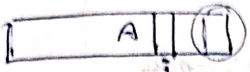
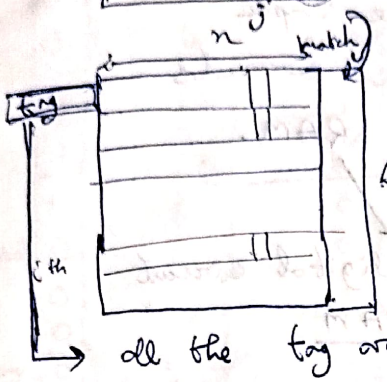
Mask register





Fully associative mapping

→ hence we need a method to search in O(1).

→ all the tag are checked

$F_{ij} = (A_j C_j + A_j' C_j')$

when M_i is true,

$$M_i = \prod_{i=1} F_{ij}$$

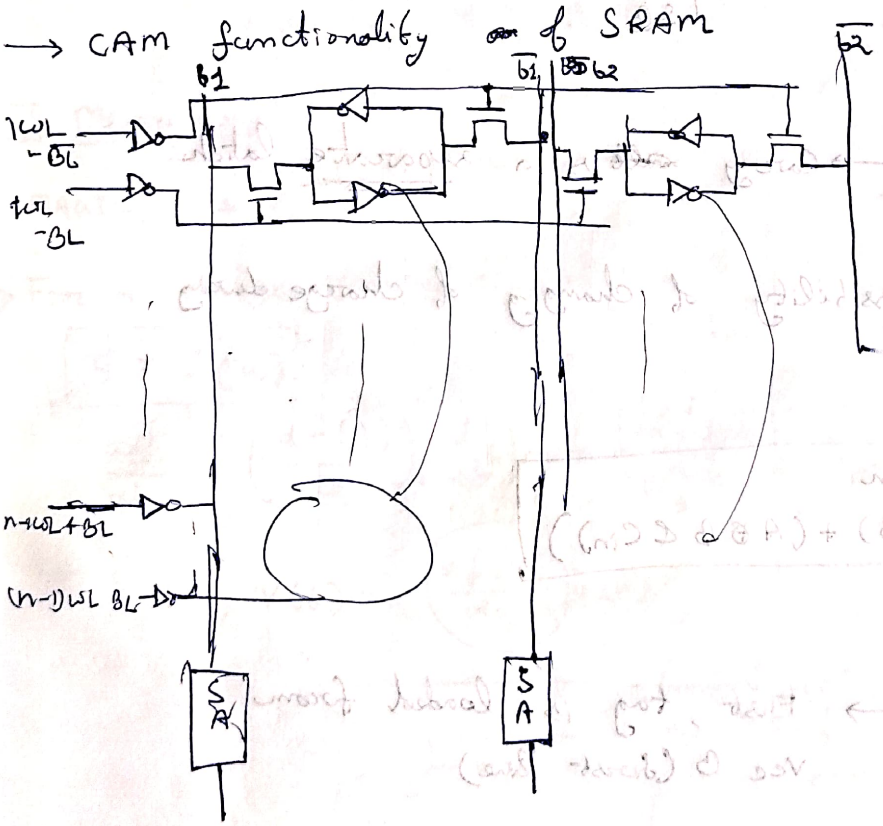
is true

CAM → Content wise searching

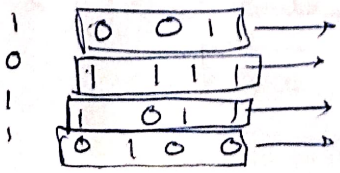
(*) This one above is like In-memory computing where we ~~have~~ what to search & in 'A'.

VII Searching in SRAM:

→ CAM functionality of SRAM



→ The key is to perform parallel search.



VI Bit-Serial Arithmetic:

