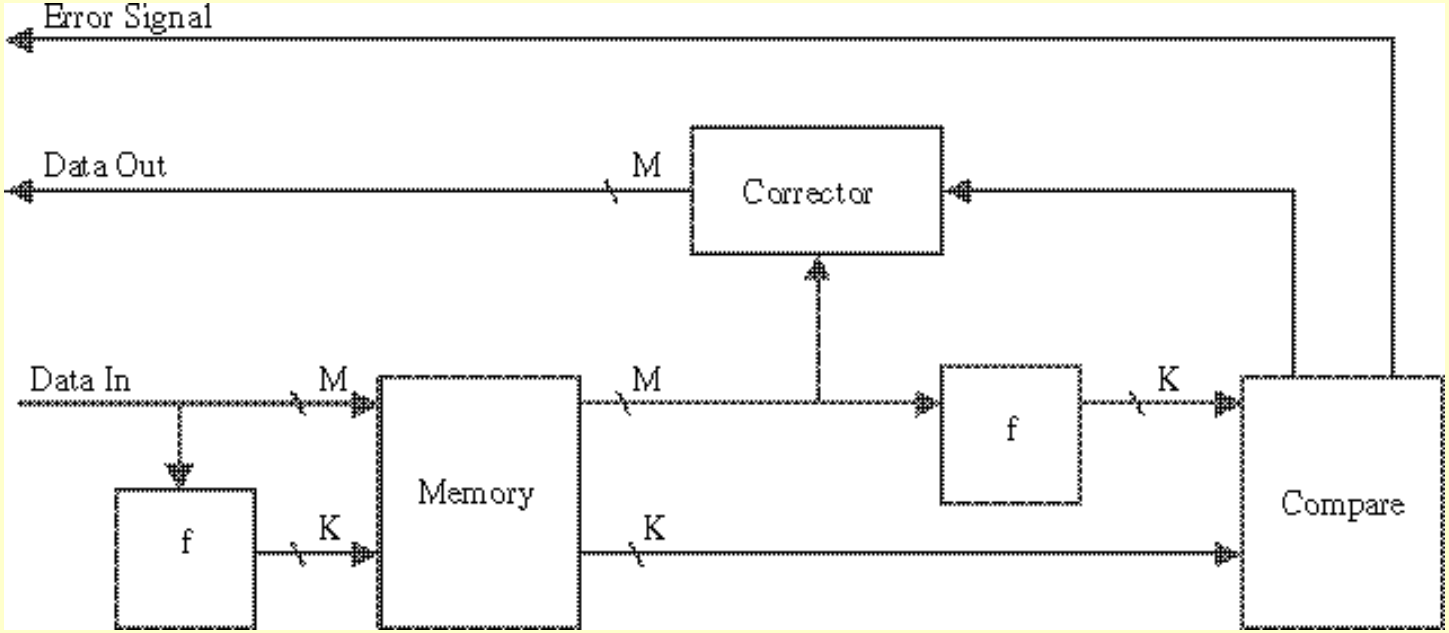
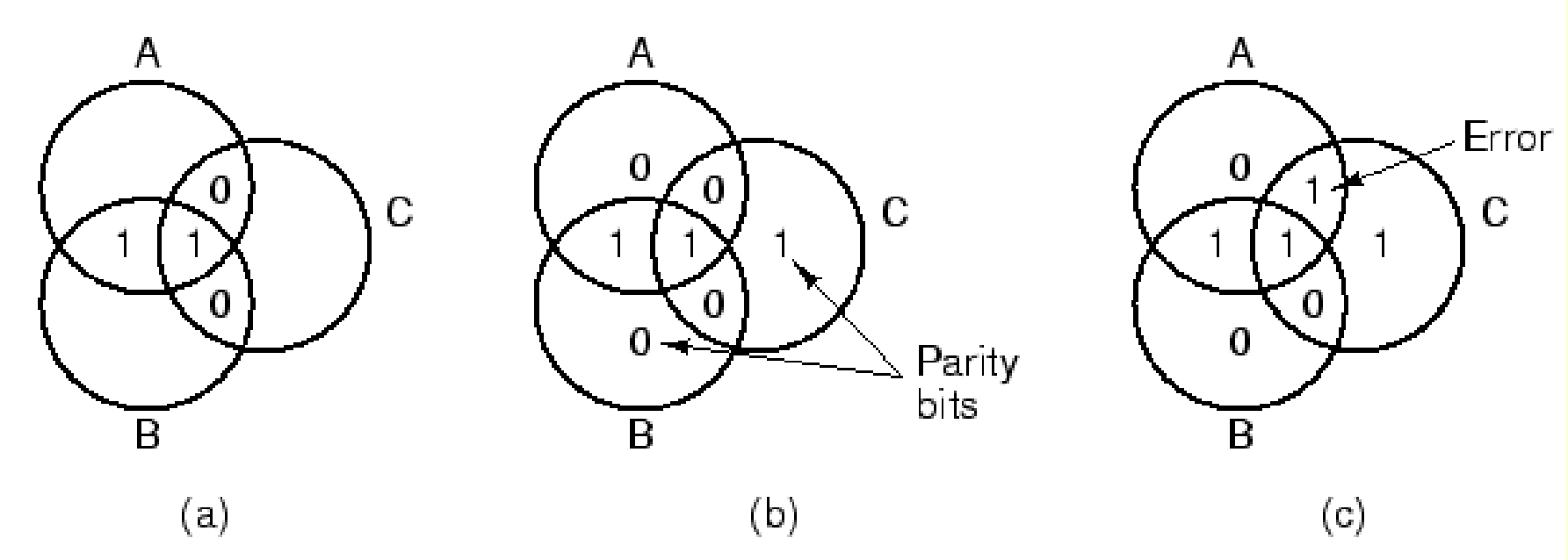
Error Detection And Correction (EDAC)



