

Error Control Coding



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

Outline



- ◇ Introduction
- ◇ Error Detection Code - Cyclic Redundancy Code (CRC)
- ◇ Convolutional Code
- ◇ Linear Block Code

Introduction



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

Error Control Strategies



- ◇ Error control for a one-way system must be accomplished using *forward error correction (FEC)*, that is, by employing error-correcting codes that automatically correct errors detected at the receiver.
- ◇ Error control for a two-way system can be accomplished using error detection and retransmission, called *automatic repeat request (ARQ)*. This is also known as the *backward error correction (BEC)*.
 - ◇ In an ARQ system, when errors are detected at the receiver, a request is sent for the transmitter to repeat the message, and this continues until the message is received correctly.
- ◇ The major advantage of ARQ over FEC is that error detection requires much simpler decoding equipment than does error correction.

Error Control Strategies



- ◇ ARQ is adaptive in the sense that information is retransmitted only when errors occur.
- ◇ When the channel error rate is high, retransmissions must be sent too frequently, and the system throughput, the rate at which newly generated messages are correctly received, is lowered by ARQ.
- ◇ In general, wire-line communications (more reliable) adopts BEC scheme, while wireless communications (relatively unreliable) adopts FEC scheme.

Error Detection Code Cyclic Redundancy Code (CRC)



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

Error Detecting Codes



- ◇ Cyclic Redundancy Code (CRC Code) – also known 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



(b)

$$\begin{array}{r}
 \\
 \\
 110001 \overline{) 111000110 } \\
 \oplus \underline{11001} \\
 \\
 \oplus \underline{00000} \\
 \\
 \oplus \underline{11001} \\
 \\
 \oplus \underline{11001} \\
 \\
 \oplus \underline{00000} \\
 \\
 \oplus \underline{110} \\
 \\
 \oplus \underline{11} \\
 \\
 \oplus \underline{0} \\
 \\
 \underline{\underline{0000}}
 \end{array}$$

Remainder = 0: no errors

$$\begin{array}{r}
 \\
 \\
 110001 \overline{) 111000110 } \\
 \oplus \underline{11001} \\
 \\
 \oplus \underline{00000} \\
 \\
 \oplus \underline{11001} \\
 \\
 \oplus \underline{11001} \\
 \\
 \oplus \underline{00000} \\
 \\
 \oplus \underline{110} \\
 \\
 \oplus \underline{11} \\
 \\
 \oplus \underline{0} \\
 \\
 \underline{\underline{1001}}
 \end{array}$$

Error burst

Remainder \neq 0: error detected

Cyclic Redundancy Code (CRC)



- ◇ Examples of CRCs used in practice:

$$\text{CRC} - 16 = X^{16} + X^{15} + X^2 + 1$$

$$\text{CRC} - \text{CCITT} = X^{16} + X^{12} + X^5 + 1$$

$$\begin{aligned} \text{CRC} - 32 &= X^{32} + X^{26} + X^{23} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 \\ &\quad + X^5 + X^4 + X^{22} + X + 1 \end{aligned}$$

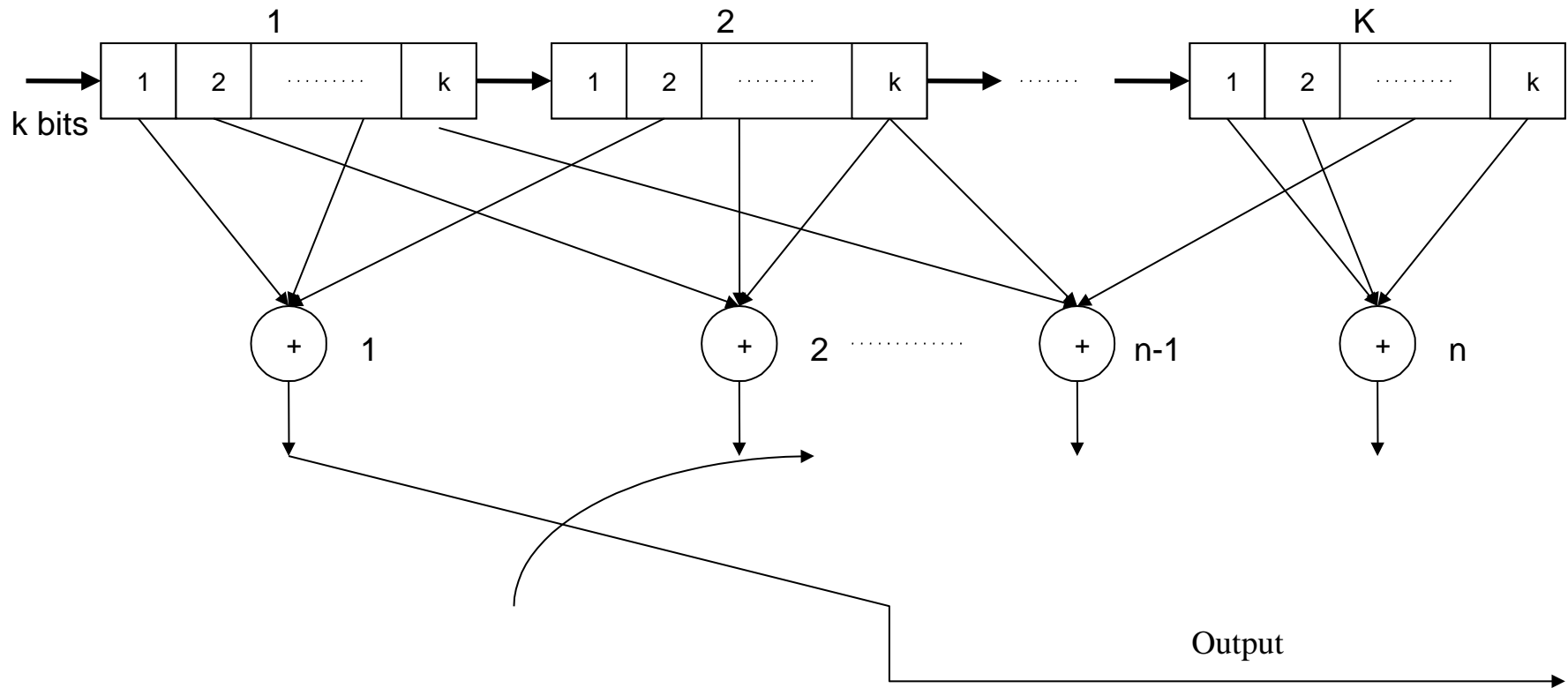
- ◇ A 16-bit checksum catches all single and double errors, all errors with an odd number of bits, all burst errors of length 16 or less, 99.997% of 17-bit error bursts, and 99.998% of 18-bit and longer bursts.

Convolutional Code



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

Structure of Convolutional Encoder



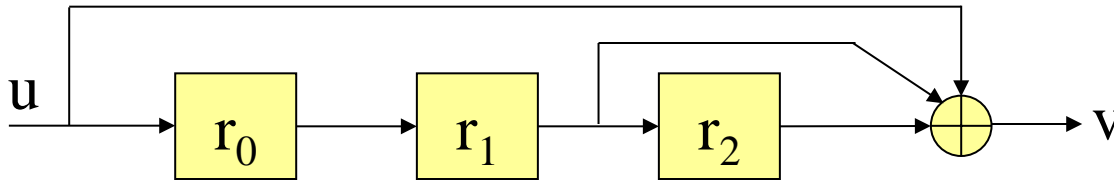
Convolutional Code



- ◇ 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

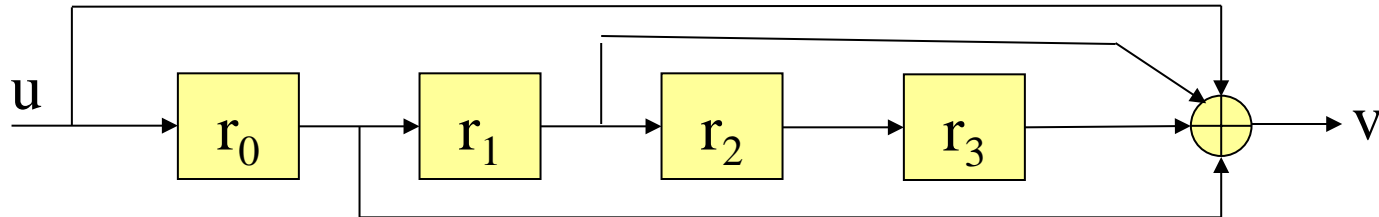
- ◇ Each encoded bit is a function of the present input bits and their past ones.