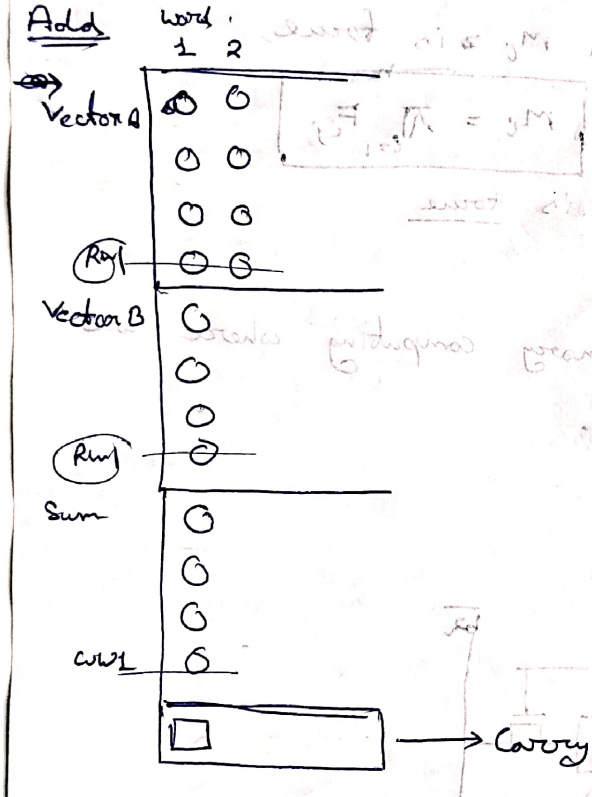
→ $S_i = x_i \oplus y_i \oplus c_{i-1}$

$c_i = x_{i-1} \oplus y_{i-1}$

→ we can do these simple things using logic circuits on peripheral & SRAM.

see digital circuit & SRAM.

Add



Vector A + Vector B = Sum

Reading from Vec A, Vec B

writing to sum.

Carry will be updated accordingly.

Sum will start from LSB.

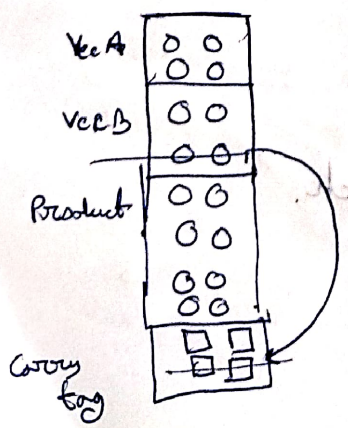
→ Carry is a separate latch

→ There's a possibility of changing of charge during changing phases.

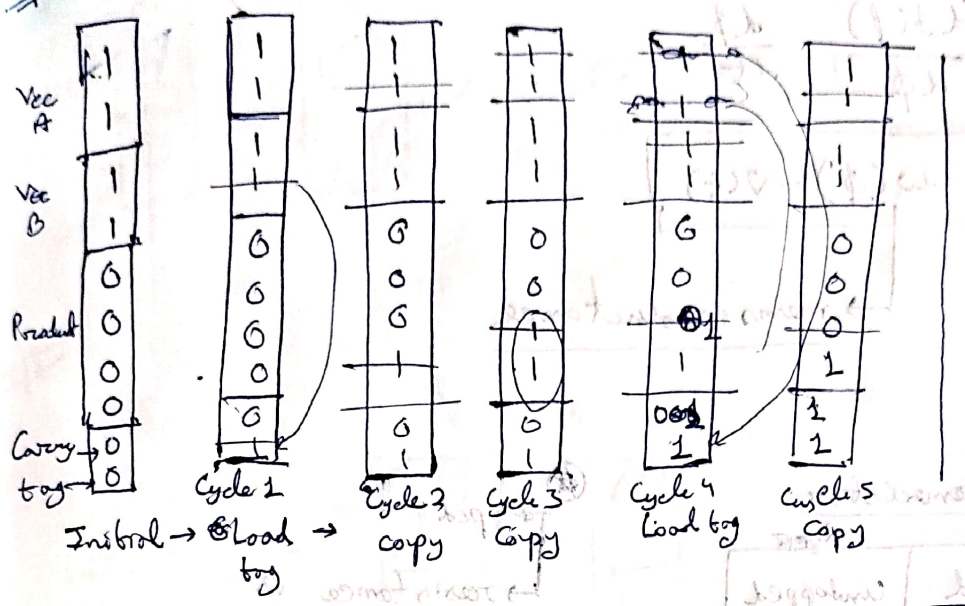
→ Sum → $A \oplus B \oplus C_{in}$

Carry → $(A \& B) + (A \oplus B \& C_{in})$

Multiplication:



→ First, tag is loaded from Vec B (lowest line)



(A) number of cycles needed n^2+5n-2 ($n \rightarrow$ number of bits)

but here in this case, the formula is not following.

