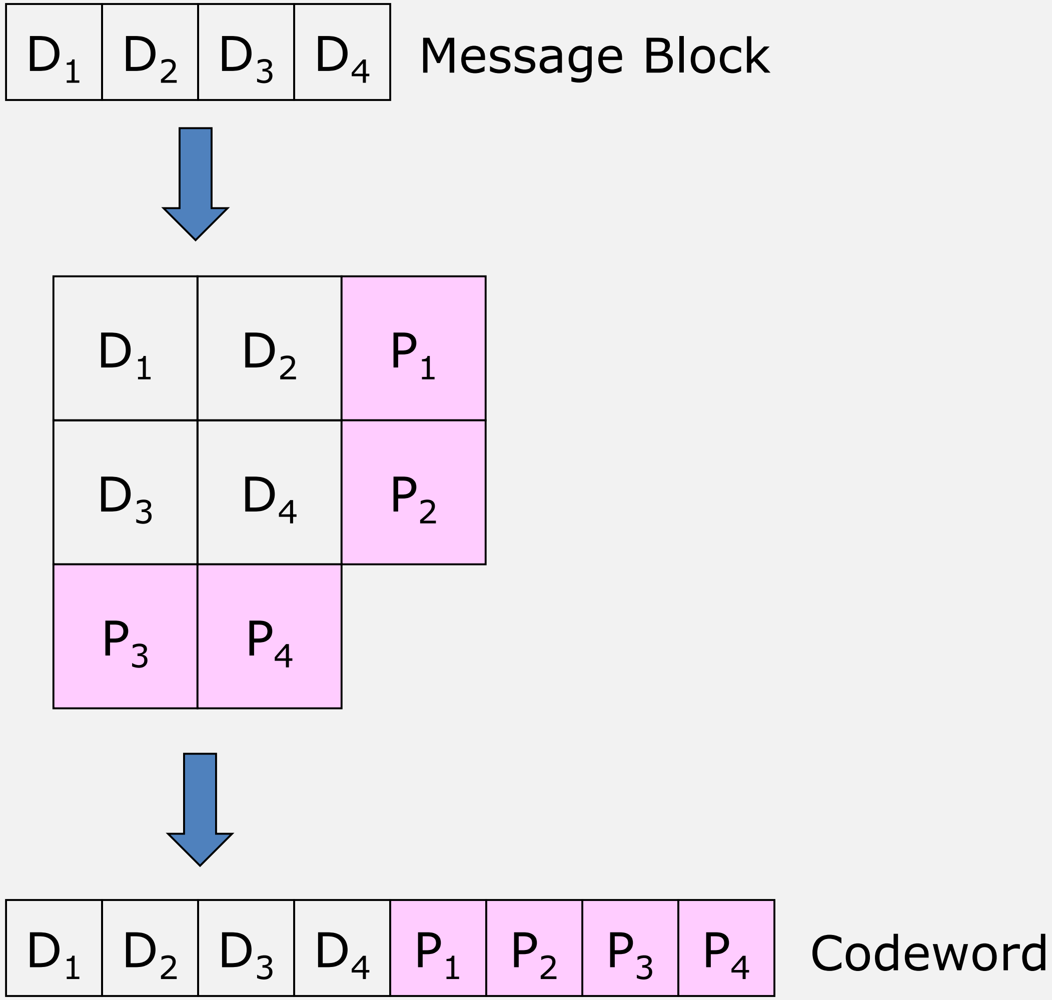
(9,4,4) Code

Review: (8,4,3) Code



D_1	D_2	D_3	D_4

Message Block

D_1	D_2	P_1
D_3	D_4	P_2
P_3	P_4	

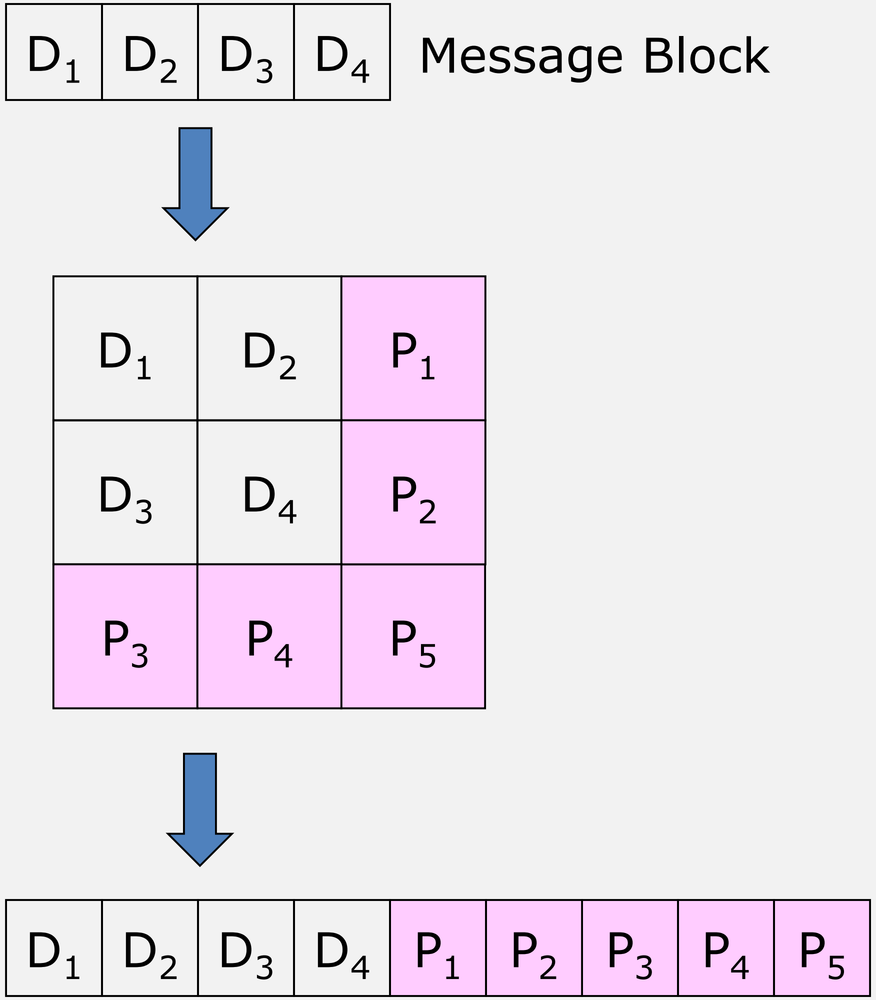
D_1	D_2	D_3	D_4	P_1	P_2	P_3	P_4

Codeword

- Arrange the message block to a 2x2 square.
- Add a parity bit (P_i) to each row or column, so that it has even parity.
  - Choose P_1 so row 1 has even parity.
  - Choose P_2 so row 2 has even parity.
  - Choose P_3 so column 1 has even parity.
  - Choose P_4 so column 2 has even parity.
- Rearrange the bits to form the final codeword.

