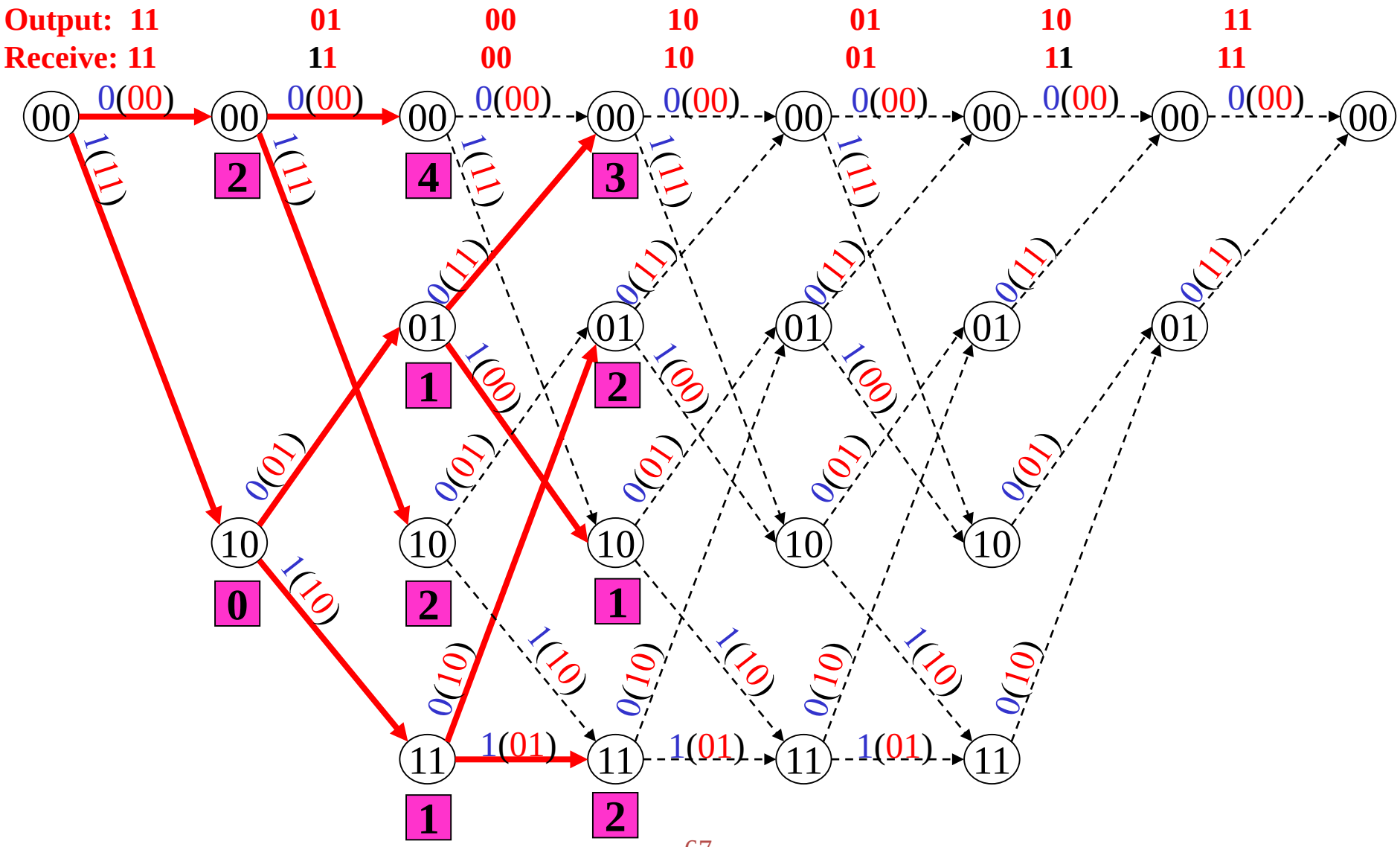
Viterbi Decoding Algorithm



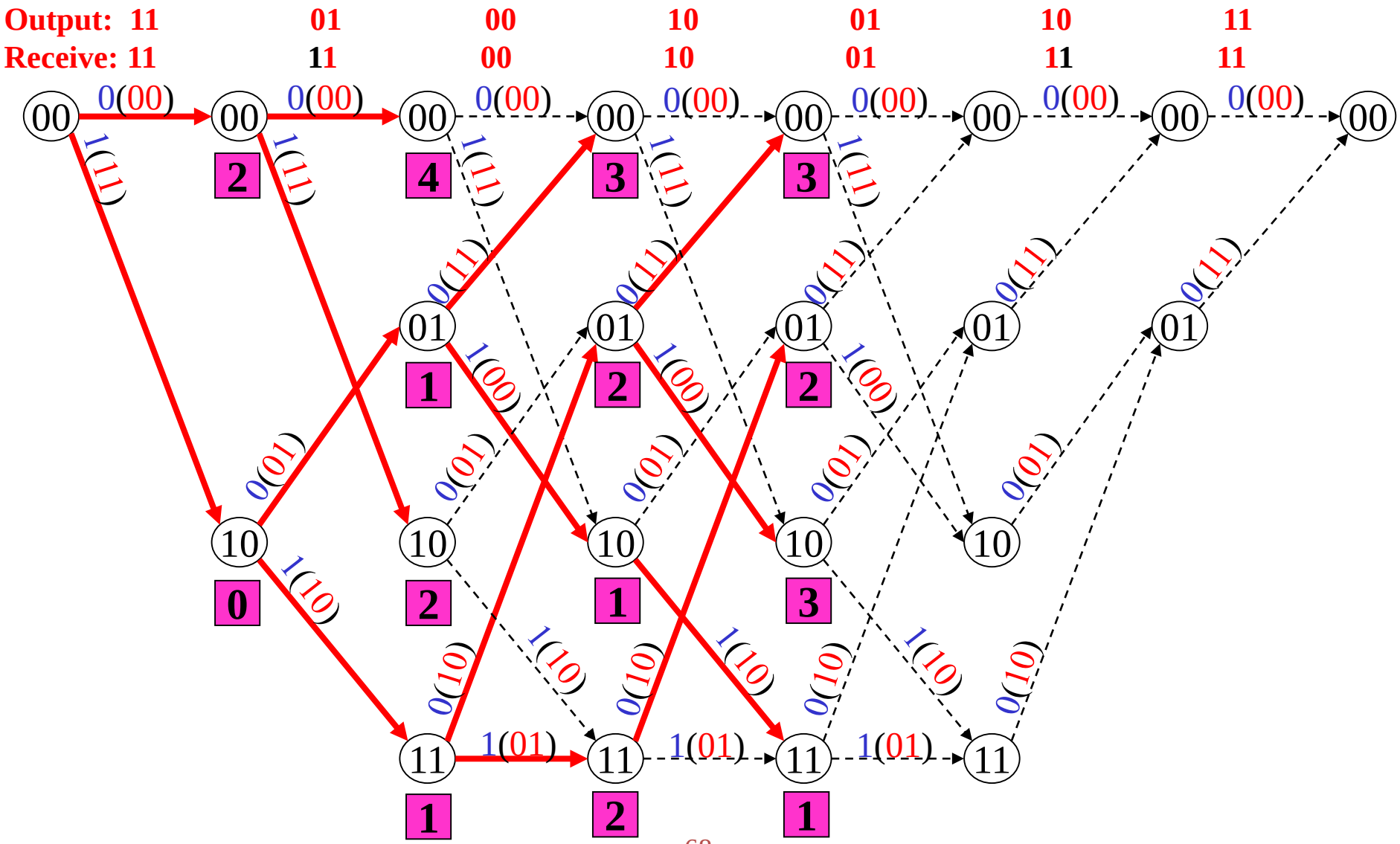
Viterbi Decoding Algorithm



