Digital Data Communication and Error Detection

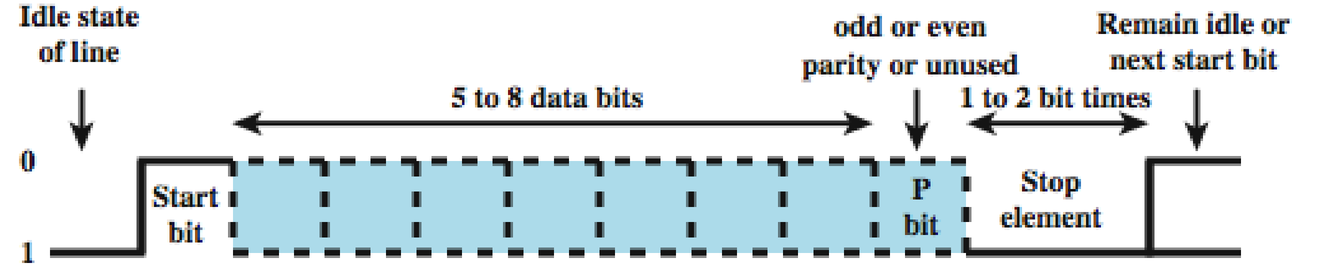
Asynchronous and Synchronous Transmission

- timing problems require a mechanism to synchronize the transmitter and receiver
  - receiver samples stream at bit intervals
  - if clocks not aligned and drifting will sample at wrong time after sufficient bits are sent
    - 1% drift causes sampling to be 0.01 bit time away from center
    - For 1Mbps rate, after 50 samples receiver reads the wrong bit.
- two solutions to synchronizing clocks
  - asynchronous transmission
  - synchronous transmission

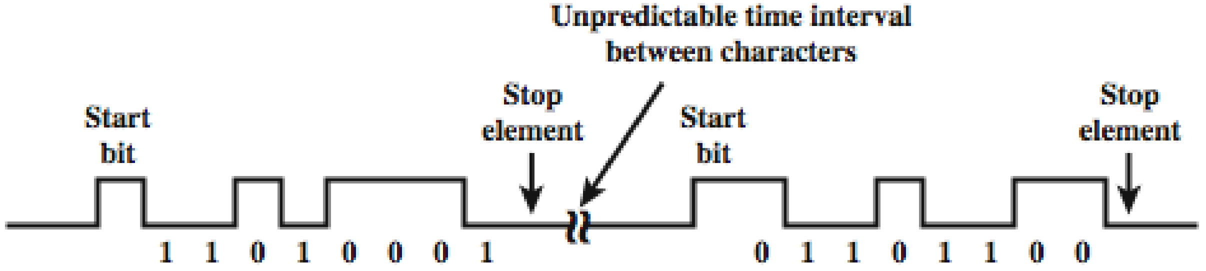
Asynchronous Transmission

- Send one character or unit of data (frame) at a time
- Start and end of transmission indicated by unique pattern
- overhead of 2 or 3 bits per char (~20%)
- good for data with large gaps (keyboard)
- Simple and cheap

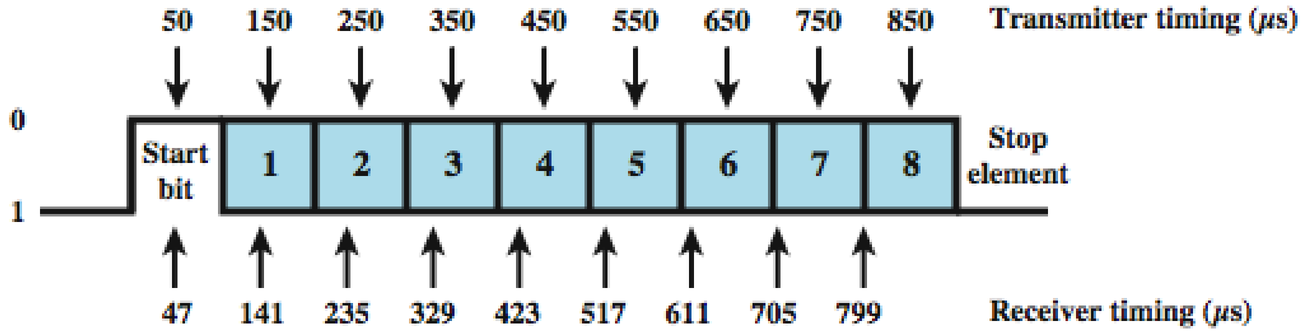
Asynchronous Transmission



(a) Character format



(b) 8-bit asynchronous character stream



(c) Effect of timing error

Synchronous Transmission

- block of data transmitted sent as a frame
- clocks must be synchronized
  - can use separate clock line
  - or embed clock signal in data
- need to indicate start and end of block
  - use preamble and postamble
- more efficient (lower overhead) than async

Types of Error

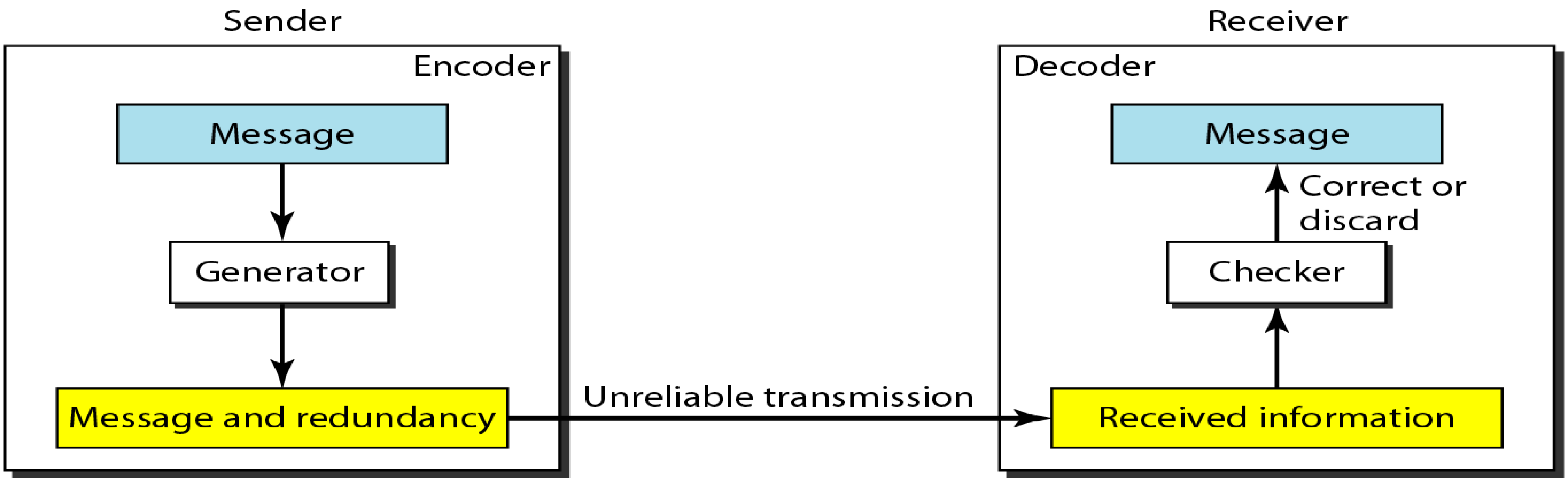
- an error occurs when a bit is altered between transmission and reception
- single bit errors
  - only one bit altered
  - caused by white noise
- burst errors
  - contiguous sequence of *B* bits in which first, last and any number of intermediate bits in error
  - caused by impulse noise or by fading in wireless
  - effect greater at higher data rates