Reduction :

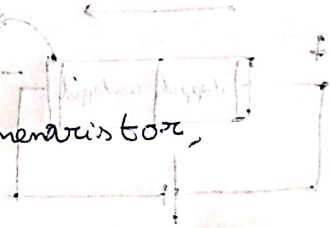
Reduction is a ~~case~~ common operation for DNNs.

28/02/24

Memristors

MAGIC, MAAJ,

For a charge controlled memristor,



$$\phi = f(q)$$

$$\frac{d\phi}{dt} = \left(\frac{df(q)}{dq}\right)\left(\frac{dq}{dt}\right) = \frac{df(q)}{dq} * i(t)$$

$$\therefore V(t) = \frac{df(q)}{dq} i(t)$$

→ Similar to $V = iR$

$$V(t) = M(q)\, i(t)$$

→ unit of $M(q)$ is ohm

For a flux controlled,

$$q = f(\phi)$$

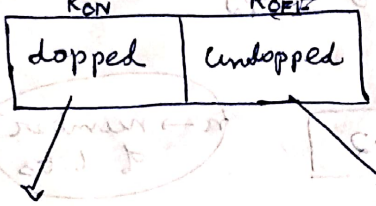
$$\frac{dv}{dt} = \frac{d f(\phi)}{d\phi} \cdot \frac{d\phi}{dt}$$

$$i(t) = W(\phi) \cdot v(t)$$

→ mems conductance

≠

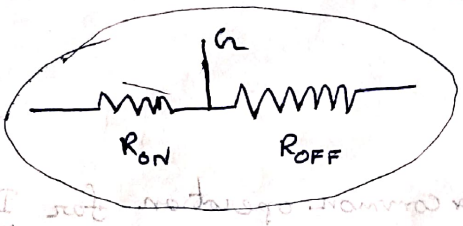
Memristor



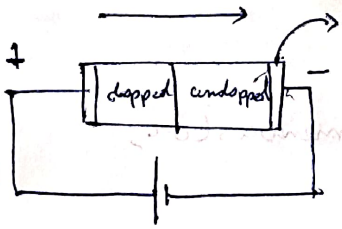
TiO_2

TiO_{2-x}

(Ti_4O_7)



Memristor can be two terminal, but we consider the 'G' then it also can be three terminal.

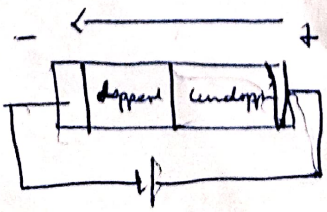


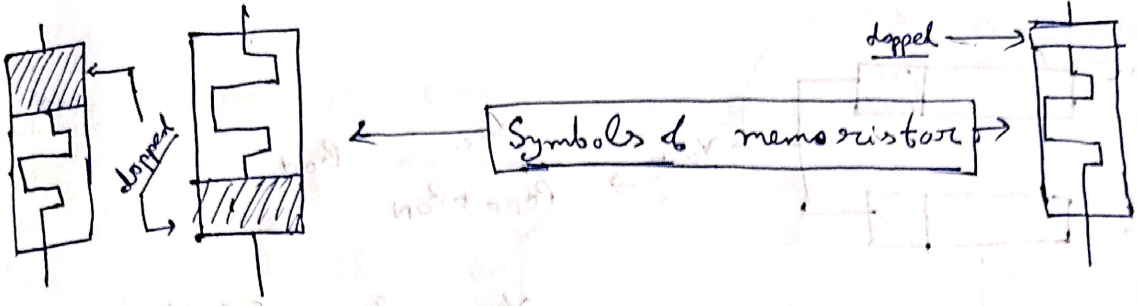
forward bias

→ Dopped area enhanced

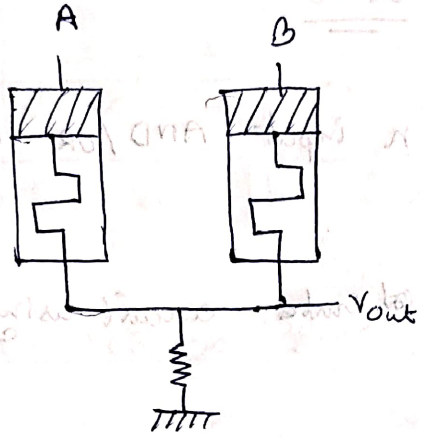
so, if current flow is as above, dopped area is enhanced & resistance is decreased. It is called forward bias.

In reverse bias, resistance is increased.