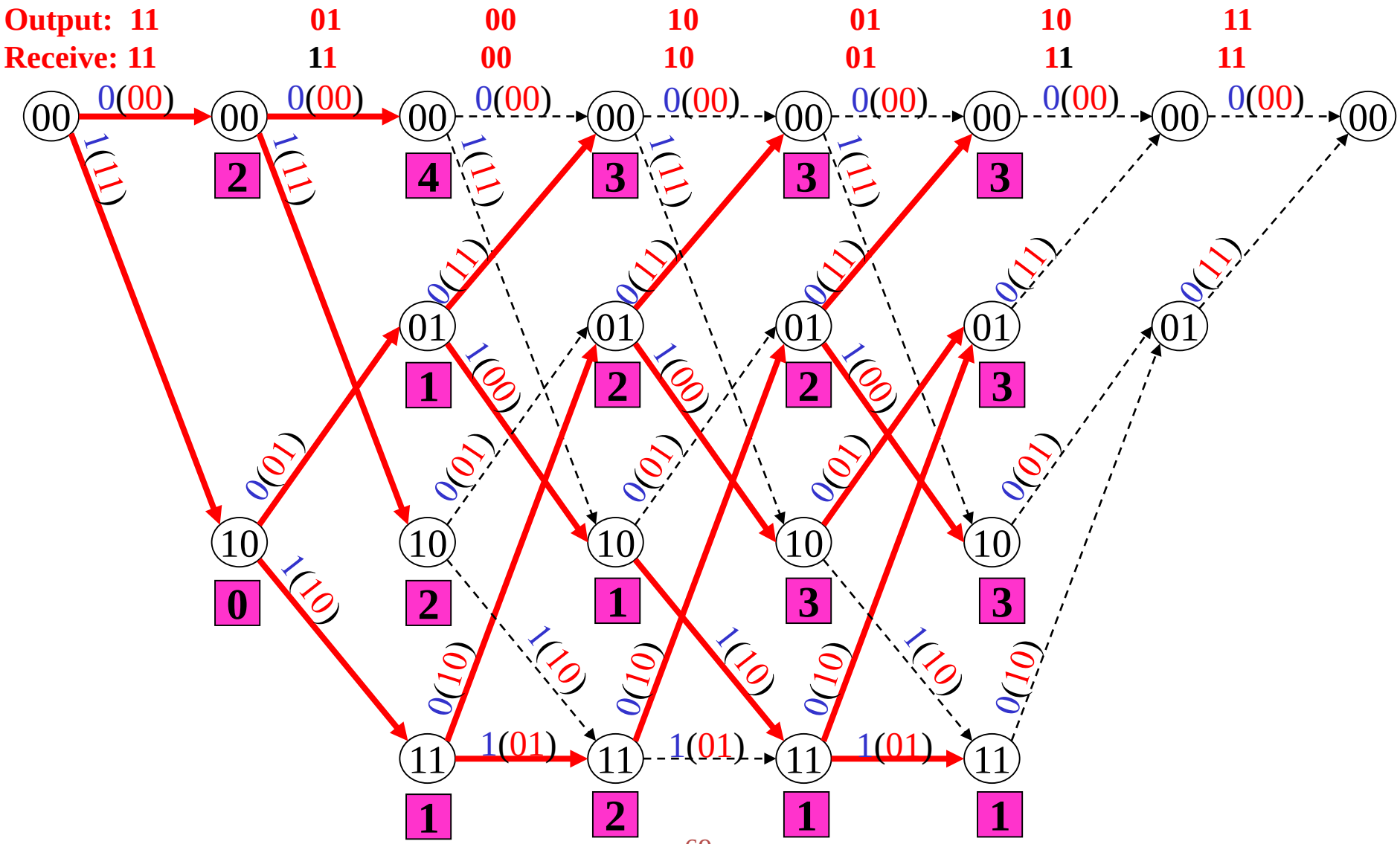
Viterbi Decoding Algorithm



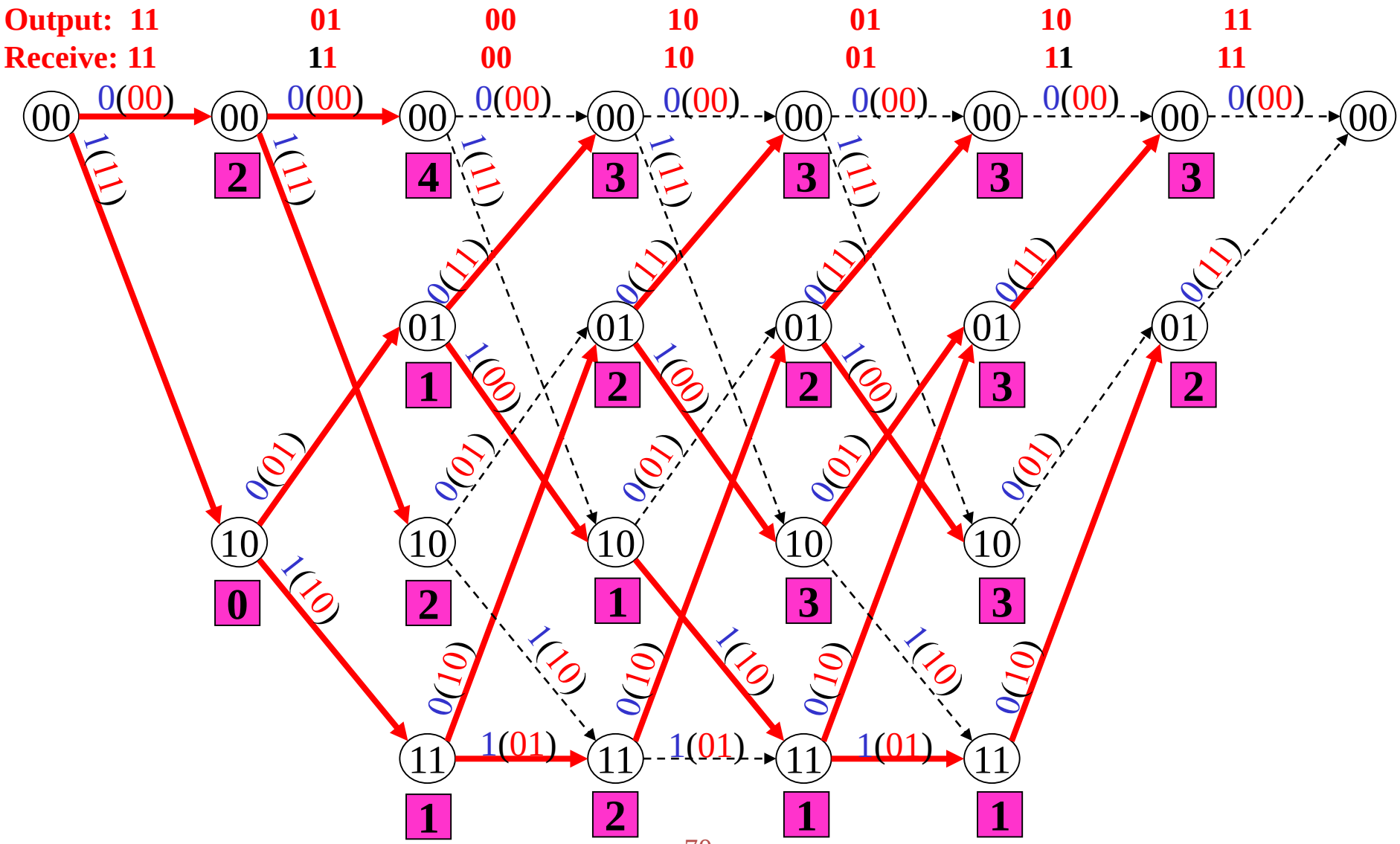
Viterbi Decoding Algorithm



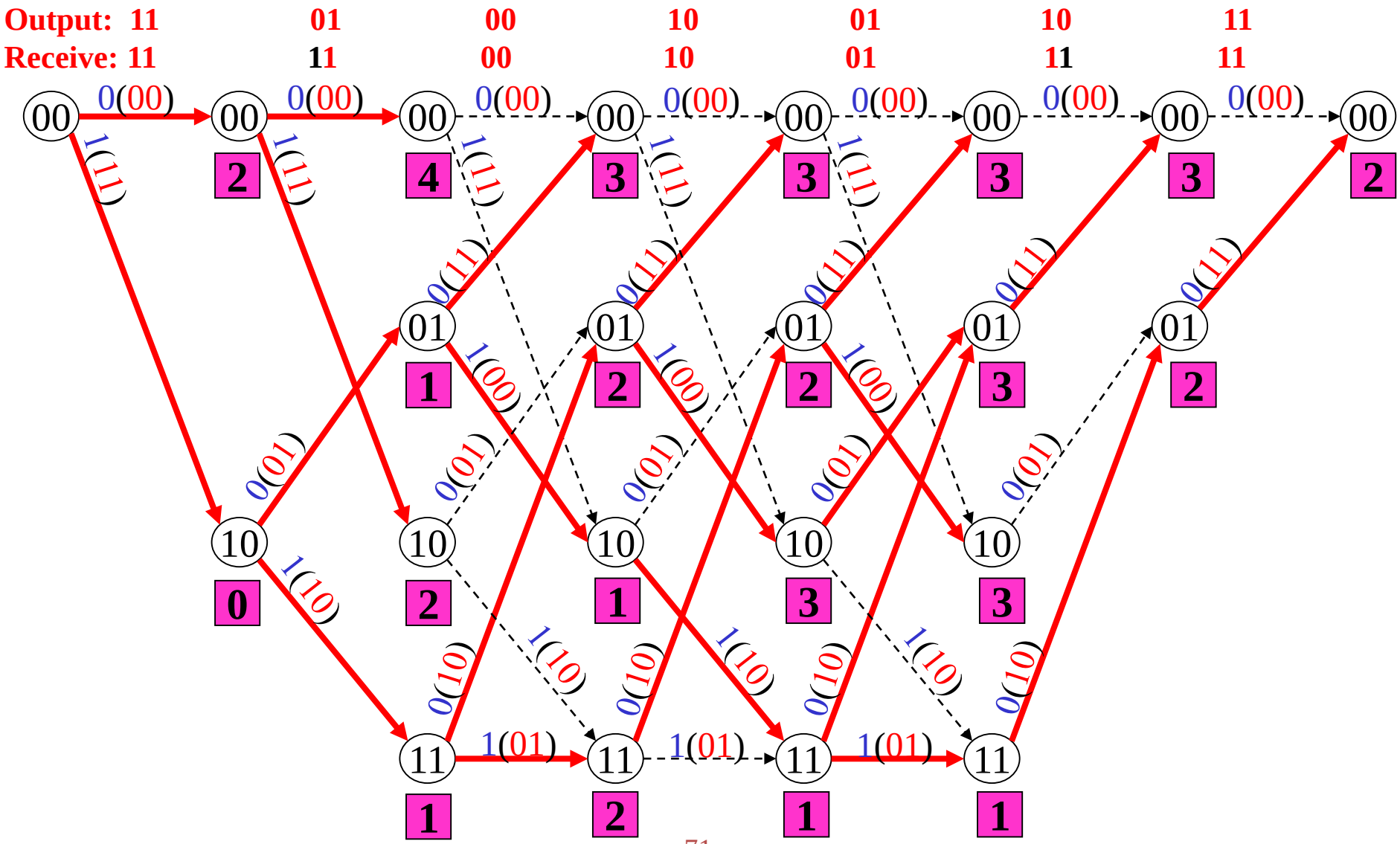