Generator Sequence



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

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

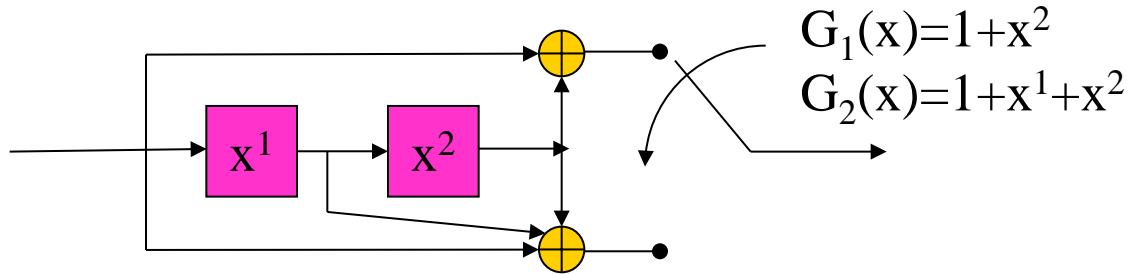


$$g_0^{(2)} = 1, g_1^{(2)} = 1, g_2^{(2)} = 1, g_3^{(2)} = 0, \text{ and } g_4^{(2)} = 1.$$

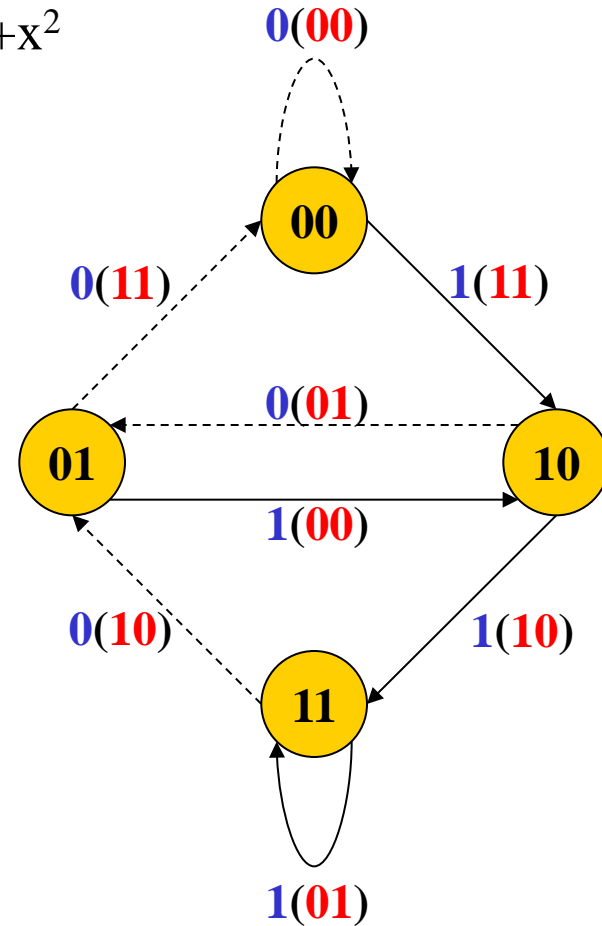
Generator Sequence: $g^{(2)} = (1 \ 1 \ 1 \ 0 \ 1)$

Convolutional Codes

An Example - (rate=1/2 with K=2)

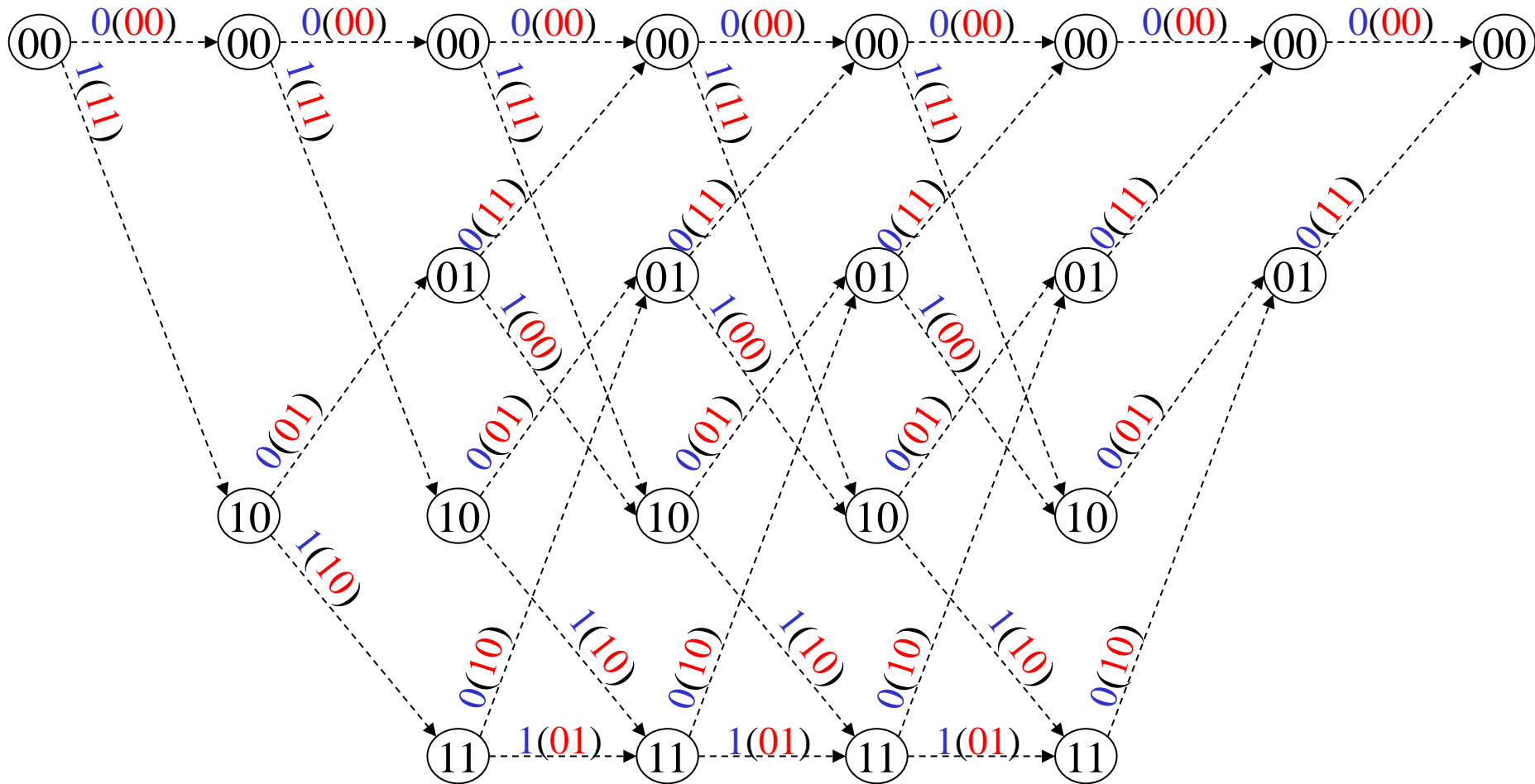


	Present	Next	Output
0	00	00	00
1	00	10	11
0	01	00	11
1	01	10	00
0	10	01	01
1	10	11	10
0	11	01	10
1	11	11	01



