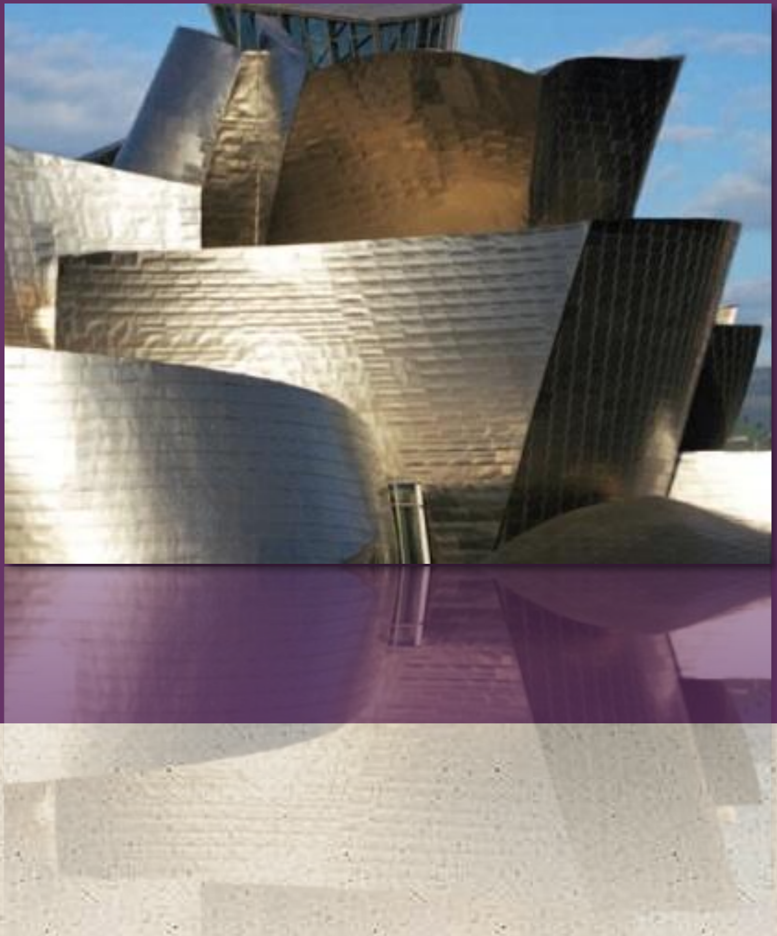
+



William Stallings
Computer Organization and Architecture
10th Edition

+

Internal Memory

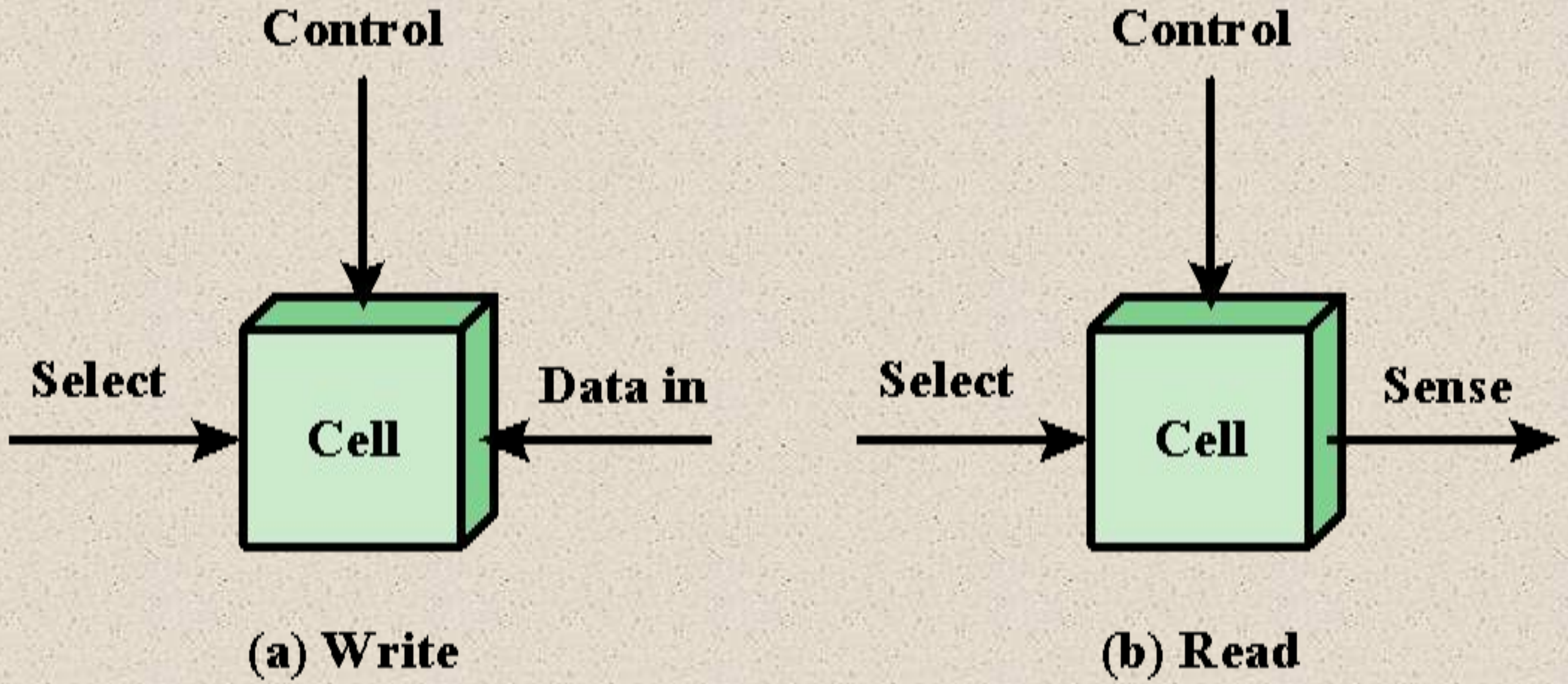
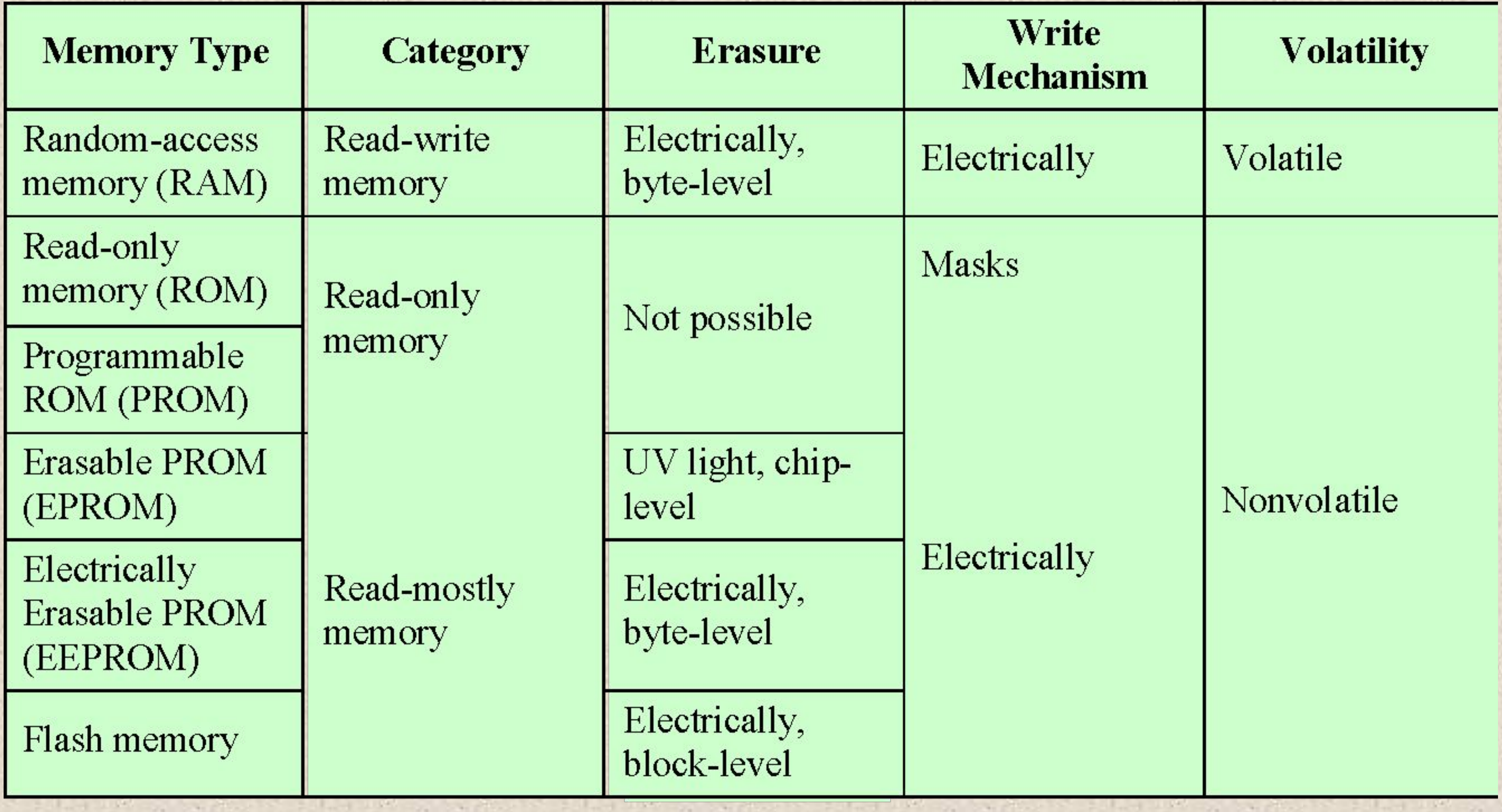


Figure 5.1 Memory Cell Operation



Memory Type	Category	Erasure	Write Mechanism	Volatility
Random-access memory (RAM)	Read-write memory	Electrically, byte-level	Electrically	Volatile
Read-only memory (ROM)	Read-only memory	Not possible	Masks	Nonvolatile
Programmable ROM (PROM)			Electrically	
Erasable PROM (EPROM)	Read-mostly memory	UV light, chip-level		
Electrically Erasable PROM (EEPROM)		Electrically, byte-level		
Flash memory		Electrically, block-level		