Viterbi Decoding Algorithm



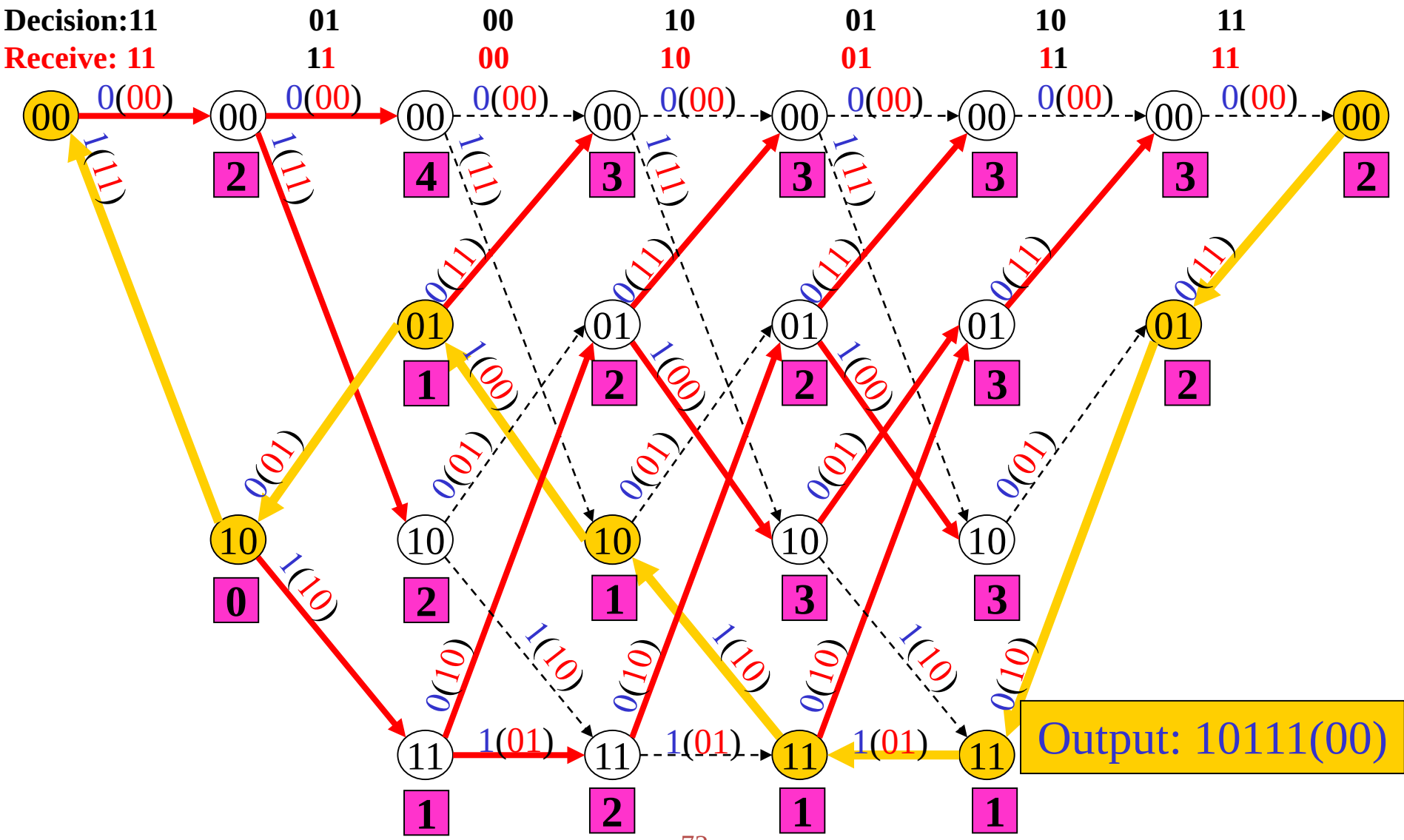
Viterbi Decoding Algorithm



Viterbi Decoding Algorithm



Viterbi Decoding Algorithm



Error Detecting Code (Cyclic Redundancy Code)

Wireless Information Transmission System Lab.
Institute of Communications Engineering
National Sun Yat-sen University

Error Detecting Codes

- Cyclic Redundancy Code (CRC Code) – also know as the polynomial code.

- Polynomial codes are based upon treating bit strings as representations of polynomials with coefficients of 0 and 1 only.
- For example, 110001 represents a six-term polynomial: $x^5+x^4+x^0$

