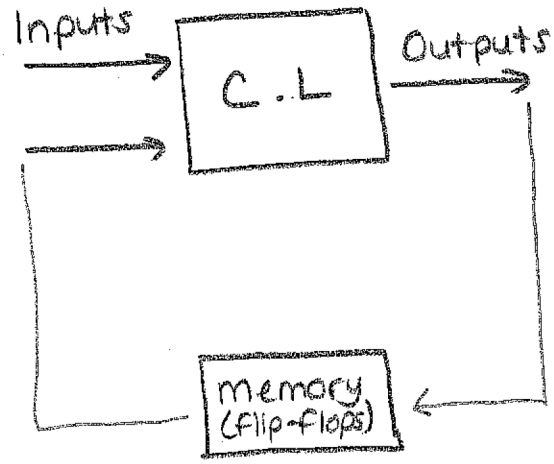
# Finite State Machines: (Sequantial Digital Circuits)




* In combinational logic circuits (C.L), we have inputs, output, gates.

Ex: (C.L)

Design a C.L circuit that outputs a "1" when the value of the inputs ABC > 5 in decimal.

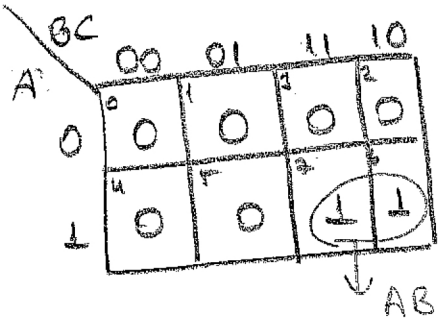
most significant bit (MSB)

least sign. bit (LSB)

| A | B | C | Q |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |
| 1 | 1 | 1 | 1 |

6 > 5

7





* independent from C.

## STEPS (for prev. ex.)

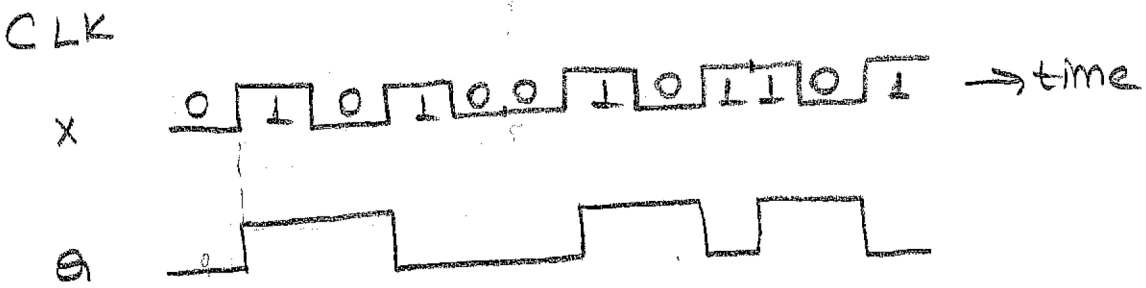
→ Draw Table

→ K-map

→ Simplify function

→ Draw the circuit

Ex: Assert output whenever input bit stream has odd # of 1's.




## STEPS

1) Understand the problem

2) Draw the state diagram

3) Constant the transition table
   ↳state

4) Simplify the table

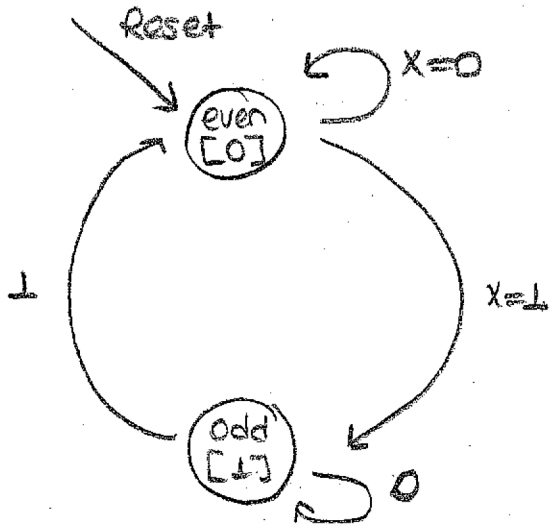
5) Draw the circuit

Answer:

2) State Diagram

State:

output value

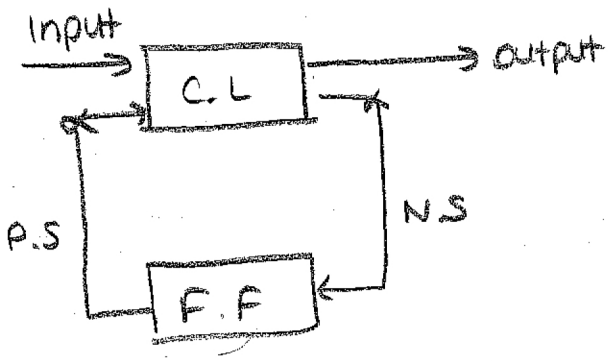


3) State Transition Table: (output determined according to the P.S)

| Present State (PS) | Input (x) | Next State (NS) | Output |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 |

↑ Determined at the P.S.

4)

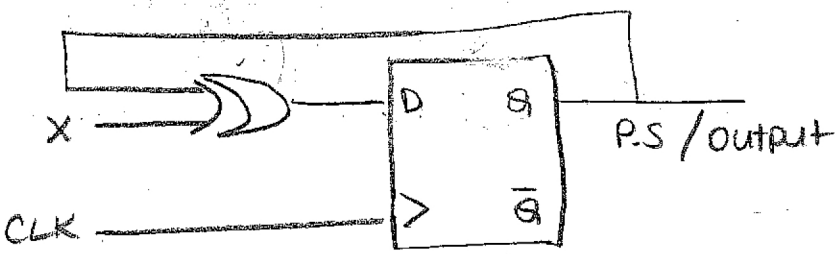


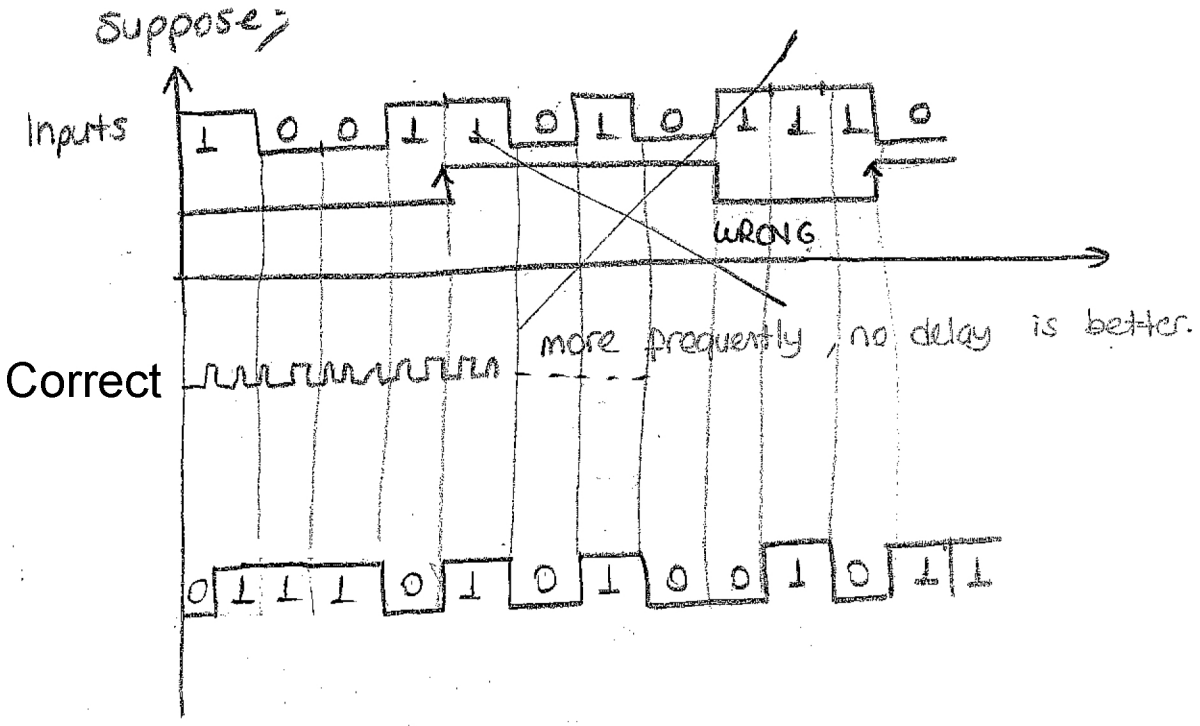

Assume D-FF implementation

| PS ⇒ $D_1$ | X | NS ⇒ $\Theta_1$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

$$\Theta_1 = D_1 \oplus X$$

XOR



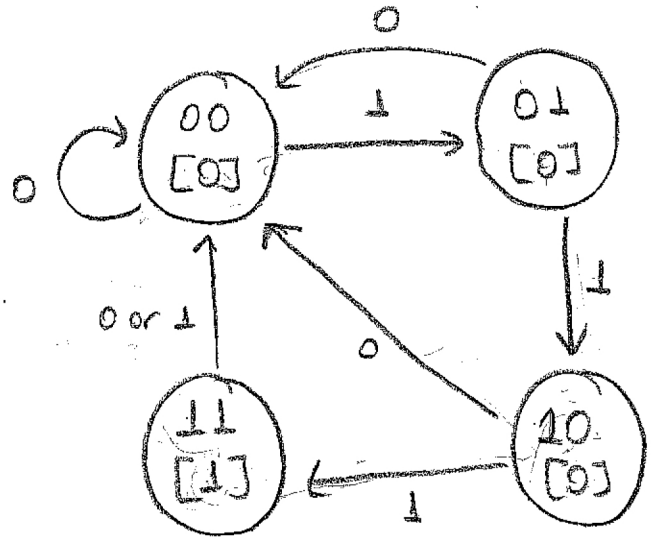

Ex: Design of vending machine.

