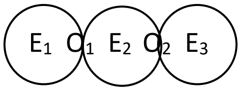
1. Parity
   a. Errors with a Hamming distance of 1

   b. *Even parity*

   c. *Odd parity*

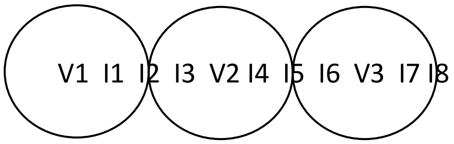
   d. Example for even parity
      i. C denotes the position of the check bit
      ii. C1001 ->
      iii. C1101 ->

   e. Even parity creates valid code words



Circles have a radius of one Hamming distance.

   f. Two bit errors



Circles have a radius of two Hamming distance.

2. Further bit checking

   a. Will look at Hamming(7,4) error correction

   b. Example: 4-bit word

111	110	101	100	011	010	001	Bit position (binary)
7	6	5	4	3	2	1	Bit position (decimal)
							Bit type (D = data, C = check / parity)

      i. C0

      ii. C1

      iii. C2

3. Hamming(7,4) examples

   a. Original data: 0110

      i. Let's calculate the code word associated with it

         1. Fill in table, then calculate check bits

111	110	101	100	011	010	001
7	6	5	4	3	2	1

b. Let’s flip one of the bits now

i. Is 0010011 valid?

1. Fill in the table, then verify check bits

111	110	101	100	011	010	001
7	6	5	4	3	2	1

ii. Was codeword valid?

c. Error correction portion

i. We know there was an error, how do we fix it?

4. How Hamming(7,4) works
    a. Each data bit must be covered (checked) by at least 2 parity bits
    b. Why we start numbering from 1
    c. Which parity bits check which data bits
        i. C0
        ii. C1
        iii. C2
        iv. If we had a C3
        v. In general
    d. Why we can XOR to determine the bit position
    e. Expanding it further
        i. Expanding to cover a larger data size
        ii. Covering more errors
            1. Given T errors: