Coding and Error Control

Chapter 8

Coping with Data Transmission Errors

- Error detection codes
  - Detects the presence of an error
- Automatic repeat request (ARQ) protocols
  - Block of data with error is discarded
  - Transmitter retransmits that block of data
- Error correction codes, or forward correction codes (FEC)
  - Designed to detect and correct errors

Error Detection Probabilities

- Definitions
  - P_b : Probability of single bit error (BER)
  - P_1 : Probability that a frame arrives with no bit errors
  - P_2 : While using error detection, the probability that a frame arrives with one or more undetected errors
  - P_3 : While using error detection, the probability that a frame arrives with one or more detected bit errors but no undetected bit errors

Error Detection Probabilities

- With no error detection

$$P_1 = (1 - P_b)^F$$

$$P_2 = 1 - P_1$$

$$P_3 = 0$$

  - F = Number of bits per frame

Example

- BER=10^{-6}
- Frame length 1000-bit on a continuously used 64Kbps communication channel
- User requires that the there can be a maximum of 1 frame error per day. Can this line be used?
- $64000*60*60*24/1000=5.529*10^{6}$ packets / day
- Probability of 1 packet being faulty
- $P_2=1/5.529*10^{6}=\mathbf{0.18 * 10^{-6}}$
  - **requirement we need to meet**
- Since $P_b=10^{-6}$, then $P_1=(1-P_b)^F=0.999$
- $P_2=1-P_1=\mathbf{10^{-3}}$
  - **offered by the system**

Finding and Fixing Errors

- The main purpose of error detection and correction is to create ”redundant bits" in the data to improve transmission performance.
- Redundant bits increase the amount of data sent and thus increase the bandwidth requirement.
- Reduces bandwidth efficiency for high SNR
- Provides very good communication performance for low SNR

Error Detection Process

- Transmitter
  - For a given frame, an error-detecting code (check bits) is calculated from data bits
  - Check bits are appended to data bits
- Receiver
  - Separates incoming frame into data bits and check bits
  - Calculates check bits from received data bits
  - Compares calculated check bits against received check bits
  - Detected error occurs if mismatch

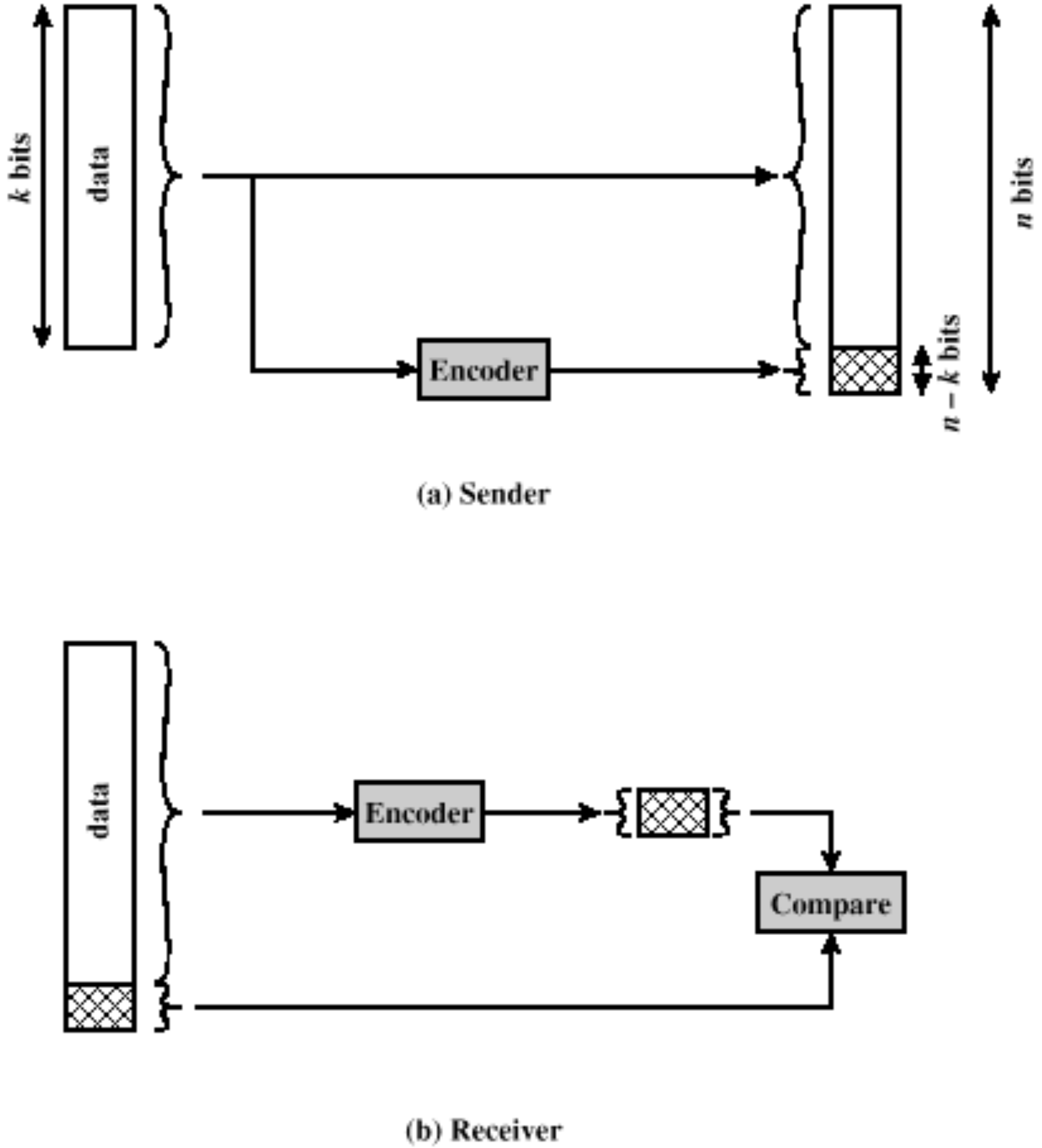


Figure 8.1 Error Detection Process

Parity Check

- Parity bit appended to a block of data
- Even parity
  - Added bit ensures an even number of 1s
- Odd parity
  - Added bit ensures an odd number of 1s
- Example, 7-bit character [1110001]
  - Even parity [11100010]
  - Odd parity [11100011]
- Cannot find the error if there are 2 erroneous bits

Cyclic Redundancy Check (CRC)

- Transmitter
  - For a *k*-bit block, transmitter generates an (*n*-*k*)-bit frame check sequence (FCS)
  - Resulting frame of *n* bits is exactly divisible by predetermined number
- Receiver
  - Divides incoming frame by predetermined number
  - If no remainder, assumes no error

Modulo 2 Arithmetic

- Addition (XOR process)
  (X) 10110100

(Y) $\underline{00101010}$ +

(Z) 10011110

CRC using Modulo 2 Arithmetic

- Exclusive-OR (XOR) operation
- Parameters:
  - T = n-bit frame to be transmitted
  - D = k-bit block of data; the first k bits of T
  - F = $(n – k)$-bit FCS; the last $(n – k)$ bits of T
  - P = pattern of $n–k+1$ bits; this is the predetermined divisor
  - Q = Quotient
  - R = Remainder

CRC using Modulo 2 Arithmetic

- For *T*/*P* to have no remainder, start with

$$T = 2^{n-k} D + F$$

- Divide $2^{n\text{-}k}D$ by *P* gives quotient and remainder

$$\frac{2^{n-k} D}{P} = Q + \frac{R}{P}$$

- Use remainder as FCS

$$T = 2^{n-k} D + R$$

CRC using Modulo 2 Arithmetic

- Does *R* cause *T*/*P* have no remainder?

$$\frac{T}{P} = \frac{2^{n-k}D + R}{P} = \frac{2^{n-k}D}{P} + \frac{R}{P}$$

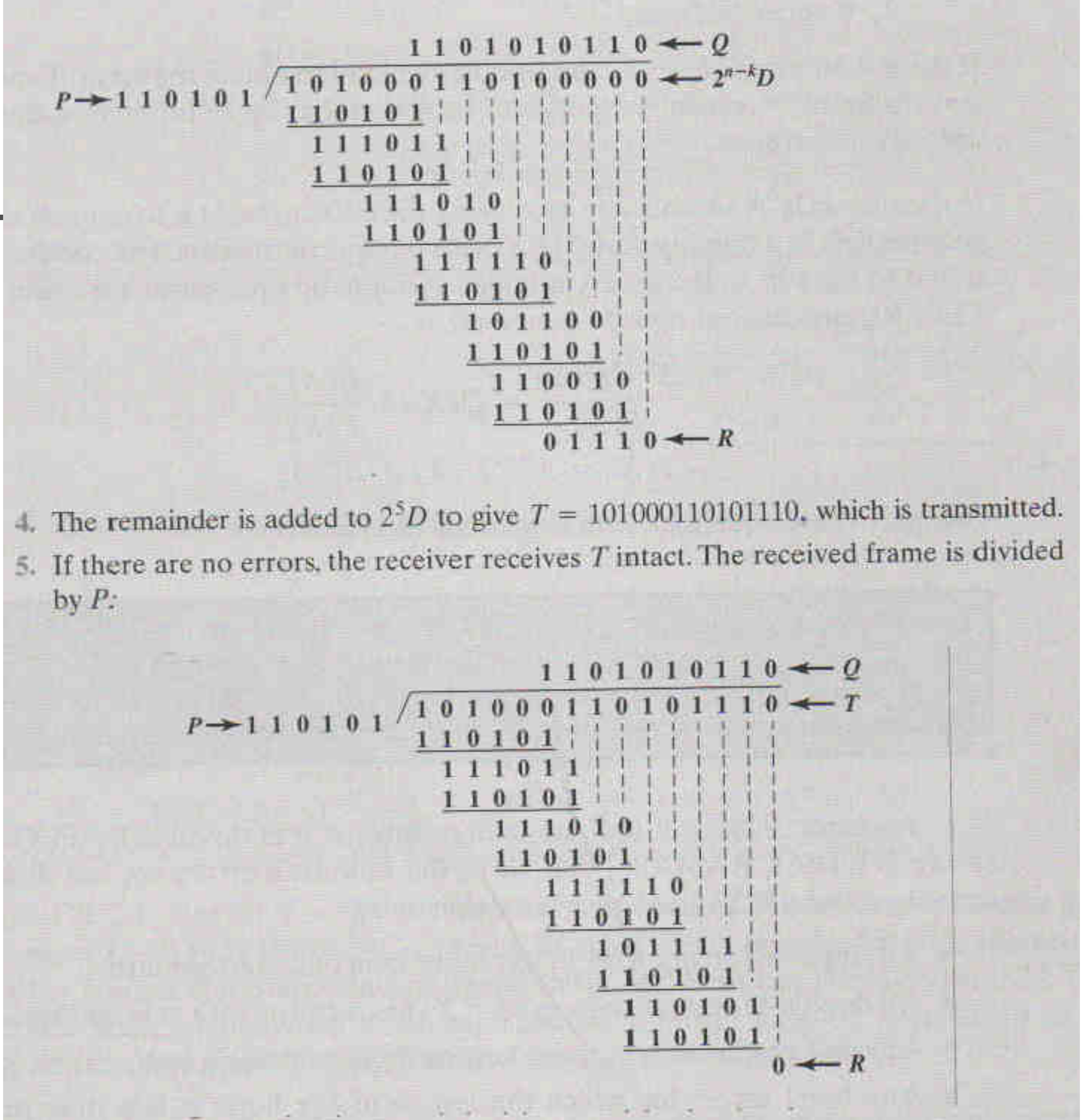
- Substituting,

$$\frac{T}{P} = Q + \frac{R}{P} + \frac{R}{P} = Q + \frac{R+R}{P} = Q$$

  - No remainder, so *T* is exactly divisible by *P*

Example

- Message D = 1010001101 (10-bit)
- Pattern P = 110101 (6-bit)
- FCS R = ?
- *Since* P *is* 6-bit, R must be 5-bit. n = 15, k = 10 and ($n-k$) = 5
- First, the message is multiplied by D 2^5
- 100000 x 1010001101 = 101000110100000



```
                            1 1 0 1 0 1 0 1 1 0 ← Q
P → 1 1 0 1 0 1 ) 1 0 1 0 0 0 1 1 0 1 0 0 0 0 0 ← 2^{n-k}D
                  1 1 0 1 0 1
                    1 1 1 0 1 1
                    1 1 0 1 0 1
                        1 1 1 0 1 0
                        1 1 0 1 0 1
                            1 1 1 1 1 0
                            1 1 0 1 0 1
                                1 0 1 1 0 0
                                1 1 0 1 0 1
                                  1 1 0 0 1 0
                                  1 1 0 1 0 1
                                      0 1 1 1 0 ← R
```

4. The remainder is added to 2^5D to give T = 101000110101110, which is transmitted.

5. If there are no errors, the receiver receives T intact. The received frame is divided by P:

```
                            1 1 0 1 0 1 0 1 1 0 ← Q
P → 1 1 0 1 0 1 ) 1 0 1 0 0 0 1 1 0 1 0 1 1 1 0 ← T
                  1 1 0 1 0 1
                    1 1 1 0 1 1
                    1 1 0 1 0 1
                        1 1 1 0 1 0
                        1 1 0 1 0 1
                            1 1 1 1 1 0
                            1 1 0 1 0 1
                                1 0 1 1 1 1
                                1 1 0 1 0 1
                                  1 1 0 1 0 1
                                  1 1 0 1 0 1
                                            0 ← R
```

CRC using Polynomials

- All values expressed as polynomials
  - Dummy variable *X* with binary coefficients

$$\frac{X^{n-k}D(X)}{P(X)} = Q(X) + \frac{R(X)}{P(X)}$$

$$T(X) = X^{n-k}D(X) + R(X)$$

Example

- Message *D* = 1010001101 → (x)=$x^9+x^7+x^3+x^2+1$
- Pattern *P* = 110101 → $P(x)=x^5+x^4+x^2+1$
- FCS *R* = 01110 → $R(x)=x^3+x^2+x$

$$
\begin{array}{rcccccccccccccccl}
 & & & & & & X^9 & +X^8 & & +X^6 & & +X^4 & & +X^2 & +X & \leftarrow Q(X) \\
\cline{2-15}
P(X) \longrightarrow X^5+X^4+X^2+1 \,\big) & X^{14} & & +X^{12} & & & & +X^8 & +X^7 & & +X^5 & & & & & \leftarrow X^5D(X) \\
 & X^{14} & +X^{13} & & +X^{11} & & +X^9 & & & & & & & & & \\
\cline{2-7}
 & & X^{13} & +X^{12} & +X^{11} & & +X^9 & +X^8 & & & & & & & & \\
 & & X^{13} & +X^{12} & & +X^{10} & & +X^8 & & & & & & & & \\
\cline{3-8}
 & & & & X^{11} & +X^{10} & +X^9 & & +X^7 & & & & & & & \\
 & & & & X^{11} & +X^{10} & & +X^8 & & +X^6 & & & & & & \\
\cline{5-10}
 & & & & & & X^9 & +X^8 & +X^7 & +X^6 & +X^5 & & & & & \\
 & & & & & & X^9 & +X^8 & & +X^6 & & +X^4 & & & & \\
\cline{7-12}
 & & & & & & & & X^7 & & +X^5 & +X^4 & & & & \\
 & & & & & & & & X^7 & +X^6 & & +X^4 & & +X^2 & & \\
\cline{9-14}
 & & & & & & & & & X^6 & +X^5 & & & +X^2 & & \\
 & & & & & & & & & X^6 & +X^5 & & +X^3 & & +X & \\
\cline{10-15}
 & & & & & & & & & & & & X^3 & +X^2 & +X & \leftarrow R(X)
\end{array}
$$

CRC using Polynomials

- Widely used versions of *P(X)*
  - CRC–12
    - $X^{12} + X^{11} + X^3 + X^2 + X + 1$
  - CRC–16
    - $X^{16} + X^{15} + X^2 + 1$
  - CRC – CCITT
    - $X^{16} + X^{12} + X^5 + 1$
  - CRC – 32
    - $X^{32} + X^{26} + X^{23} + X^{22} + X^{16} + X^{12} + X^{11} + X^{10} + X^8 + X^7 + X^5 + X^4 + X^2 + X + 1$

CRC using Digital Logic

- Dividing circuit consisting of:
  - XOR gates
    - Up to $n - k$ XOR gates
    - Presence of a gate corresponds to the presence of a term in the divisor polynomial $P(X)$
  - A shift register
    - String of 1-bit storage devices
    - Register contains $n - k$ bits, equal to the length of the FCS

Digital Logic CRC

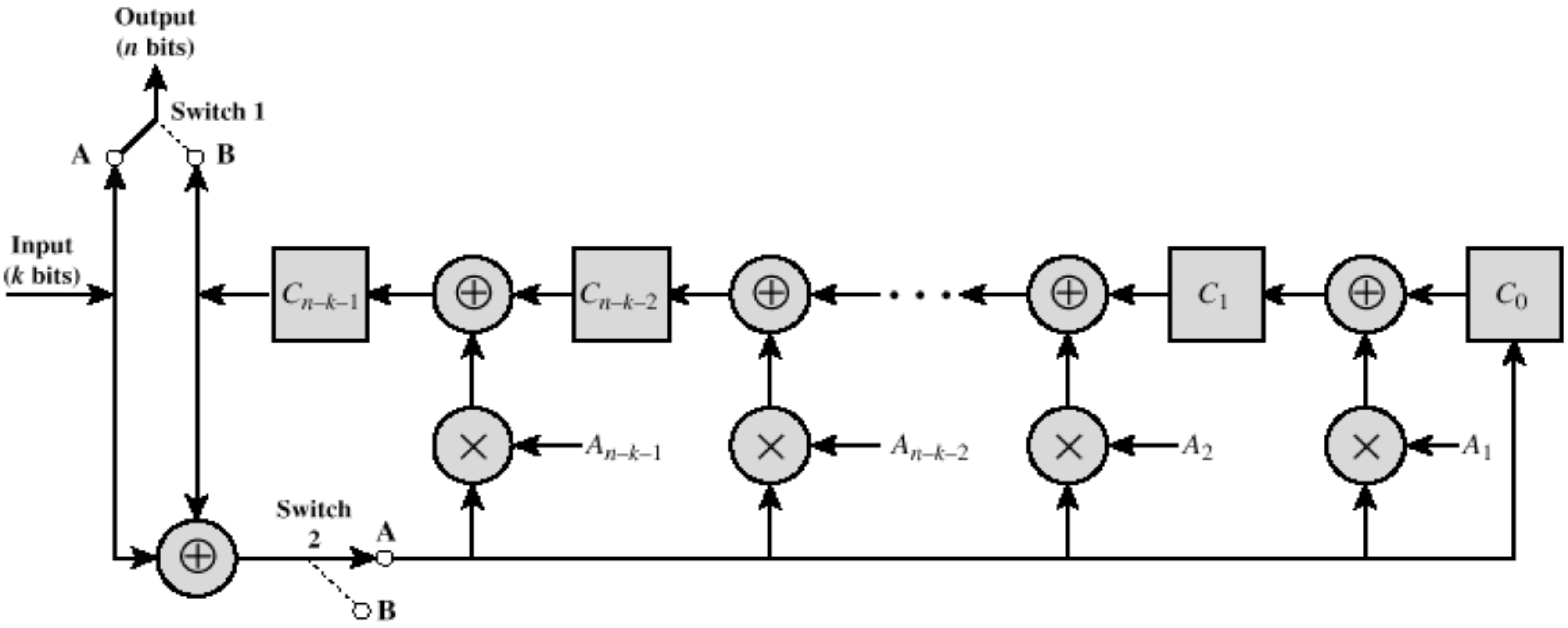


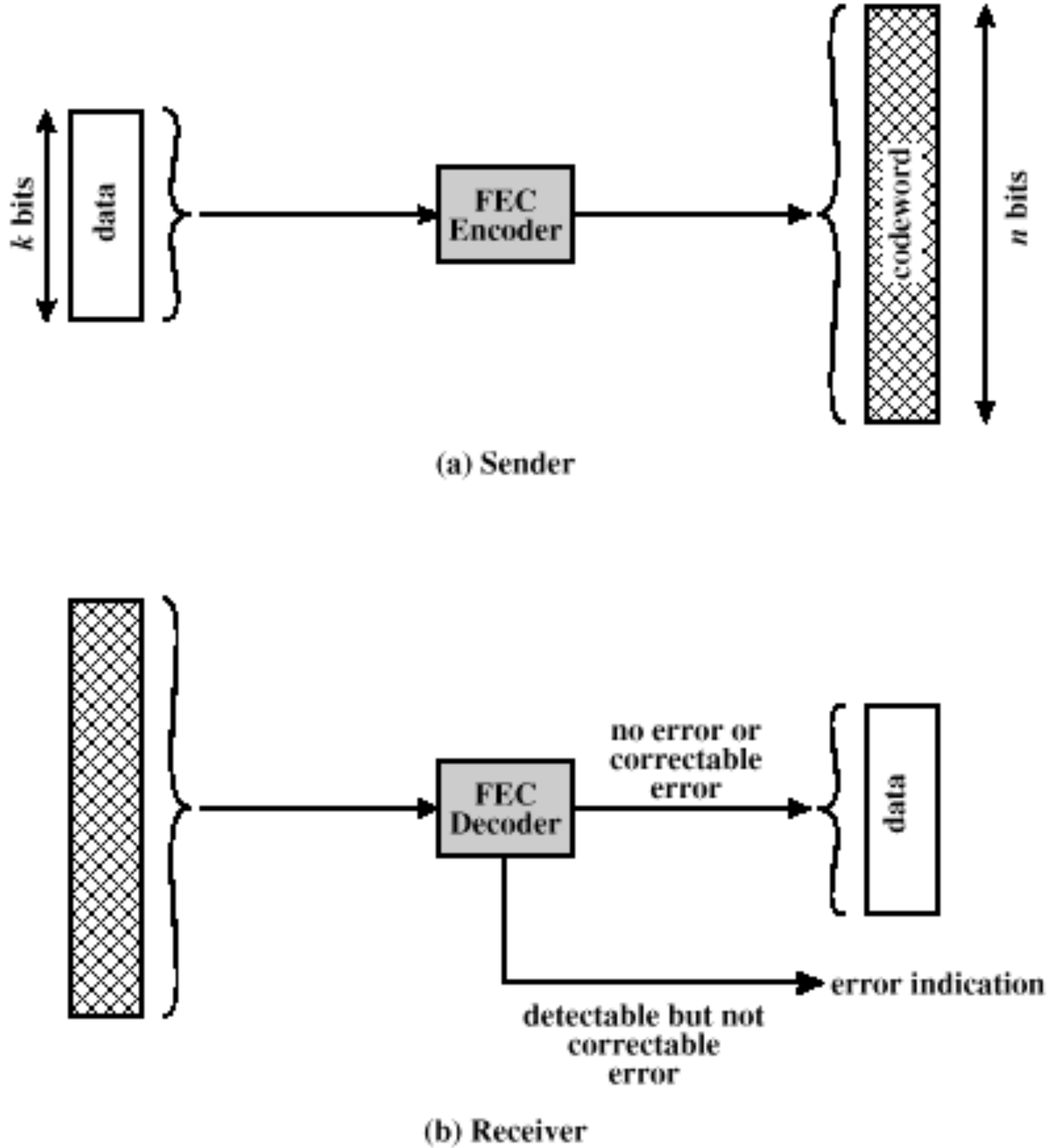
Figure 8.4 General CRC Architecture to Implement Divisor
$1 + A_1X + A_2X^2 + \ldots + A_{n-1}X^{n-k-1} + X^{n-k}$

Wireless Transmission Errors

- Error detection requires retransmission
- Detection inadequate for wireless applications
  - Error rate on wireless link can be high, results in a large number of retransmissions
  - Long propagation delay compared to transmission time

Block Error Correction Codes

- Transmitter
  - Forward error correction (FEC) encoder maps each k-bit block into an n-bit block codeword
  - Codeword is transmitted; analog for wireless transmission
- Receiver
  - Incoming signal is demodulated
  - Block passed through an FEC decoder



SS

Figure 8.5 Forward Error Correction Process

FEC Decoder Outcomes

- No errors present
  - Codeword produced by decoder matches original codeword
- Decoder detects and corrects bit errors
- Decoder detects but cannot correct bit errors; reports uncorrectable error
- Decoder detects no bit errors, though errors are present

Block Code Principles

- Hamming distance – for 2 *n*-bit binary sequences, the number of different bits
  - E.g., v_1=011011; v_2=110001; d(v1, v_2)=3
- Redundancy – ratio of redundant bits to data bits
- Code rate – ratio of data bits to total bits
- Coding gain – the reduction in the required E_b/N_0 to achieve a specified BER of an error-correcting coded system

Block Code Principles

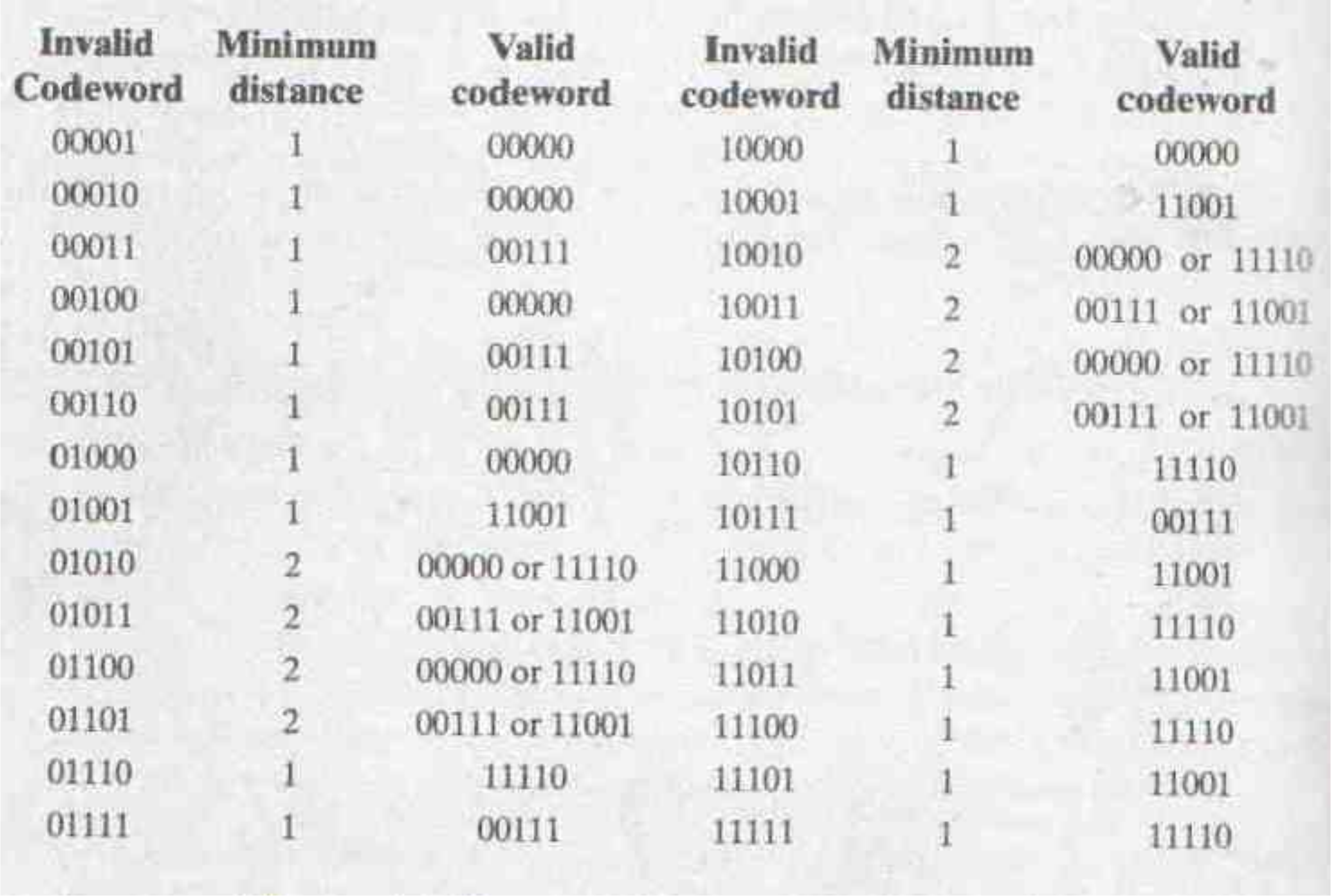
- For k = 2 and n = 5 the following mapping can be made

Data Block	**Code Block**
00	00000
01	00111
10	11001
11	11110

- The bit sequence received by the receiver is 00100
- Can this error be fixed?

Block Code Principles

- There are 28 invalid codes



Invalid Codeword	Minimum distance	Valid codeword	Invalid codeword	Minimum distance	Valid codeword
00001	1	00000	10000	1	00000
00010	1	00000	10001	1	11001
00011	1	00111	10010	2	00000 or 11110
00100	1	00000	10011	2	00111 or 11001
00101	1	00111	10100	2	00000 or 11110
00110	1	00111	10101	2	00111 or 11001
01000	1	00000	10110	1	11110
01001	1	11001	10111	1	00111
01010	2	00000 or 11110	11000	1	11001
01011	2	00111 or 11001	11010	1	11110
01100	2	00000 or 11110	11011	1	11001
01101	2	00111 or 11001	11100	1	11110
01110	1	11110	11101	1	11001
01111	1	00111	11111	1	11110

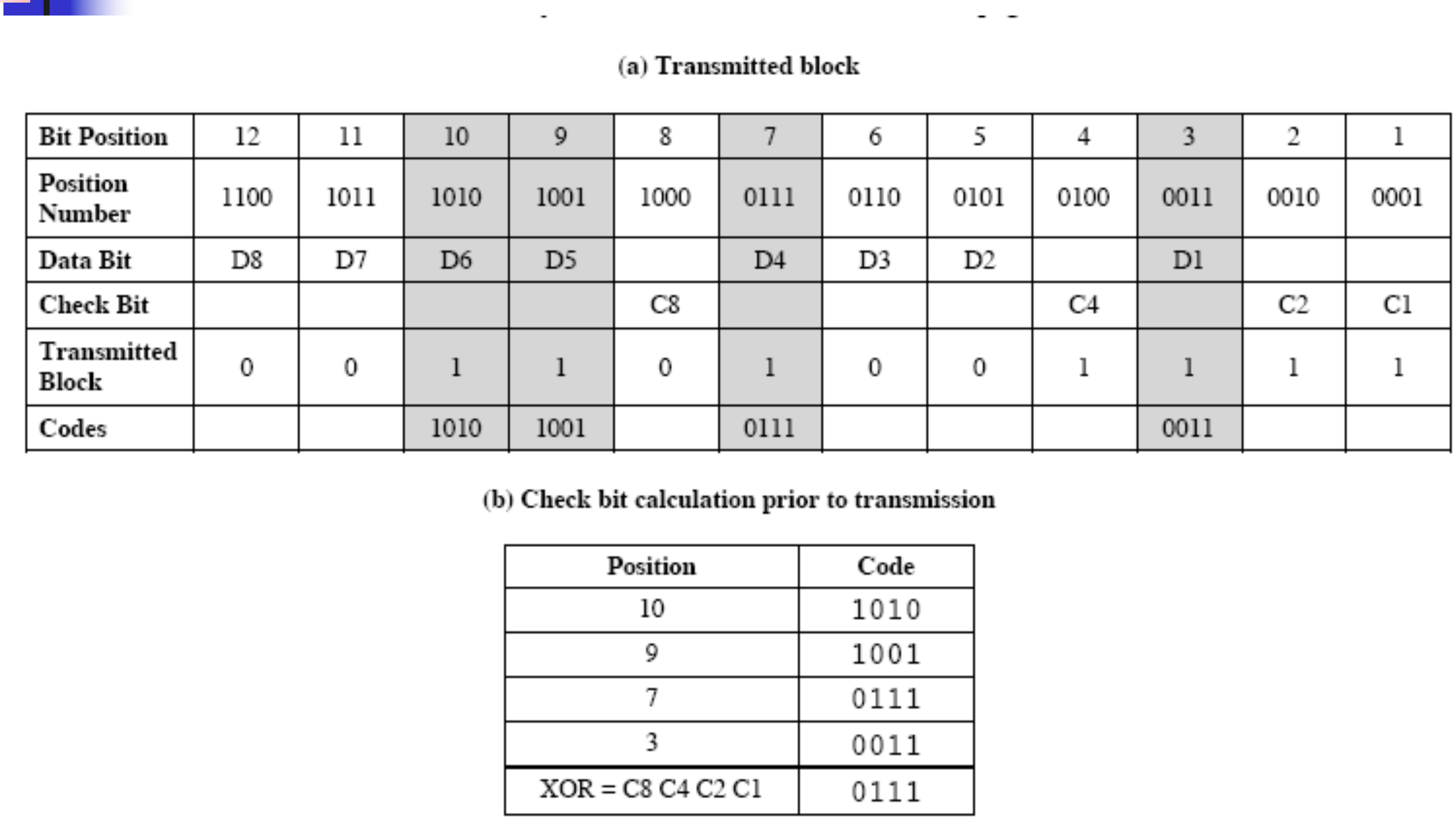
Hamming Code

- Designed to correct single bit errors
- Family of (n, k) block error-correcting codes with parameters:
  - Block length: $n = 2^m - 1$
  - Number of data bits: $k = 2^m - m - 1$
  - Number of check bits: $n - k = m$
  - Minimum distance: $d_{\min} = 3$
- Single-error-correcting (SEC) code
  - SEC double-error-detecting (SEC-DED) code

Hamming Code Process

- Encoding: *k* data bits + (*n* -*k*) check bits
  - Check bits are placed in bit positions that are the power of 2 in the frame.
- Decoding: compares received (*n* -*k*) bits with calculated (*n* -*k*) bits using XOR
  - Resulting (*n* -*k*) bits called *syndrome word*
  - Syndrome range is between 0 and $2^{(n-k)}-1$
  - Each bit of syndrome indicates a match (0) or conflict (1) in that bit position

Example



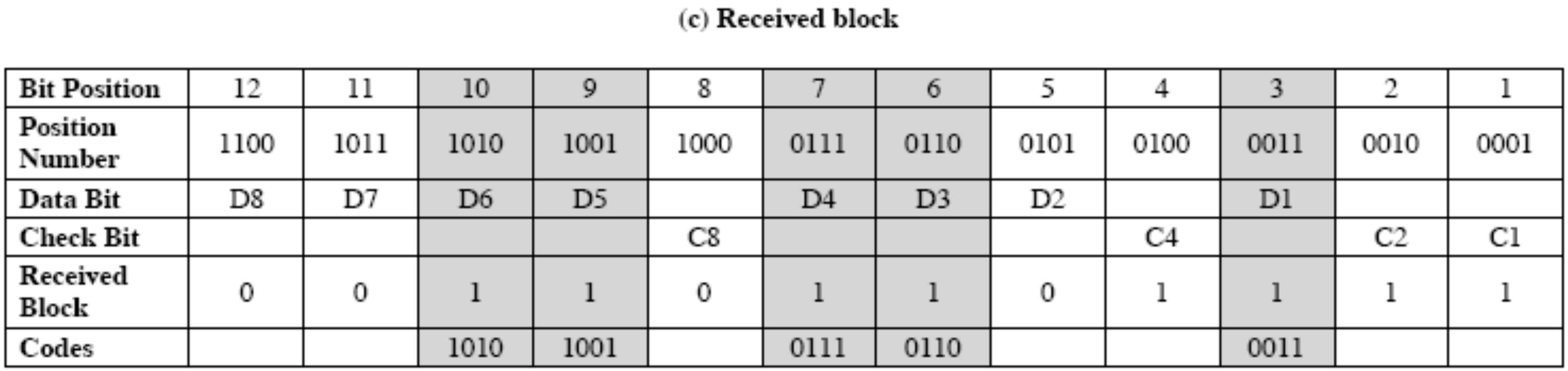
(a) Transmitted block

Bit Position	12	11	10	9	8	7	6	5	4	3	2	1
Position Number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data Bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check Bit					C8				C4		C2	C1
Transmitted Block	0	0	1	1	0	1	0	0	1	1	1	1
Codes			1010	1001		0111				0011		