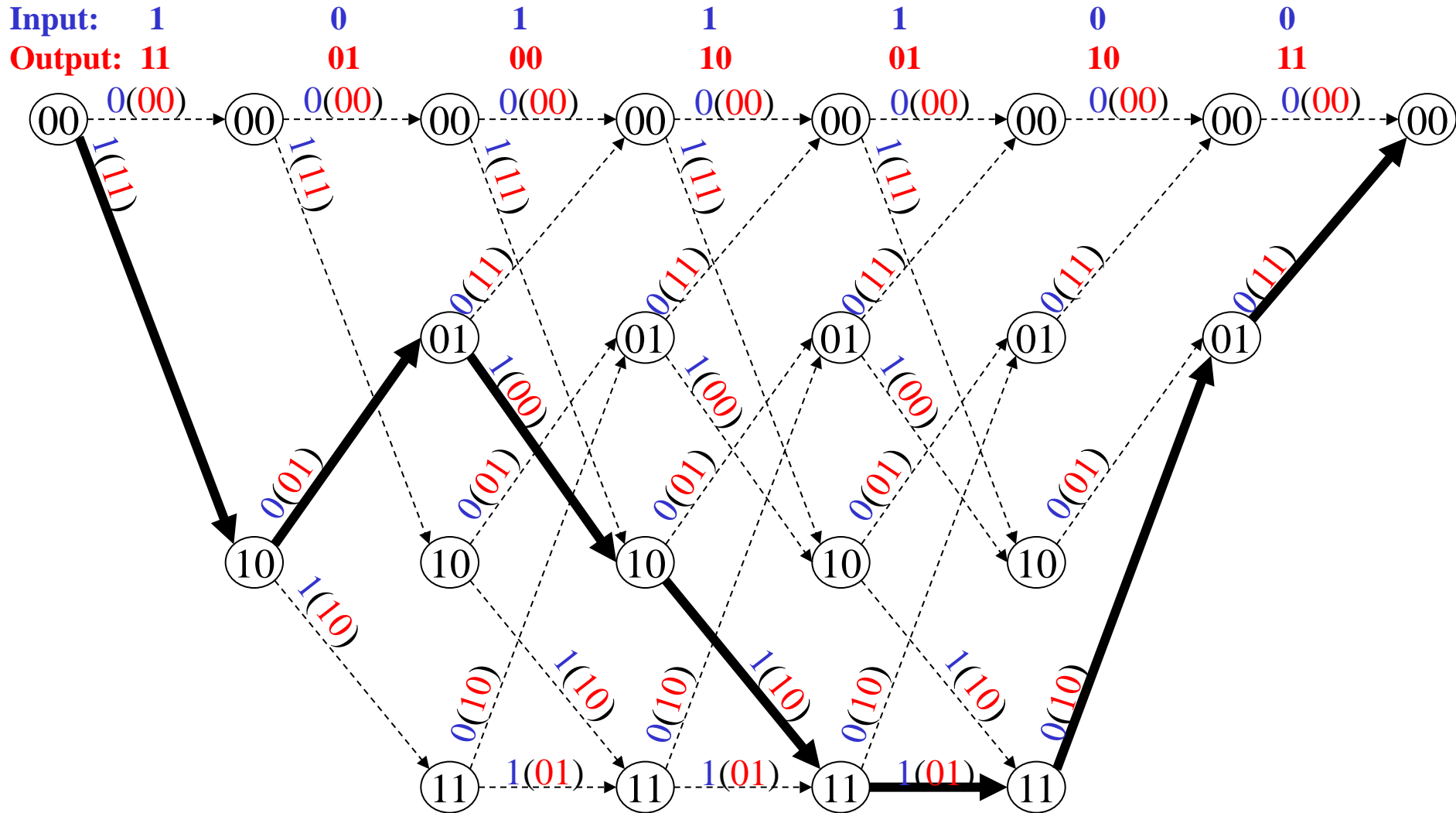
State Diagram

Trellis Diagram Representation



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

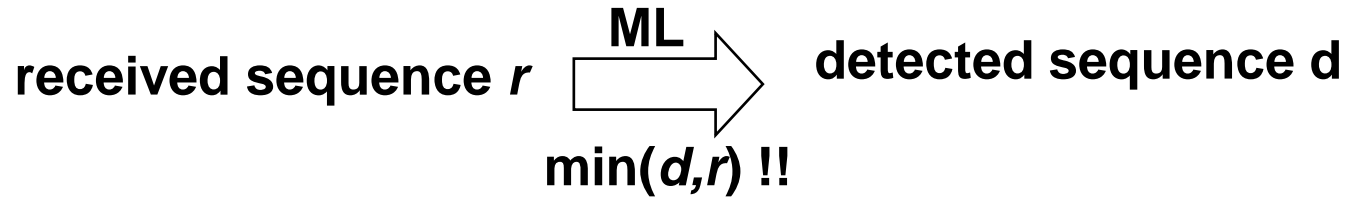
Encoding Process



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

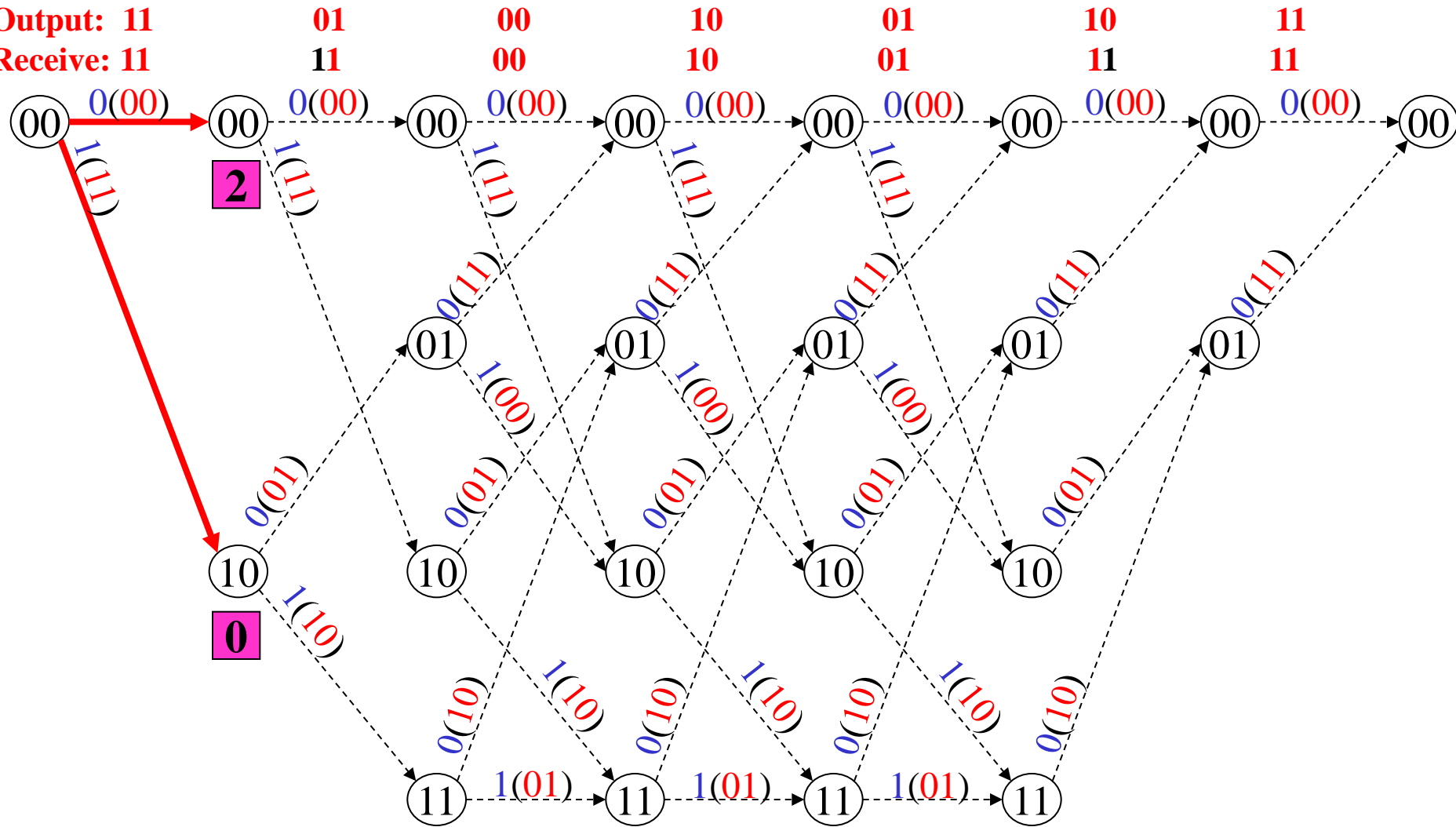


- ◇ Basic concept
 - ◇ Generate the code trellis at the decoder
 - ◇ The decoder penetrates through the code trellis level by level in search for the transmitted code sequence
 - ◇ At each level of the trellis, the decoder computes and compares the metrics of all the partial paths entering a node
 - ◇ 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