$$\text{code rate} = \frac{1}{2}$$

A (9,4,4) Code



D_1	D_2	D_3	D_4	Message Block

D_1	D_2	P_1
D_3	D_4	P_2
P_3	P_4	P_5

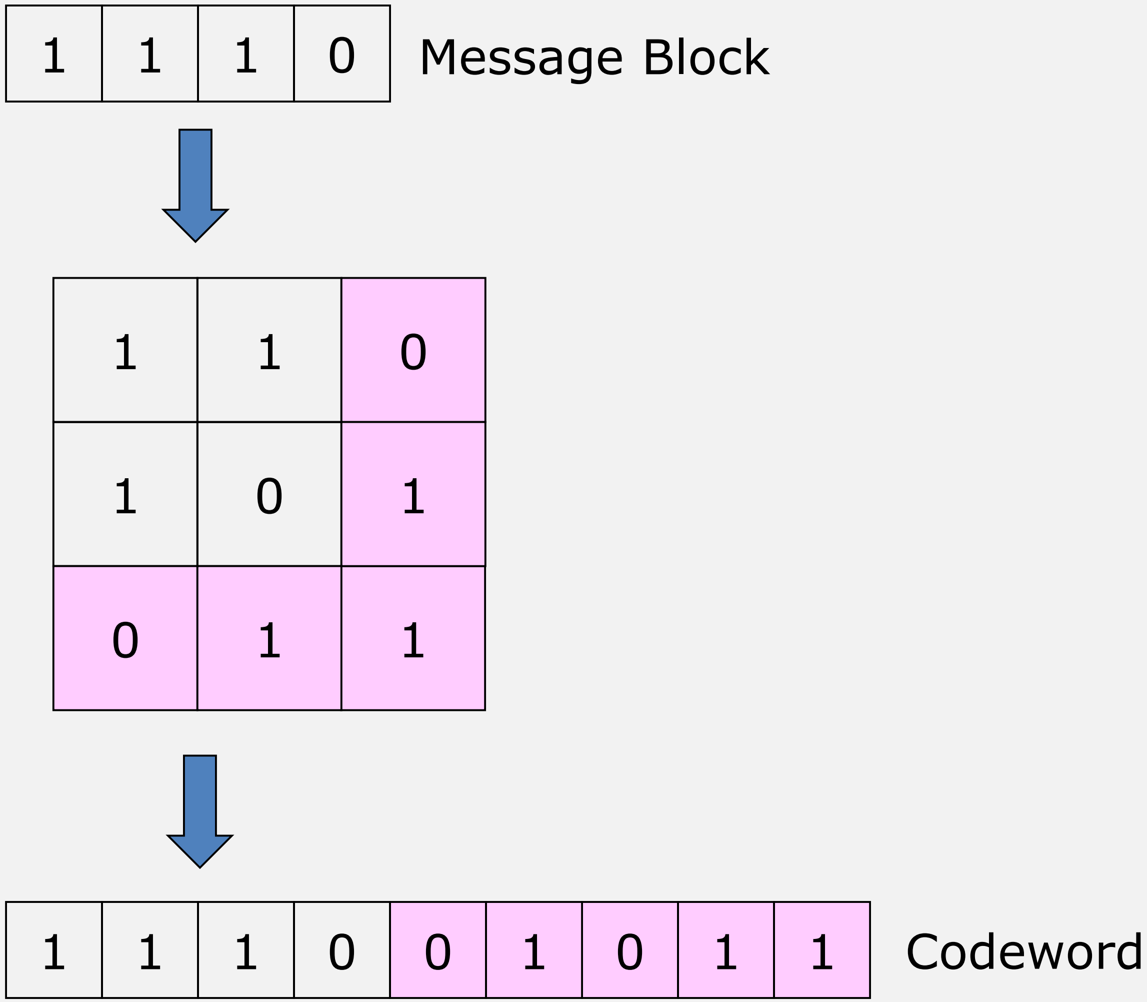
D_1	D_2	D_3	D_4	P_1	P_2	P_3	P_4	P_5

- We can increase the minimum Hamming distance in the (8,4,3) code to 4 by adding an overall parity bit, which is chosen so that the codeword always has even parity.

- The (9,4,4) code allows us to either
  - Correct 1 bit errors
    and detect 2 bit errors

    OR
  - Detect 1, 2 and 3 bit errors

Example



1	1	1	0	Message Block

1	1	0
1	0	1
0	1	1

1	1	1	0	0	1	0	1	1	Codeword

- Arrange the message block to a 2x2 square.

- Parity bits
  - P_1: parity for D_1,D_2
  - P_2: parity for D_3,D_4
  - P_3: parity for D_1,D_3
  - P_4: parity for D_2,D_4
  - P_5: parity for D_1,D_2,D_3,D_4,P_1,P_2,P_3,P_4

- Rearrange the bits to form the final codeword.