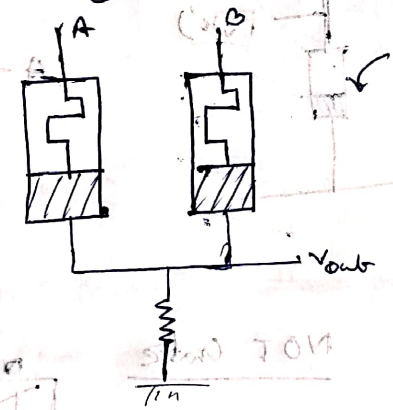




#



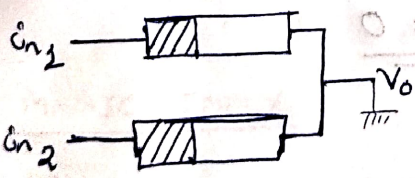
→ if we think memristor as diode, then the above picture is OR ~~Gate~~ Gate using memristor.



now this one cannot make 'AND'

→ Hence, this is not a viable design.

#



if

A	B	
0	0	Vout → 0
1	0	Vout → 1

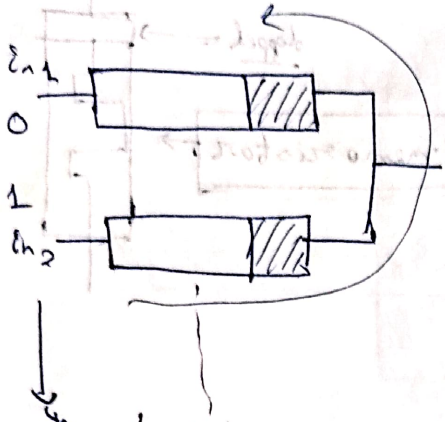
$$\frac{V_{cc}}{R_{ON} + R_{OFF}} \cdot R_{OFF}$$

$R_{ON} << R_{OFF}$

$= V_{cc}$

'OR' ~~Gate~~ Gate

→

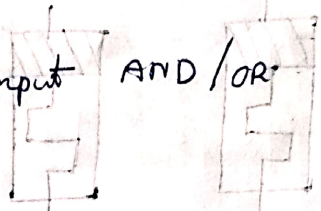


$$V_{out} \rightarrow \frac{V_{cc}}{R_{OFF} + R_{ON}} \cdot R_{ON}$$

$$\frac{V_{cc}}{R_{OFF}} \cdot R_{ON} \ll V_{cc}$$

AND' Gate

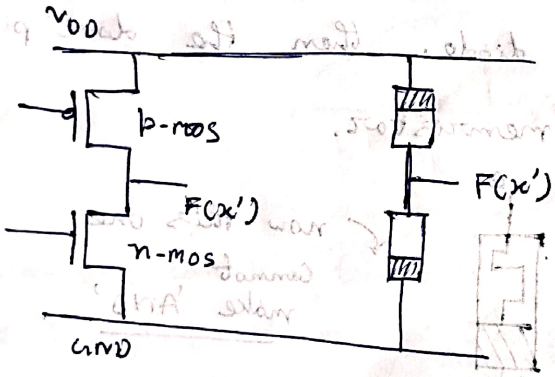
$V_{out} = 0$

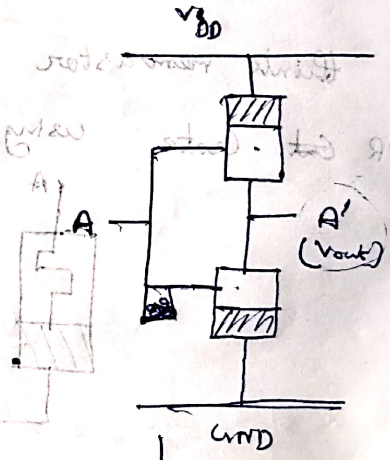


→ we can make these n input AND / OR Gate also.

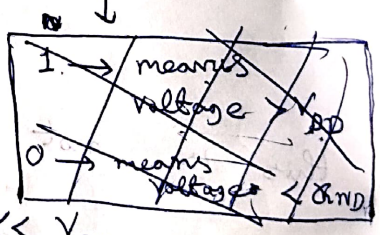
→ we can add 'NOT' Gate circuit using CMOS.