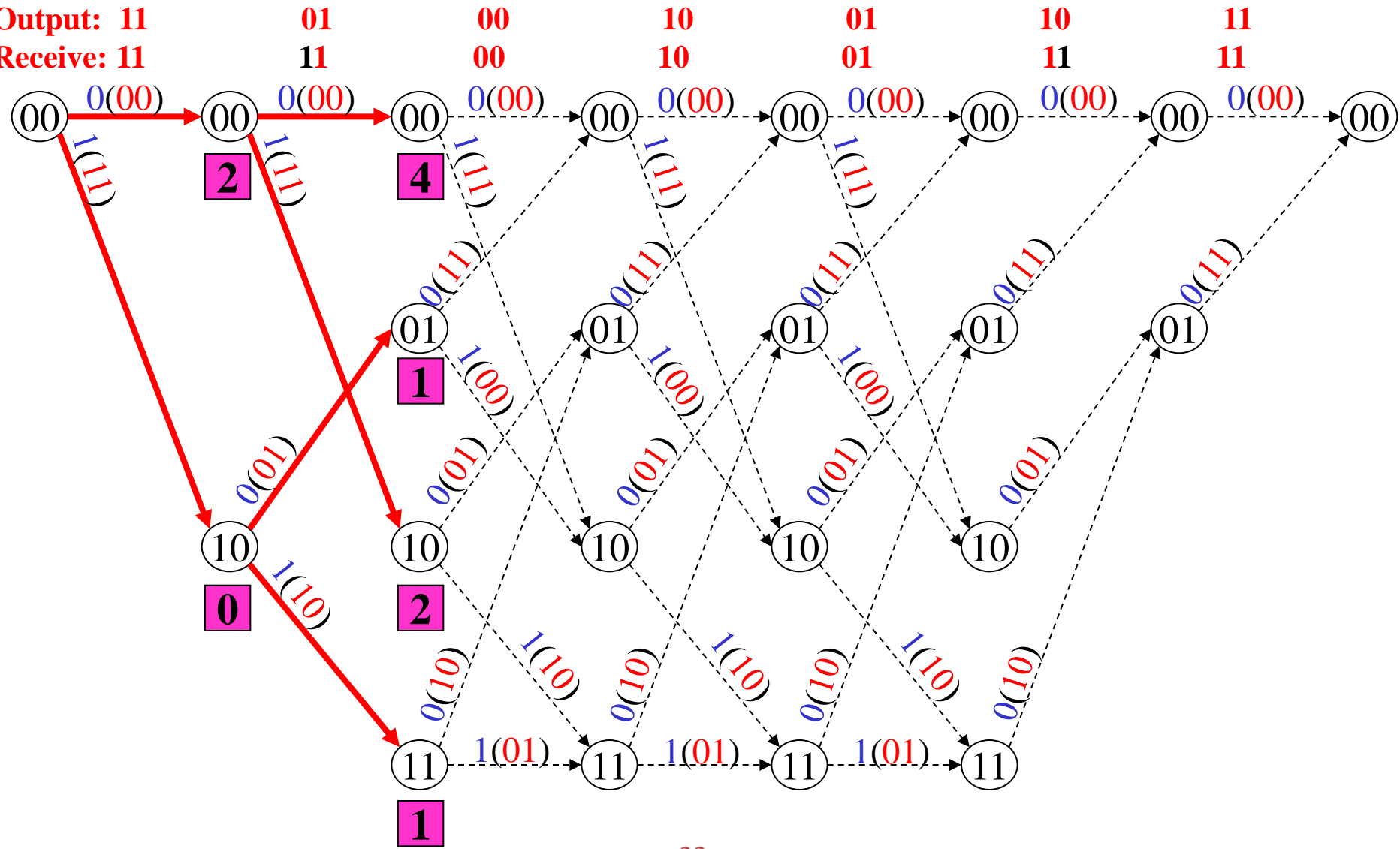
Output: 11
Receive: 11



Viterbi Decoding Algorithm



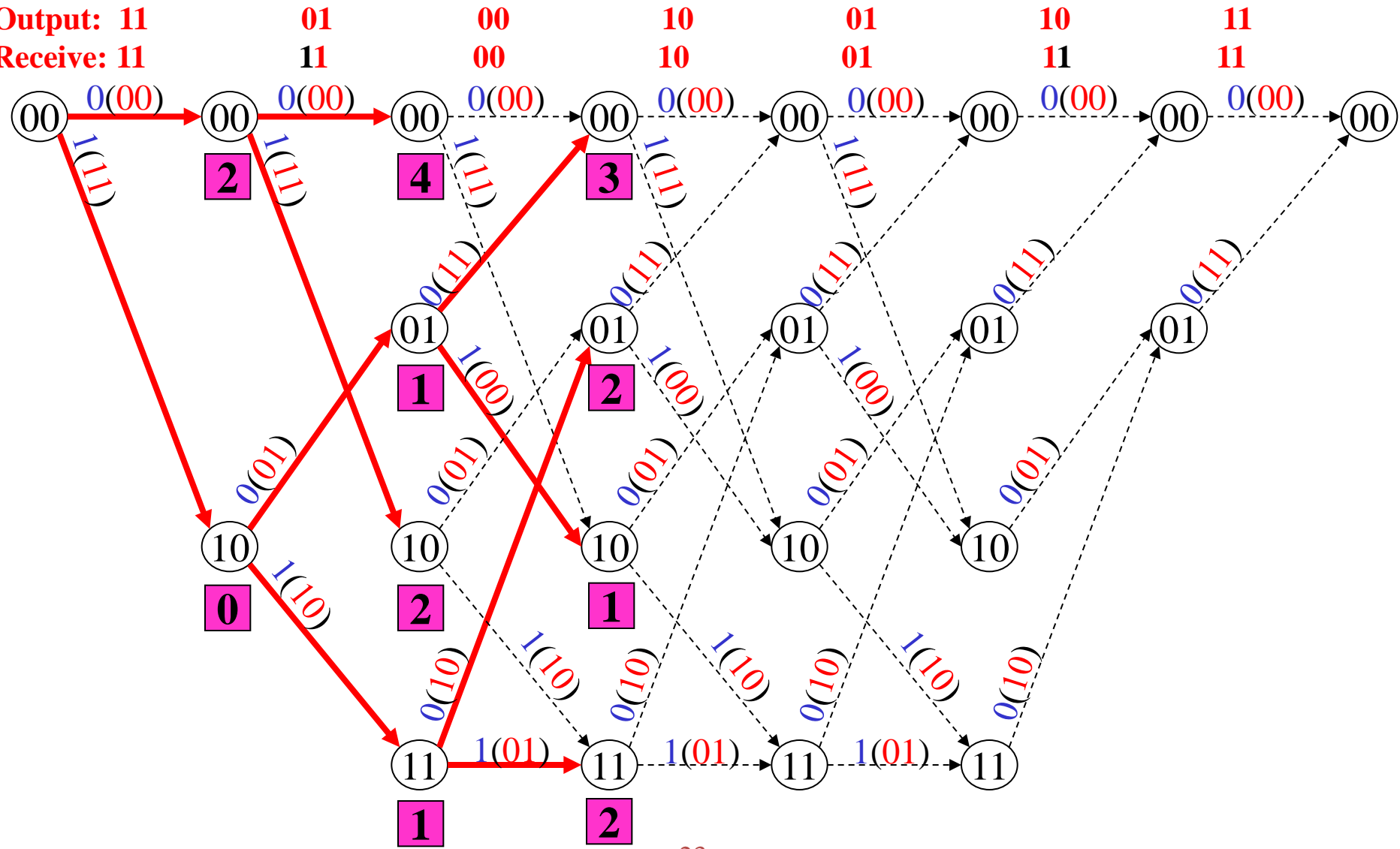
Output: 11
Receive: 11



Viterbi Decoding Algorithm



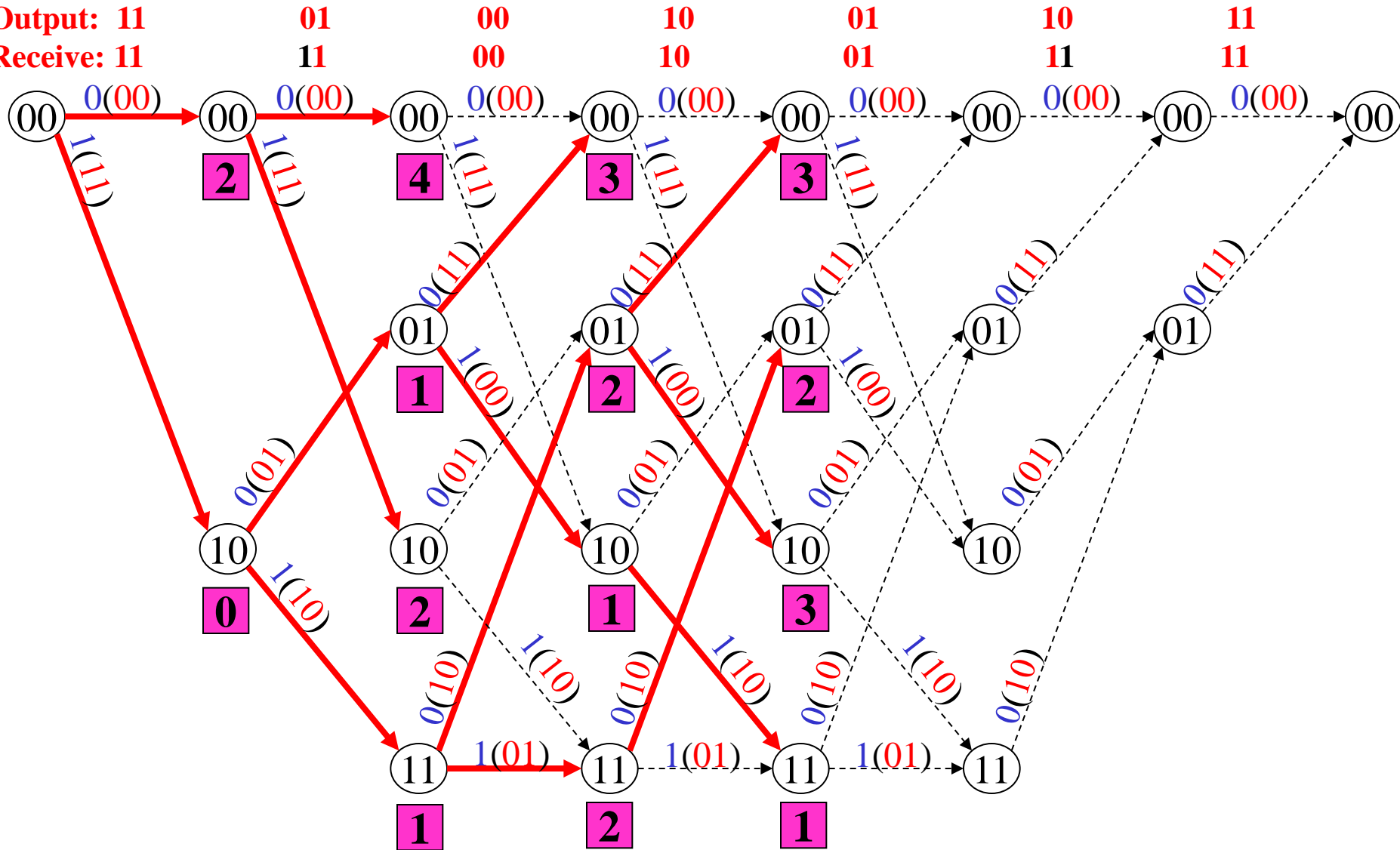
Output: 11
Receive: 11



Viterbi Decoding Algorithm



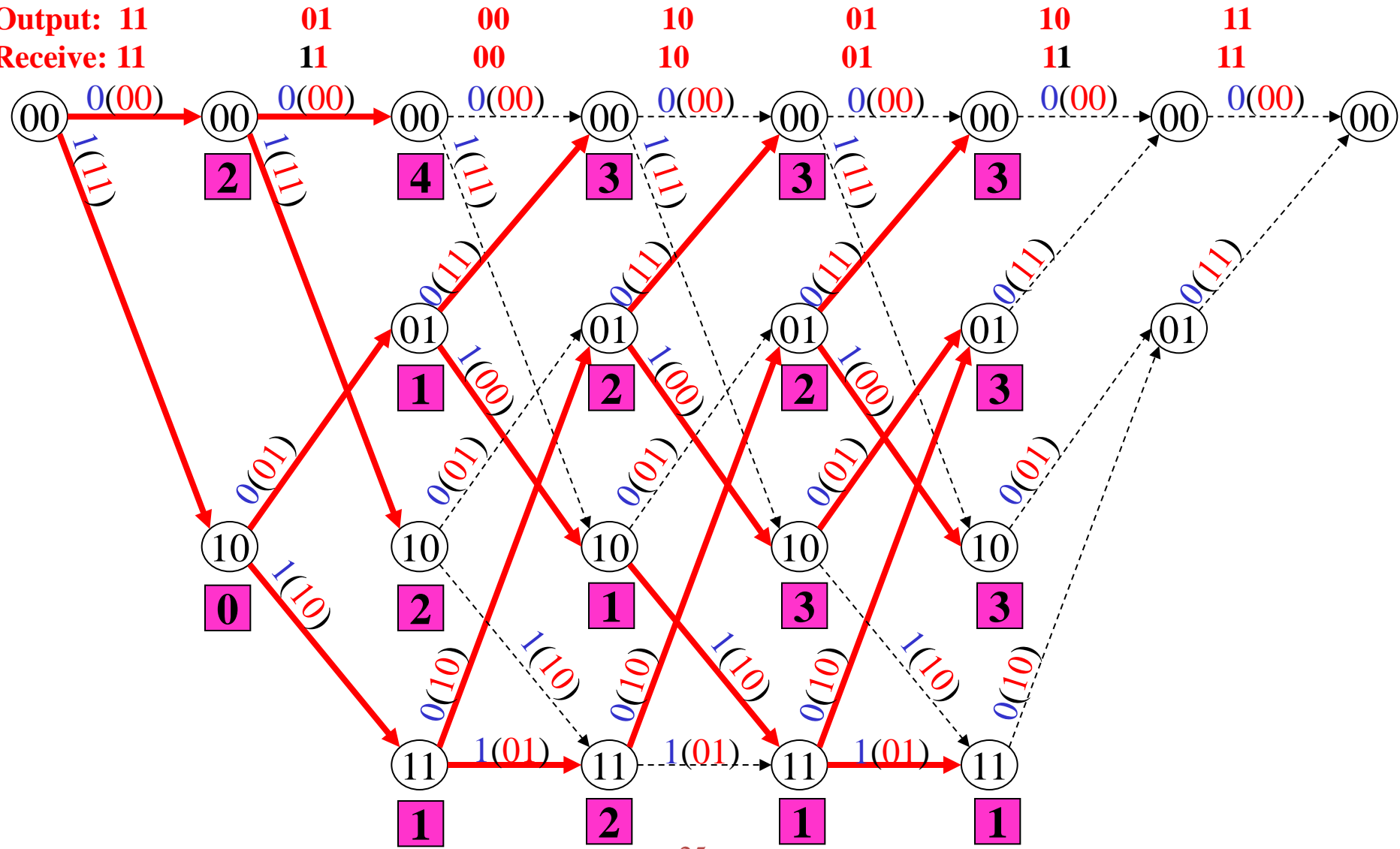
Output: 11
Receive: 11



Viterbi Decoding Algorithm



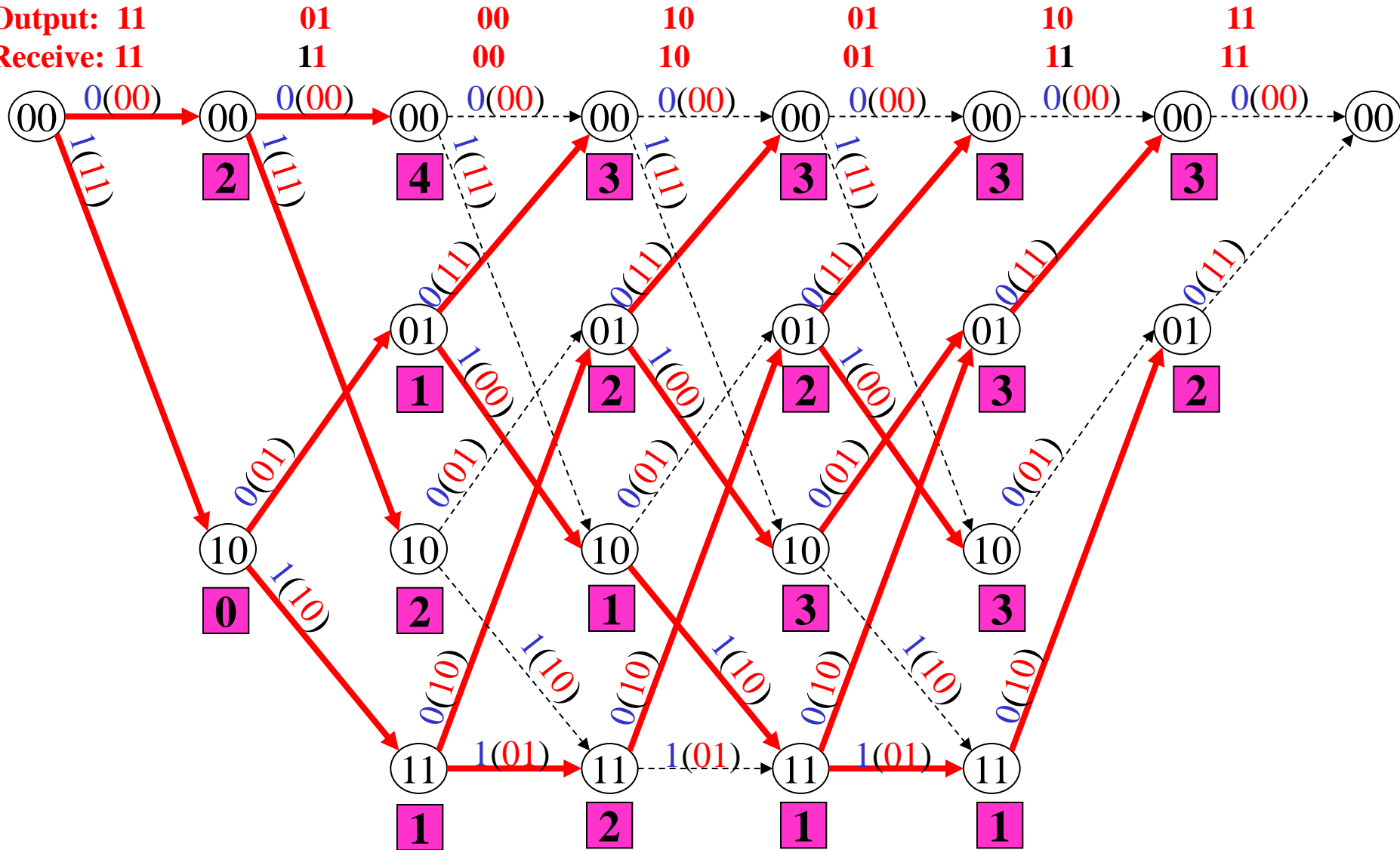
Output: 11
Receive: 11



Viterbi Decoding Algorithm



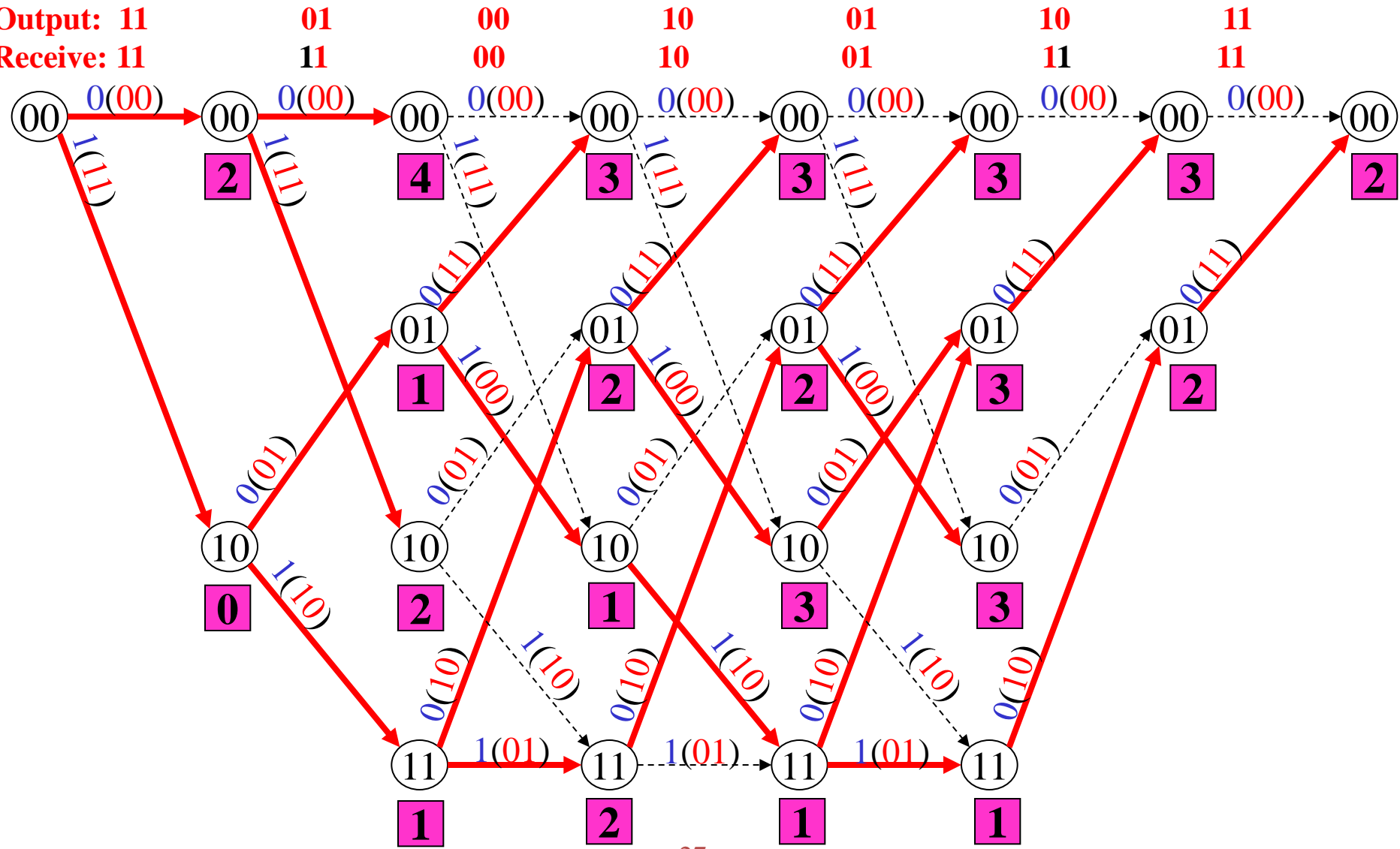
Output: 11
Receive: 11



Viterbi Decoding Algorithm



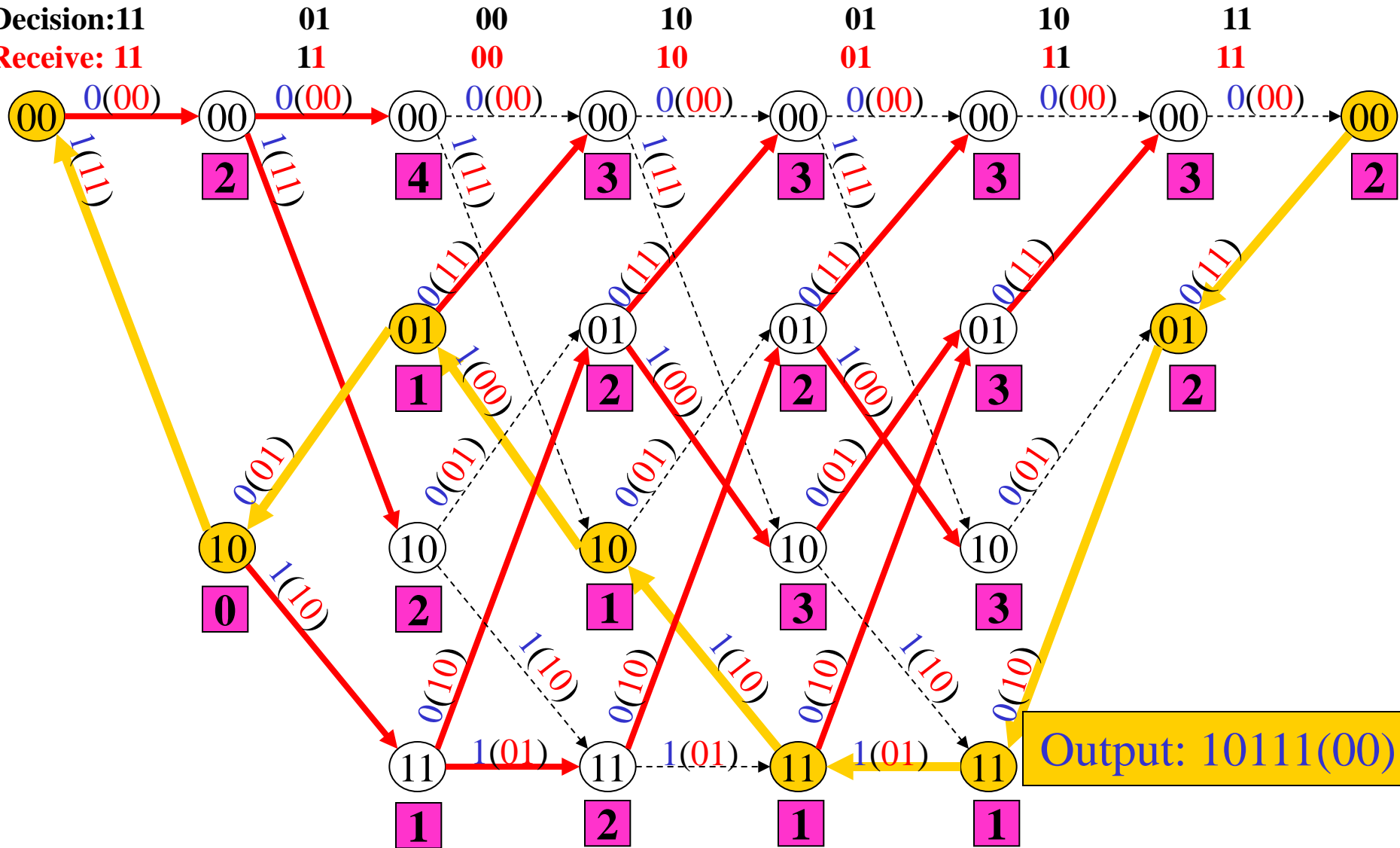
Output: 11
Receive: 11



Viterbi Decoding Algorithm



Decision: 11
 Receive: 11



Linear Block Code



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

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
 - ◇ $\mathbf{0}$ must be code word.

Generator Matrix