Error Detection

- Link transmission will have errors
- P_b: Probability of bit error (BER)
- P_f: Probability of frame not in error = $(1-P_b)^F$, F is frame length in bits
- detected using error-detecting code
  - calculated as a function of data bits
  - recalculated and checked by receiver
- still chance of undetected error

Redundancy

- To detect or correct errors, redundant bits of data must be added



Error Coding

- Process of adding redundancy for error detection or correction
- Two types:
  - Block codes
    - Divides the data to be sent into a set of blocks
    - Extra information attached to each block
    - Memoryless
  - Convolutional codes
    - Treats data as a series of bits, and computes a code over a continuous series
    - The code computed for a set of bits depends on the current and previous input

XOR Operation

- Main operation for computing error detection/correction codes
- Similar to modulo-2 addition

$0 \oplus 0 = 0 \qquad 1 \oplus 1 = 0$

a. Two bits are the same, the result is 0.

$0 \oplus 1 = 1 \qquad 1 \oplus 0 = 1$

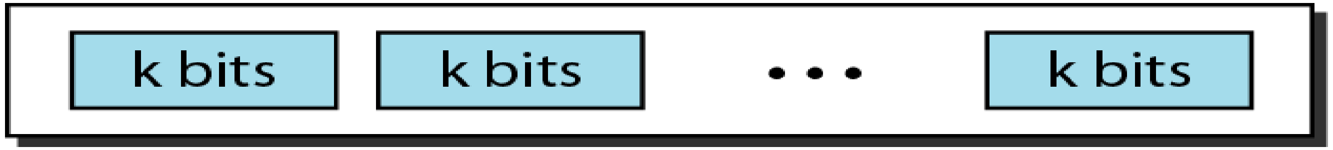
b. Two bits are different, the result is 1.

$$\begin{array}{cccccc} & 1 & 0 & 1 & 1 & 0 \\ \oplus & 1 & 1 & 1 & 0 & 0 \\ \hline & 0 & 1 & 0 & 1 & 0 \end{array}$$

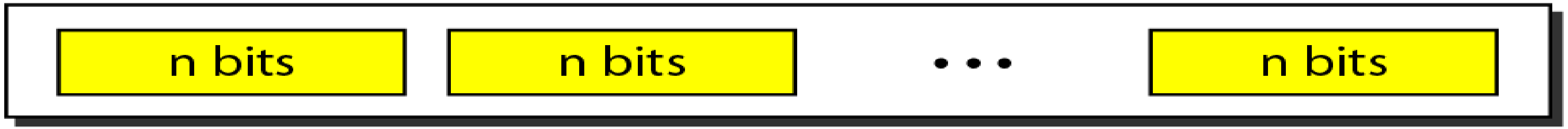
c. Result of XORing two patterns

Block Coding

- Message is divided into *k*-bit blocks
  - Known as *datawords*
- *r* redundant bits are added
  - Blocks become *n*=*k*+*r* bits
  - Known as *codewords*

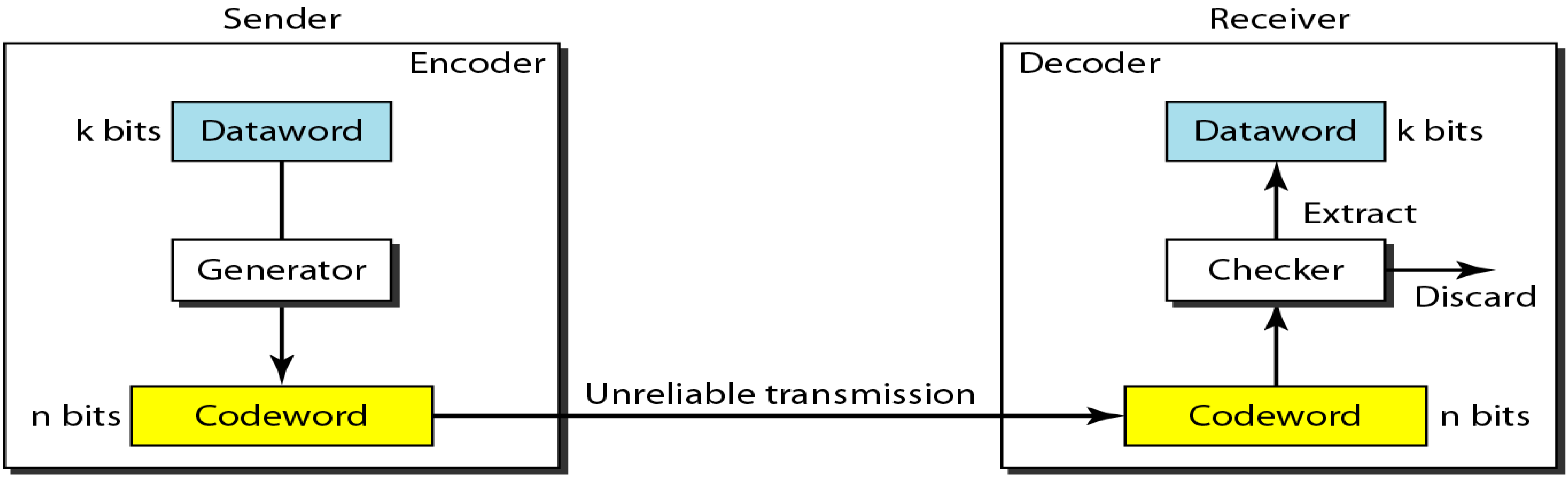


2^k Datawords, each of k bits



2^n Codewords, each of n bits (only 2^k of them are valid)

Error Detection in Block Coding



Notes

- An error-detecting code can detect
  only the types of errors for which it is designed
  - Other types of errors may remain undetected.
- There is no way to detect every possible error

Common Detection Methods

- Parity check
- Checksum
- Cyclic Redundancy Check

Parity Check

- Most common, least complex
- Single bit is added to a block
- Two schemes:
  - Even parity – Maintain even number of 1s
    - E.g., 1011 $\rightarrow$ 1011**<u>1</u>**
  - Odd parity – Maintain odd number of 1s
    - E.g., 1011 $\rightarrow$ 1011**<u>0</u>**

