

LECTURE 5

ERROR DETECTION AND CORRECTION

Example 4

Suppose the following block of 16 bits is to be sent using a checksum of 8 bits.

10101001 00111001

The numbers are added as:

10101001

00111001

Sum 11100010

Checksum 00011101

The pattern sent is 10101001 00111001 00011101

Example 5

Now suppose the receiver receives the pattern sent in Example 7 and there is no error.

10101001 00111001 00011101

When the receiver adds the three sections, it will get all 1s, which, after complementing, is all 0s and shows that there is no error.

10101001

00111001

00011101

Sum

11111111

Complement

00000000 means that the pattern is OK.

Example 6

Now suppose there is a burst error of length 5 that affects 4 bits.

10101111 11111001 00011101

When the receiver adds the three sections, it gets

| | | |
|-------------|------------|---------------------------|
| | 10101111 | |
| | 11111001 | |
| | 00011101 | |
| Partial Sum | 1 11000101 | |
| Carry | | 1 |
| Sum | 11000110 | |
| Complement | 00111001 | the pattern is corrupted. |

Correction

Stop and wait

Go Back N

Sliding Window

Hamming Code

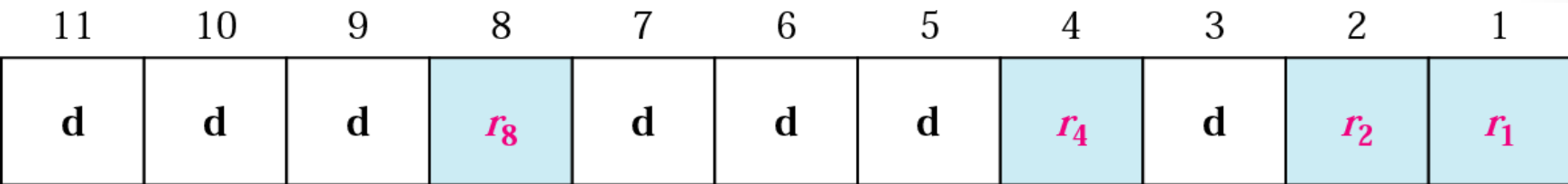
Hamming Code

Data and redundancy bits

| Number of data bits m | Number of redundancy bits r | Total bits $m + r$ |
|----------------------------|----------------------------------|-----------------------|
| 1 | 2 | 3 |
| 2 | 3 | 5 |
| 3 | 3 | 6 |
| 4 | 3 | 7 |
| 5 | 4 | 9 |
| 6 | 4 | 10 |
| 7 | 4 | 11 |

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

Positions of redundancy bits in Hamming code



$$2^3, 2^2, 2^1, 2^0$$

Redundancy bits calculation

r_1 will take care of these bits.

| | | | | | | | | | | |
|----|---|---|-------|---|---|---|-------|---|-------|-------|
| 11 | | 9 | | 7 | | 5 | | 3 | | 1 |
| d | d | d | r_8 | d | d | d | r_4 | d | r_2 | r_1 |

r_2 will take care of these bits.

| | | | | | | | | | | |
|----|----|---|-------|---|---|---|-------|---|-------|-------|
| 11 | 10 | | | 7 | 6 | | | 3 | 2 | |
| d | d | d | r_8 | d | d | d | r_4 | d | r_2 | r_1 |

r_4 will take care of these bits.

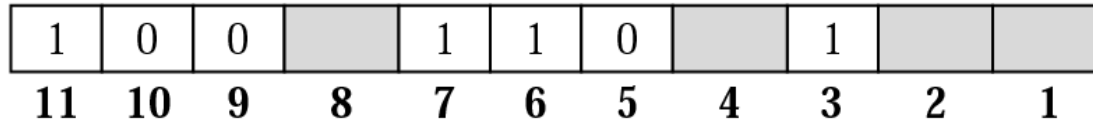
| | | | | | | | | | | |
|---|---|---|-------|---|---|---|-------|---|-------|-------|
| | | | | 7 | 6 | 5 | 4 | | | |
| d | d | d | r_8 | d | d | d | r_4 | d | r_2 | r_1 |

r_8 will take care of these bits.

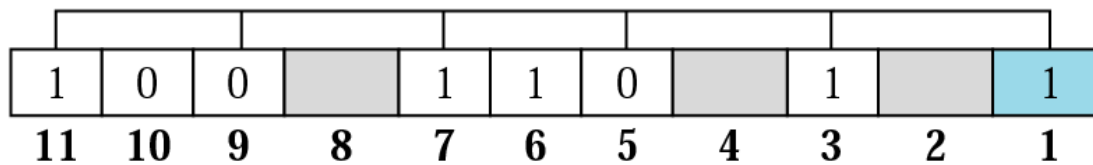
| | | | | | | | | | | |
|----|----|---|-------|---|---|---|-------|---|-------|-------|
| 11 | 10 | 9 | 8 | | | | | | | |
| d | d | d | r_8 | d | d | d | r_4 | d | r_2 | r_1 |