=





NOT Gate



if, $A = 1$,

$$V_{out} = \frac{V_{DD}}{R_{ON} + R_{OFF}} \cdot R_{OFF} \approx \frac{V_{DD}}{R_{OFF}} \ll V_{DD}$$

hence, $A' = 0$

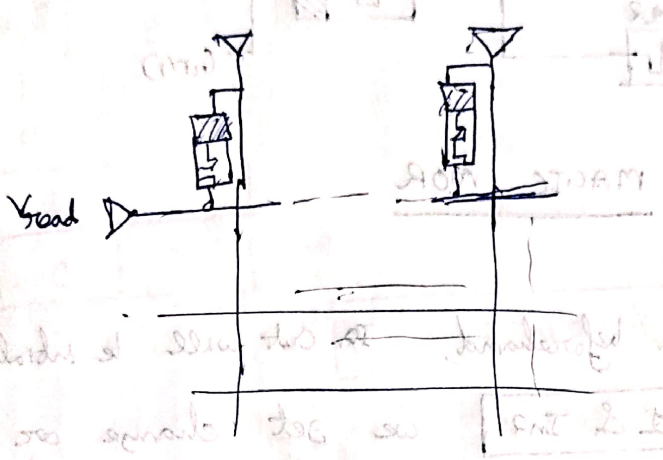
if, $A = 0$

$$V_{out} = \frac{V_{DD}}{R_{ON} + R_{OFF}} R_{OFF} \approx \frac{V_{DD}}{R_{OFF}} R_{OFF}$$

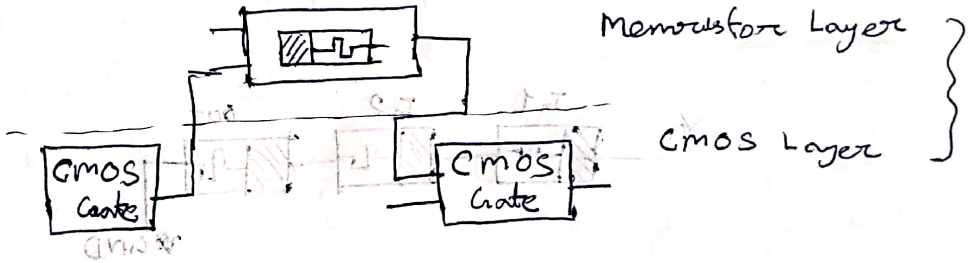
hence, $A' = 1$

05/03/24

Memristor Crossbar structure



Hybrid memristor - CMOS Logic

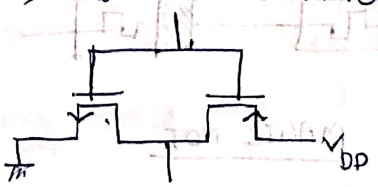


Memristor aided logic (MRL)

Shown previously. ~~(last pag~~

(Previous Page's)

To make NOR/NAND, add CMOS Inverter

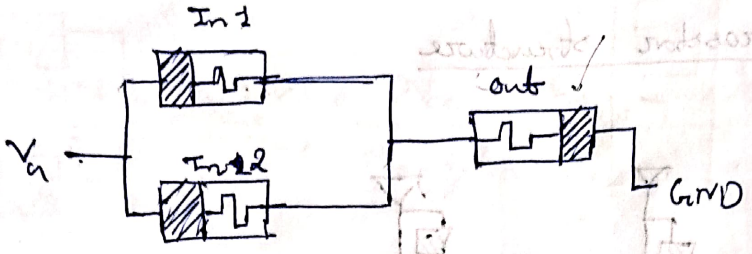


MAGIC, IMPLY

↳ Logic & memristor logic operation

MAGIC → memristor aided logic.

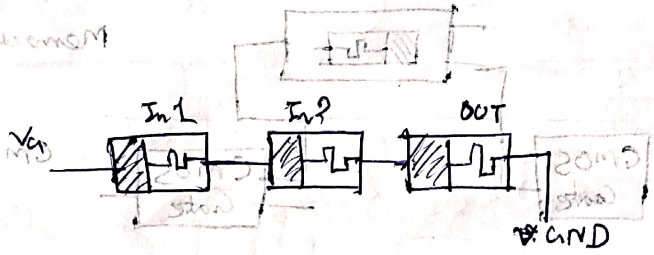
Magic is a stateful logic family



MAGIC NOR

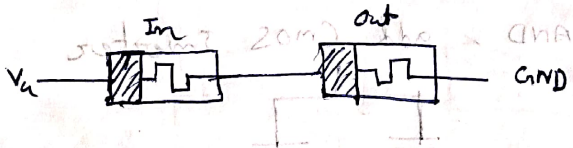
The values are stored beforehand, out will be initialised, then depending on In1 & In2 we get change or unchanged value of Out.