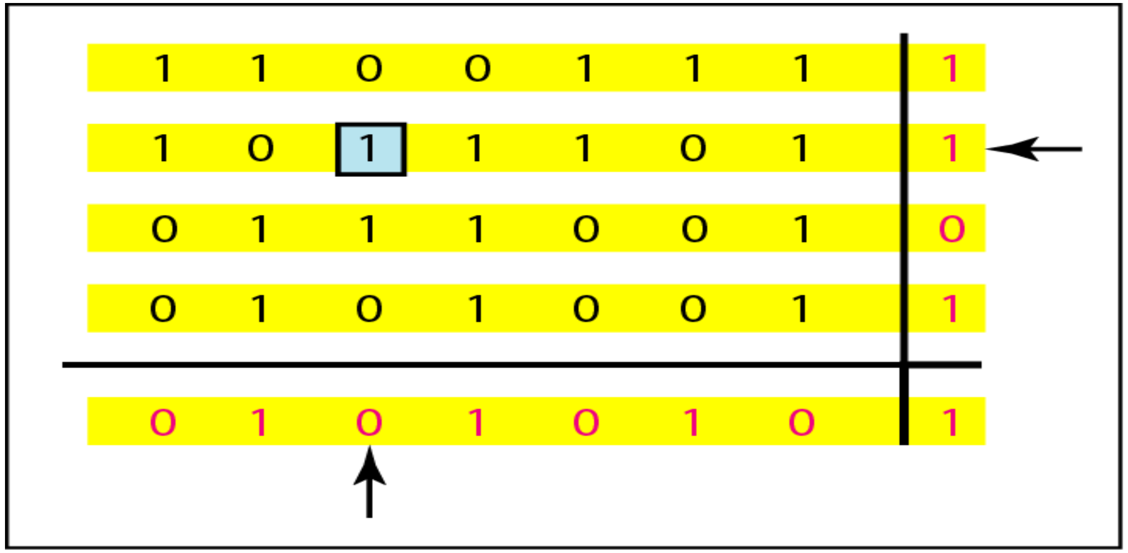
2D Parity Check

1	1	0	0	1	1	1	1	Row parities
1	0	1	1	1	0	1	1	
0	1	1	1	0	0	1	0	
0	1	0	1	0	0	1	1	
0	1	0	1	0	1	0	1	

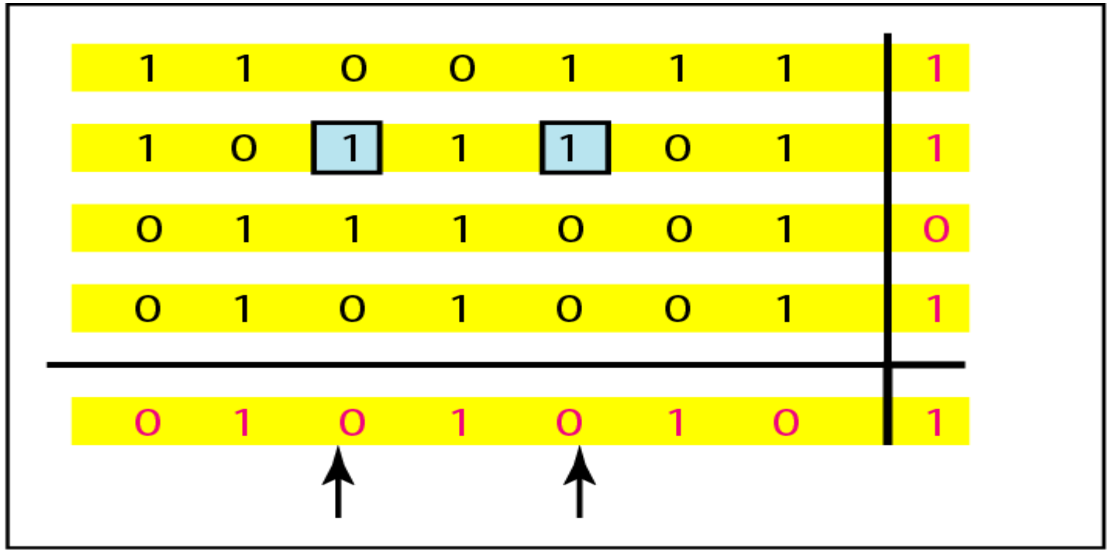
Column parities

What is its performance?

