Parity Bit Based Codes

Single Parity Bit Code

- Given a k-bit message, we can create a (k+1,k,2) block code by adding a single bit.
  - The bit is chosen so that the sum of the (k+1) bits in the codeword is even.
  - Equivalently, the bit is 1 if the sum of the k message bit is odd, and 0 otherwise.
  - The bit is called a parity bit.
  - The resulting codeword is said to have even parity.

- Example: (8,7,2) code

parity bit

message block: 0 1 1 0 0 1 0
codeword: 0 1 1 0 0 1 0 **1**

message block: 1 0 1 1 1 0 0
codeword: 1 0 1 1 1 0 0 **0**

$$\text{code rate} = \frac{k}{k+1} = \frac{7}{8}$$

Error Detection

- With the (k+1,k,2) parity bit code, we can detect single bit errors.
- If the received codeword has an even number of ones,
  then, we assume no bit errors have occurred
  otherwise, we assume a one bit error has occurred
- Example: (8,7,2) code

message block:	0 1 1 0 0 1 0
sent codeword:	0 1 1 0 0 1 0 1
received codeword:	0 1 1 0 1 1 0 1 $\rightarrow$ single bit error (detected)
received codeword:	0 1 1 1 1 1 0 1 $\rightarrow$ two bit error (not detected)

An (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	

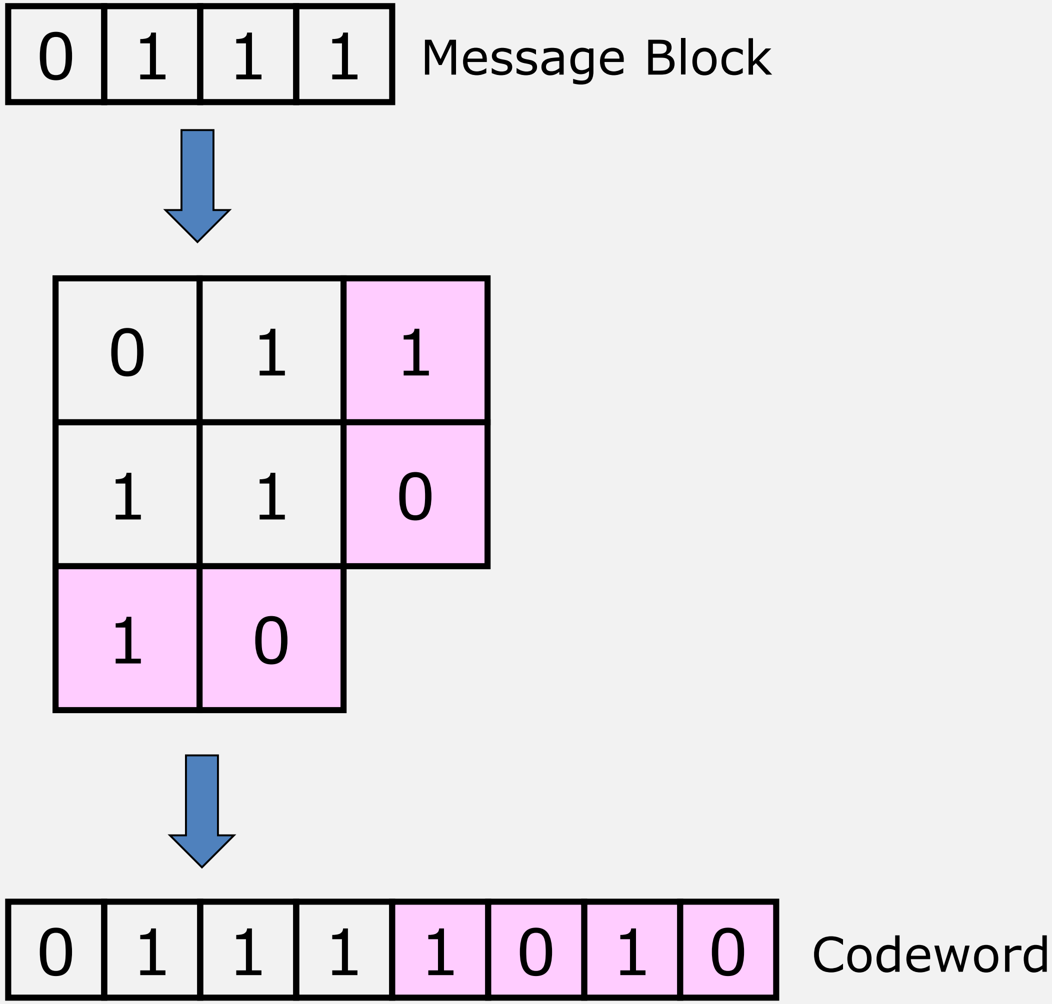
D_1	D_2	D_3	D_4	P_1	P_2	P_3	P_4

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.