◇ 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, $\mathbf{g}_0, \mathbf{g}_1, \dots, \mathbf{g}_{k-1}$ in C

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

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

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

Generator Matrix



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

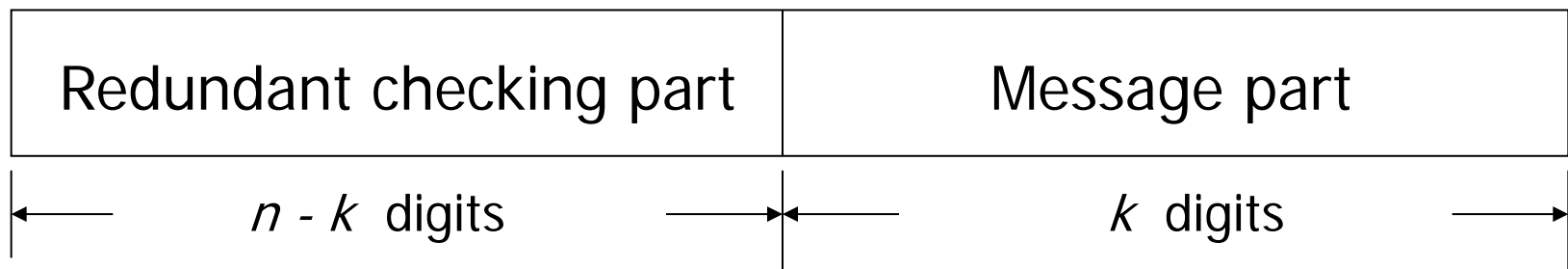
$$\mathbf{v} = \mathbf{u} \cdot \mathbf{G} = (u_0, u_1, \dots, u_{k-1}) \cdot \begin{bmatrix} \mathbf{g}_0 \\ \mathbf{g}_1 \\ \vdots \\ \mathbf{g}_{k-1} \end{bmatrix} = u_0 \mathbf{g}_0 + u_1 \mathbf{g}_1 + \dots + u_{k-1} \mathbf{g}_{k-1}$$

- ◇ 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.

Systematic Code



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



Systematic format of a code word

Parity Check Matrix