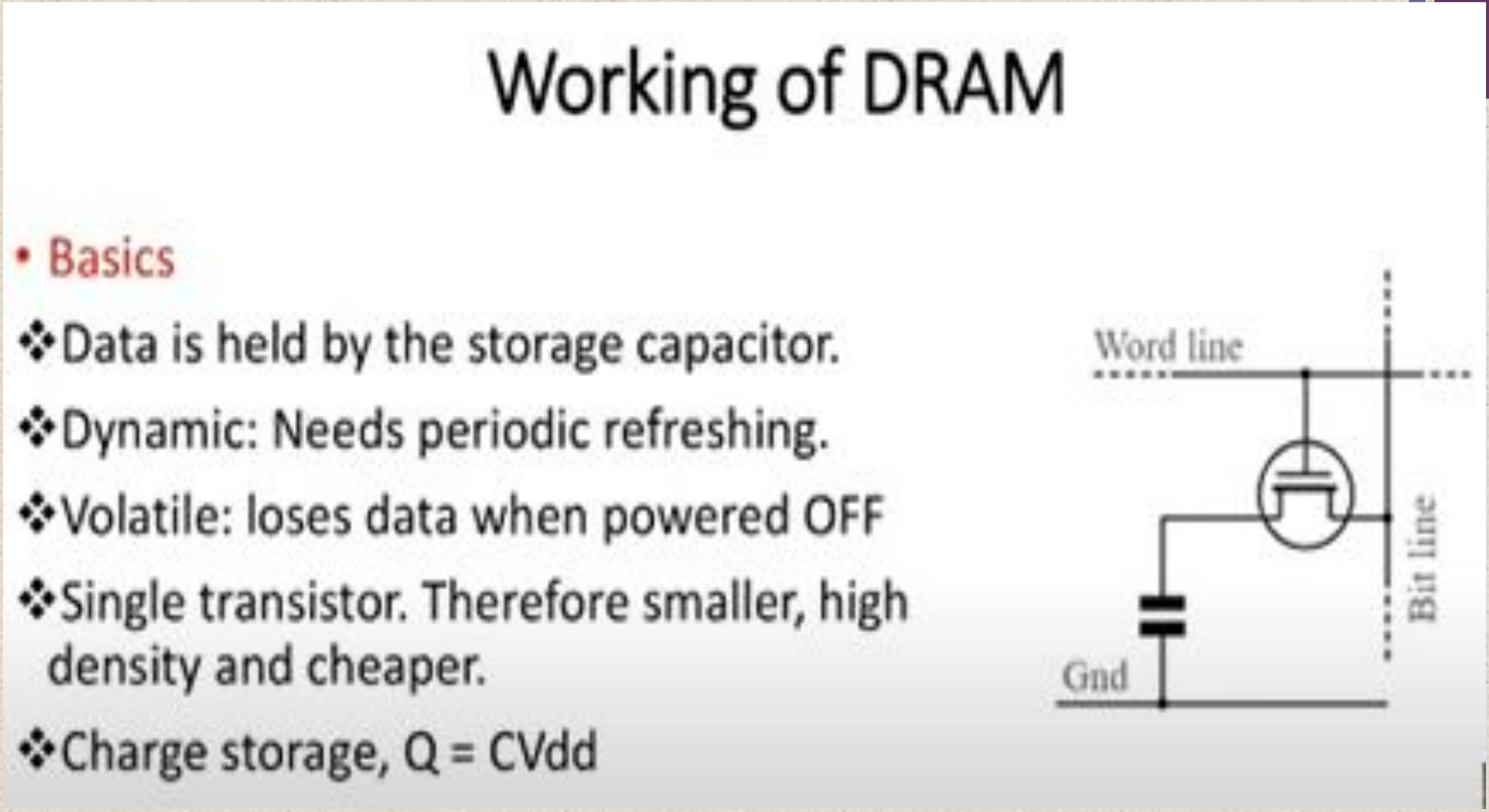
Table 5.1
Semiconductor Memory Types

Dynamic RAM (DRAM)

- RAM technology is divided into two technologies:
  - Dynamic RAM (DRAM)
  - Static RAM (SRAM)
- DRAM
  - Made with cells that store data as charge on capacitors
  - Presence or absence of charge in a capacitor is interpreted as a binary 1 or 0
  - Requires periodic charge refreshing to maintain data storage
  - The term *dynamic* refers to tendency of the stored charge to leak away, even with power continuously applied

+

Dynamic RAM (DRAM)



Working of DRAM

- Basics

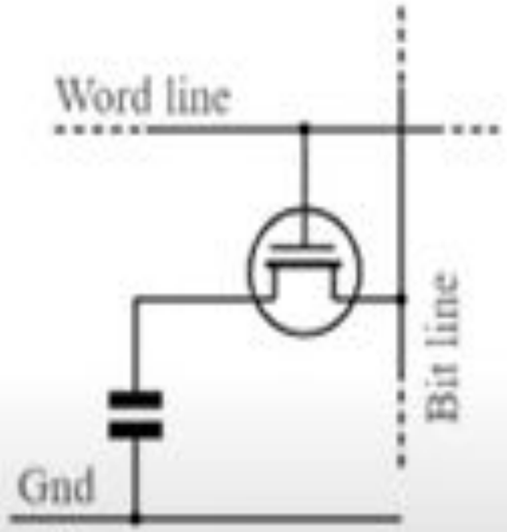
❖Data is held by the storage capacitor.

❖Dynamic: Needs periodic refreshing.

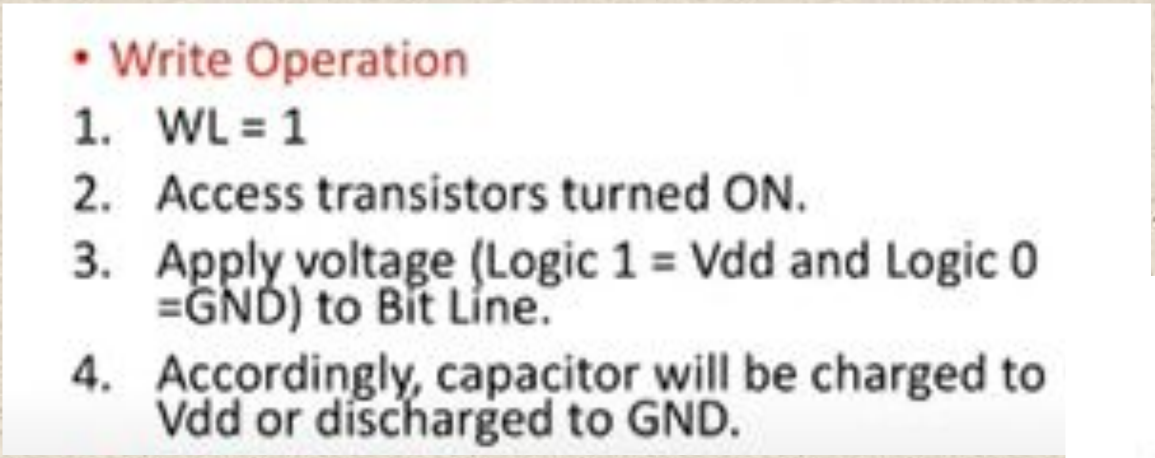
❖Volatile: loses data when powered OFF

❖Single transistor. Therefore smaller, high density and cheaper.

❖Charge storage, Q = CVdd

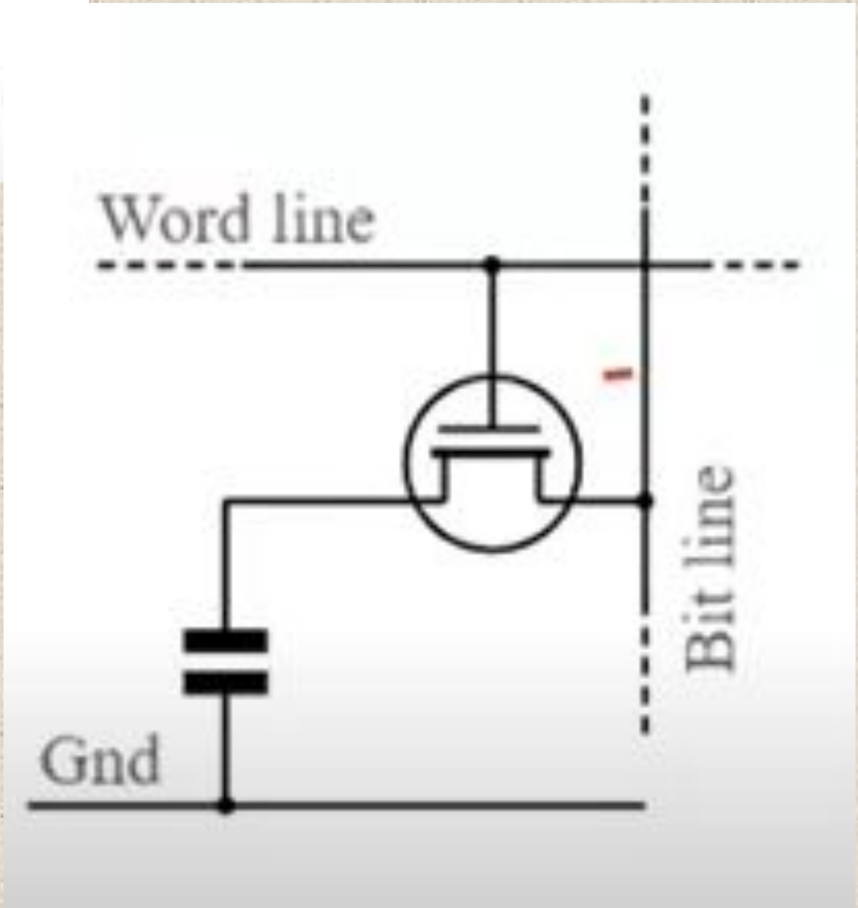


Dynamic RAM (DRAM)