Compute Syndrome Bits

0	1	1	0
1	1	0	0
1	0	1	
0	0		0

0	1	1	0
1	0	0	1
1	0	1	
0	1		1

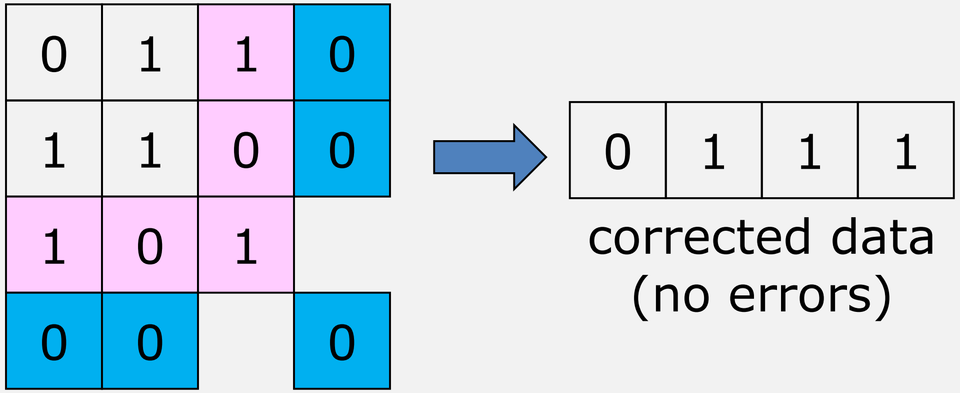
0	1	1	0
1	1	1	1
1	0	1	
0	0		1

D_1	D_2	P_1	S_1
D_3	D_4	P_2	S_2
P_3	P_4	P_5	
S_3	S_4		S_5

- To correct errors, we assume at most 2 bits have errors.

- Compute 5 syndrome bits by checking
  - the first two rows for even parity (S_1,S_2)
  - the first two columns for even parity (S_3,S_4)
  - the entire codeword for even parity (S_5)

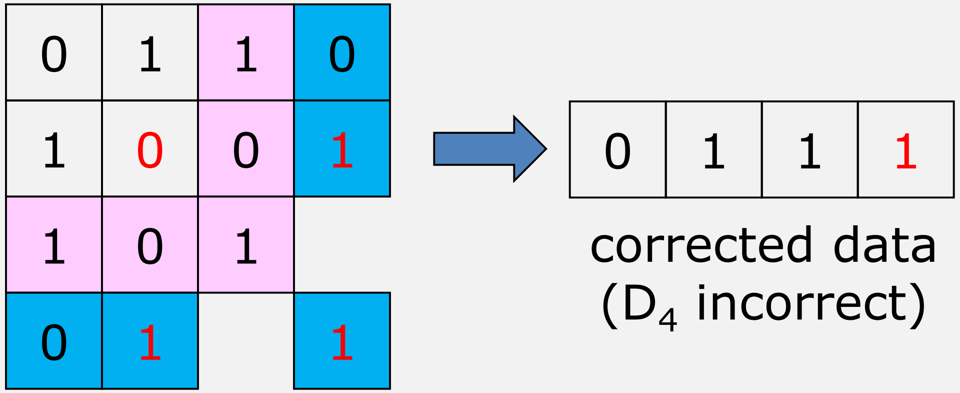
Performing Error Correction



0	1	1	0
1	1	0	0
1	0	1	
0	0		0

0	1	1	1

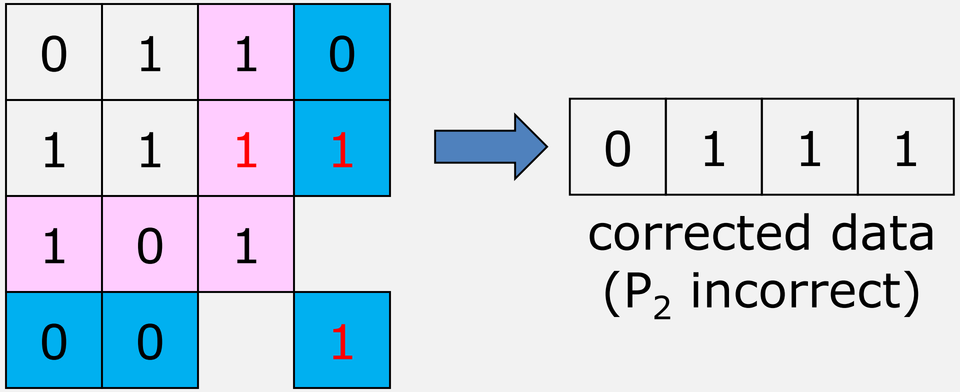
corrected data
(no errors)



0	1	1	0
1	0	0	1
1	0	1	
0	1		1

0	1	1	1

corrected data
(D_4 incorrect)



0	1	1	0
1	1	1	1
1	0	1	
0	0		1

0	1	1	1

corrected data
(P_2 incorrect)