2D Parity Check: Performance

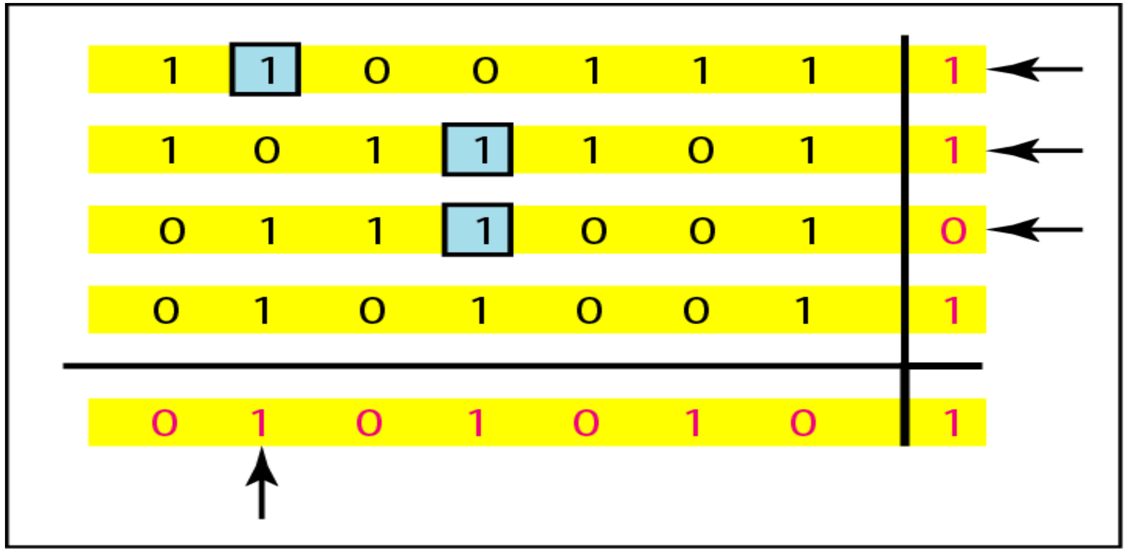


b. One error affects two parities

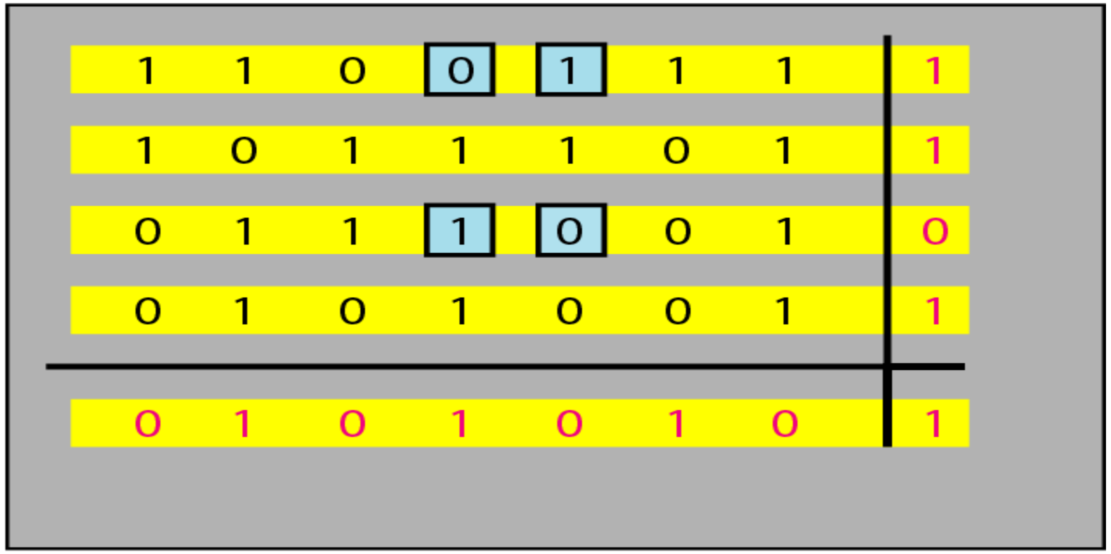


c. Two errors affect two parities

2D Parity Check: Performance



d. Three errors affect four parities



e. Four errors cannot be detected

Internet Checksum

- Used in network and transport layer
- Treat datawords to be protected as numbers and sum them using ones complement addition (wrap around leftmost digit)
- Send the negative of sum along with data
- Receiver adds all datawords (including checksum) and checks if zero
- Cannot protect against errors that don’t affect the sum

Internet Checksum: Sender

- The message is divided into 16-bit words.
- The value of the checksum word is set to 0.
- All words including the checksum are added using one’s complement addition.
- The sum is complemented and becomes the checksum.
- The checksum is sent with the data.

Internet Checksum: Receiver

- The message (including checksum) is divided into 16-bit words.
- All words are added using one’s complement addition.
- The sum is complemented and becomes the new checksum.
- If the value of checksum is 0, the message is accepted; otherwise, it is rejected.

Figure 10.25 *Example 10.23*

1	0	1	3	Carries
4	6	6	F	(Fo)
7	2	6	7	(ro)
7	5	7	A	(uz)
6	1	6	E	(an)
0	0	0	0	Checksum (initial)
8	F	C	6	Sum (partial)
			1	
8	F	C	7	Sum
7	0	3	8	Checksum (to send)

a. Checksum at the sender site

1	0	1	3	Carries
4	6	6	F	(Fo)
7	2	6	7	(ro)
7	5	7	A	(uz)
6	1	6	E	(an)
7	0	3	8	Checksum (received)
F	F	F	E	Sum (partial)
			1	
8	F	C	7	Sum
0	0	0	0	Checksum (new)