- Semiconductor memory is prone to errors both **soft** (transient) and **hard** (permanent). Errors can be both detected and corrected (within certain limitations) by employing appropriate logic in the system
  - The function **f** is applied to the **M** bits being stored and this generates a **K** bit code which is stored along with the data. On readout the same function is applied to the retrieved M bits and the resulting K-bit code is compared to the retrieved code. **Either**:
    - No error is detected. The M fetched bits are sent, **or**
    - An correctable error is detected. The data bits plus error-correcting information are sent to the corrector which reconstructs the original M bits, **or**
    - A non-correctable error is detected. The error is reported.

Hamming Codes



A **Hamming code** operates as follows:

a)The data bits are each assigned to one of the inner areas of a **Venn diagram**.

b)Parity bits (aka **check bits**) are assigned so that each circle contains an even number of 1s.

c)If a single-bit error occurs, it is easy to determine which is the bad bit

d)Which can then be inverted to restore the original data

Hamming Codes

- The following layout of data and check bits satisfies the three conditions:

Bit_{10}	$Position_2$	Check	Data
12	1100		M8
11	1011		M7
10	1010		M6
9	1001		M5
8	1000	C8	
7	0111		M4
6	0110		M3
5	0101		M2
4	0100	C4	
3	0011		M1
2	0010	C2	
1	0001	C1	

Bits whose positions are powers of two are designated as check bits. They are calculated as follows (# indicates an exclusive-or operation, which is, of course, associative, so no brackets are needed):

C1 = M1 # M2 # M4 # M5 # M7
C2 = M1 # M3 # M4 # M6 # M7
C4 = M2 # M3 # M4 # M8
C8 = M5 # M6 # M7 # M8

Each check bit operates on every data bit position whose position number contains a 1 in the corresponding column position. So, data bit positions 3, 5, 7, 9 and 11 all contain the term 2^0 and are checked by check bit 1. Alternatively, bit position **n** is checked by all those bits C**i** such that the **i**s sum to **n**: e.g. bit 11is checked by C1, C2 and C8 (because 8 + 2 + 1 = 11).

Hamming Codes

- An Example

  The 8-bit input word is 00111001 and bit M1 is the rightmost bit.

  The check bit calculations are then:

  C1 = 1 # 0 # 1 # 1 # 0 = 1
  C2 = 1 # 0 # 1 # 1 # 0 = 1
  C4 = 0 # 0 # 1 # 0 = 1
  C8 = 1 # 1 # 0 # 0 = 0

  If data bit 4 is erroneously changed from 1 to 0, the recalculation of the check produces:

  C1 = 1 # 0 # 0 # 1 # 0 = 0
  C2 = 1 # 0 # 0 # 1 # 0 = 0
  C4 = 0 # 0 # 0 # 0 = 0
  C8 = 1 # 1 # 0 # 0 = 0

  The syndrome word is produced by XORing the two sets of check bits together:

	C8	C4	C2	C1
	0	1	1	1
#	0	0	0	0
=>	0	1	1	1

  The resultant value, 0111, indicates that bit position 7, i.e. data bit 4, is incorrect.

  The code just described is known as an **SEC** (**Single Error Correcting**) code.

Hamming Codes

- Detecting multiple-bit errors
  - An SEC (Single Error Correcting) Code cannot cope with multiple-bit errors. However, the SEC can be extended so that it can not only correct single-bit errors but also detect (but **not** correct) double-bit errors. This is called a **Single Error Correcting-Double Error Detecting** (**SEC-DED**) Hamming Code.
    (a) Inner segments are assigned the data bits
    (b) Outer segments are calculated, as is extra bit.
    (c) Two bits are garbled during retrieval.
    (d) SEC algorithm compounds the error: identifies a correct bit as erroneous.
    (e) Correcting the "error" bit makes the parities within the circles correct
    (f) Now the overall parity bit is wrong; the algorithm signals an unrecoverable error.