We have threesold valtage, to know what is '0' & what '1'.



MAGIC NAND

If current flow is above some valeue then state & Out resistor is changed.



MAGIC NOT

▷ MAGIC NOR in a ~~memristo~~ memristive memory.

1 ⟶ R_{ON} in memristor
0 ⟶ R_{OFF} in memristor

VII

Imply:

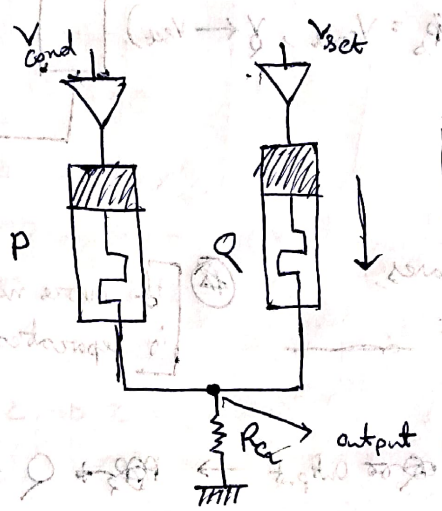
06/203/24

$P \rightarrow Q$: if P then Q

P(t)	Q(t)	Q(t + t_0)
0	0	1
0	1	1
1	0	0
1	1	1

P \ Q	0	1
0	1	1
1	0	1

$\therefore P \rightarrow Q \equiv \sim P \wedge Q$



$V_{cond} < V_{Th} < V_{set}$

$R_{ON} < R_{eq} < R_{OFF}$

→ in case, P = 1, Q = 0, the output voltage is similar to V_{cond}, hence output is '0'. ($V_{cond} < V_{Th}$)

Realize NAND:

→ result will be stored in S

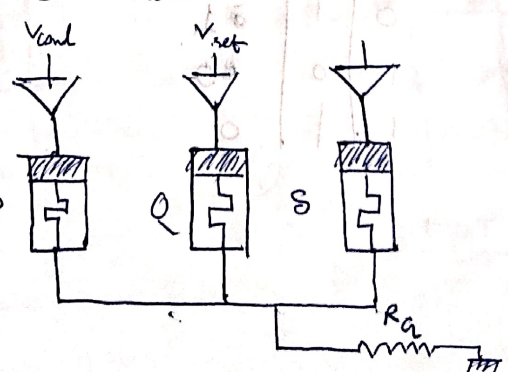
~~FALSE(S)~~ Step 1: FALSE (S) ($V_S = V_{CLEAR}$)

as if, we are doing NAND b/w 'P' & 'Q'

Step 2: P $\xrightarrow[\text{Imply}]{}$ S (V_P = Cond, $V_S = V_{set}$)

Step 3: Q $\xrightarrow[\text{Imply}]{}$ S ($V_Q = V_{cond}$, $V_S = V_{set}$)

Three steps to realize ~~non~~ NAND



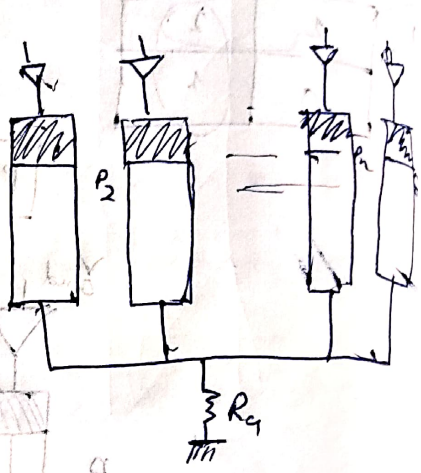
So, to put it simply,

1. $S \leftarrow 0$
2. $S \leftarrow \overline{P} + S$
3. $S \leftarrow \overline{Q} + S$

} this will convert into $\equiv \overline{QP}$ NAND

≠

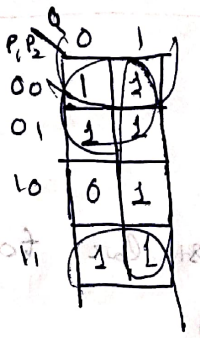
NOR: $Q = (P_1 + P_2 + \dots + P_n)'$



*Step 1 : FALSE (Q) ($V_Q \leftarrow V_{clear}$)