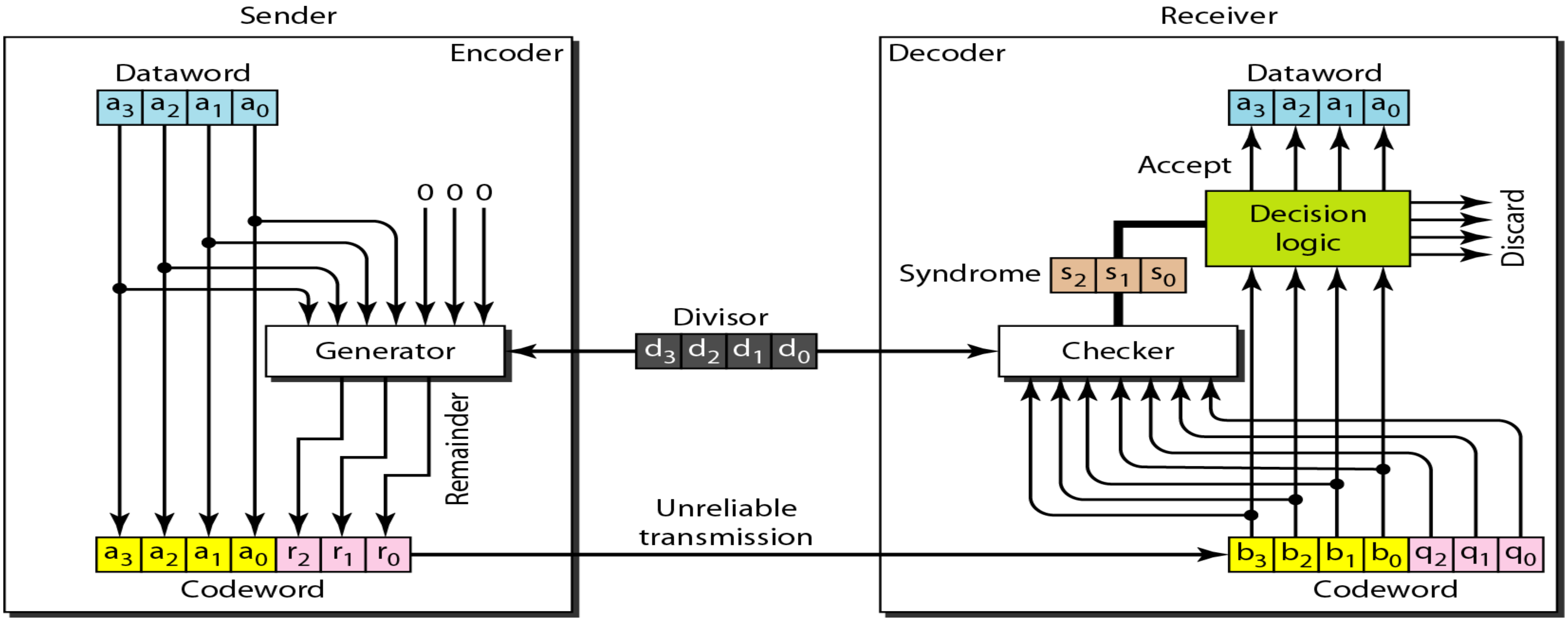
a. Checksum at the receiver site

Cyclic Redundancy Check

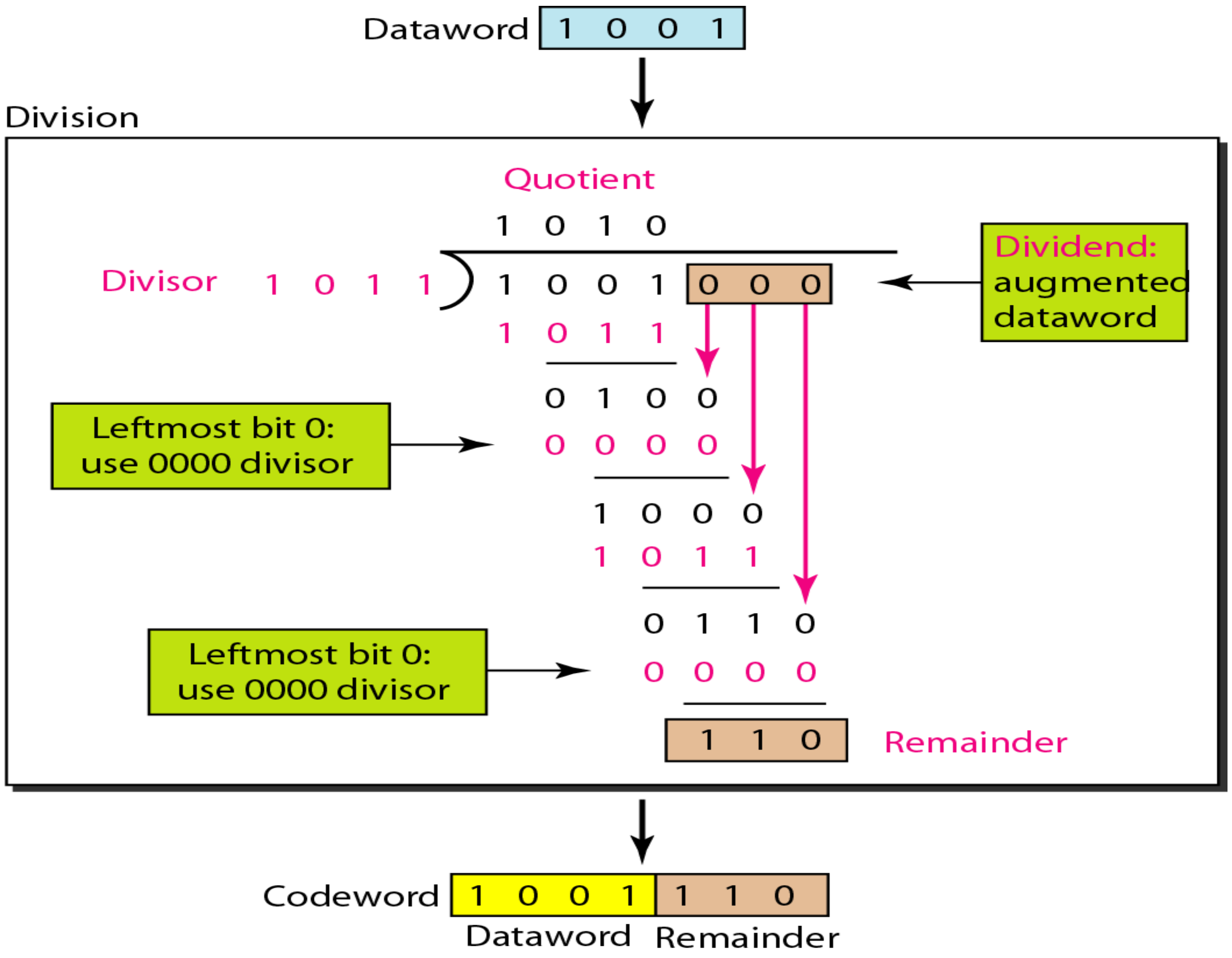
- In a cyclic code, rotating a codeword always results in another codeword
- Example:

Dataword	*Codeword*	*Dataword*	*Codeword*
0000	0000000	1000	1000101
0001	0001011	1001	1001110
0010	0010110	1010	1010011
0011	0011101	1011	1011000
0100	0100111	1100	1100010
0101	0101100	1101	1101001
0110	0110001	1110	1110100
0111	0111010	1111	1111111

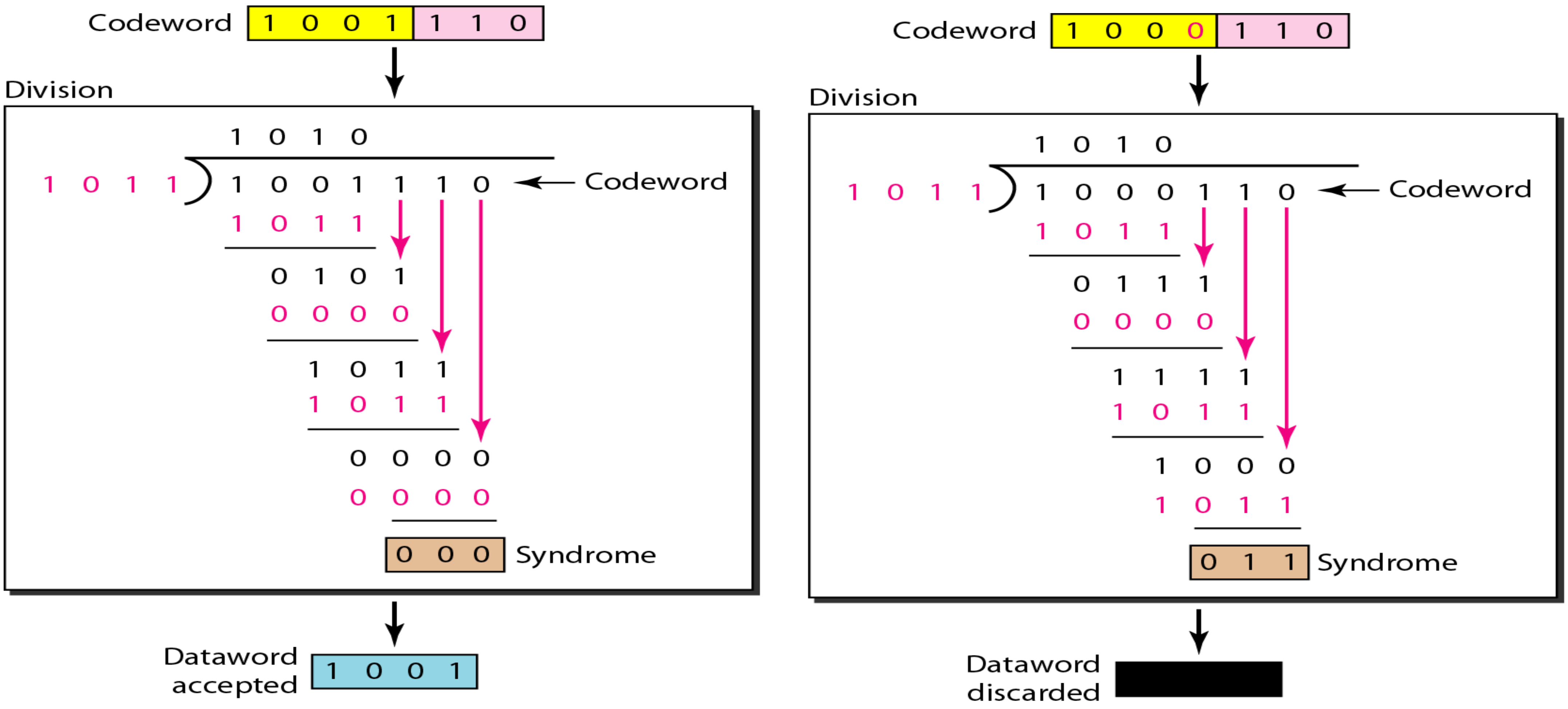
CRC Encoder/Decoder



CRC Generator

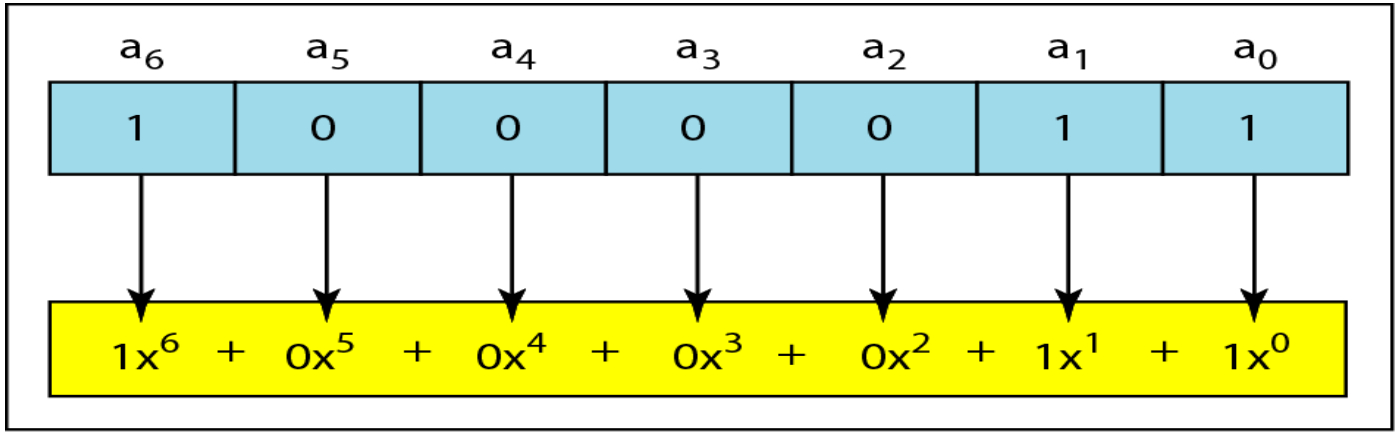


Checking CRC

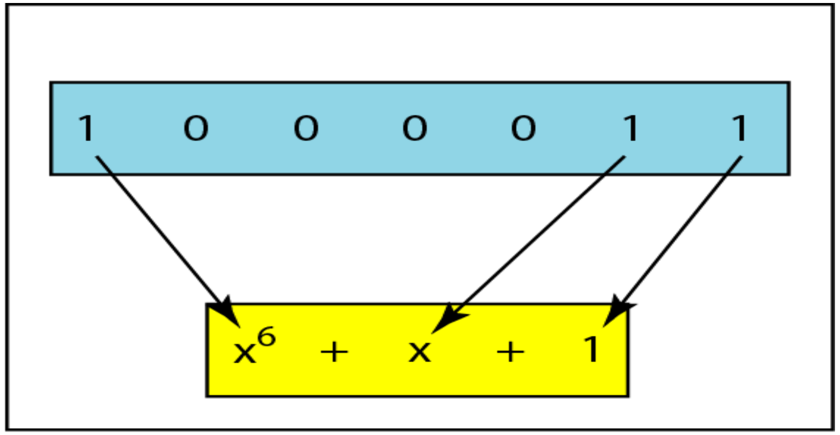


Polynomial Representation

- More common representation than binary form
- Easy to analyze
- Divisor is commonly called *generator polynomial*

