Week 3: Hamming code

Hamming codes are a generalisation of the parity and Gray code ideas, to cover many bits, not just one.

- A Hamming distance of 3 can correct for 1 bit error
  – if 2 bit flips are possible, and we receive 011, we don't know if it should be 111 (one bit flipped) or 000 (two bits flipped)
- For detecting a single bit error, we use the formula:
  $2^p >= m + p + 1$
  m = number of data bits
  p = number of check bits
  to implement a Hamming Code. with m data bits and p check bits
  • the above example (000 and 111) would have m = 3, p = 3
  • So we need 7 bits to encode 3 bits of data
- e.g. assume we have 4 data bits, find the number of check bits in order to detect and correct a single bit error
- How many check bits for m=4 data bits?
  – try p = 2. So 2^p = 2^2 = 4,
  • but m + p + 1 = 4 + 2 + 1 = 7 and 4 is not >= 7
  – try p = 3. So 2^p = 2^3 = 8
  • m + p + 1 = 4 + 3 + 1 = 8
  • so 3 check bits are needed to encode 4 bits of data
- Check bits are usually interspersed with data bits in a set pattern
  – if they were all at the front (or end), we'd have to know in advance how long the original data bit string was
- For example, it is common to place check bits at positions that are powers of 2 (1,2,4,8,16 etc) from LSB to MSB

Bit Position	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Data/Parity	D6	D5	P4	D4	D3	D2	P3	D1	P2	P1

- Note that P1,P2,P3,P4 is sometimes written as P1,P2,P4,P8 (ie powers of 2, corresponding to their bit position)
  – This placement pattern allows a mathematical trick to be used to determine which parity bits protect which data bits
  – Consider P1: what binary numbers have a 1 at bit position 1?
  • 1, 11, 101, 111, 1001, 1011, 1101, 1111, 10001, …
  – Read those binary numbers as base 10, those are the bits being protected by P1
  • bit positions: 1, 3, 5, 7, 9, 11, 13, 15, 17, …
- example cont.
  – Consider P2: what binary numbers have a 1 at bit position 2?
  • 10, 11, 110, 111, 1010, 1011, 1110, 1111, 10010, …
  – Read those binary numbers as base 10, those are the bits being protected by P2
  • bit positions: 2, 3, 6, 7, 10, 11, 14, 15, 18
  – Consider P3: what binary numbers have a 1 at bit position 3?
  • 100, 101, 110, 111, 1100, 1101, 1110, 1111, 10100
  – Read those binary numbers as base 10, those are the bits being protected by P3
  • bit positions: 4, 5, 6, 7, 12, 13, 14, 15, 20, …
- example cont.
  – Consider P4: what binary numbers have a 1 at bit position 4?
  • 1000,1001,1010…
  – Read those binary numbers as base 10, those are the bits being protected by P2
  • bit positions: 8,9,10…
- So for example, bit position 5 (i.e. D2) is protected by both P1 and P3
  – if we were using even parity, and bit position 5 was 0, then P1 and P3 should both be 1
  – if bit position 5 was 1, then one of P1 or P3 would need to be 1
- e.g. determine the single bit error-correcting Hamming code required for the data 1011_2, using even parity – 1011 is 4 data bits, so from the formula we need p=3

Bit Position	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Data/Parity	D4	D3	D2	P3	D1	P2	P1
Data	1	0	1		1		
P1	1		1		1		?
P2	1	0			1	?	
P3	1	0	1	?			

  – P1 checks bits 1,3,5,7 = ?111. For even parity, P1 = 1
  – P2 checks bits 2,3,6,7 = ?101 so P2 = 0
  – P3 checks bits 4,5,6,7 = ?101 so P3 = 0
  – So the final code is 1010101

- e.g. assume we have a Hamming 4-bit (data) code with even parity. We have received code 1000100.
  – find and repair the error (if any)

Bit Position	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Data/Parity	D4	D3	D2	P3	D1	P2	P1
Data	1	0	0	0	1	0	0
P1	1		0		1		?
P2	1	0			1	?	
P3	1	0	0	?			

  – P1 checks bits 1,3,5,7 = 0101 so P1 is correctly = 0
  – P2 checks bits 2,3,6,7 = 0101 so P2 is correctly = 0
  – P3 checks bits 4,5,6,7 = 0001 so having P3 = 0 is error
  • error must be one of bits 4,5,6,7
  • bit 5 is correct (via P1 being correct).
  • bits 6 & 7 must be correct (as P1 and P2 are correct)
  • error must be bit 4 (a check bit)
  – corrected code is 1001100
  • correct data = 1001

- e.g. assume we have a Hamming 4-bit (data) code with even parity. We have received code 1000101.
  – find and repair the error (if any)

Bit Position	**7**	**6**	**5**	**4**	**3**	**2**	**1**
Data/Parity	D4	D3	D2	P3	D1	P2	P1
Data	1	0	0	0	1	0	1
P1	1		0		1		?
P2	1	0			1	?	
P3	1	0	0	?			

  – P1 checks bits 1,3,5,7 = 1101 so P1 is in error
  – P2 checks bits 2,3,6,7 = 0101 so P2 is correctly = 0
  – P3 checks bits 4,5,6,7 = 0001 so P3 is in error
  • P1 and P3 both check bits 5 and 7
  • but P2 also checks bit 7, and P2 is OK
  • error must be bit 5
  – corrected code is 1010101

    - corrected data = 1011
  - Because of where the check bits are positioned, there is a shortcut
    - Simply add the bit value for the P's in error. More later…..