- Deliver a package of gum after 1.5 ₺
- Single coin slot
- 1 ₺ and 0.5 ₺
- No change (geri verme yok)

Lab. Question Answer:

Her 3'de 1 yapacak sonra 0.

Diagram:




00 → No one

01 → first one

10 → Second one

11 → Third one

Step 3: State Transition Table

| P.S $Q_1 Q_0$ | Input X | N.S $D_1 D_0$ | Output Z |
|---|---|---|---|
| 0 0 | 0 | 0 0 | 0 |
| 0 0 | 1 | 0 1 | 0 |
| 0 1 | 0 | 0 0 | 0 |
| 0 1 | 1 | 1 0 | 0 |
| 1 0 | 0 | 0 0 | 0 |
| 1 0 | 1 | 1 1 | 0 |
| 1 1 | 0 | 0 0 | 1 |
| 1 1 | 1 | 0 0 | 1 |

(D1)

| X \ $Q_1Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 1 |

$D_1 =$

→ No simplification

$= X \cdot Q_1' \cdot Q_0 + X \cdot Q_1 \cdot Q_0'$

(D0)

| X \ $Q_1Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 |

$D_0 =$

→ $X \cdot Q_0'$

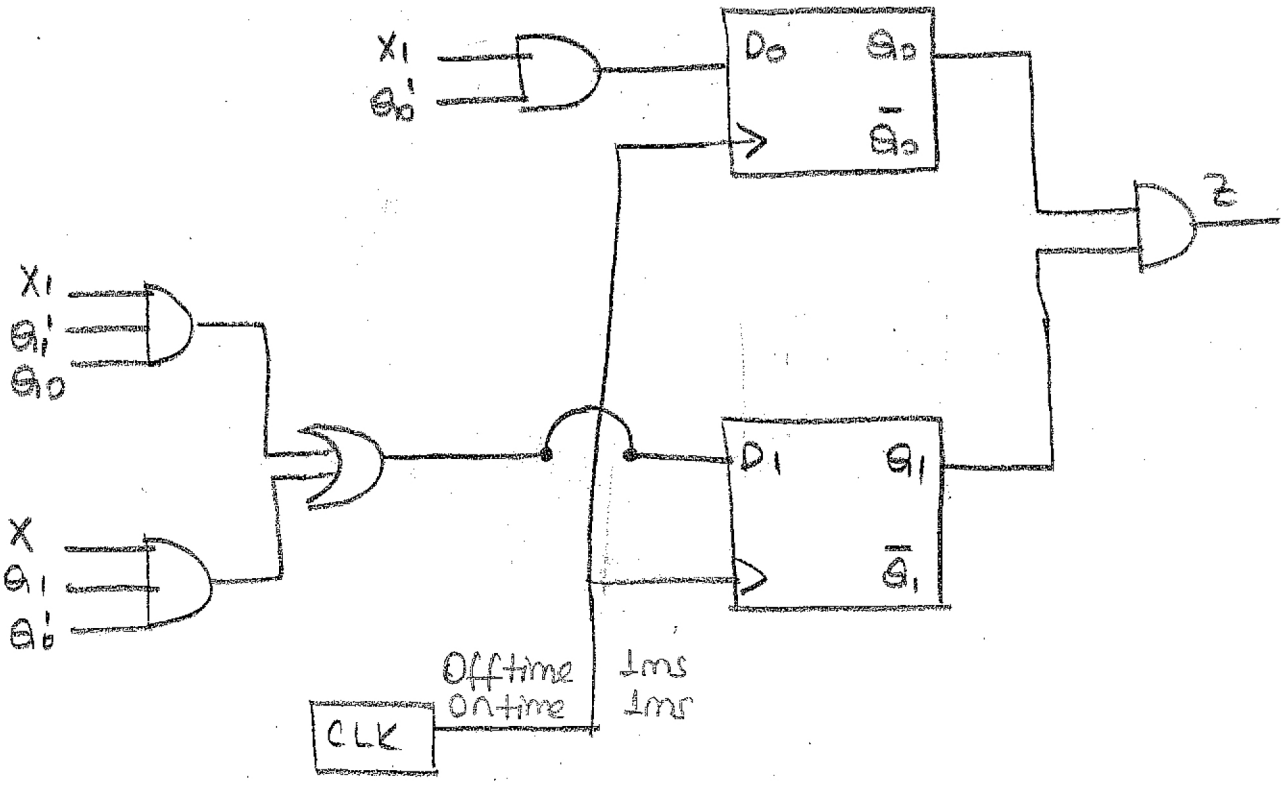
(Z)

| X \ $Q_1Q_0$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 |

$Z =$

→ $Q_1 \cdot Q_0$

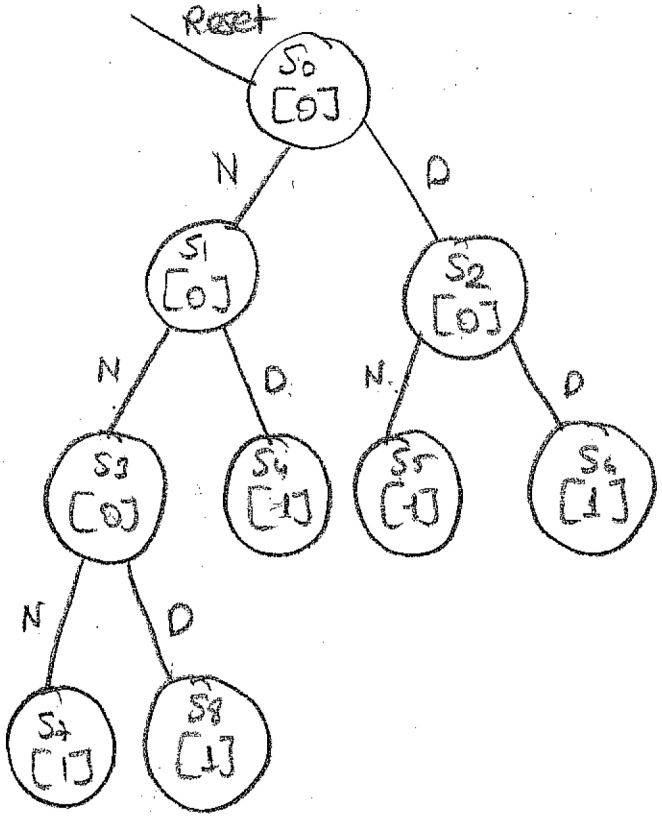
Circuit Implementation:

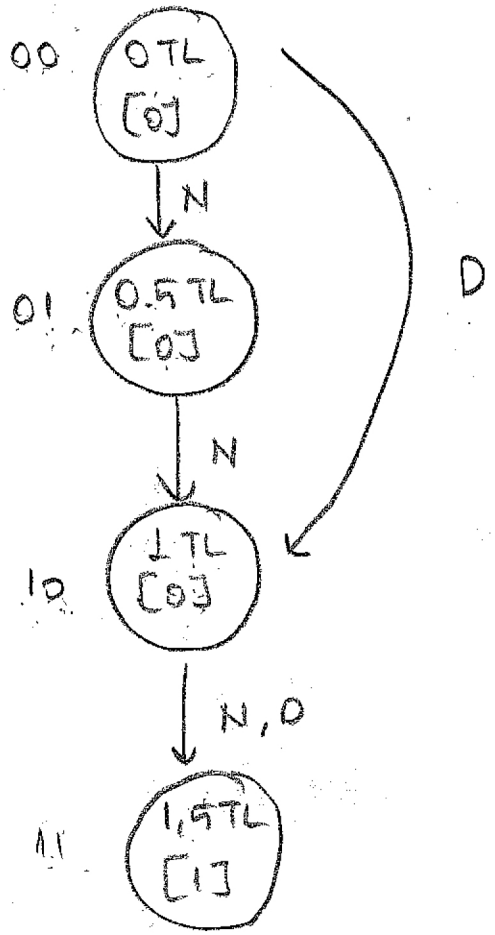


# Vending machine

1 TL = D

0.5 TL = N



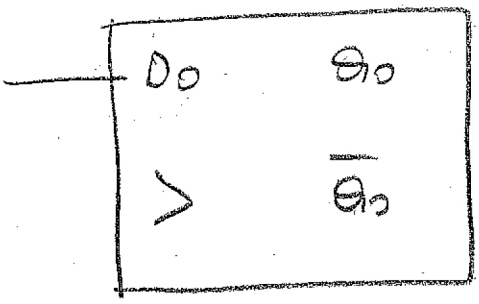
| P.S $Q_1\ Q_0$ | Inputs D N | N.S $D_1\ D_0$ | Outputs |
|---|---|---|---|
| 0 0 | 0 0 | 0 0 | 0 |
| | 0 1 | 0 1 | 0 |
| | 1 0 | 1 0 | 0 |
| | 1 1 | X X | |
| 0 1 | 0 0 | 1 0 | 0 |
| | 0 1 | 1 0 | 0 |
| | 1 0 | 1 1 | 0 |
| | 1 1 | X X | X |

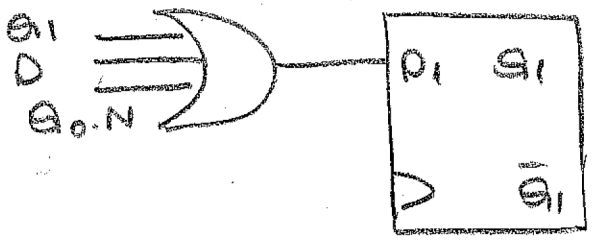
| $Q_1\ Q_0$ | D N | $D_1\ D_0$ | Outputs |
|---|---|---|---|
| 1 0 | 0 0 | 1 0 | 0 |
| | 0 1 | 1 1 | 0 |
| | 1 0 | 1 1 | 0 |
| | 1 1 | X X | X |
| 1 1 | 0 0 | 1 1 | 1 |
| | 0 1 | 1 1 | 1 |
| | 1 0 | 1 1 | 1 |
| | 1 1 | X X | X |

$$D_1 = Q_1 + D + Q_0 N$$

$$D_0 = N \bar{Q}_0 + Q_0 . \bar{N} + Q_1 . N + Q_1 . D$$

$$\text{Output} = Q_1 . Q_0$$




## CH: 6

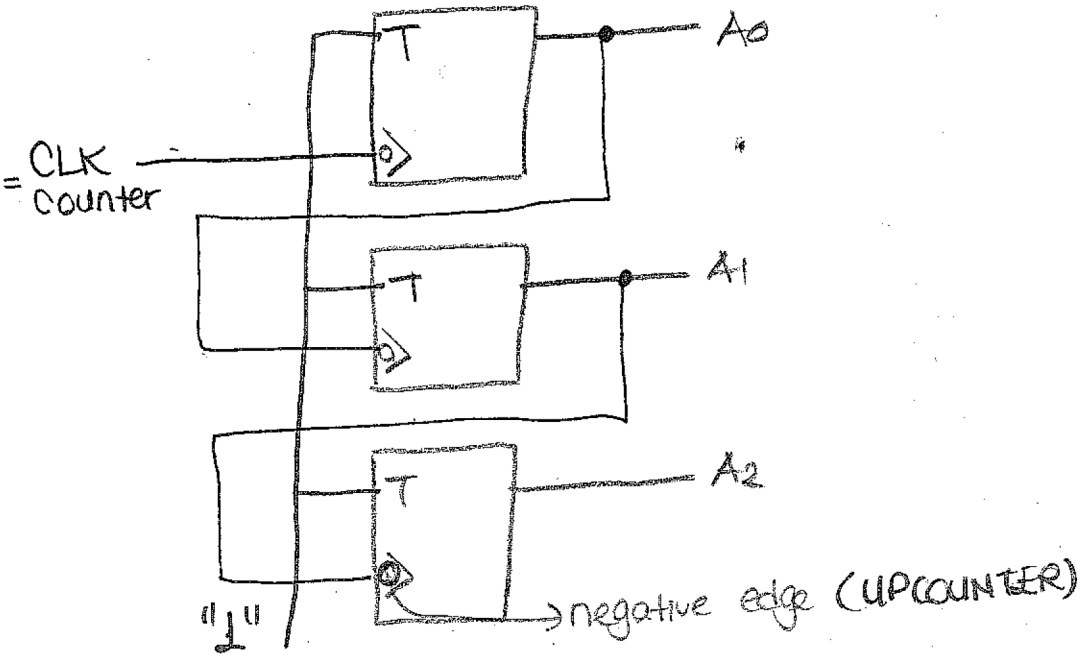
# REGISTERS & COUNTERS

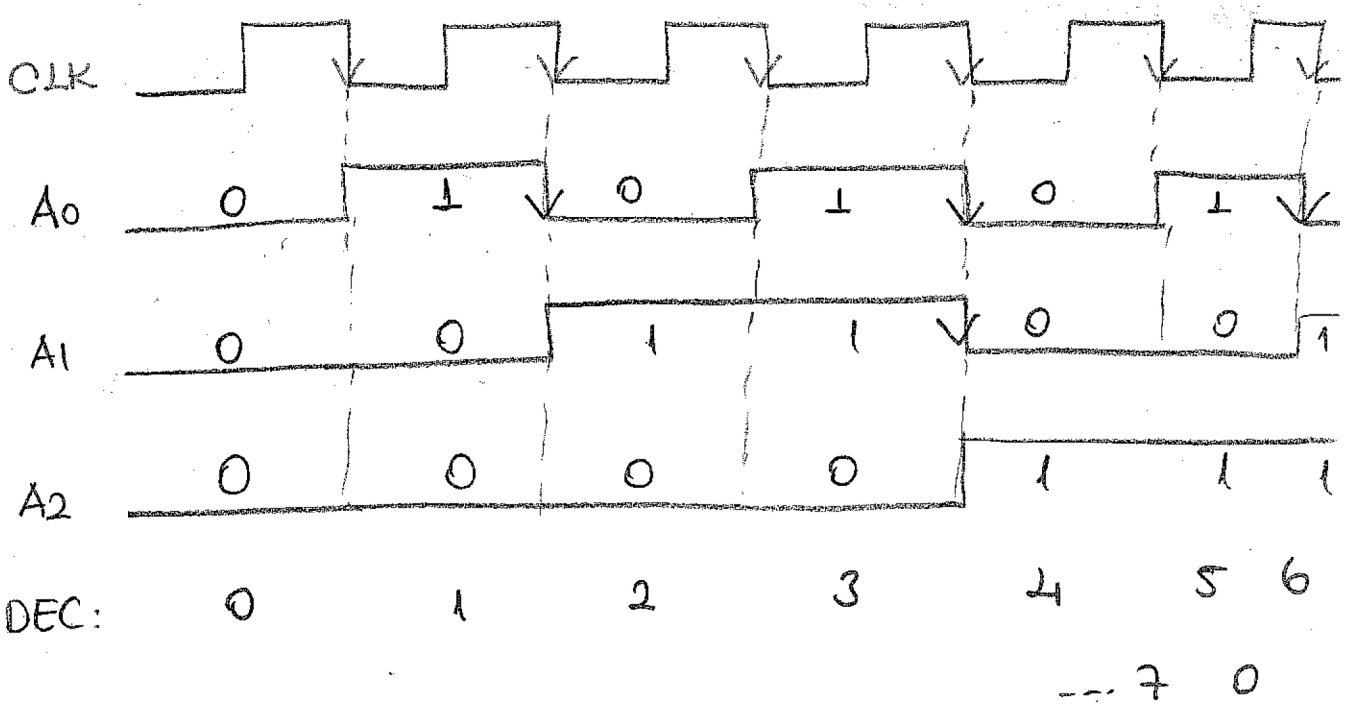
**Counters:** Circuit that follows the binary number sequence.

- Ripple Counters
- Synchronous Counters

## Ripple Counter:

### Using TFF:

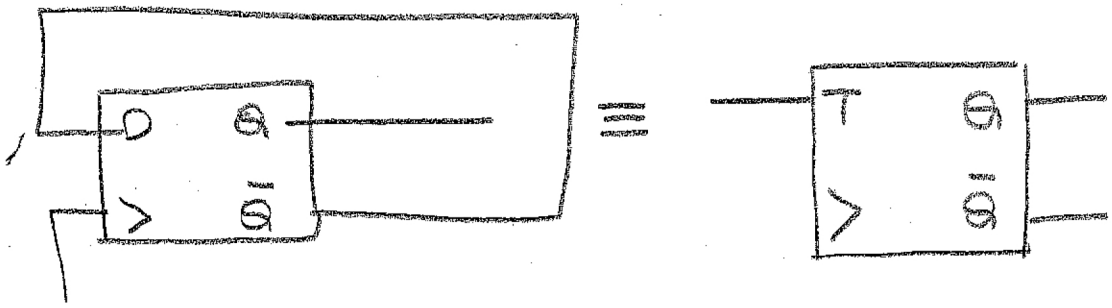





NOT: If we use (+) edge TFF's, then we obtain down counters.

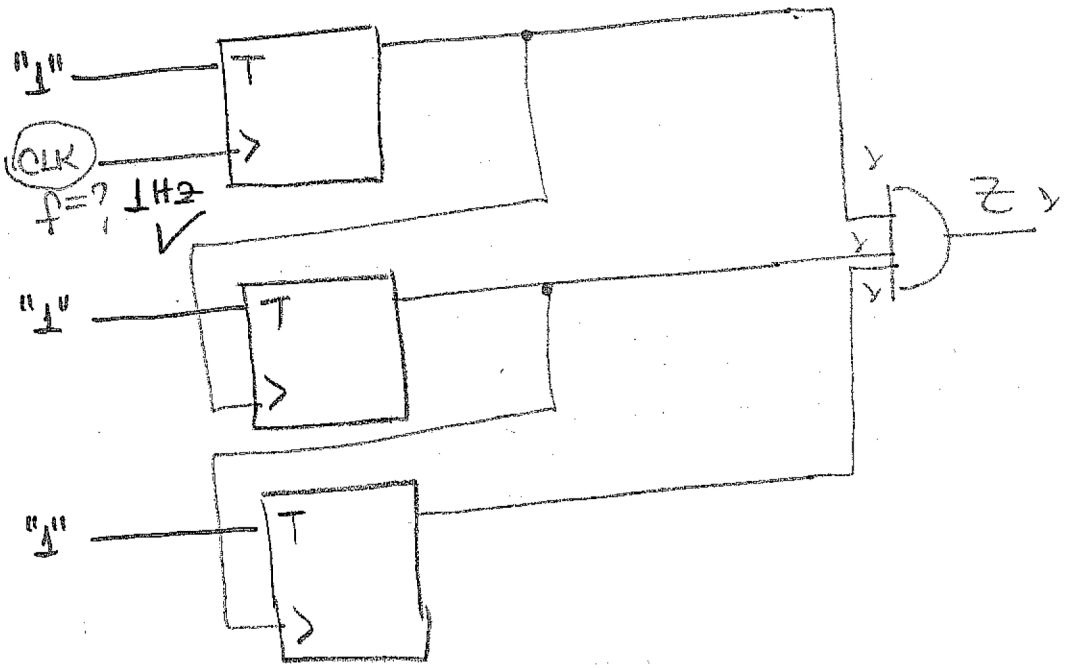
Also note that DFF if you connect

*



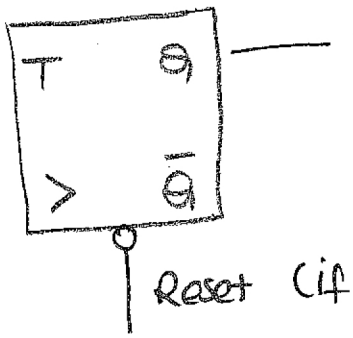
Ex: Design a counter circuit that waits for 8 seconds and outputs a "1".

Ans: Use the TFF counter as before




Question: What if we want to make it count to a certain number?

Then, one possible solution is to use FF's with reset such as,




whenever reset=0, $Q_1=0$ immediately.

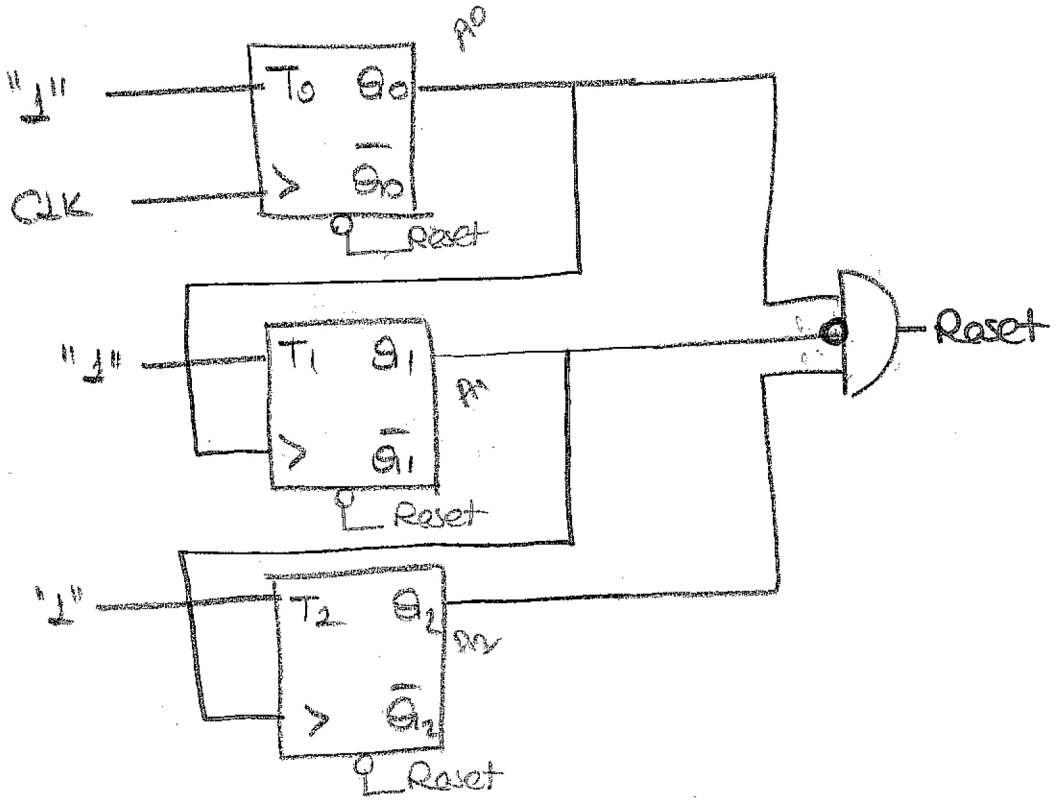
Reset (if it's zero, outputs are zero)

EX: Design a ripple counter that counts to decimal 5.

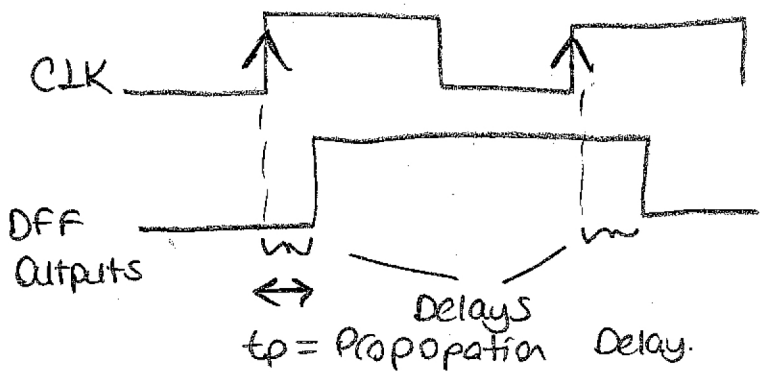
Ans:

5 = 101

000, 001, 010, 011, 100, 101, 000




Ripple counters are problematic in the way that they generate delays:



The remedy is to use

## Synchronous Counters:

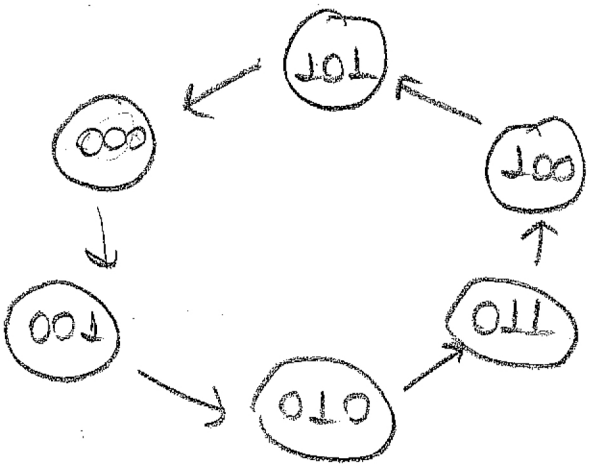
We can use the first design techniques for synchronous counter design.

Ex: Design a sync counter that counts,

000, 001, 010, 011, 100, 101, 000- ..

Ans:

State Diagram: (NO input)




count from 0 to 5.

| $Q_2Q_1Q_0$ P.S | $D_2D_1D_0$ N.S | Output |
|---|---|---|
| 000 | 001 | 0 |
| 001 | 010 | 0 |
| 010 | 011 | 0 |
| 011 | 100 | 0 |
| 100 | 101 | 0 |
| 101 | 000 | 1 |
| 110 | 111 | 0 |
| 111 | 000 | 0 |

} 9

# MEMORY & REGISTERS

Register: 1 bit storage device (FF)

Memory → Random Access memory (RAM):

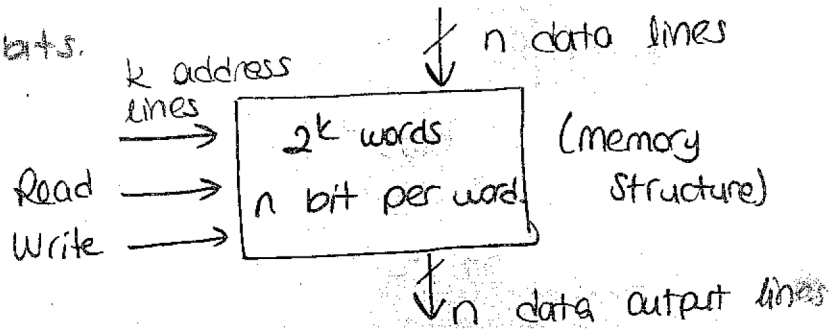
Memory → Read - Only memory (ROM):

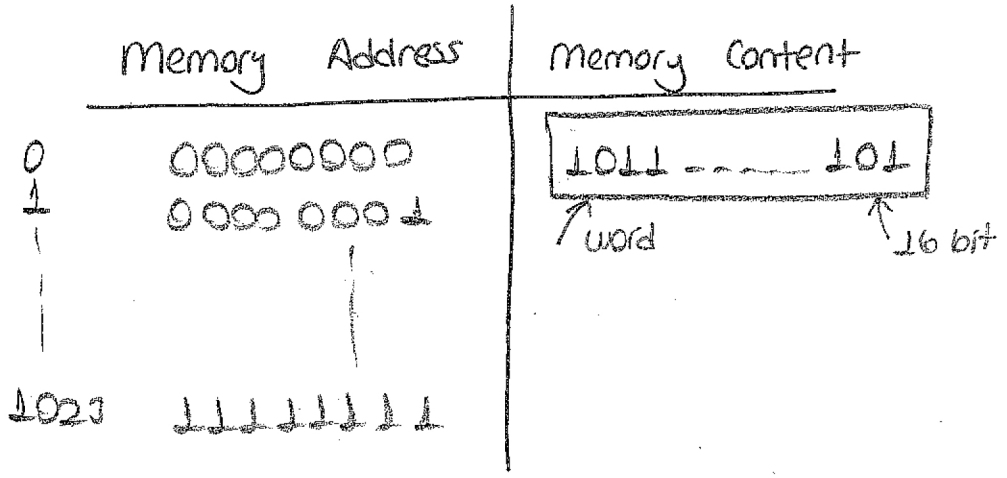
RAM:

Definitions:

Words = binary information in groups.

Byte: Groups of 8 bits.






Address → $0 - 2^K - 1$ where K is # of address lines

Ex: How many bytes does A memory with 1K words of 16 bits each can support?

Ans:

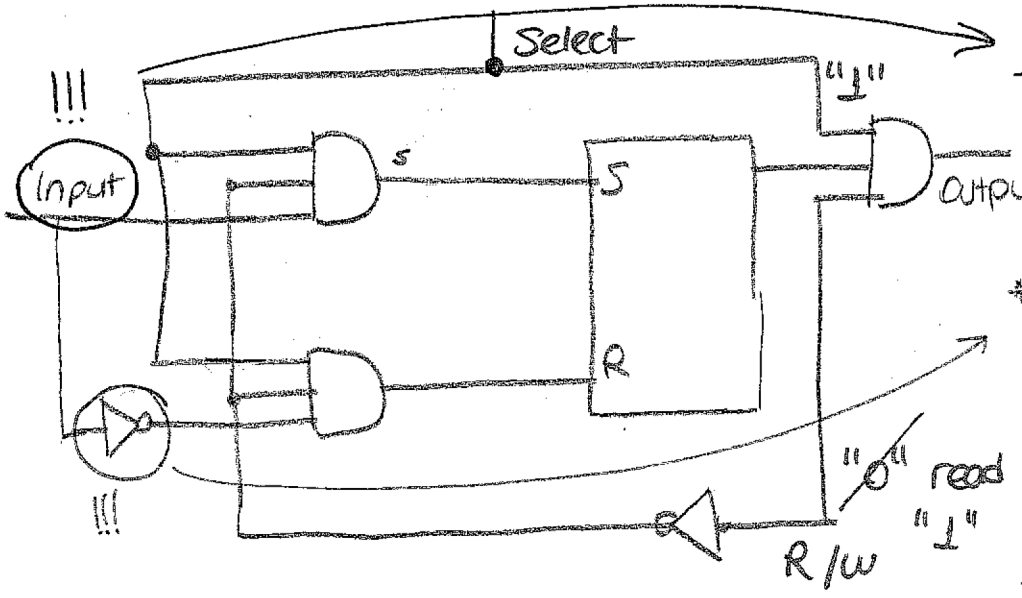
$1K = 1024 = 2^{10}$

since each word contains 16 bit = 2 bytes

$\Rightarrow 1k \times 2 = 2k = 2048$ bytes.

## Internal Construction of RAM:

RAm with m words and n bits per word consists of $m \times n$ binary storage cells. (bsc)



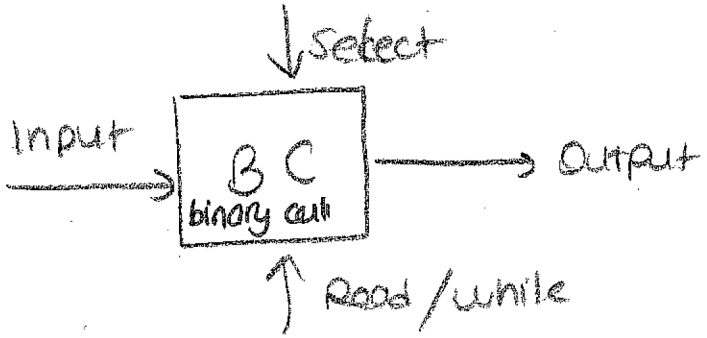

* Whatever input is, S will be same. if input=1, S=1

* Whatever complement is, R becomes same in writing mode.

S R 00

NO CHANGE

We can also show it as,




* BC stores a bit

select = 0 , output = 0

When select = 1 , R/W mode is enabled

↳ If R/W = 0:

Write is performed

↳ If R/W = 1:

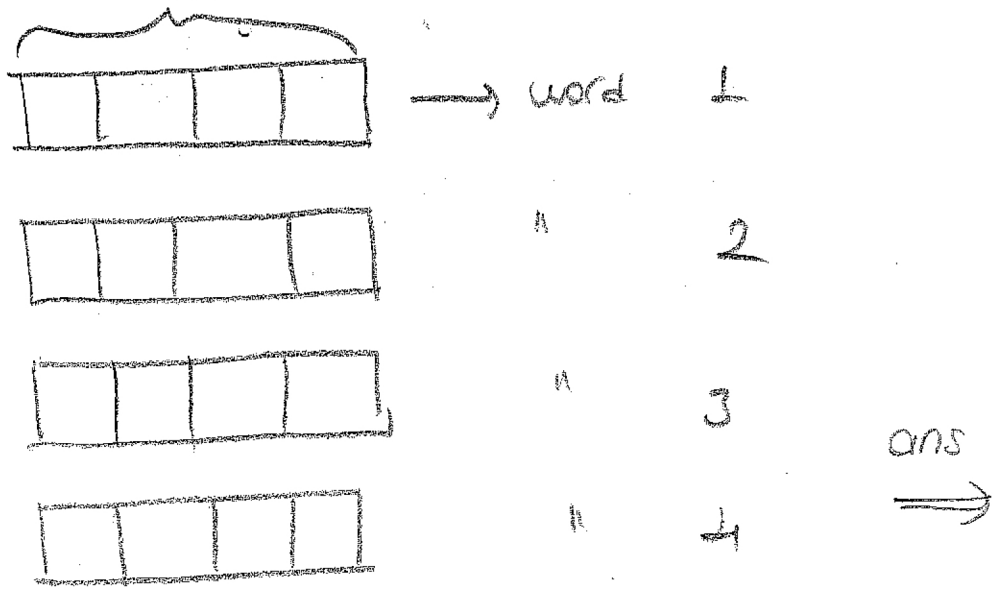
Read is performed.

In reading mode, latched, does not change

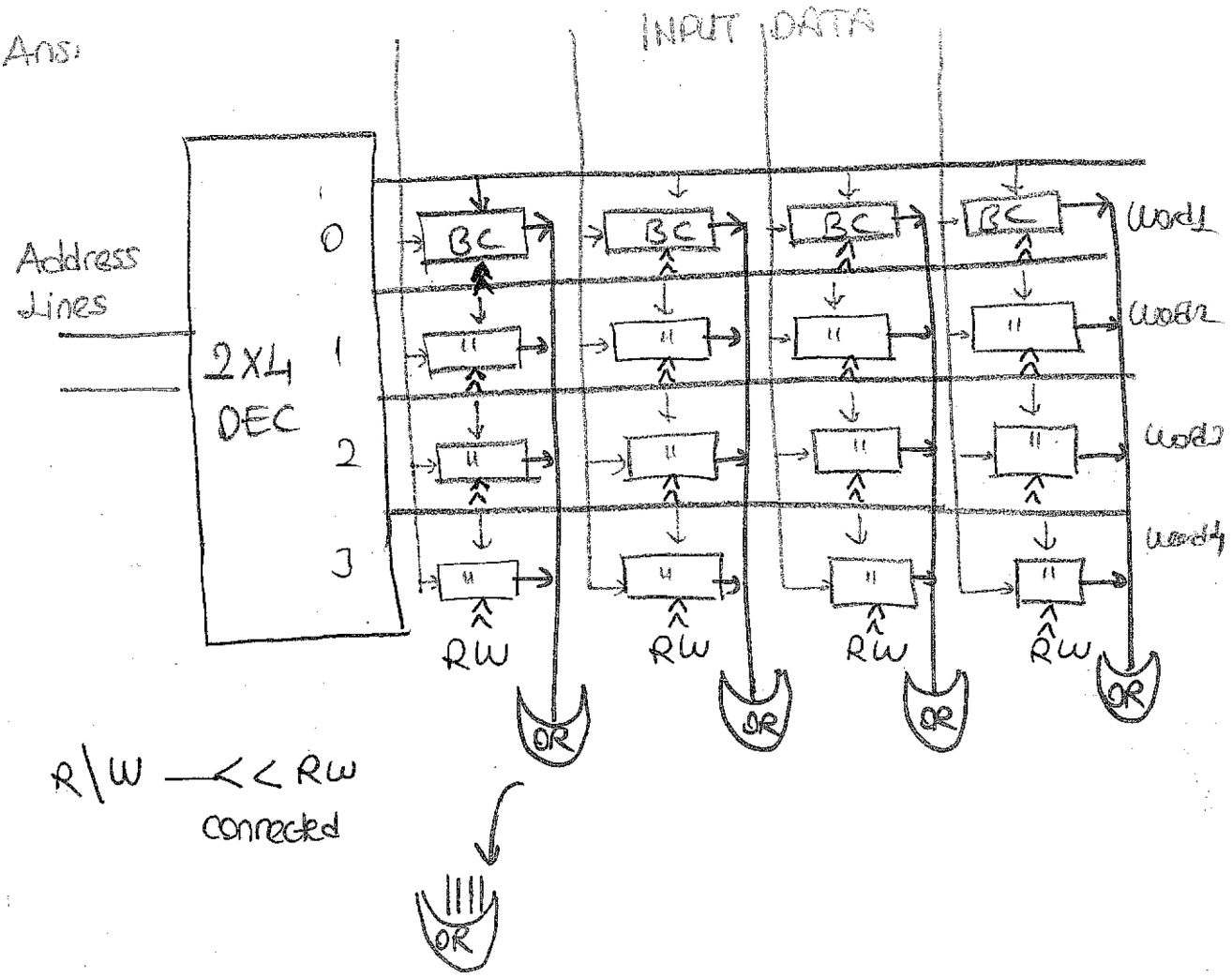
(The data inside BC, is latched)

EX: Design a 4x4 RAM

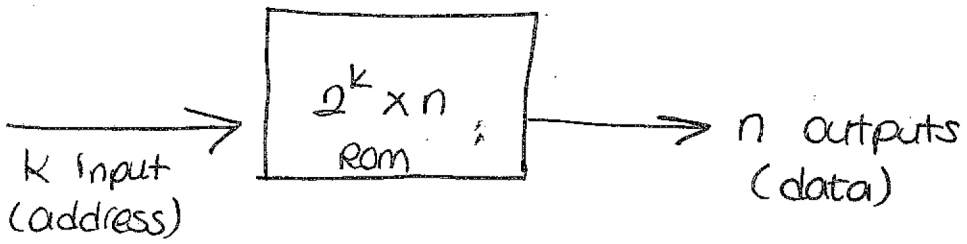
each word contains 4 bits.



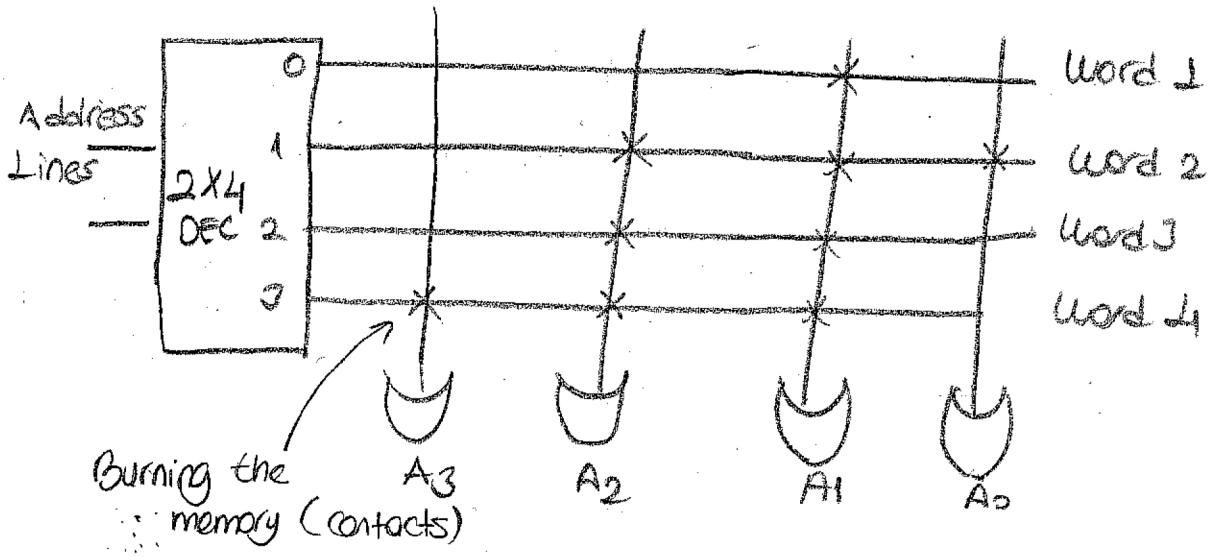
Ans:


INPUT DATA
Address lines
2X4 DEC
0
1
2
3
BC
RW
OR
word1
word2
word3
word4
R/W —<< RW
connected

# Read Only Memory (ROM)




Ex: Design a 4x4 ROM and store 0010, 0111, 0110, 1110.




* In ROM, once you write, you can't change it

— ROM is not erasible (silinebilir). The data is permanent. The read speed is fast.

— RAM is erasible, R/W is possible. The read/write speed is slower than ROM.

— If we modify a ROM to be erasible, This is called EPROM

To erase EPROM, one needs to put it on UV light.

— There is also a 2nd type of erasible Rom.

EEPROM or $E^2$PROM (transistors, switches...)

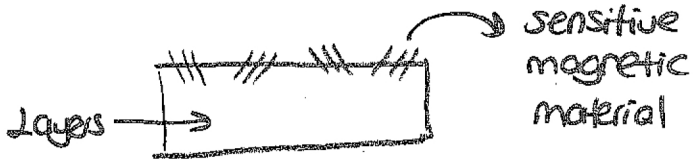
The difference from EPROM is that instead of using UV light to erase the memory, we use electrical signals?

RAM: All these RAM and Rom are semiconductor (yarı iletken) memories. by electronic devices.

Non-volatile memory: when the power is off, the data remains inside the memory.

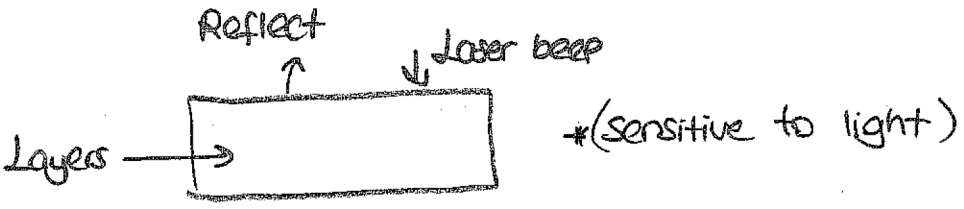
Other non-volatile memories.

1) Hard-disk: magnetic field based operation.



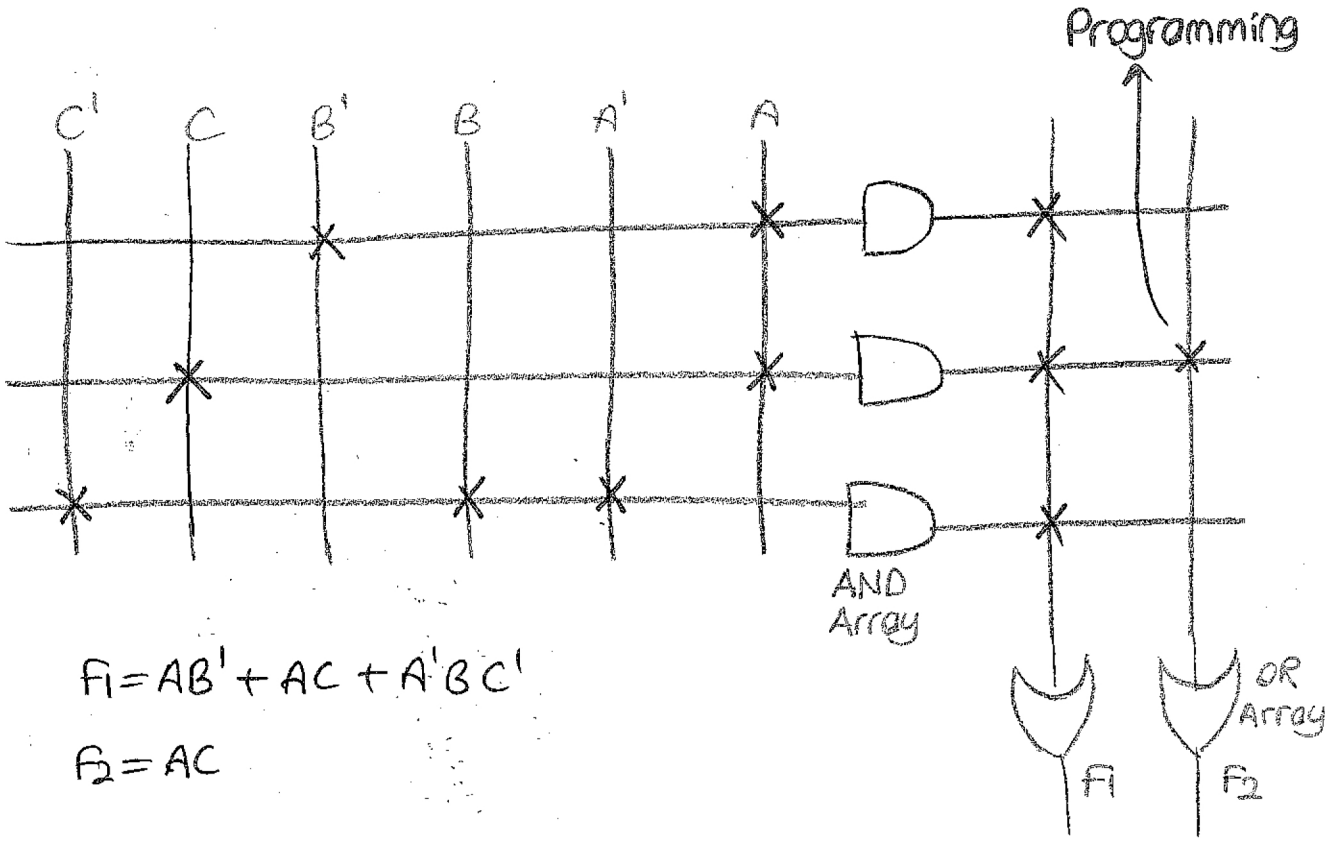

!! PERMENANT

2) CD-Rom, DVD-Rom: Optical waves are used.



# PLD's: Programmable Logic Devices:

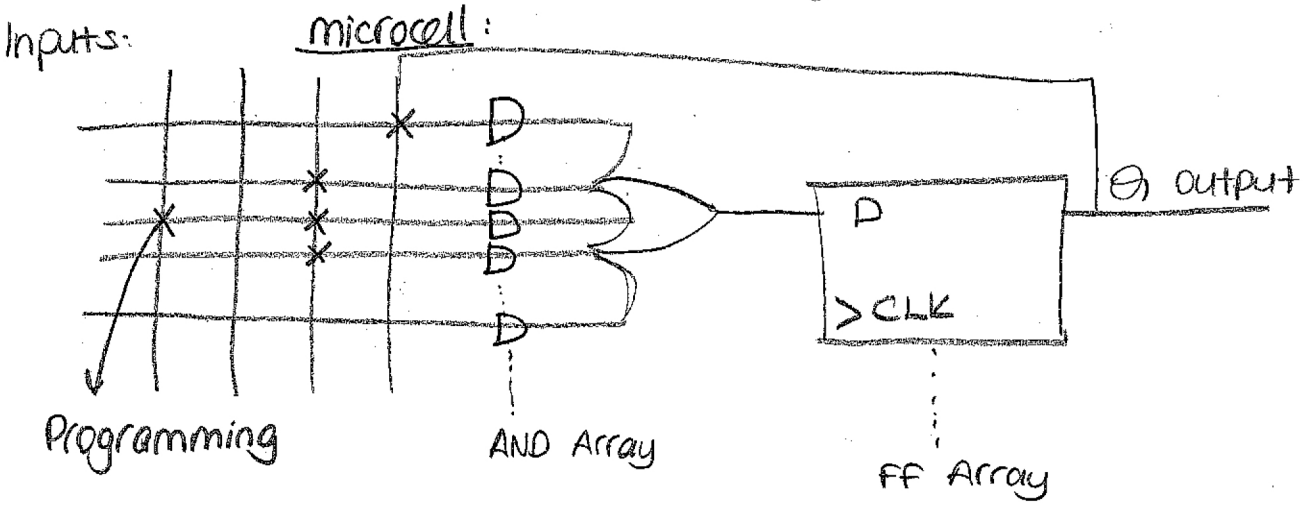
## PLA: Programmable Logic Array:




$F_1 = AB' + AC + A'BC'$

$F_2 = AC$

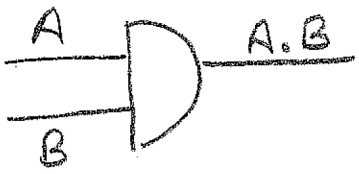
## SPLD: Sequential Prog. Logic Devices:

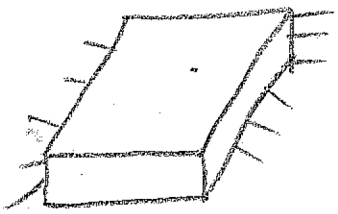


FPGA: field Programmable Gate Array

↓

Electric field



(IC)

Digital Integrated Circuits:

IC Digital Logic Families:

DTL – Diode – Transistor Logic

TTL – Transistor – Transistor Logic

MOS – Metal - Oxide Semiconductor

CMOS – Complementary Metal-Oxide Semiconductor

|

RTL, ECL...

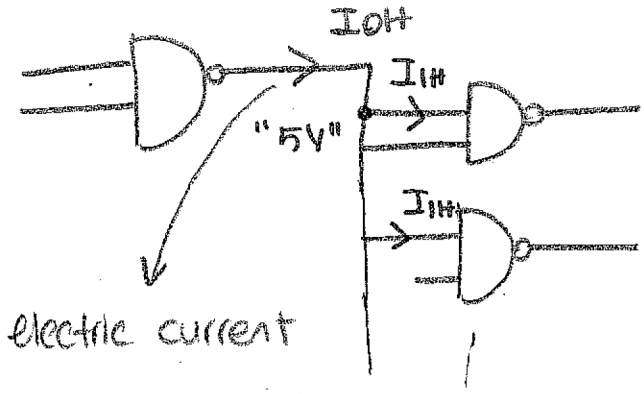
## Special Characteristics:

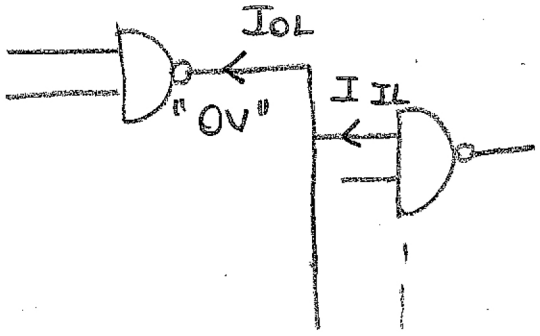
### 1-) Fan-out:

max. number of gates that can be connected to a single gate.

$$\text{Fanout} = \frac{|I_{OH}|}{|I_{IH}|} = \frac{|I_{OL}|}{|I_{IL}|}$$

$I_{OH}$: output high, $I_{OL}$: output low, $I_{IH}$: input high, $I_{IL}$: input low




EX: Given, $I_{OH} = 400\mu A$, $I_{IH} = 40\mu A$, $I_{OL} = 16mA$,

$I_{IL} = 1.6mA$ Find the fan-out.

$$F.O = \frac{400\mu A}{40\mu A} \frac{16mA}{1.6mA} = 10$$

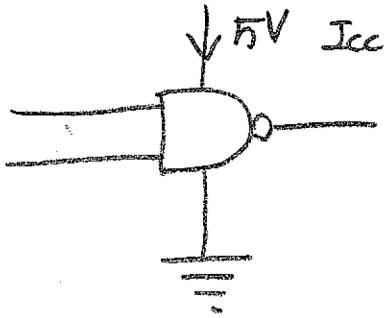
Power Dissipation:

$$I_{cc}(avg) = \frac{I_{CCH} + I_{CCL}}{2}$$

where $I_{CL}$ = current

$$\text{Power Consumption} = P_D(avg) = I_{cc}(avg) \times V_{cc}$$

EX: Find the ave power dissipation of a standart TTL. NAND gate that uses a supply voltage of 5V.



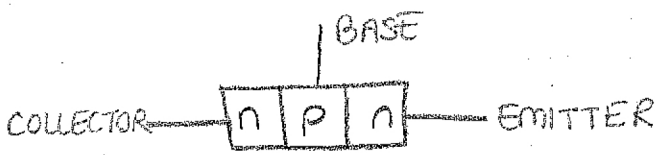

$I_{CCL} = 3mA$

$I_{CCH} = 1mA$

$$P_{Avg} = \frac{(3+1)\,mA}{2} * (5V) = 10mW$$

# Bipolar Transistor Characteristics:

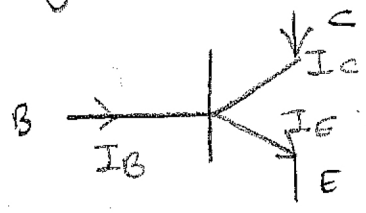
Transistor is an electronic device:




n: negative charged semiconductor

p: positive " "

Symbol:




$I_B > 0$ → Transistor is ON.

$I_E = I_B + I_C$

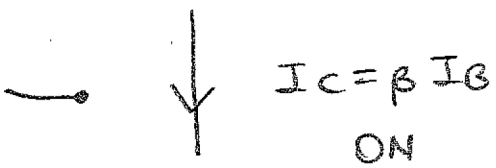
But $I_C = \beta . I_B$, where $\beta$ = gemin of the transistor, typically 100 or greater

Therefore, $I_E = I_B + \beta I_B \simeq \beta I_B$

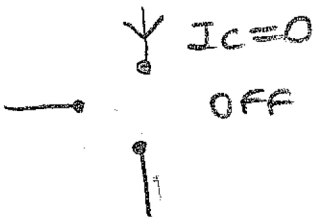
$I_E \approx I_C$

when $I_B = 0$, $I_C = 0$ means that transistor is OFF.

$I_B > 0$

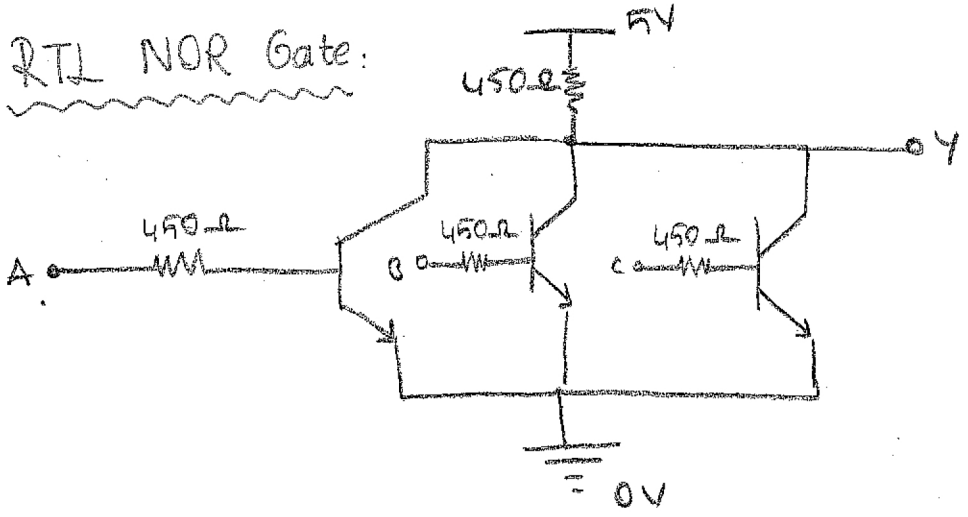



$I_B = 0$



# RTL Circuits:

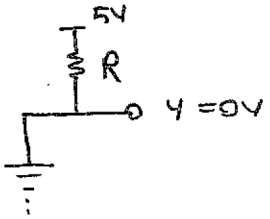
## RTL NOR Gate:



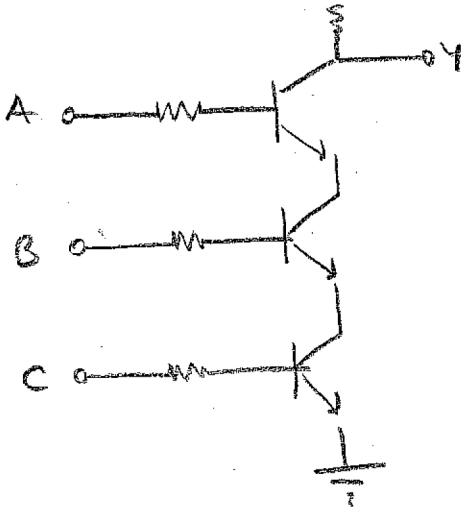

| A | B | C | $(A+B+C)'$ |
|---|---|---|---|
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 0 |
| ⋮ | ⋮ | ⋮ | ⋮ |
| 1 | 1 | 1 | 0 |

- When any of the inputs = 1

  Then;



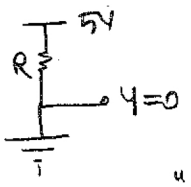

## RTL NAND Gate:




- When A = B = C = "1".

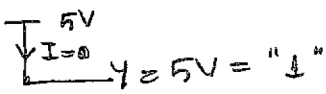
  All trans. are ON.

  and



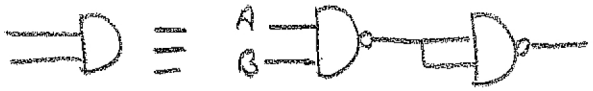

- If any input is "0", Y = 1

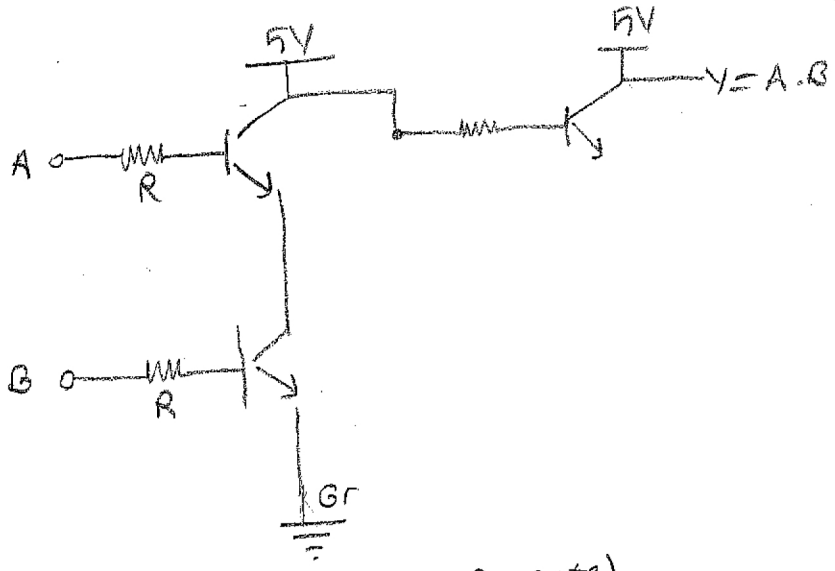
  say A = 0;



EX: Implement F = A.B by using RTL logic

The AND gate is;




(RTL imp. of AND gate)

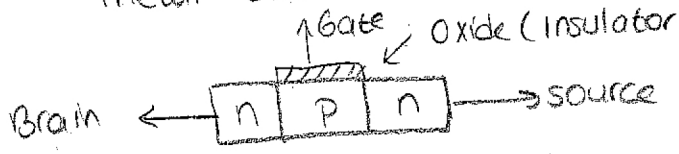
In computer architerture, cmos logic is used, mostly becames of low power consuption and fast switching.

## cmos logic:

Complementary MOSFET Transistor logic

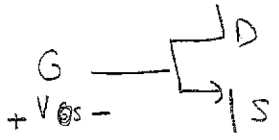
## MOSFET:

Metal Oxide Substate field Effect Transistor




Treshhold voltage ≃ 1V

The Symbole:



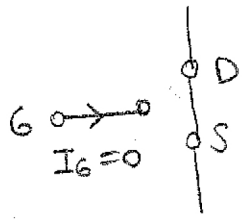

$V_{GS} < V_+$

Trans is OFF!

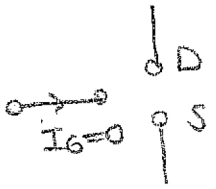
$V_{GS} > V_+$

Trans is ON!

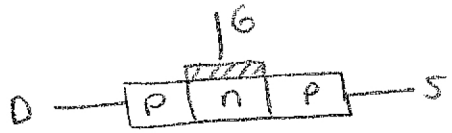
Transistor is ON:



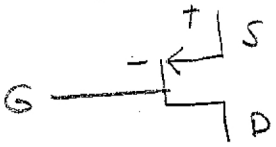

Transistor is OFF:




There is also; Pmos;




The symbole is;



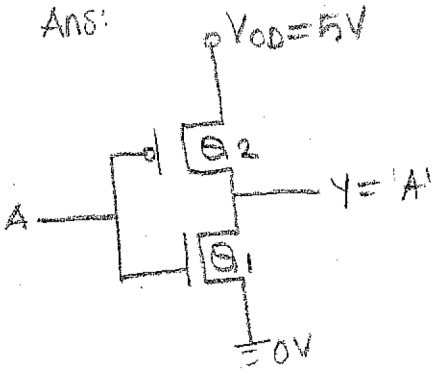

$V_{DS} > V_+$ Trans is ON!

$V_{DS} < V_+$ Trans is OFF!

$$\boxed{\text{CMOS} = \text{NMOS} + \text{PMOS}}$$

Ex: Implement an inverter using CMOS logic

Ans:

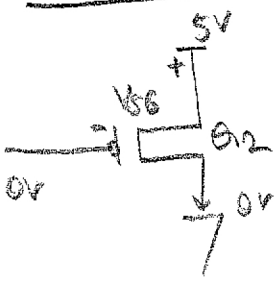



when A=0=0V


$V_{AS} = V_A - V_S = 0 - 0 = 0V$

$0V < V_+ = 1V$ , Then $\Theta_1$ is OFF.

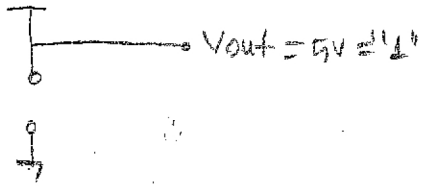
when A=0




$V_{SG} = V_S - V_G$

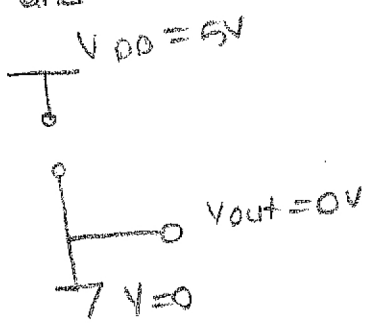
$= 5 - 0 = 5V > V_+ \Rightarrow \Theta_2$ is ON

So when A=0;



When $A=1$, $\Theta_1$ is ON, $\Theta_2$ is OFF.

and




Thus;

| A | Y |
|---|---|
| 0 | 1 |
| 1 | 0 |

**EX:** Build an $F = A \oplus B$ by using a cmos Logic (Assume complemented inputs are available)

Ans: $F = A \oplus B = AB' + A'B$ (serial, parallel)