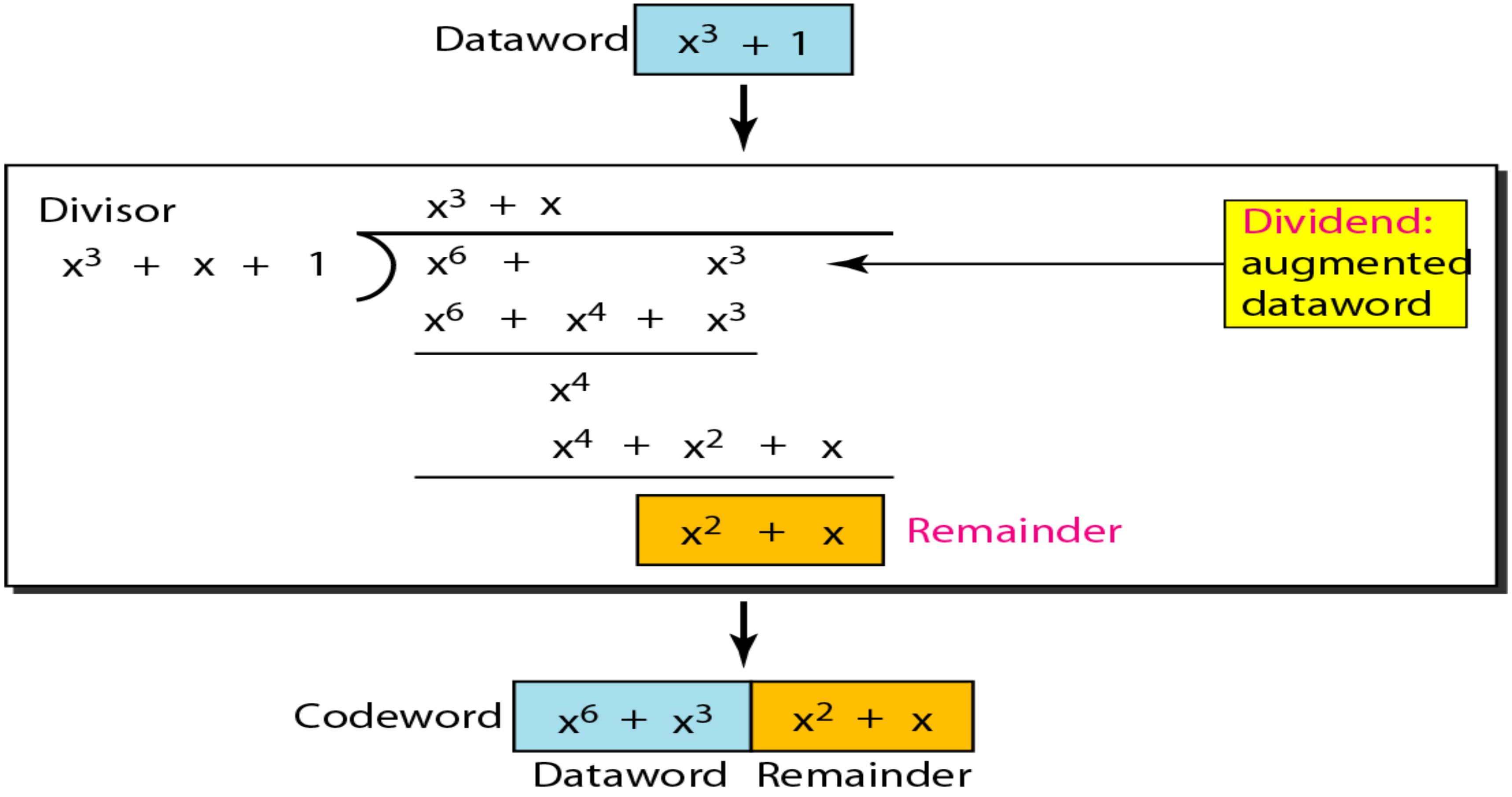


a. Binary pattern and polynomial



b. Short form

Division Using Polynomial



Strength of CRC

- Can be analyzed using polynomial
  - $\boldsymbol{M(x)}$ – Original message
  - $\boldsymbol{G(x)}$ – Generator polynomial of degree n
  - $\boldsymbol{R(x)}$ – Generated CRC

$$\boxed{\boldsymbol{M(x) \cdot x^n = Q(x) \cdot G(x) + R(x)}}$$

- Transmitted message is

$$\boldsymbol{M(x) \cdot x^n - R(x)}$$

which is divisible by $\boldsymbol{G(x)}$

Strength of CRC

- Received message is

$$\boldsymbol{M(x) \cdot x^n - R(x) + E(x)}$$

  where $\boldsymbol{E(x)}$ represents bit errors
- Receiver does not detect any error when $\boldsymbol{E(x)}$ is divisible by $\boldsymbol{G(x)}$, which means either:
  - $\boldsymbol{E(x)} = 0 \rightarrow$ No error
  - $\boldsymbol{E(x)} \neq 0 \rightarrow$ Undetectable error

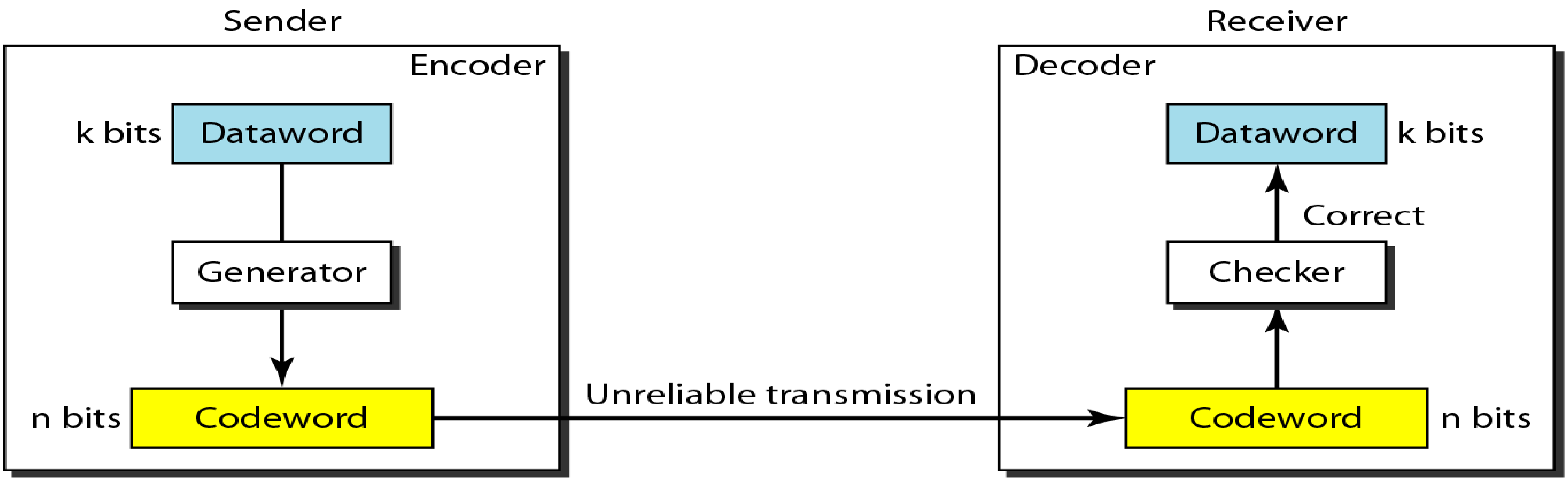
Strength of CRC

- If $\boldsymbol{G(x)}$ contains at least two terms, then all single-bit errors can be detected
- If $\boldsymbol{G(x)}$ cannot divide $x^t + 1$ $(0 \le t < n)$, then all isolated double errors can be detected
- If $\boldsymbol{G(x)}$ contains a factor of $(x+1)$, all odd-numbered errors can be detected

CRC's Strength Summary

- All burst errors with $L \leq n$ will be detected
- All burst errors with $L = n + 1$ will be detected with probability $1 - (1/2)^{n-1}$
- All burst errors with $L > n + 1$ will be detected with probability $1 - (1/2)^{n}$

Error Correction



Example: *Error Correction Code*

Dataword	*Codeword*
00	00000
01	01011
10	10101
11	11110

$k, r, n = ?$

The receiver receives 01001, what is the original dataword?

Hamming Distance

Hamming Distance between two words is the number of differences between corresponding bits.

- d(01, 00) = ?
- d(11, 00) = ?
- d(010, 100) = ?
- d(0011, 1000) = ?
- How many 8-bit words are *n* bits away from 10000111?

Minimum Hamming Distance

The minimum Hamming distance is the smallest Hamming distance between all possible pairs in a set of words.