(b) Check bit calculation prior to transmission

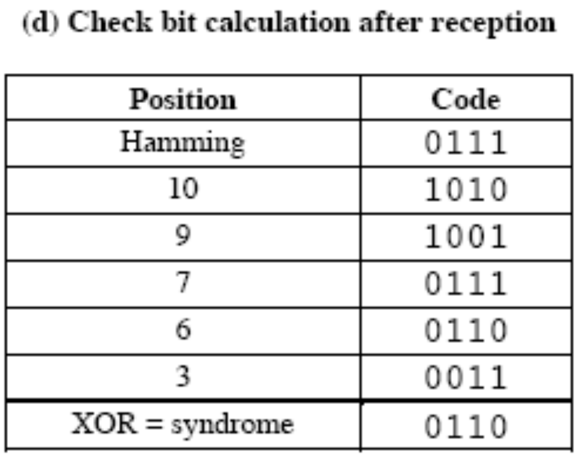
Position	Code
10	1010
9	1001
7	0111
3	0011
XOR = C8 C4 C2 C1	0111

Example



(c) Received block

Bit Position	12	11	10	9	8	7	6	5	4	3	2	1
Position Number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data Bit	D8	D7	D6	D5		D4	D3	D2		D1		
Check Bit					C8				C4		C2	C1
Received Block	0	0	1	1	0	1	1	0	1	1	1	1
Codes			1010	1001		0111	0110			0011		

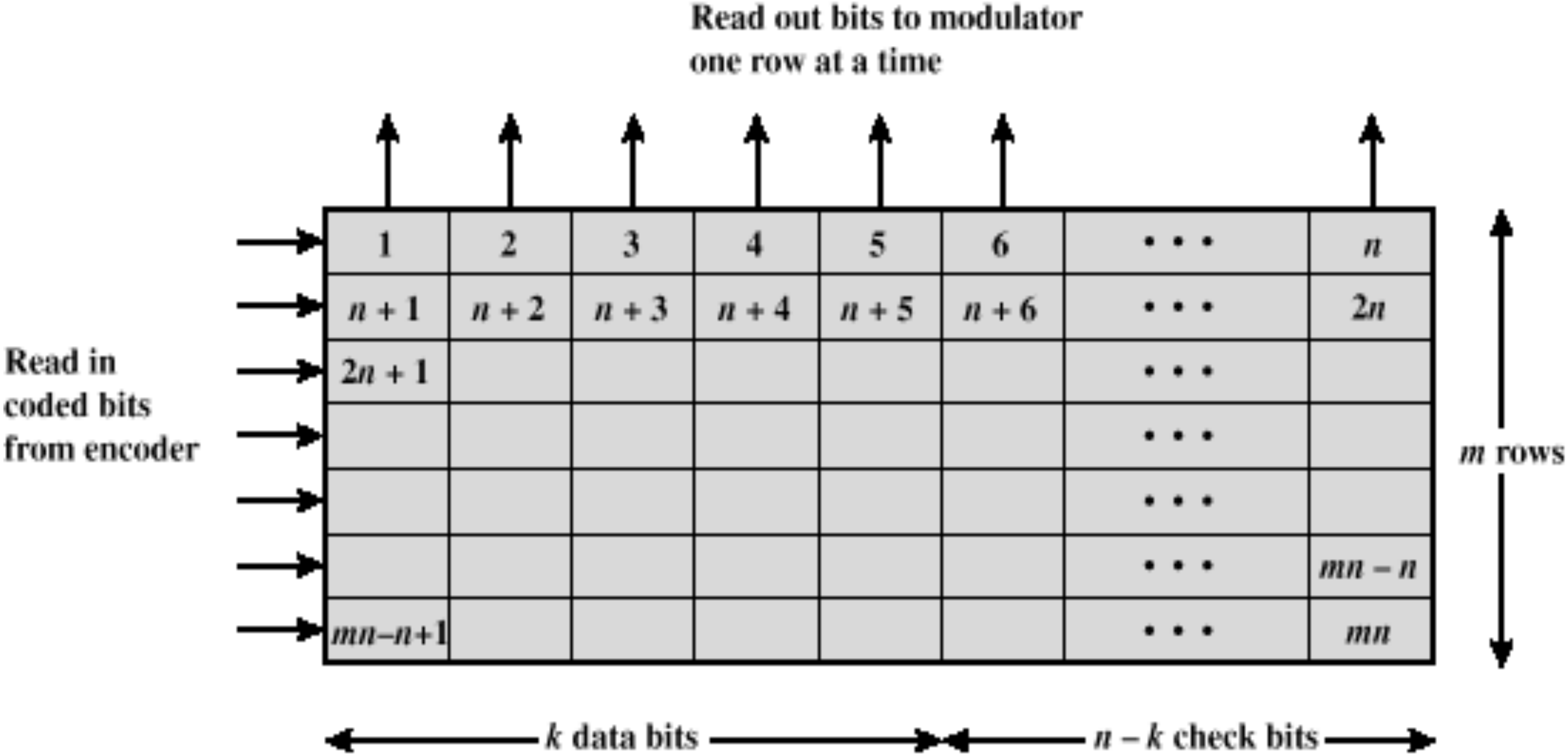


(d) Check bit calculation after reception

Position	Code
Hamming	0111
10	1010
9	1001
7	0111
6	0110
3	0011
XOR = syndrome	0110

Block Interleaving

- Data written to and read from memory in different orders
- Data bits and corresponding check bits are interspersed with bits from other blocks
- At receiver, data are deinterleaved to recover original order
- A burst error that may occur is spread out over a number of blocks, making error correction possible