- ◇ For any $k \times n$ matrix \mathbf{G} with k linearly independent rows, there exists an $(n-k) \times n$ matrix \mathbf{H} with $n-k$ linearly independent rows such that any vector in the row space of \mathbf{G} is orthogonal to the rows of \mathbf{H} and any vector that is orthogonal to the rows of \mathbf{H} is in the *row space* of \mathbf{G} .
- ◇ An n -tuple \mathbf{v} is a code word in the code generated by \mathbf{G} if and only if $\mathbf{v} \cdot \mathbf{H}^T = 0$
- ◇ This matrix \mathbf{H} is called a *parity-check matrix* of the code
- ◇ The 2^{n-k} linear combinations of the rows of matrix \mathbf{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 \mathbf{G}
- ◇ C_d is called the *dual code* of C

Parity Check Matrix



- ◇ If the generator matrix of an (n,k) linear code is in the systematic form of (*) in page 34, the parity-check matrix may take the following form :

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

$$= \begin{bmatrix} 1 & 0 & 0 & \cdot & \cdot & \cdot & 0 & p_{00} & p_{10} & \cdot & \cdot & \cdot & p_{k-1,0} \\ 0 & 1 & 0 & \cdot & \cdot & \cdot & 0 & p_{01} & p_{11} & \cdot & \cdot & \cdot & p_{k-1,1} \\ 0 & 0 & 1 & \cdot & \cdot & \cdot & 0 & p_{02} & p_{12} & \cdot & \cdot & \cdot & p_{k-1,2} \\ \cdot & & & & & & & & & & & & \\ \cdot & & & & & & & & & & & & \\ \cdot & & & & & & & & & & & & \\ 0 & 0 & 0 & \cdot & \cdot & \cdot & 1 & p_{0,n-k-1} & p_{1,n-k-1} & \cdot & \cdot & \cdot & p_{k-1,n-k-1} \end{bmatrix}$$

Parity Check Matrix



- ◇ Let \mathbf{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 $\mathbf{G} \cdot \mathbf{H}^T = \mathbf{0}$

Syndrome



- ◇ The generator matrix \mathbf{G} is used in the encoding operation at the transmitter. On the other hand, the parity-check matrix \mathbf{H} is used in the decoding operation at the receiver.