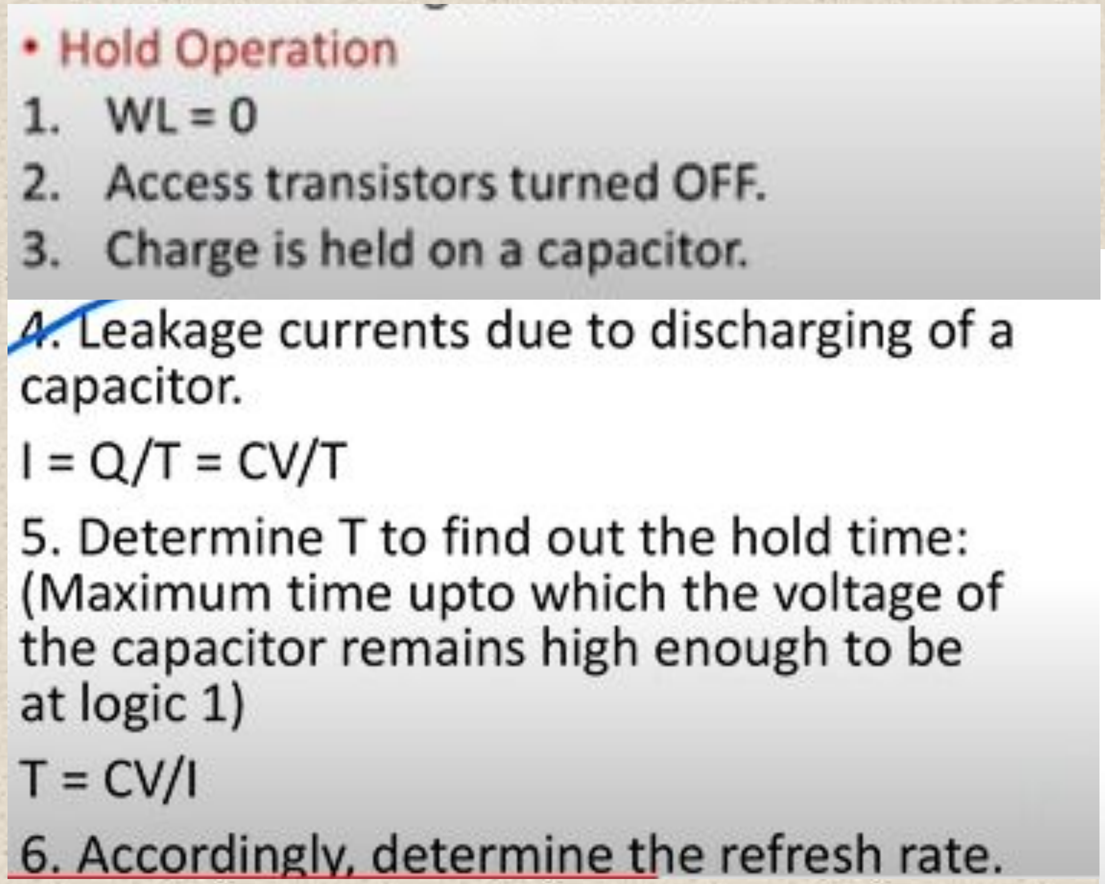


- Write Operation

1. WL = 1
2. Access transistors turned ON.
3. Apply voltage (Logic 1 = Vdd and Logic 0 =GND) to Bit Line.
4. Accordingly, capacitor will be charged to Vdd or discharged to GND.



Dynamic RAM (DRAM)



- Hold Operation

1. WL = 0
2. Access transistors turned OFF.
3. Charge is held on a capacitor.

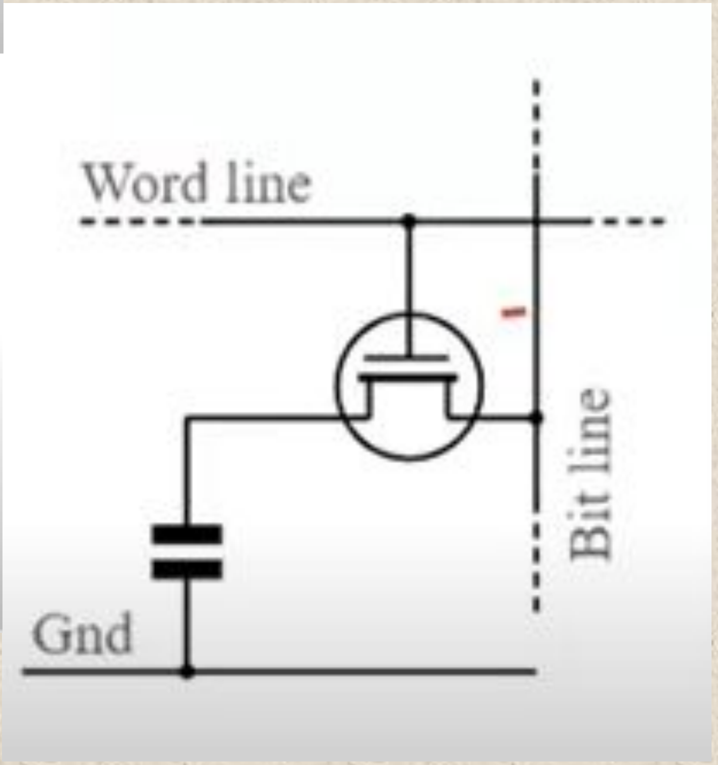
4. Leakage currents due to discharging of a capacitor.

$I = Q/T = CV/T$

5. Determine T to find out the hold time: (Maximum time upto which the voltage of the capacitor remains high enough to be at logic 1)

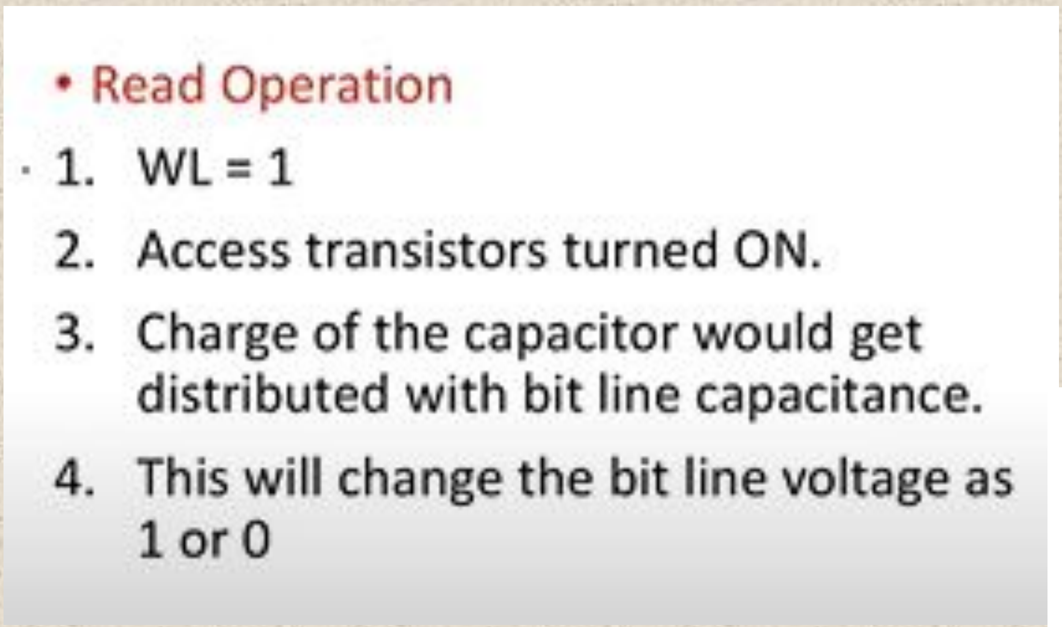
$T = CV/I$

6. Accordingly, determine the refresh rate.



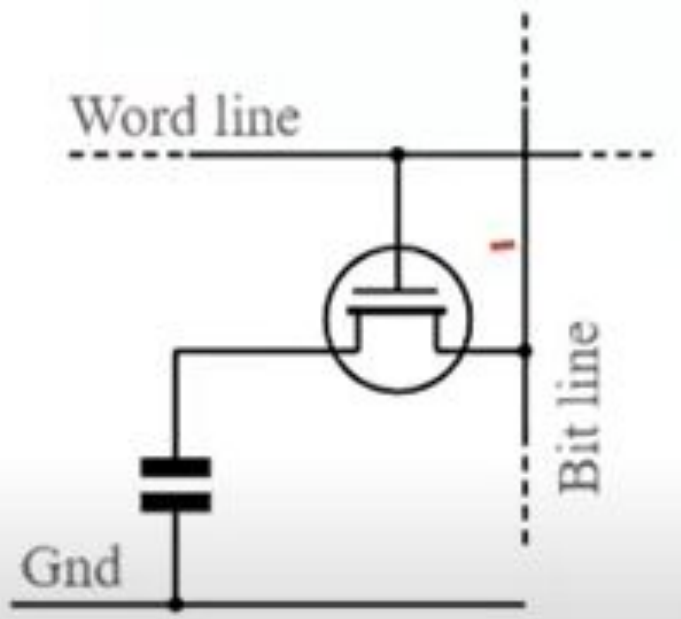
+

Dynamic RAM (DRAM)



- Read Operation

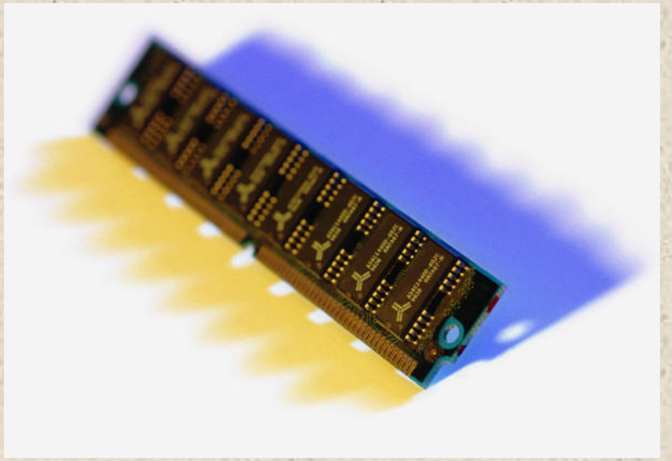
1. WL = 1
2. Access transistors turned ON.
3. Charge of the capacitor would get distributed with bit line capacitance.
4. This will change the bit line voltage as 1 or 0



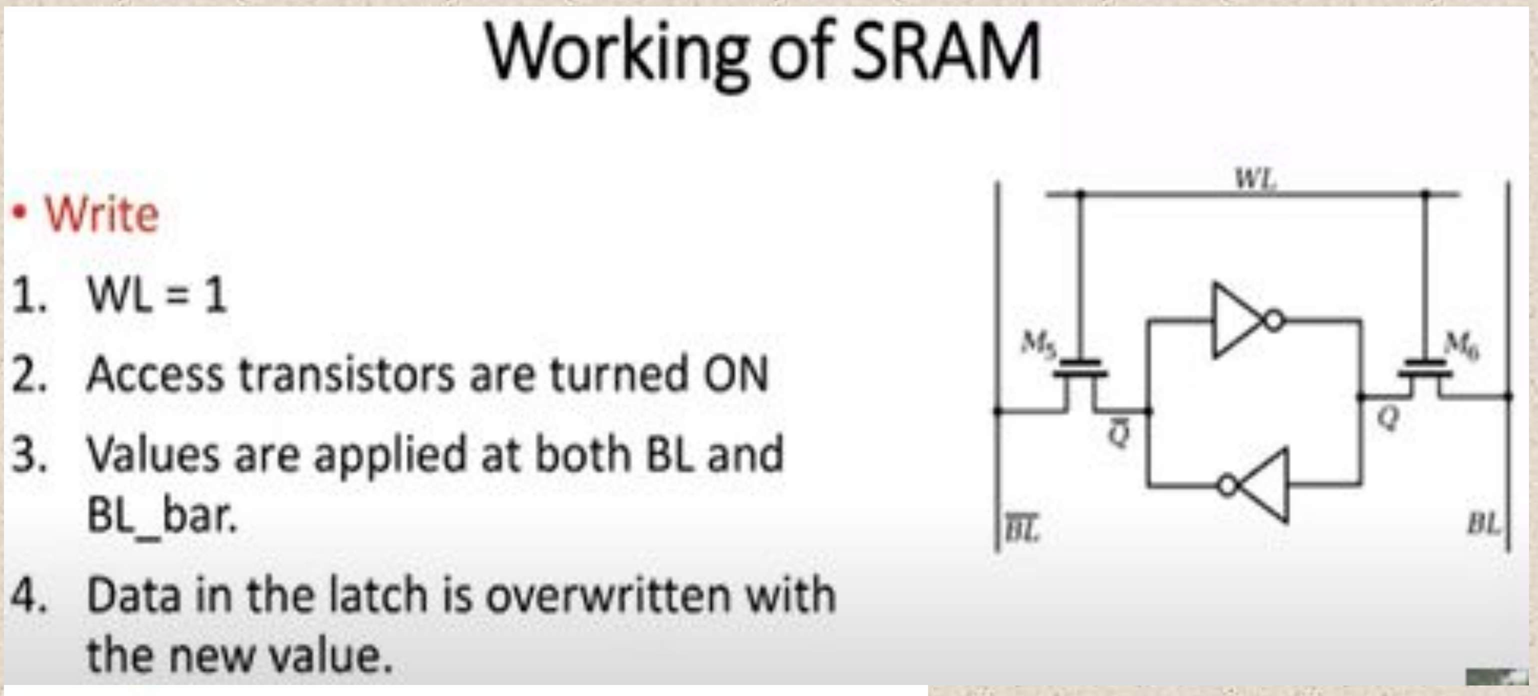
+

Static RAM (SRAM)

- Digital device that uses the same logic elements used in the processor
- Binary values are stored using traditional flip-flop logic gate configurations
- Will hold its data as long as power is supplied to it



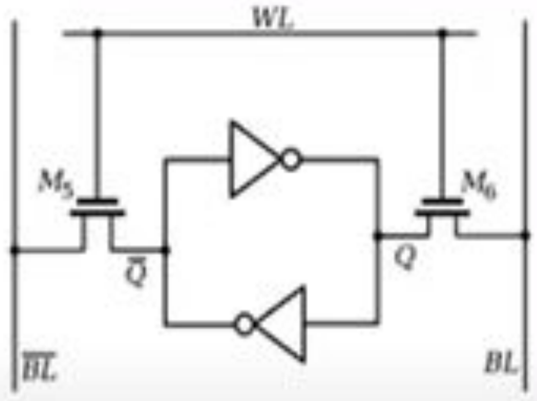
Static RAM (SRAM)

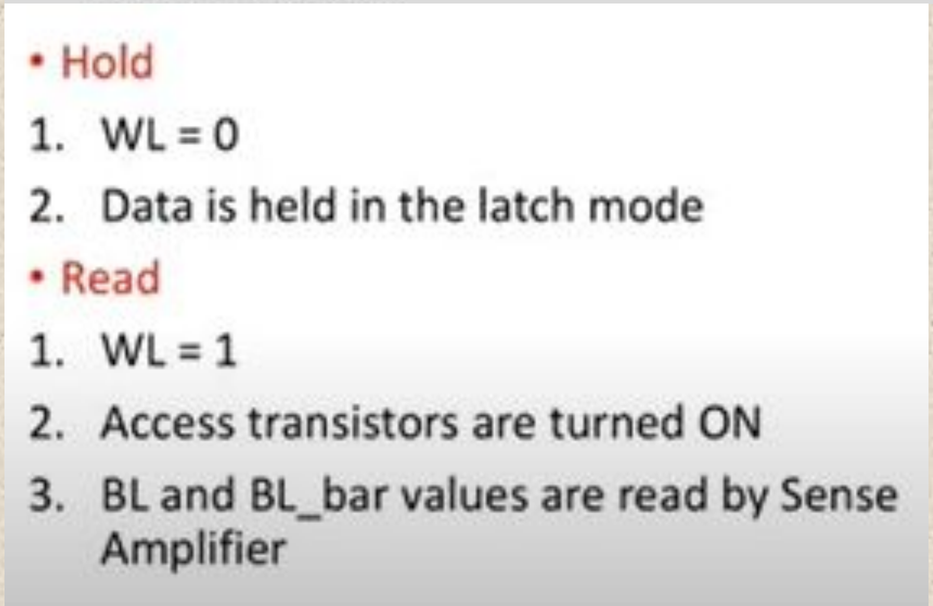


Working of SRAM

- Write

1. WL = 1
2. Access transistors are turned ON
3. Values are applied at both BL and BL_bar.
4. Data in the latch is overwritten with the new value.





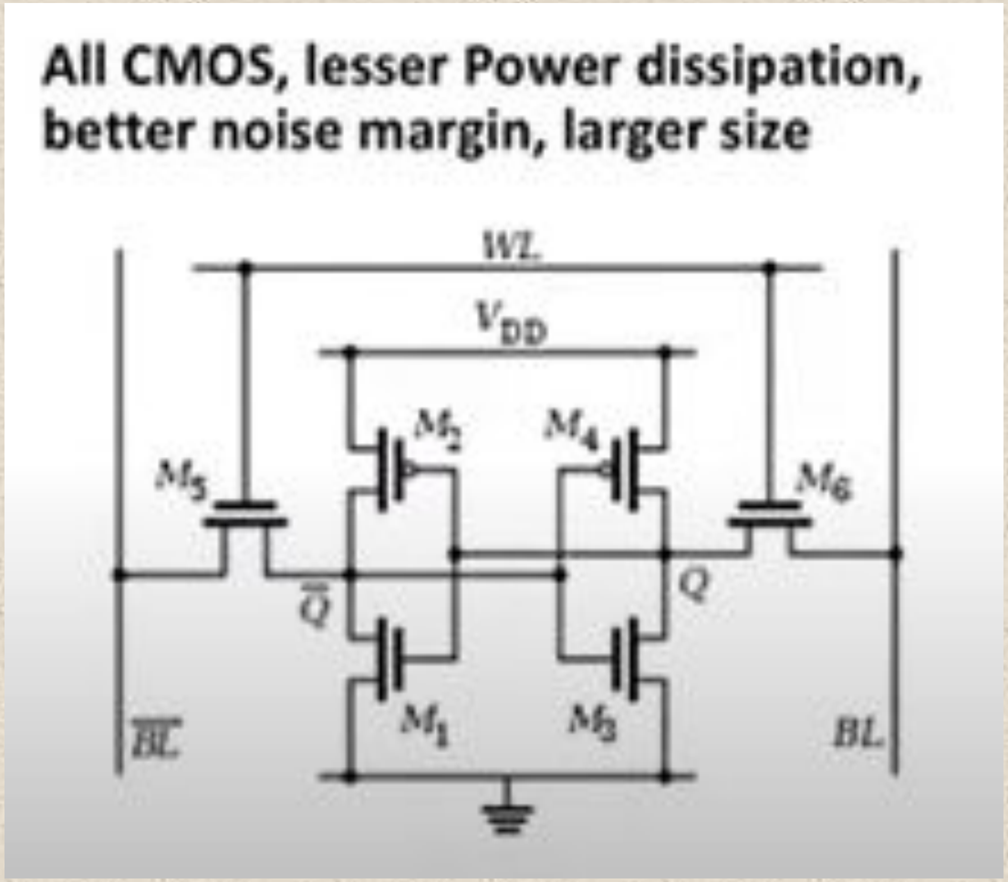
- Hold

1. WL = 0
2. Data is held in the latch mode

- Read

1. WL = 1
2. Access transistors are turned ON
3. BL and BL_bar values are read by Sense Amplifier

Static RAM (SRAM)

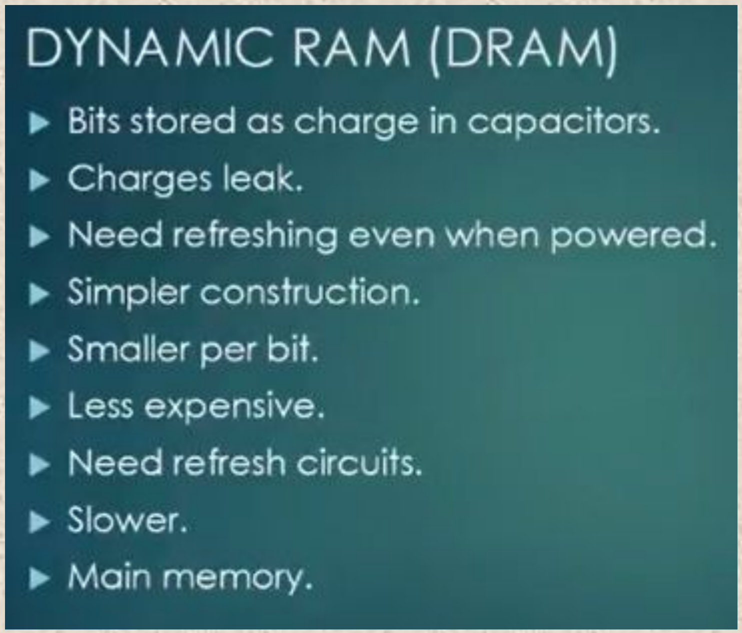


All CMOS, lesser Power dissipation, better noise margin, larger size

SRAM versus DRAM

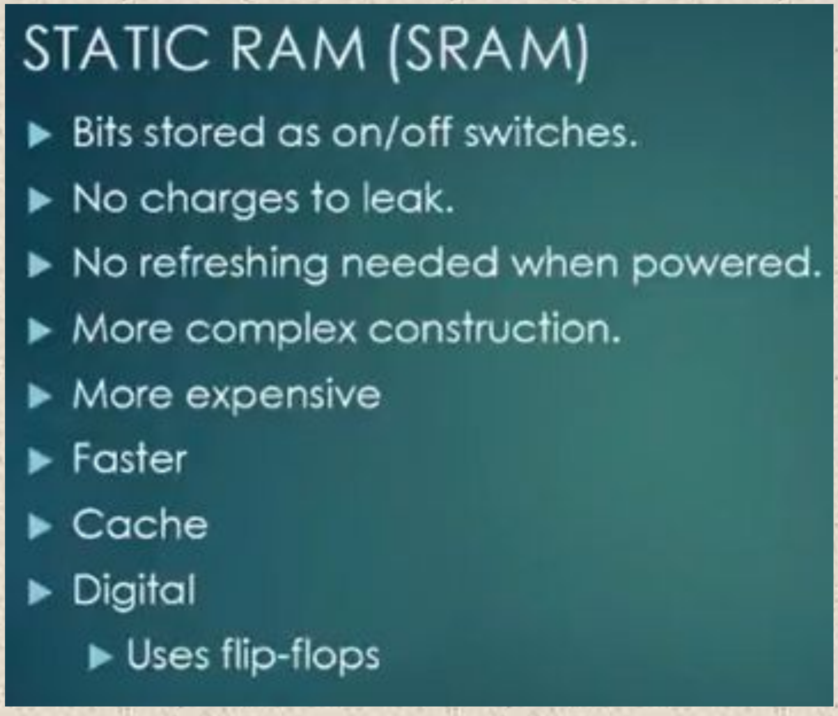
- Both volatile
  - Power must be continuously supplied to the memory to preserve the bit values

SRA
M



DYNAMIC RAM (DRAM)

- Bits stored as charge in capacitors.
- Charges leak.
- Need refreshing even when powered.
- Simpler construction.
- Smaller per bit.
- Less expensive.
- Need refresh circuits.
- Slower.
- Main memory.



STATIC RAM (SRAM)

- Bits stored as on/off switches.
- No charges to leak.
- No refreshing needed when powered.
- More complex construction.
- More expensive
- Faster
- Cache
- Digital
  - Uses flip-flops

DRA
M

Astatic RAM (SRAM) is a digital device that uses the same logic elements used in the processor.

In a SRAM, **binary values are stored using traditional flip-flop logic-gate** configurations

SRAM structure for an individual cell. Four transistors are **(T_1 T_2 T_3 T_4)** cross connected in an **arrangement that produces a stable logic state.**

In logic state 1, C_1 point is high and C_2 point is low; in this state, **T_1** and **T_4** are off and **T_2** and **T_3** are on.

In logic state 0, C_2 point is high and C_1 point is low; in this state, **T_1** and **T_4** are on and **T_2** and **T_3** are off.

Both states are stable as long as the direct current (dc) voltage is applied. Unlike the DRAM, no refresh is needed to retain data.

Astatic RAM (SRAM) is a digital device that uses the same logic elements used in the processor.

In a SRAM, **binary values are stored using traditional flip-flop logic-gate** configurations

SRAM structure for an individual cell. Four transistors are **(T_1 T_2 T_3 T_4)** cross connected in an **arrangement that produces a stable logic state.**

In logic state 1, C_1 point is high and C_2 point is low; in this state, **T_1** and **T_4** are off and **T_2** and **T_3** are on.

In logic state 0, C_2 point is high and C_1 point is low; in this state, **T_1** and **T_4** are on and **T_2** and **T_3** are off.

Both states are stable as long as the direct current (dc) voltage is applied. Unlike the DRAM, no refresh is needed to retain data.

SRAM **address line is used to open or close a switch**.

The address line controls two transistors **(T_5 and T_6).**

When a signal is applied to this line, **the two transistors are switched on**, allowing a **read or write operation**.

+

Error Correction

The simplest of the error-correcting codes is the Hamming code devised by Richard Hamming at Bell Laboratories.

- **Hard Failure**
  - Permanent physical defect
  - Memory cell or cells affected cannot reliably store data but become stuck at 0 or 1 or switch erratically between 0 and 1
  - Can be caused by:
    - Harsh environmental abuse
    - Manufacturing defects
    - Wear
- **Soft Error**
  - Random, non-destructive event that alters the contents of one or more memory cells
  - No permanent damage to memory
  - Can be caused by:
    - Power supply problems
    - Alpha particles

Error Correction

Hamming code is **a block code that is capable of detecting up to two simultaneous bit errors and correcting single-bit errors**.

In Hamming code, the source encodes the message by adding redundant bits in the message. These redundant bits are mostly inserted and generated at certain positions in the message to accomplish error detection and correction process.

Application of Hamming code

Satellites

Computer Memory

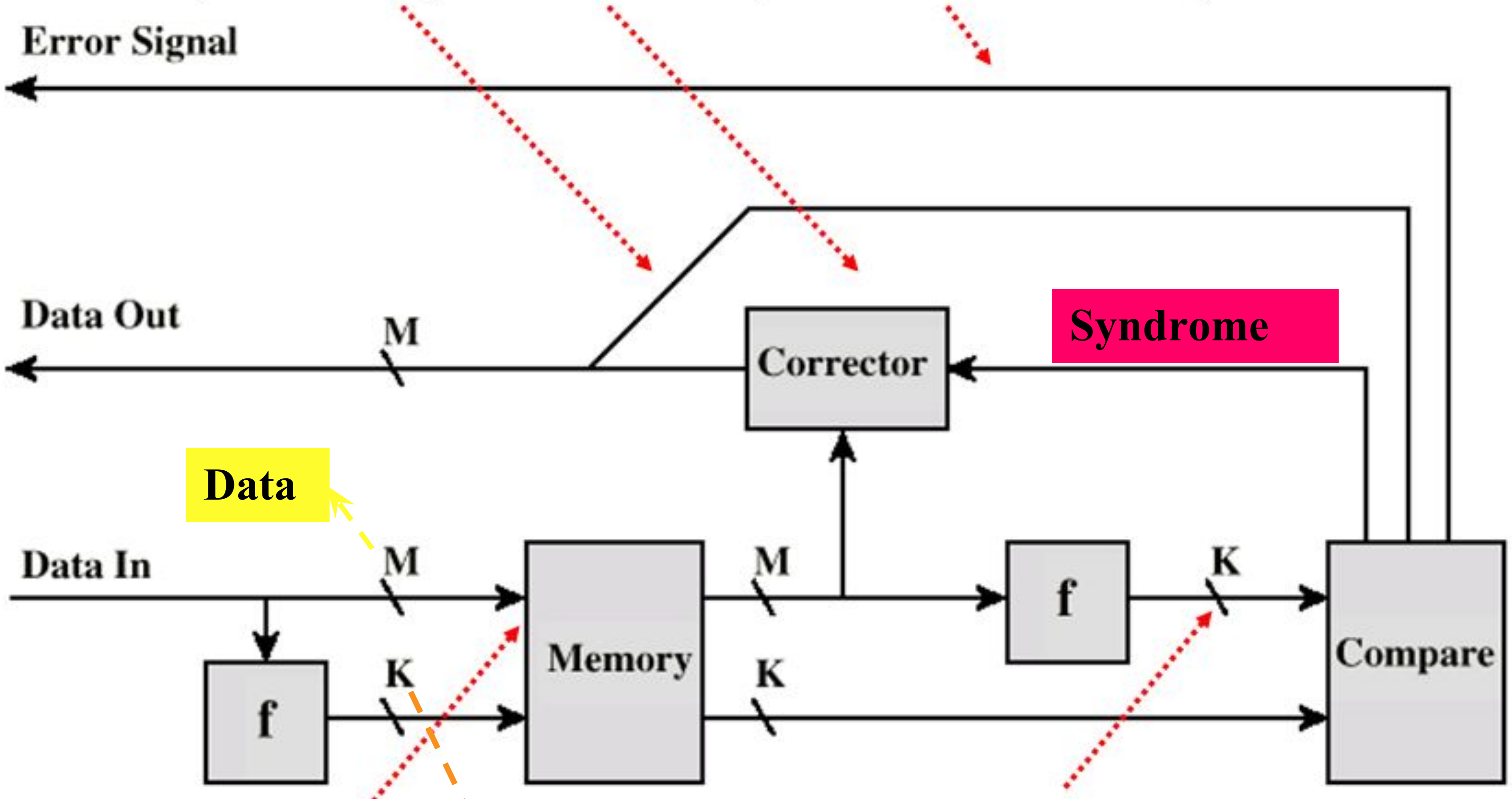
Modems

Embedded Processor

Error Correcting Code Function

Comparison yields 3 results

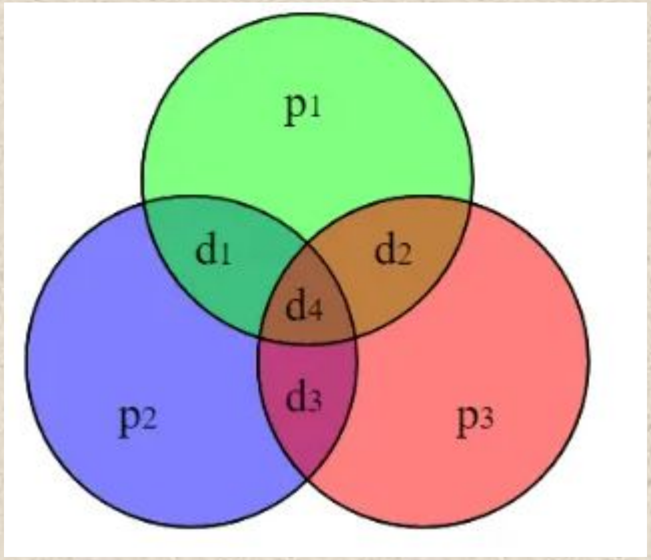
a) No errors b) corrected errors c) errors detected but not possible to correct it.



Both data (M bits) and code generated by f (K bits) are stored.

During fetch, new K code bits generated from the M data bits by f and compared with fetched code bits

Venn diagrams to illustrate the use of this code on 4-bit words



Data : 1110

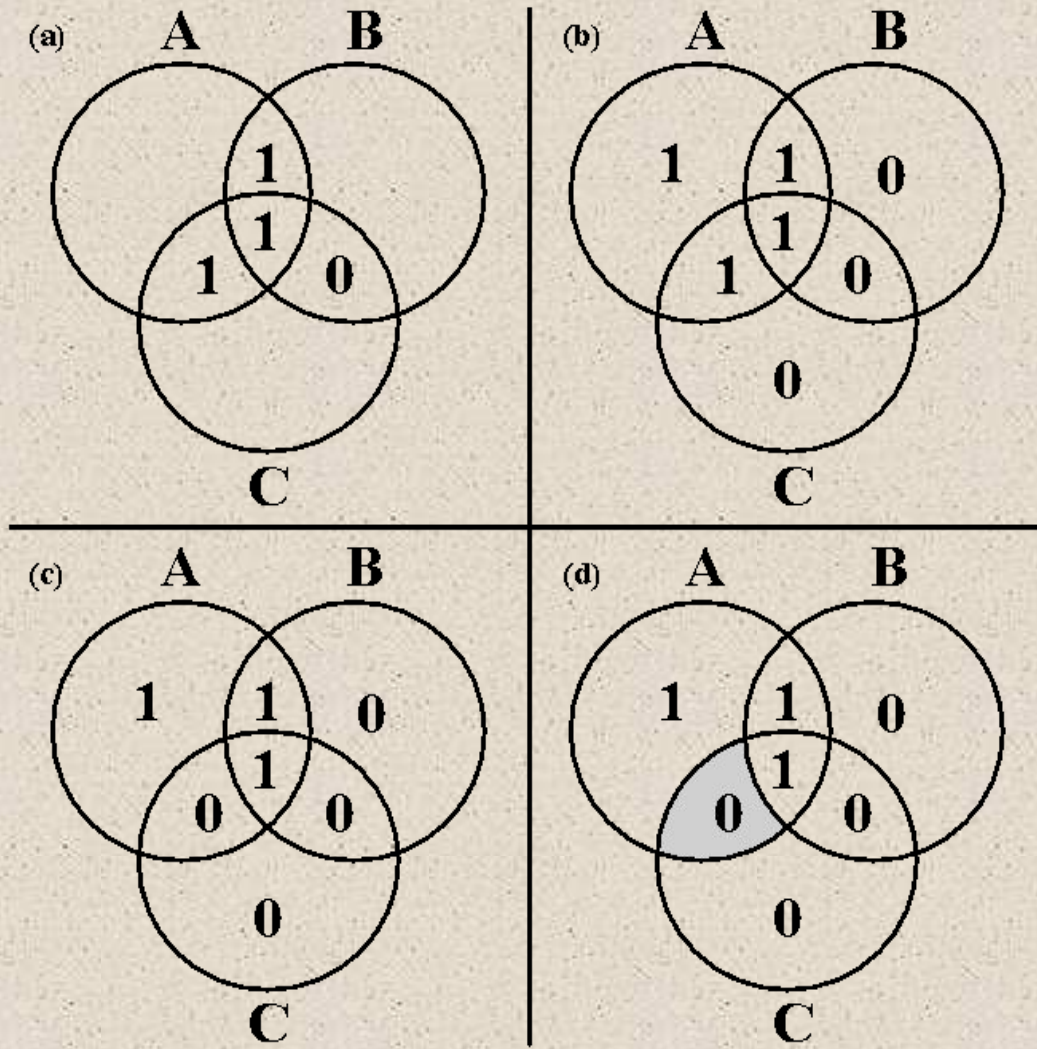
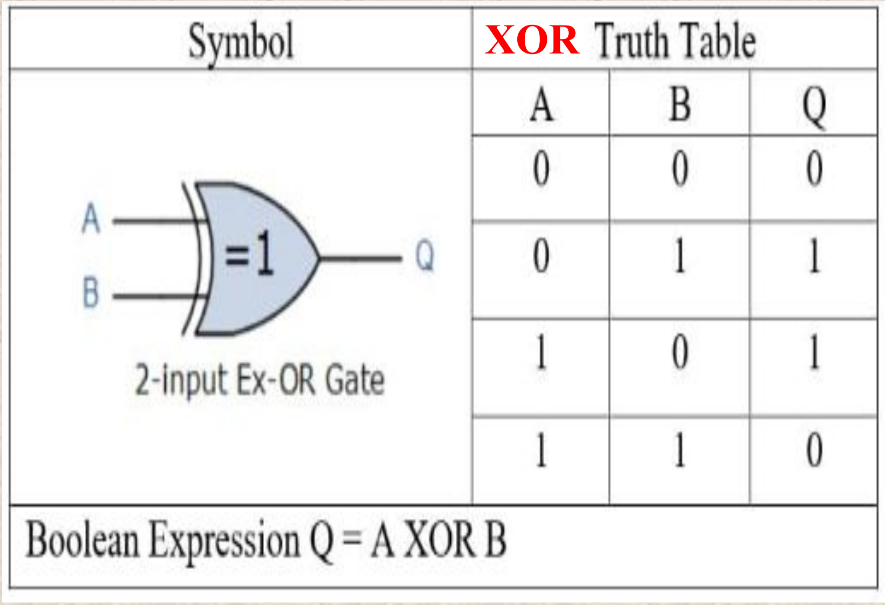


Figure 5.8 Hamming Error-Correcting Code

General Single-Bit Error Correction(SEC)

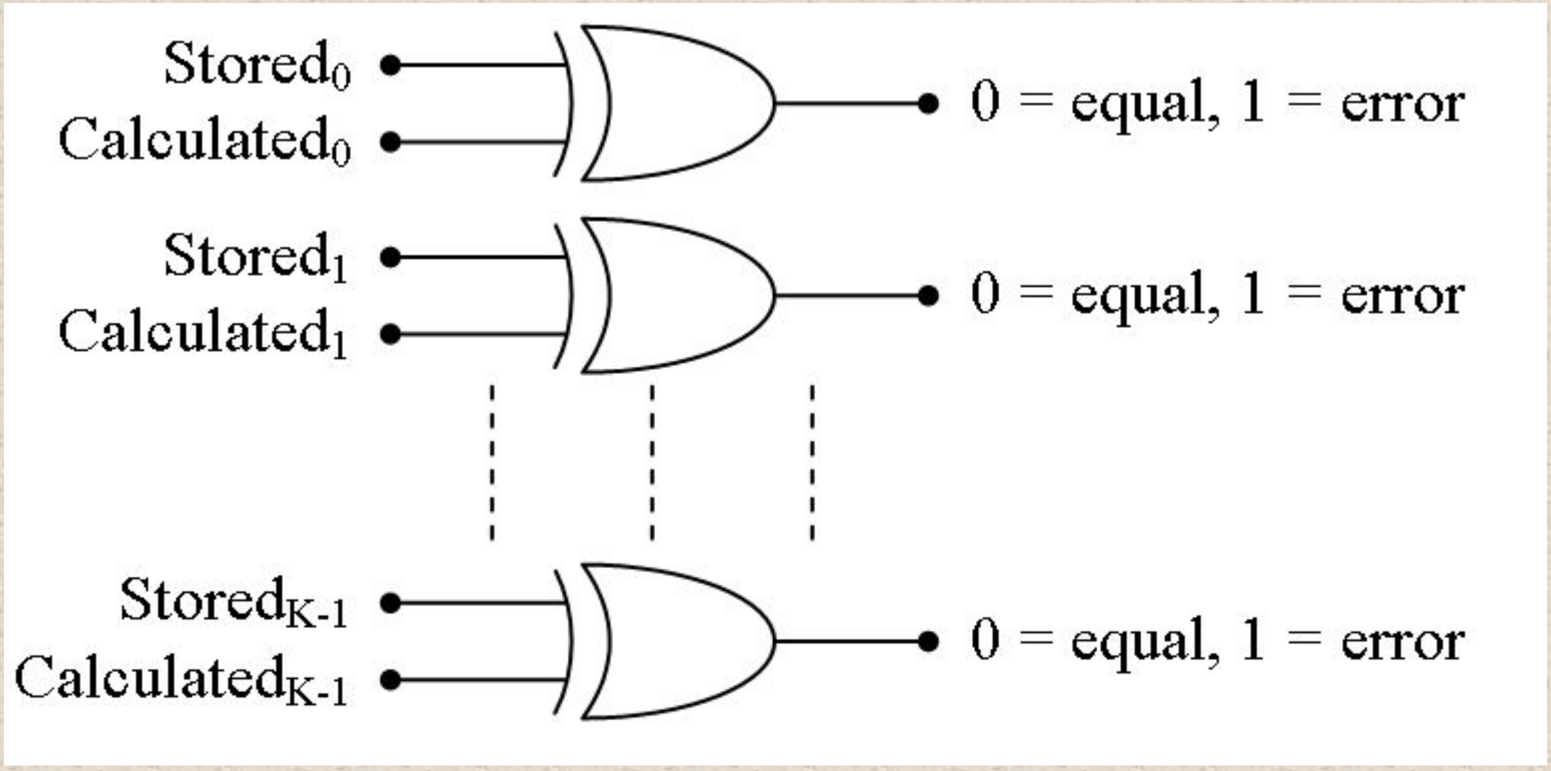
- The mechanics of the typical error correction/detection system are created with XOR gates
  - Odd number of ones input to an XOR □ 1 output
  - Even number of ones input to an XOR □ 0 output
- Upon data retrieval, two K-bit values are generated:
  - The stored K-bit value
  - The K-bit value generated from the stored data
- A bit-by-bit comparison is performed on these two values generating a K-bit result
  - 0’s in bit positions where there is no error
  - 1’s in bit positions where two bits disagree
  - K-bit result is called a *syndrome word*


A
B
=1
Q
2-input Ex-OR Gate

Symbol	XOR Truth Table		
	A	B	Q
	0	0	0
	0	1	1
	1	0	1
	1	1	0

Boolean Expression Q = A XOR B

Generation of Syndrome Word



+

Syndrome Word

- All zeros means that the data was successfully retrieved
- For data with M bits and K code bits, then there are M+K possible single bit errors, i.e., there could be an error in the data OR the K-bit code

SEC (Single Error Correction) Code Example (continued)

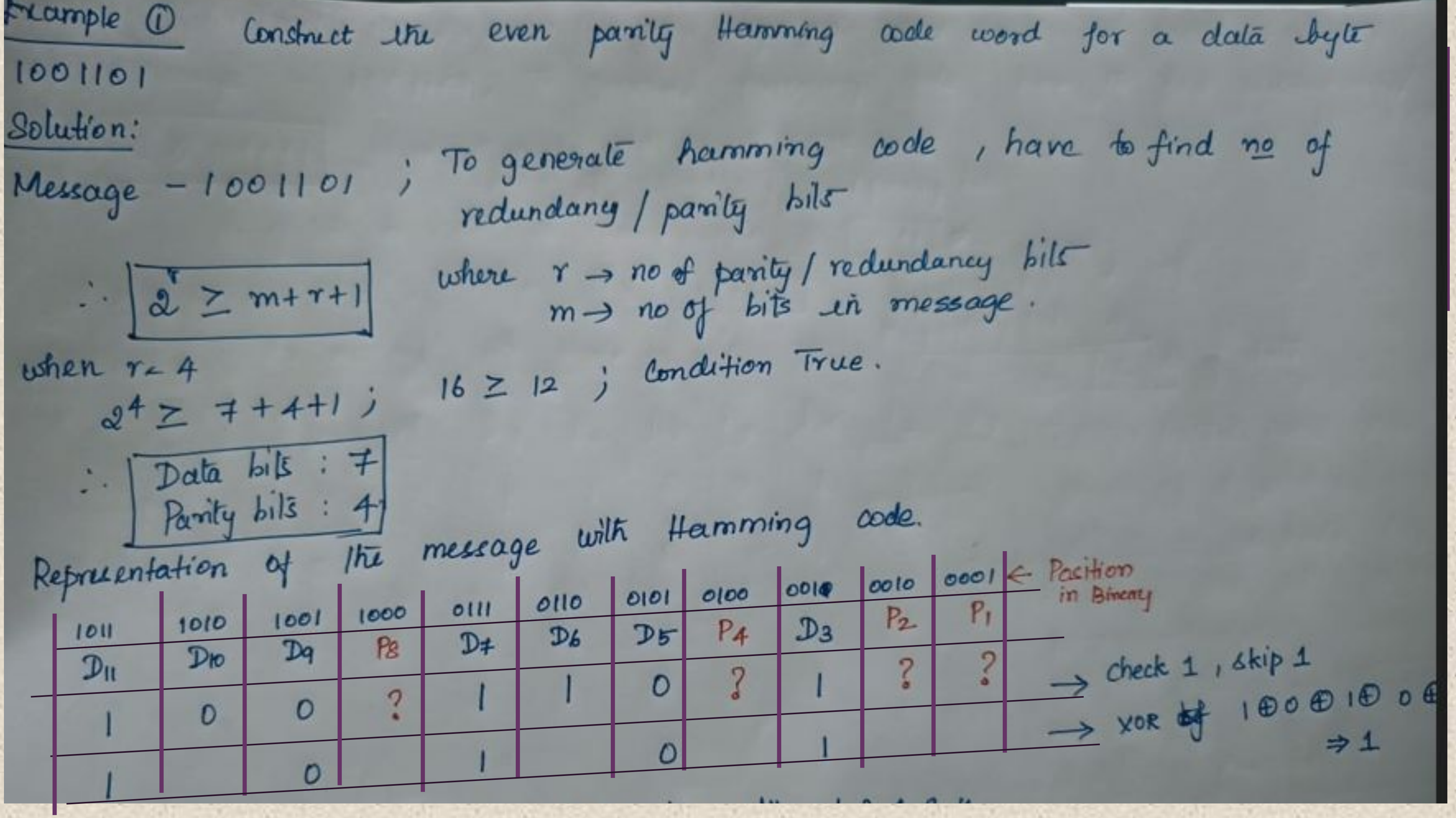
The table below is used to identify which bits of the M+K bits of the combined data and syndrome word are associated with which possible values of the syndrome word.

M+K Bit position	12	11	10	9	8	7	6	5	4	3	2	1
Position number	1100	1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
Data bits	D8	D7	D6	D5		D4	D3	D2		D1		
Code bits					C8				C4		C2	C1

+

SEC Code Example (continued)

- We need a system such that the XOR-ing of the stored code or check bits with the code or check bits calculated identifies the position number from the table above.
- ***This means that when a bit changes in the data, then ones need to appear in the digits identifying that position.***
- Each code bit C8, C4, C2, and C1 is calculated by XOR-ing all of the bits in that position that have a 1.



Example ① Construct the even parity Hamming code word for a data byte 1001101

Solution:

Message - 1001101 ; To generate hamming code, have to find no of redundancy / parity bits

$\therefore \boxed{2^r \geq m+r+1}$ where $r \rightarrow$ no of parity / redundancy bits
$m \rightarrow$ no of bits in message.

when $r = 4$
$2^4 \geq 7+4+1$; $16 \geq 12$; Condition True.

$\therefore$ Data bits : 7
Parity bits : 4

Representation of the message with Hamming code.

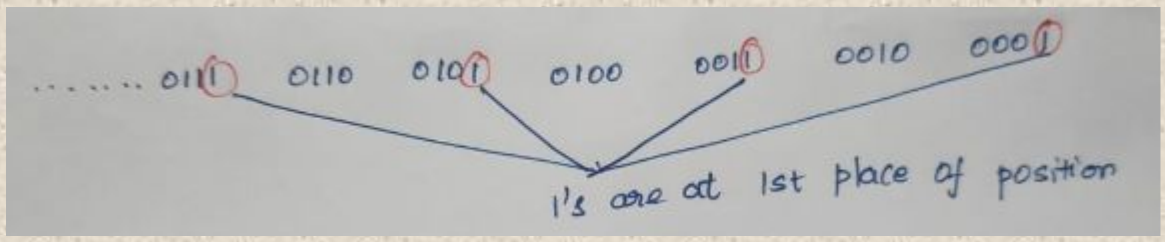
1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
D_{11}	D_{10}	D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	0	0	?	1	1	0	?	1	?	?
1		0		1		0		1		

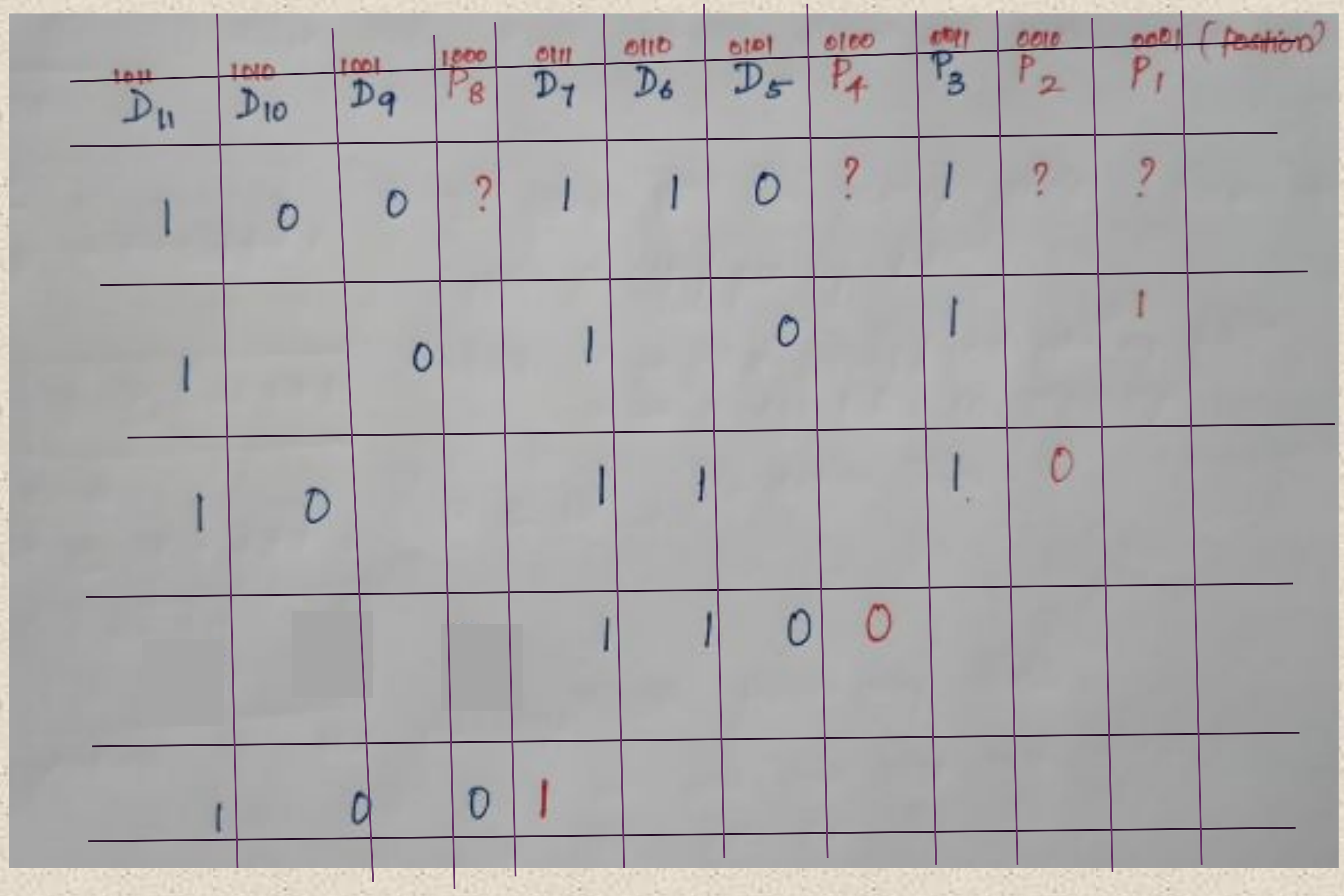
← Position in Binary

→ check 1, skip 1

→ XOR ~~of~~ $1 \oplus 0 \oplus 1 \oplus 0 \oplus$
$\Rightarrow 1$

1) Parity bits are placed on the 2 power's positions like 1,2,4,8,16,…,
2) P1 is calculated by finding the XOR of the bits available at the position which has 1 at its 1st place





1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001	(position)
D_{11}	D_{10}	D_9	P_8	D_7	D_6	D_5	P_4	P_3	P_2	P_1	
1	0	0	?	1	1	0	?	1	?	?	
1		0		1		0		1		1	
1	0			1	1			1	0		
				1	1	0	0				
1	0	0	1								

+

3) P2 is calculated by finding the XOR of the bits available at the position which has 1 at its 2nd place
4) P4 is calculated by finding the XOR of the bits available at the position which has 1 at its 3rd place
5) P8 is calculated by finding the XOR of the bits available at the position which has 1 at its 4th place

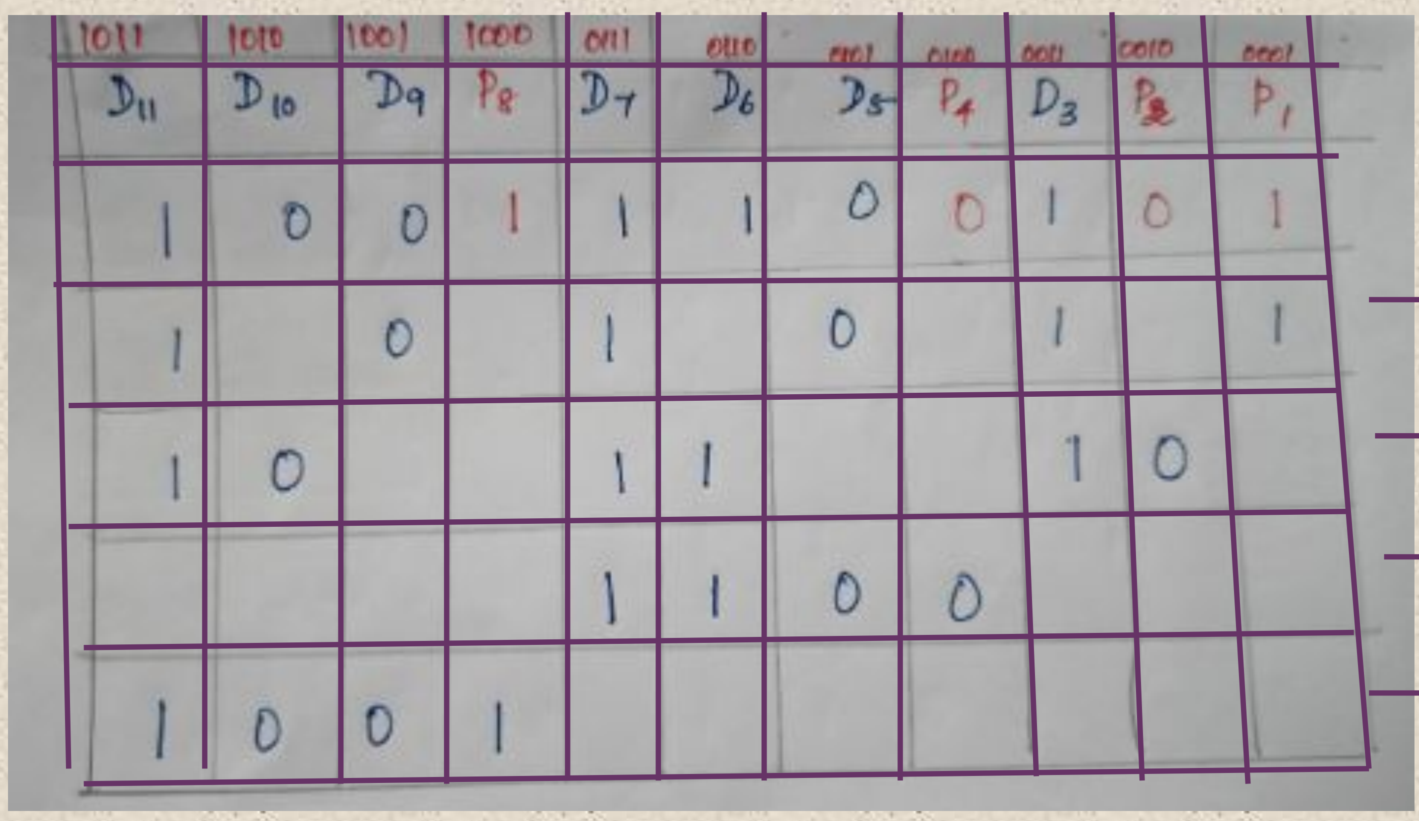
Therefore, code is

P8	P4	P2	P1
1	0	0	1

Now the final message with hamming code is **10011100101**.

This code can be checked for occurrence of error once received by the receiver. This approach id called is syndrome word calculation.

+Syndrome word calculation:



1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001	
D_{11}	D_{10}	D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1	
1	0	0	1	1	1	0	0	1	0	1	
1		0		1		0		1		1	C1
1	0			1	1			1	0		C2
				1	1	0	0				C3
1	0	0	1								C4

C1= 1 ⊕ 0 ⊕ 1 ⊕ 0 ⊕ 1 ⊕ 1 = 0

C2= 1 ⊕ 0 ⊕ 1 ⊕ 1 ⊕ 1 ⊕ 0 = 0

C4= 1 ⊕ 1 ⊕ 0 ⊕ 0 = 0

C8= 1 ⊕ 0 ⊕ 0 ⊕ 1 = 0

<u>Syndrome word</u>

C8	C4	C2	C1
0	0	0	0

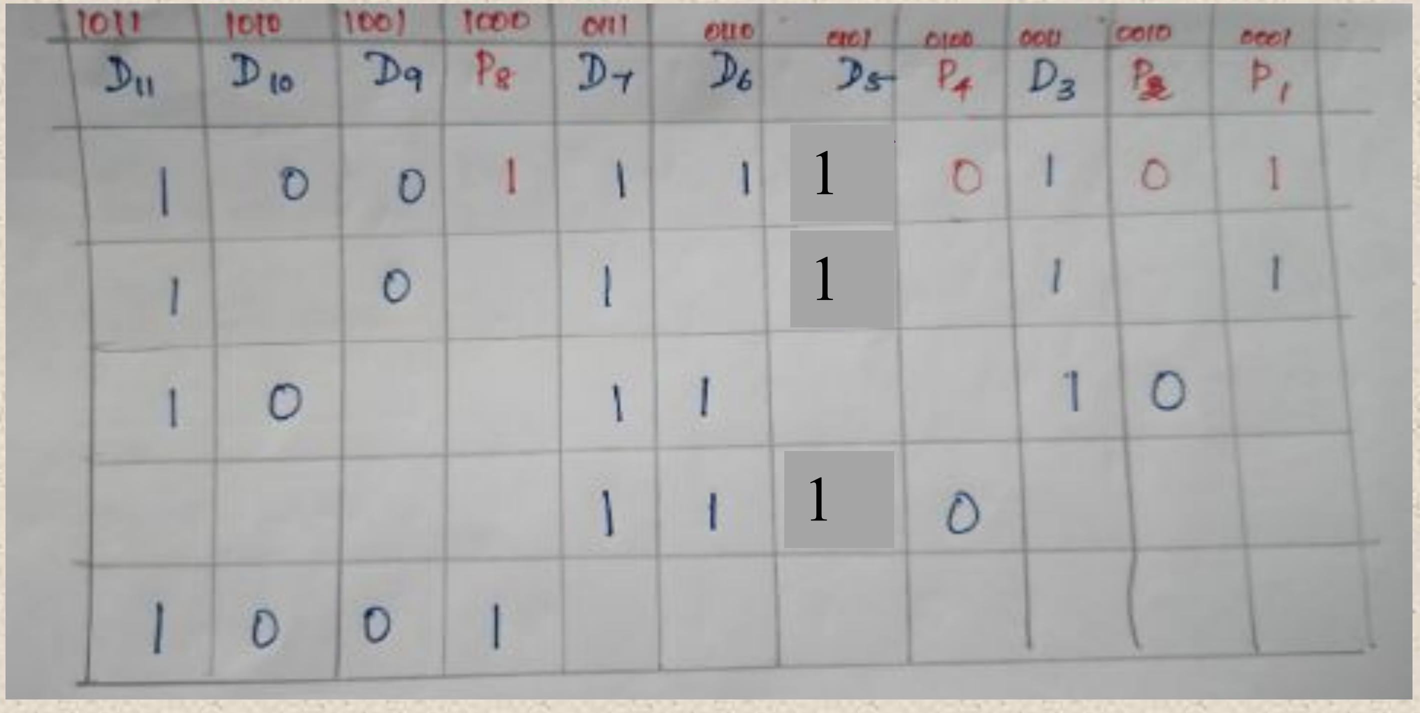
0

1) C0 is calculated by finding the XOR of the bits at the position which has 1 at its 1st place including P1 bit
2) C1 is calculated by finding the XOR of the bits at the position which has 1 at its 2nd place including P2 bit
3) C4 is calculated by finding the XOR of the bits available at the position which has 1 at its 3rd place including P4 bit
4) C8 is calculated by finding the XOR of the bits available at the position which has 1 at its 4th place including P8 bit.

Observations from syndrome word,

- If the syndrome is all zeros, there were **no errors**, i.e. the bits were exactly alike
- If the is only one 1 bit somewhere, there **exist error in parity bit**.
- If there exists more than one 1 bit, then there is **error in data bit.**

1) Suppose if there is error in D5 bit, Syndrome word is calculated for the error message 10011110101



1011	1010	1001	1000	0111	0110	0101	0100	0011	0010	0001
D_{11}	D_{10}	D_9	P_8	D_7	D_6	D_5	P_4	D_3	P_2	P_1
1	0	0	1	1	1	1	0	1	0	1
1		0		1		1		1		1
1	0			1	1			1	0	
				1	1	1	0			
1	0	0	1							

$C1 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 1 = 1$

$C2 = 1 \oplus 0 \oplus 1 \oplus 1 \oplus 1 \oplus 0 = 0$

$C4 = 1 \oplus 1 \oplus 1 \oplus 0 = 1$

$C8 = 1 \oplus 0 \oplus 0 \oplus 1 = 0$

$$\begin{array}{cccc} C8 & C4 & C2 & C1 \\ 0 & 1 & 0 & 1 \end{array}$$

$$0 \quad \underbrace{1 \quad 0 \quad 1}_{\text{5th position bit}}$$

The error is in data bit at position 5, so error is corrected by changing that bit 1 to 0.