Example of redundancy bit calculation

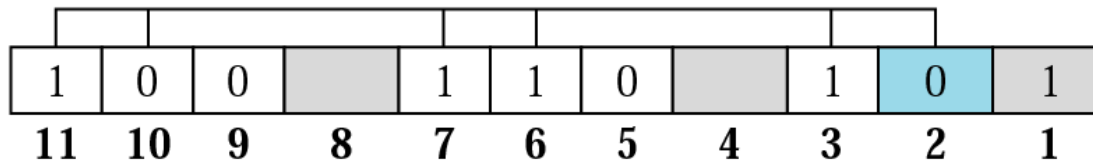
Data:
1001101



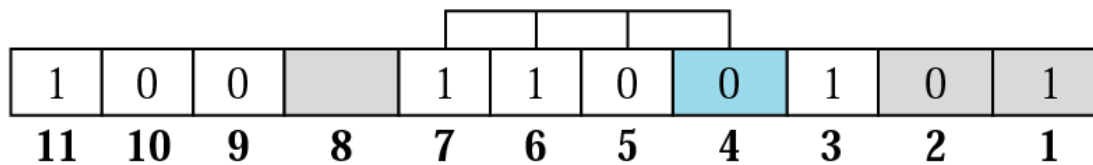
Adding r_1



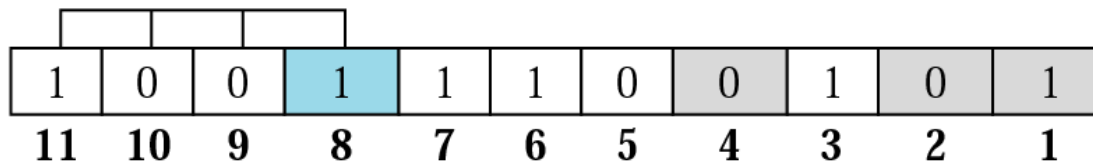
Adding r_2



Adding r_4

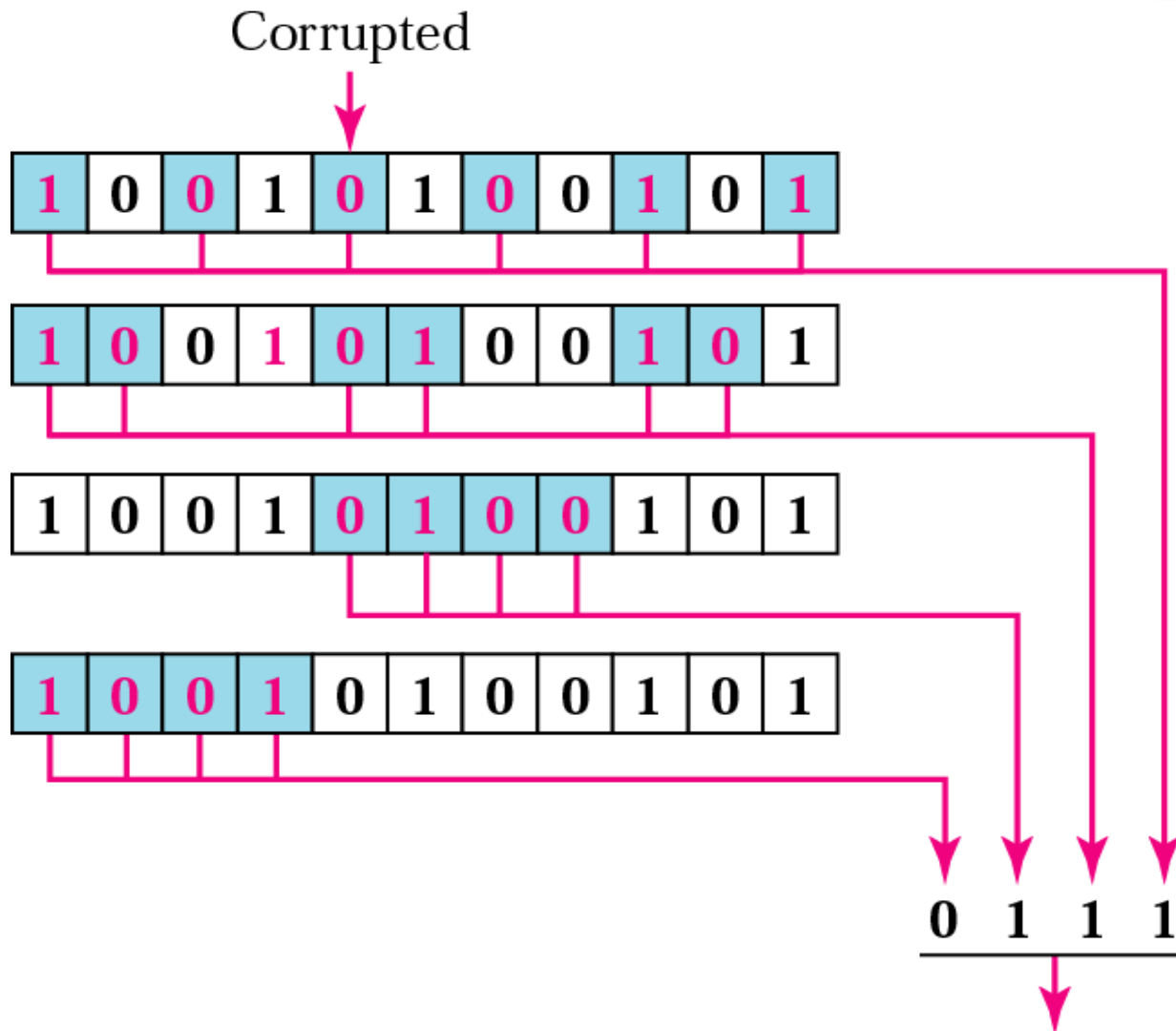


Adding r_8



Code:
10011100101

Error detection using Hamming code



The bit in position 7 is in error. **7**

Burst error correction example

Error → 1111?000011

Error → 1010?011111

11111001100

Error → 011?1011001

Error → 011?1010110

Error → 011?1001111

Received data

Direction of transmission ↑

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |

Data in transition

11111000011

10101011111

11111001100

01101011001

01101010110

01111001111

Data before being sent