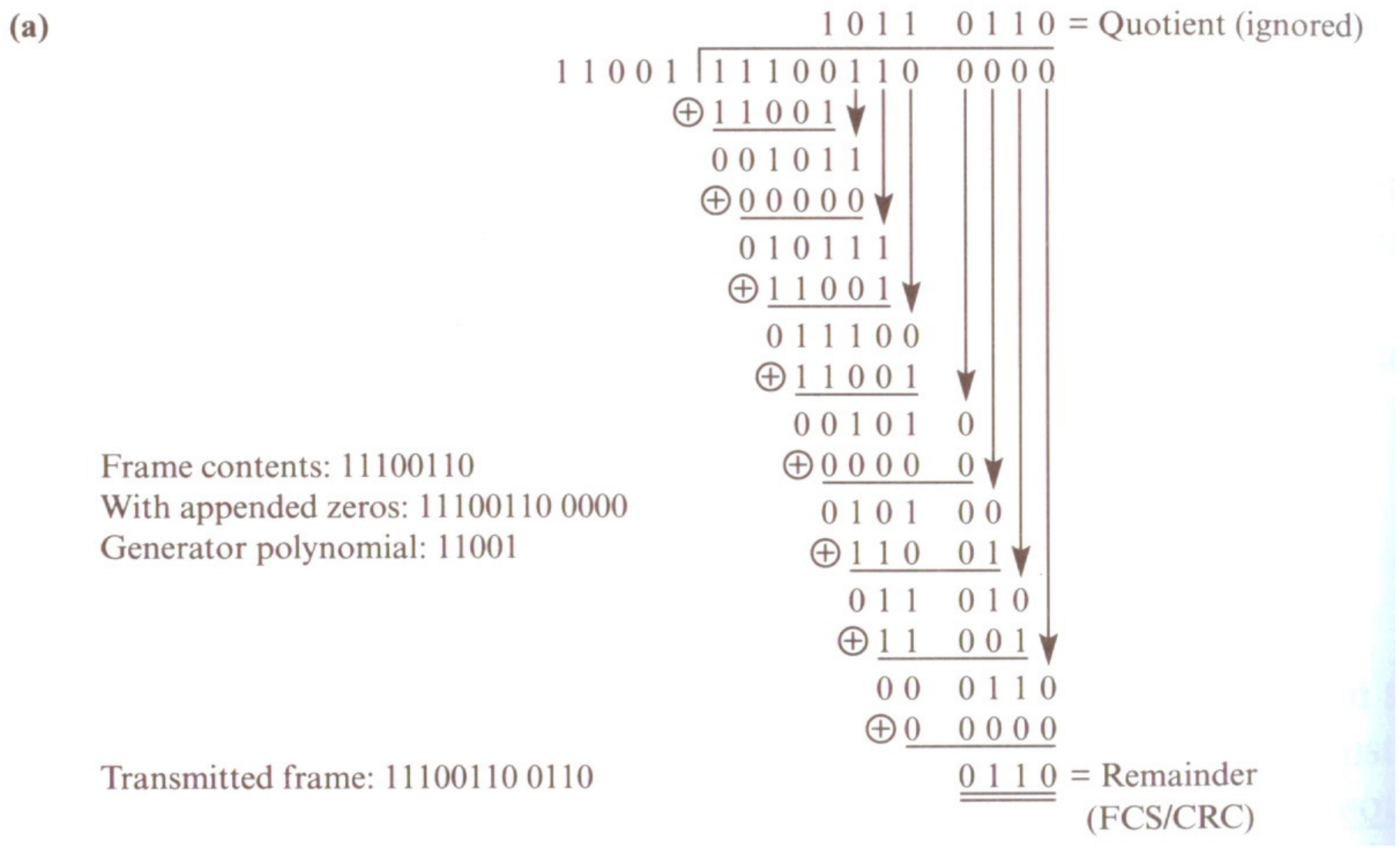
- When the polynomial code method is employed, the sender and receiver must agree upon a **generator polynomial**, $G(x)$, in advance.
- To compute the **checksum** for some frame with m bits, corresponding to the polynomial $M(x)$, the frame must be longer than the generator polynomial.