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

Example



- 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.

Syndrome Bits

D_1	D_2	D_3	D_4	P_1	P_2	P_3	P_4

Received codeword

$\Downarrow$ Rearrange the codeword

D_1	D_2	P_1
D_3	D_4	P_2
P_3	P_4	

$\Downarrow$ Compute syndrome bits

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

- The P_i have been chosen so that each row or column of the rearranged codeword should have an even number of bits.

- Syndrome bits S_i check this condition in the received code word.
  - $S_i = 1$ indicates the condition for parity bit P_i is violated.

Example Syndrome Bit Calculations

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

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

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

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

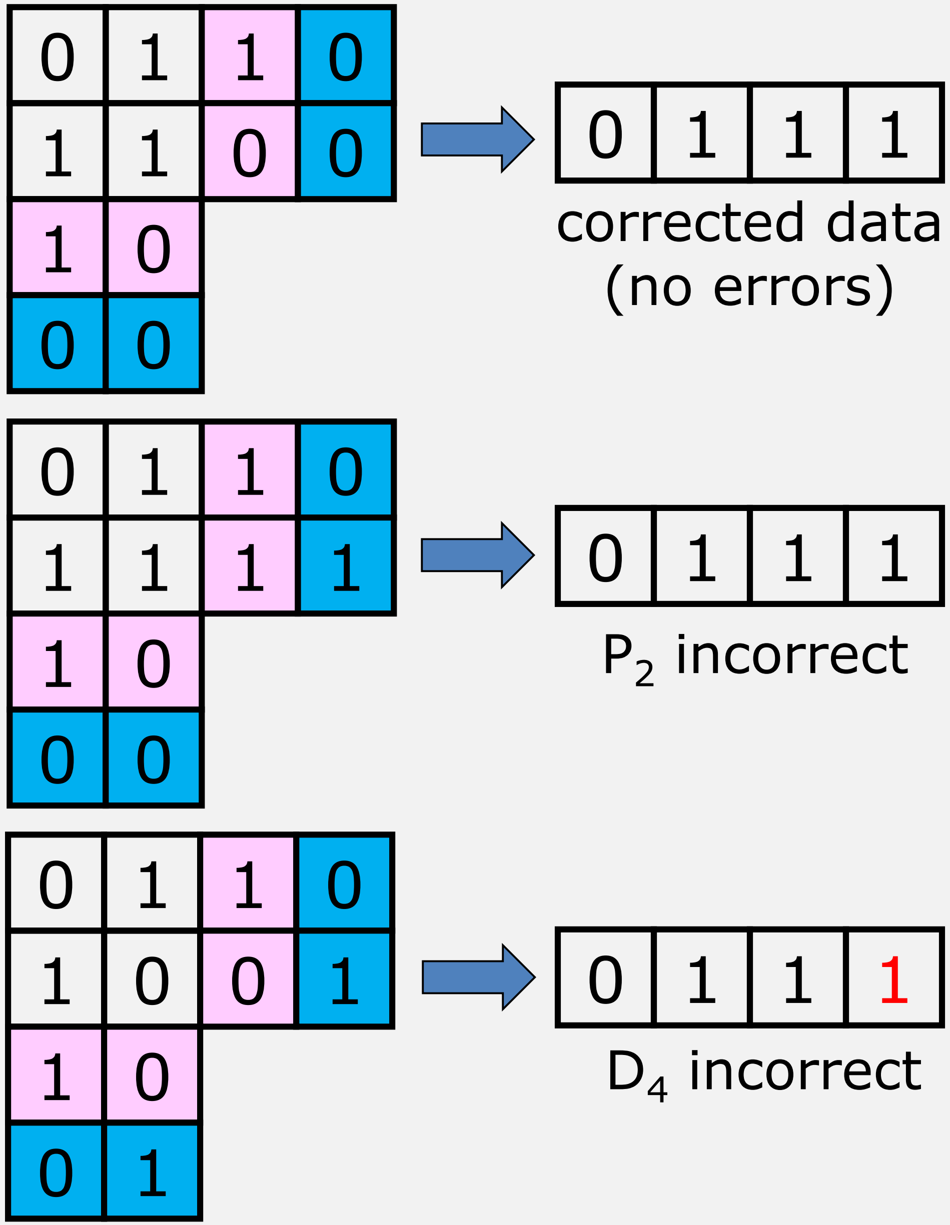
If $D_1 + D_2 + P_1$ is even
then $S_1 = 0$
else $S_1 = 1$