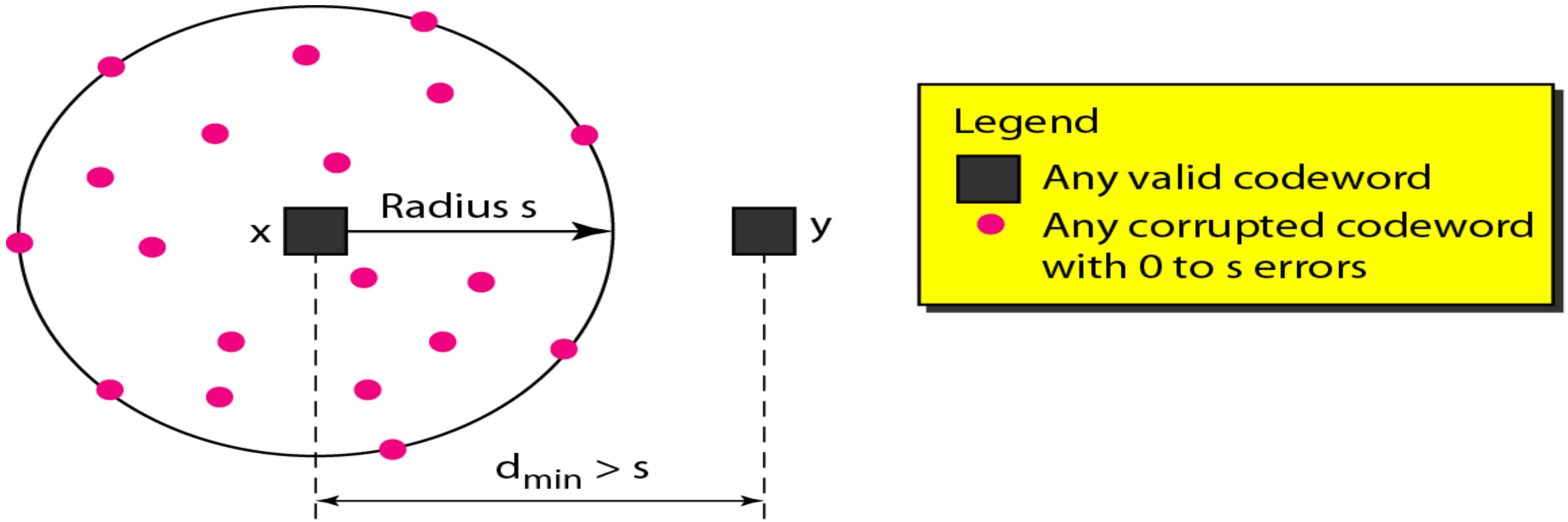
- Find the minimum Hamming Distance of the following codebook

00000
01011
10101
11110

Detection Capability of Code

- To guarantee the detection of up to *s*-bit errors, the minimum Hamming distance in a block code must be

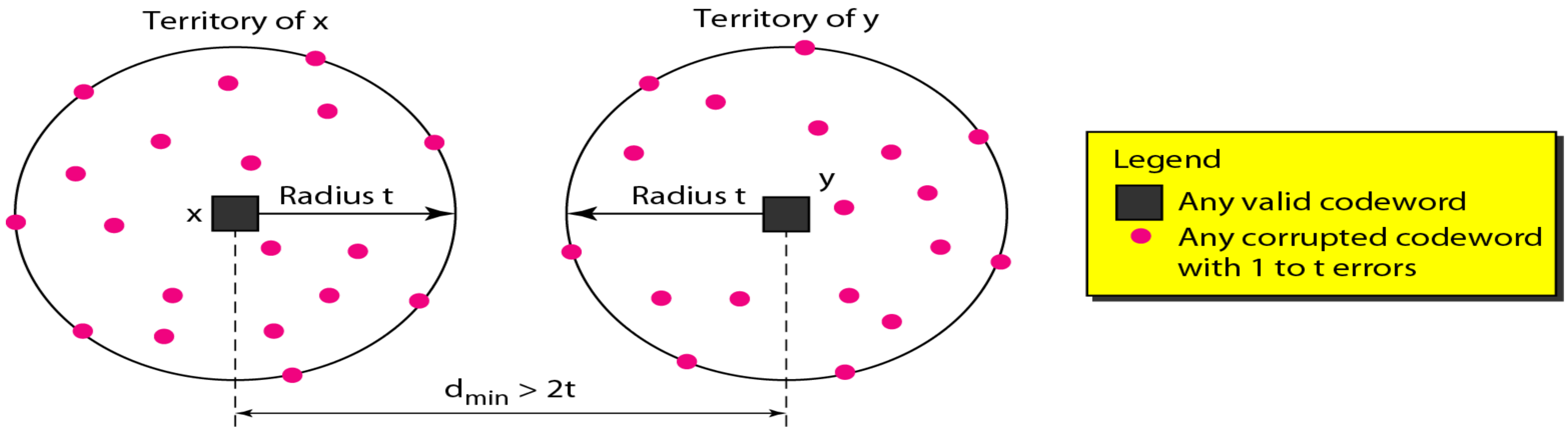
$$\boxed{d_{min} = s + 1}$$



Correction Capability of Code

- To guarantee the correction of up to *t*-bit errors, the minimum Hamming distance in a block code must be

$$\boxed{d_{min} = 2t + 1}$$



Example: *Hamming Distance*

- *A code scheme has a Hamming distance d_{min} = 4. What is the error detection and correction capability of this scheme?*

Error Correction

- Two methods
  - Retransmission after detecting error
  - Forward error correction (FEC)

Forward Error Correction

- Consider only a single-bit error in *k* bits of data
  - *k* possibilities for an error
  - One possibility for no error
  - #possibilities = *k* + 1
- Add r redundant bits to distinguish these possibilities; we need

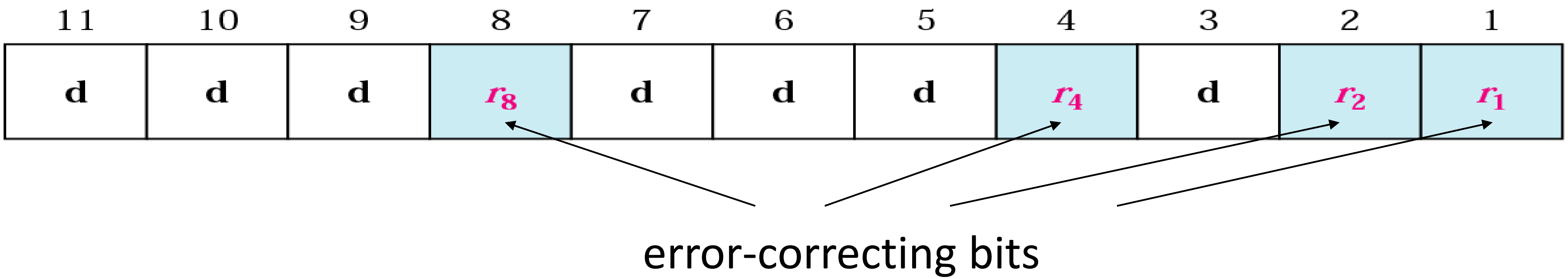
$$2^r \geq k+1$$

- But the r bits are also transmitted along with data; hence

$$2^r \geq k+r+1$$

Hamming Code

- Simple, powerful FEC
- Widely used in computer memory
  - Known as ECC memory



Example: *Hamming Code*

Data:
1 0 0 1 1 0 1

Code:
1 0 0 1 1 1 0 0 1 0 1

Example: *Correcting Error*

- Receiver receives 10010100101