Error Detecting Codes

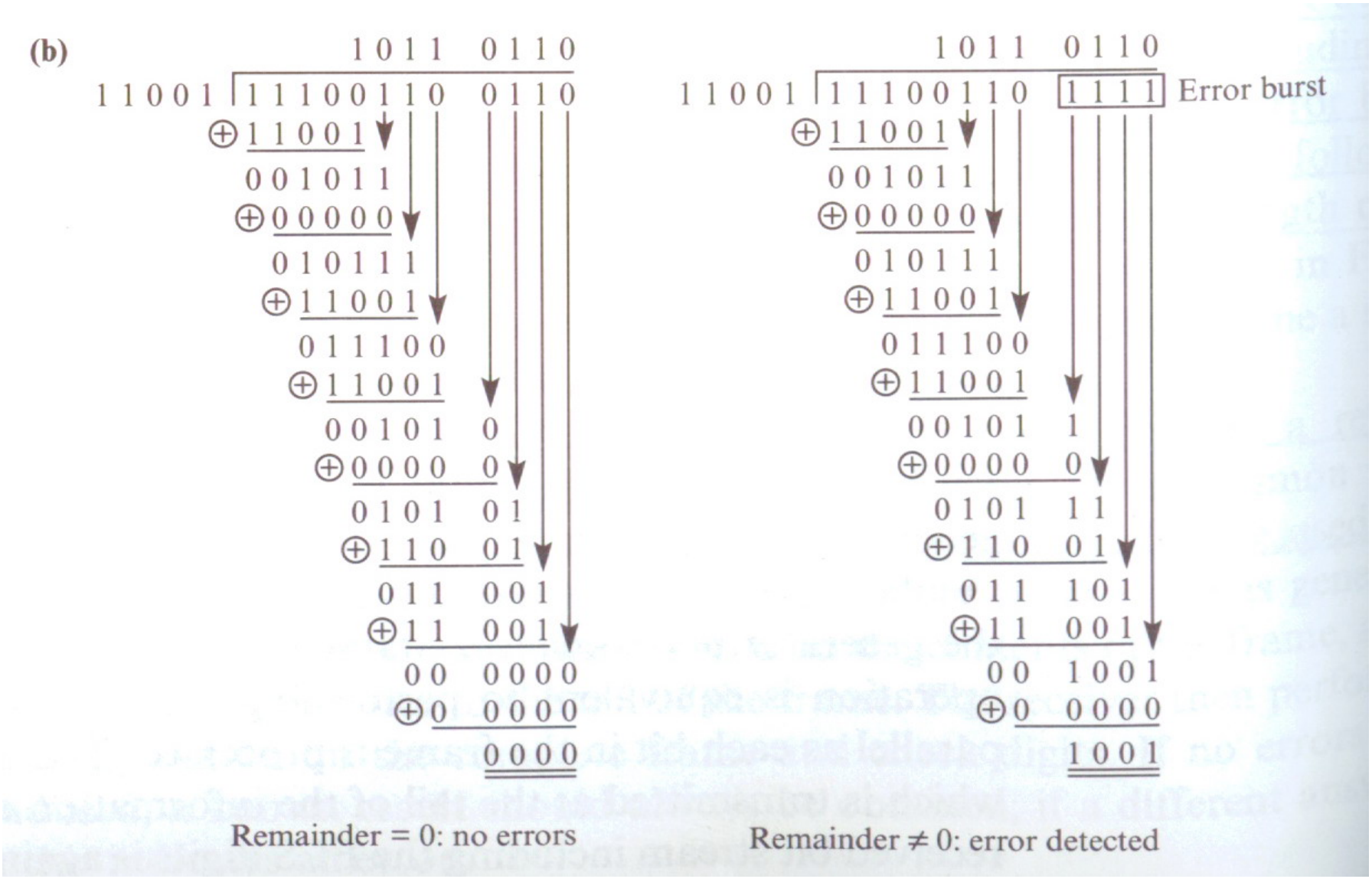
- The idea is to append a checksum to the end of the frame in such a way that the polynomial represented by the checksummed frame is divisible by $G(x)$.
- When the receiver gets the checksummed frame, it tries dividing it by $G(x)$. If there is a remainder, there has been a transmission error.
- The algorithm for computing the checksum is as follows:

1. Let r be the degree of $G(x)$. Append r zero bits to the low-order end of the frame, so it now contains $m + r$ bits and corresponds to the polynomial $x^r M(x)$.
2. Divide the bit string corresponding to $G(x)$ into the bit string corresponding to $x^r M(x)$ using modulo 2 division.
3. Subtract the remainder (which is always r or fewer bits) from the bit string corresponding to $x^r M(x)$ using modulo 2 subtraction. The result is the checksummed frame to be transmitted. Call its polynomial $T(x)$.

Calculation of the polynomial code checksum



Calculation of the polynomial code checksum



Cyclic Redundancy Code (CRC)

- Examples of CRCs used in practice:

$$\text{CRC} - 16 = X^{16} + X^{15} + X^2 + 1$$
$$\text{CRC} - \text{CCITT} = X^{16} + X^{12} + X^5 + 1$$
$$\text{CRC} - 32 = X^{32} + X^{26} + X^{23} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^{22} + X + 1$$

- A 16-bit checksum catches all single and double errors, all errors with an odd number of bits, all burst errors of length 16 or less, 99.997% of 17-bit error bursts, and 99.998% of 18-bit and longer bursts.