- ◇ Let ,

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

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

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

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

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

Syndrome



- ◇ The receiver has to decode the code vector \mathbf{c} from the received \mathbf{r} .
- ◇ The algorithm commonly used to perform this decoding operation starts 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 \mathbf{r} , the syndrome is defined as

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

- ◇ $\mathbf{s}=\mathbf{0}$ if and only if \mathbf{r} is a code word and receiver accepts \mathbf{r} as the transmitted code word.
- ◇ $\mathbf{s}\neq\mathbf{0}$ if and only if \mathbf{r} is not a code word and presence of errors has been detected.
- ◇ When the error pattern \mathbf{e} is identical to a nonzero code word (i.e., \mathbf{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



- ◇ 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

$$\begin{aligned} \mathbf{s} &= (\mathbf{c} + \mathbf{e})\mathbf{H}^T \\ &= \mathbf{cH}^T + \mathbf{eH}^T \\ \mathbf{cH}^T &= \mathbf{0} \quad \left\{ \begin{array}{l} = \mathbf{cH}^T + \mathbf{eH}^T \\ = \mathbf{eH}^T \end{array} \right. \\ &= (s_0, s_1, \dots, s_{n-k-1}) \end{aligned} \quad (10.81)$$

- ◇ Hence, the parity-check matrix \mathbf{H} of a code permits us to compute the syndrome \mathbf{s} , which depends only upon the error pattern \mathbf{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 \mathbf{c}_i , $i=1, 2, \dots, 2^k$. For any error pattern \mathbf{e} , we define the 2^k distinct vectors \mathbf{e}_i as

$$\mathbf{e}_i = \mathbf{e} + \mathbf{c}_i, \quad i = 1, \dots, 2^k \quad (10.82)$$

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

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

which is independent of index i .

Syndrome decoding - I



- ◇ The set of vectors $\{e_i, i=1, 2, \dots, 2^k\}$ 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 \mathbf{s} as:

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

$$s_1 = e_0 + e_{n-k} p_{00} + e_{n-k+1} p_{10} + \dots + e_{n-1} p_{k-1,1}$$

$$s_2 = e_1 + e_{n-k} p_{01} + e_{n-k+1} p_{11} + \dots + e_{n-1} p_{k-1,2}$$

$$\vdots$$

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

(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 \mathbf{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 \mathbf{s} reduce the search for the true error pattern \mathbf{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 \mathbf{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

$$d_{\min} = \min \{ d(\mathbf{c}_1, \mathbf{c}_2) : \mathbf{c}_1, \mathbf{c}_2 \in C, \mathbf{c}_1 \neq \mathbf{c}_2 \}$$
- ◇ 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

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

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 \mathbf{H} of the code.
- ◇ Let the matrix \mathbf{H} be expressed in terms of its columns as follows:

$$\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, \dots, \mathbf{h}_n] \quad (10.85)$$


- ◇ From Eq.(10.77) we get

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

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

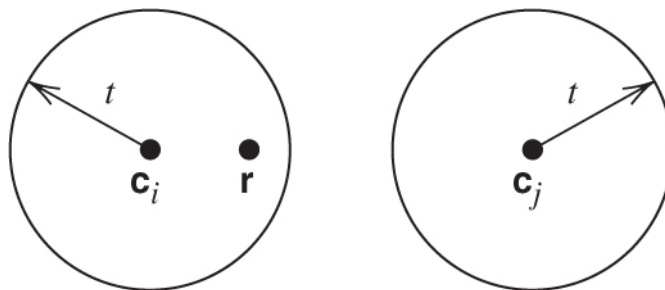
- ◇ Because $d_{\min} = w_{\min}$, the smallest Hamming weight equals the minimum distance of the code. Hence, *the minimum distance of a linear block code is defined by the minimum number of rows of the matrix \mathbf{H}^T whose sum is equal to the zero vector.*

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 $w(\mathbf{e})\leq t$.
- 
- ◇ 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} .

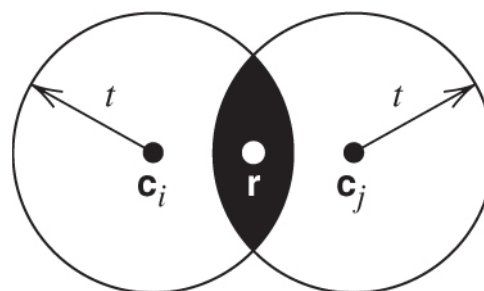


(a)

Figure 10.23

Minimum Distance Consideration

- ◇ Let these two spheres 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

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

- ◇ 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,

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

Syndrome Decoding - II



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

- ◇ Let $\mathbf{c}_1, \mathbf{c}_2, \dots, \mathbf{c}_{2^k}$ 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, \dots, D_{2^k} , D_i is the i th 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 $\mathbf{e}_2, \dots, \mathbf{e}_{2^{n-k}}$ are called *coset leaders*.

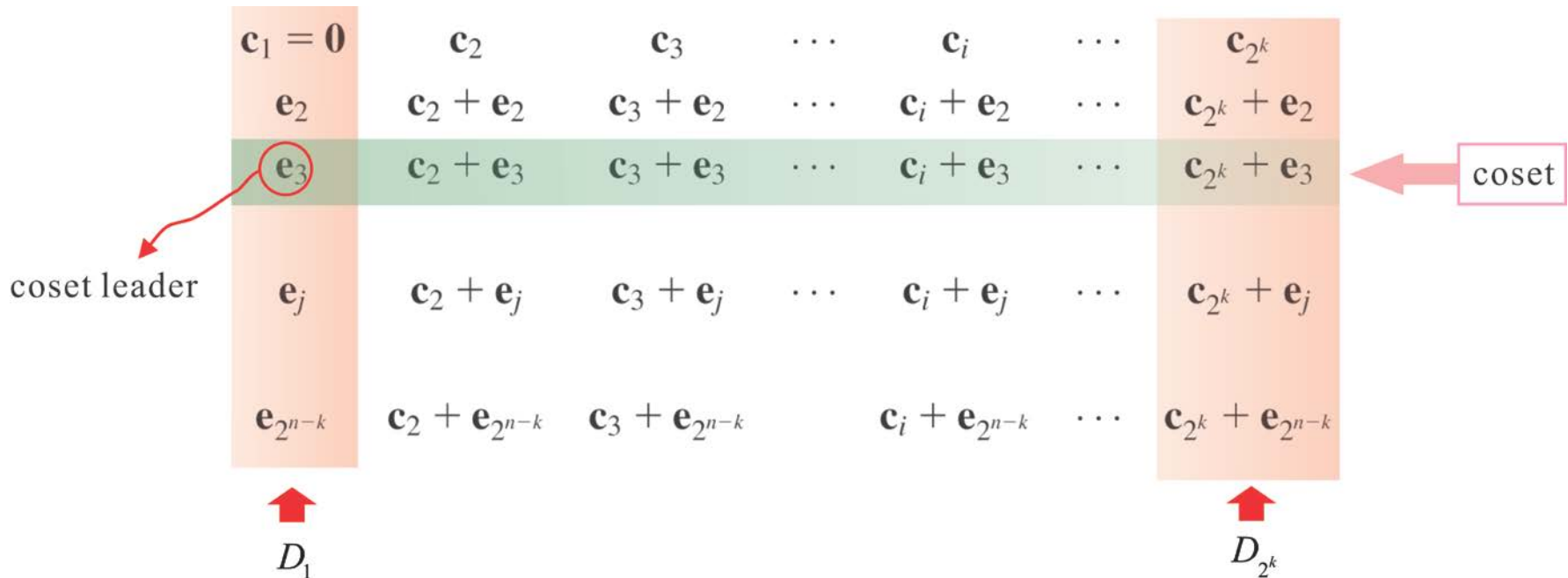


Figure 10.24 Stand array for an (n, k) block code

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 \mathbf{r} , compute the syndrome $\mathbf{s}=\mathbf{r}\mathbf{H}^T$.
2. Within the coset characterized by the syndrome \mathbf{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
$$\mathbf{c} = \mathbf{r} + \mathbf{e}_0 \quad (10.87)$$

as the decoded version of the received vector \mathbf{r} .