If $D_3 + D_4 + P_2$ is even
then $S_2 = 0$
else $S_2 = 1$

If $D_1 + D_3 + P_3$ is even
then $S_3 = 0$
else $S_3 = 1$

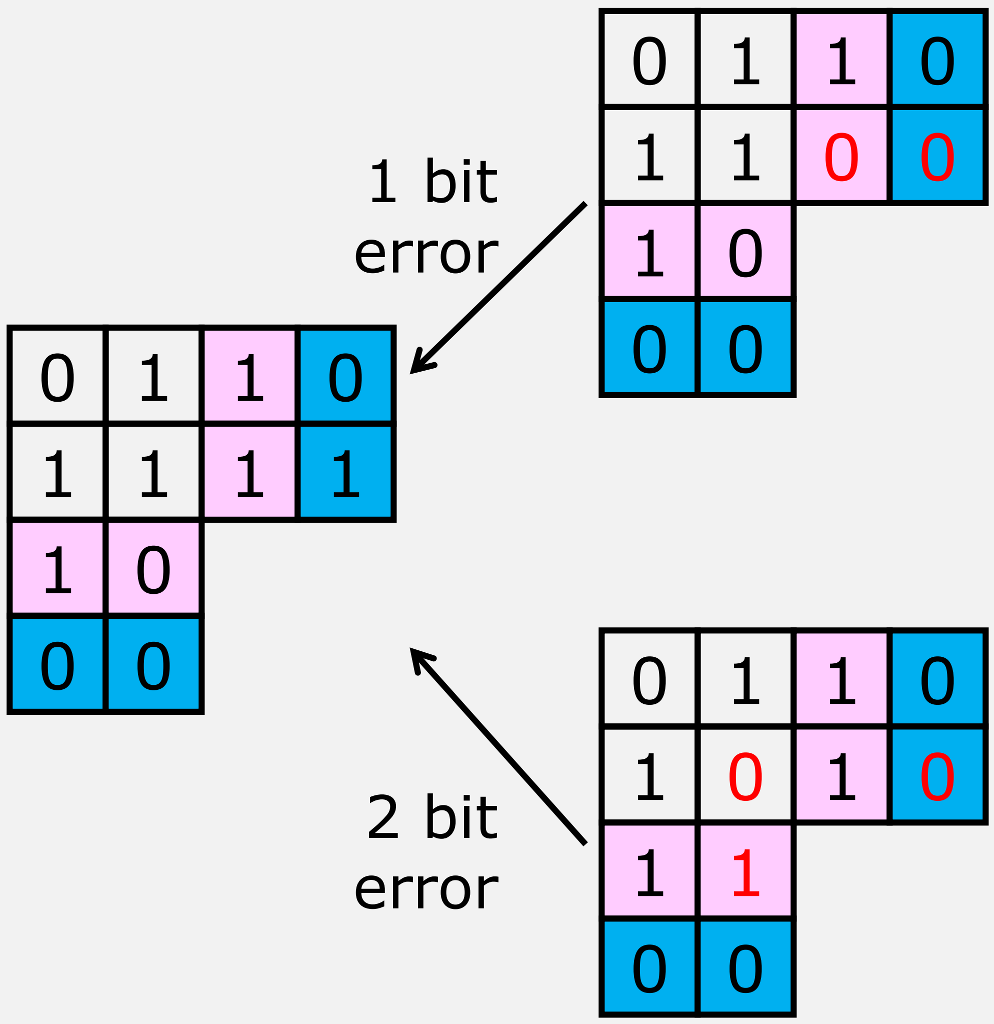
If $D_2 + D_4 + P_4$ is even
then $S_4 = 0$
else $S_4 = 1$

Performing Error Correction



- Since d=3, we can detect and correct (d-1)/2=1 bit errors.
- Check the syndrome bits
  - If all S_i = 0, we assume no error.
  - If only one S_i= 1, we assume an error in parity bit P_i.
  - If syndrome bits for column i and row j are 1, we assume a bit error in the data bit at position i,j

Performing Error Detection



- With the (8,4,3) code, since d=3, we can detect 1 or 2 bit errors.

- Check the syndrome bits
  - If all Si = 0, we assume no error.
  - Otherwise, we assume there has been an error in at least one bit

Error Correction Summary

011011101101

0110
1110
1101

01101111
11100101
11010110

01100111
11110101
11000110

0110 1110 1101

1. Take an input message stream:
2. Break the message stream into k-bit blocks (e.g. k = 4).
3. Add (n-k) parity bits to form n-bit codeword (e.g. n = 8)
4. Transmit data through noisy channel and receive codewords with some errors
5. Perform error correction
6. Extract the k=4 message bits from each corrected codeword.

Summary

- Noise, always present in communication systems, leads to bit errors.
- Error Correcting Codes can be used to reduce bit error rate.
- With (n,k,d) block codes, we use n bits to encode k message bits, where n > k.
- The minimum Hamming distance, d, between the codewords indicates how many bit errors the code can detect or correct.