Note: The numbers in the matrix indicate the order in which bits are read in.
Interleaver output sequence: 1, n + 1, 2n + 1, …

Figure 8.8 Block Interleaving

Automatic Repeat Request

- Mechanism used in data link control and transport protocols
- Relies on use of an error detection code (such as CRC)
- Flow Control
- Error Control

Flow Control

- Assures that transmitting entity does not overwhelm a receiving entity with data
- Protocols with flow control mechanism allow multiple PDUs in transit at the same time
- PDUs arrive in same order they're sent
- Sliding-window flow control
  - Transmitter maintains list (window) of sequence numbers allowed to send
  - Receiver maintains list allowed to receive

Flow Control

- Reasons for breaking up a block of data before transmitting:
    - Limited buffer size of receiver
    - Retransmission of PDU due to error requires smaller amounts of data to be retransmitted
    - On shared medium, larger PDUs occupy medium for extended period, causing delays at other sending stations

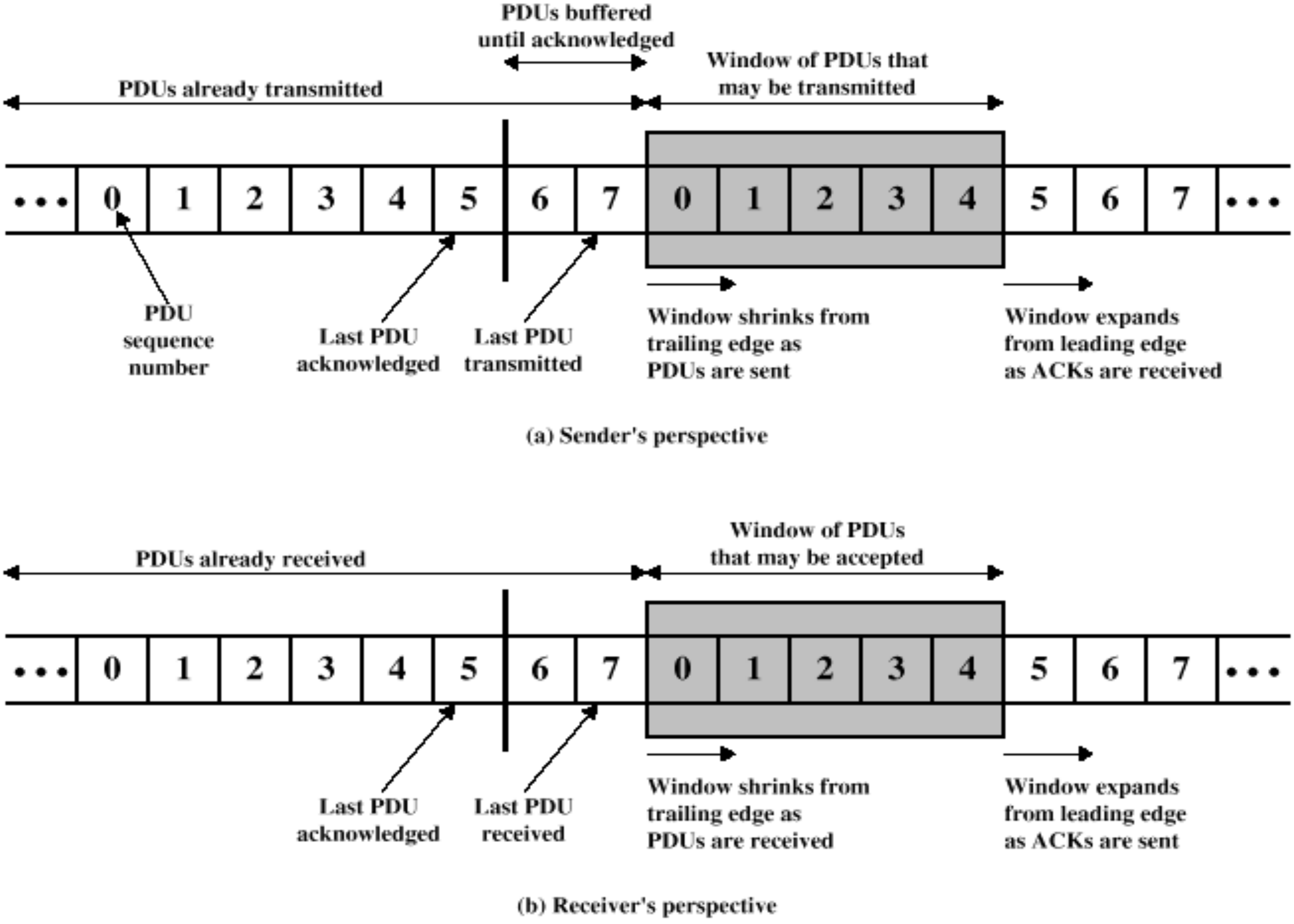

PDUs buffered until acknowledged
Window of PDUs that may be transmitted
PDUs already transmitted
• • •
0
1
2
3
4
5
6
7
0
1
2
3
4
5
6
7
• • •
PDU sequence number
Last PDU acknowledged
Last PDU transmitted
Window shrinks from trailing edge as PDUs are sent
Window expands from leading edge as ACKs are received
(a) Sender's perspective
Window of PDUs that may be accepted
PDUs already received
• • •
0
1
2
3
4
5
6
7
0
1
2
3
4
5
6
7
• • •
Last PDU acknowledged
Last PDU received
Window shrinks from trailing edge as PDUs are received
Window expands from leading edge as ACKs are sent
(b) Receiver's perspective

Figure 8.17 Sliding-Window Depiction

Error Control

- Mechanisms to detect and correct transmission errors
- Types of errors:
  - Lost PDU : a PDU fails to arrive
  - Damaged PDU : PDU arrives with errors

Error Control Requirements

- Error detection
  - Receiver detects errors and discards PDUs
- Positive acknowledgement
  - Destination returns acknowledgment of received, error-free PDUs
- Retransmission after timeout
  - Source retransmits unacknowledged PDU
- Negative acknowledgement and retransmission
  - Destination returns negative acknowledgment to PDUs in error