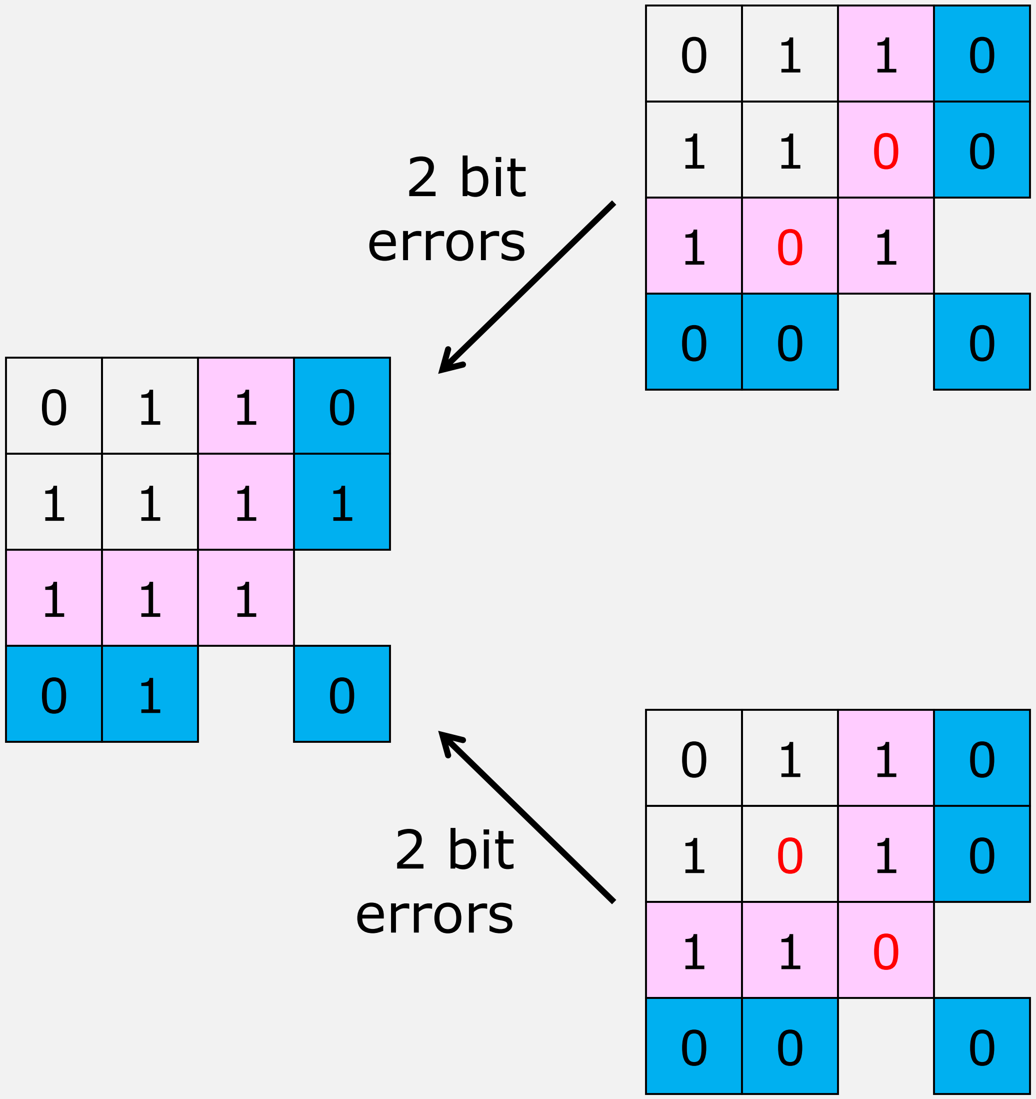
- **Check the syndrome bits.**

if all $S_i = 0$
no error

else if $S_5 = 1$, % error in one bit
check the other syndrome bits to see which bit to correct

else % error in more than one bit
we can only detect this.

Detecting Two Bit Errors



0	1	1	0
1	1	0	0
1	0	1	
0	0		0

2 bit errors

0	1	1	0
1	1	1	1
1	1	1	
0	1		0

2 bit errors

0	1	1	0
1	0	1	0
1	1	0	
0	0		0

- We cannot correct 2 bit errors because there are two valid codewords that differ from the received codeword by 2 bits.