Step 2 : $P_s \rightarrow Q$ ($V_{P_s} = V_{cond}$, $V_Q \leftarrow V_{set}$)

→ if we consider two cases,

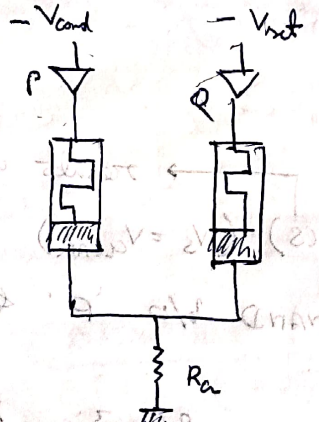


∴ Output → Q +

(#) If more number of inputs R_c is separated for all inputs

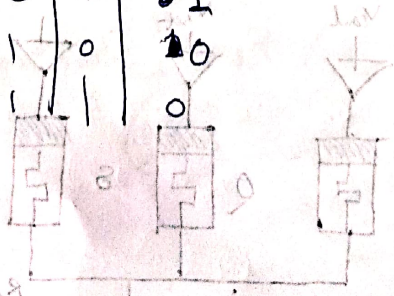
PB-Imp → Positive Biased Implies

≠



The implied logic negative representation

P	Q	Q (Left)
0	0	0
0	1	1
1	0	0
1	1	0



IMPLY XOR

12/03/24

FALSE(M1), FALSE(Z), → X IMPLY

We need 13 steps to make XOR

1 → FALSE(M1) ⇒ M1 = 0

2 → FALSE(Z) ⇒ Z = 0

3 → X IMPLY Z ⇒ Z = X'

4 → Z IMPLY M1 ⇒ M1 = X (X is copied to M1)

5 → FALSE(M2) ⇒ M2 = 0

6 → FALSE(Z) ⇒ Z = 0

7 → Y IMPLY Z ⇒ Z = Y'

8 → Z IMPLY M2 ⇒ M2 = Y (Y is now copied to M2)

9 → Y IMPLY M1 ⇒ M1 = Y' + X

10 → FALSE(Z) ⇒ Z = 0

11 → M1 IMPLY Z ⇒ Z = (Y' + X)'

12 → X IMPLY M2 ⇒ M2 = X' + Y'

13 → M2 IMPLY Z ⇒ Z = (X' + Y')' + (Y' + X')' = XY' + X'Y

IMPLY with Crossbar: (⟶)

IMPLY adders

→ $S_i = (A_i \oplus B_i) \oplus C_{i-1}$

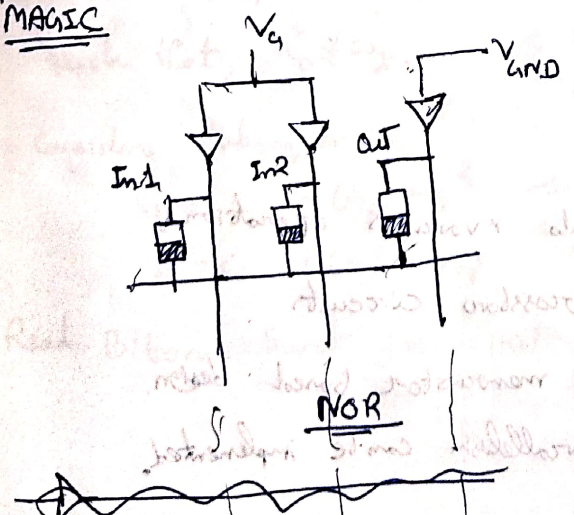
→ $C_i = (A_i \to (B_i \to '0'))$ ~~$C_i = (A_i \to (B_i \to '0'))$~~

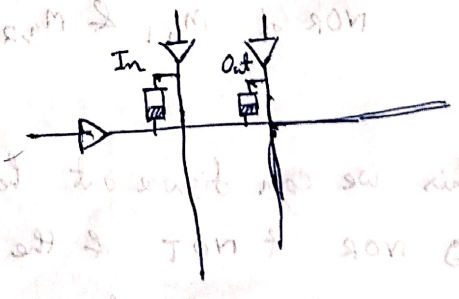
→ $C_i = (A_i \to (B_i \to '0')) \to ((C_{i-1} \to ((A \oplus B_i) \to '0')) \to '0')$

MUX with IMPLY

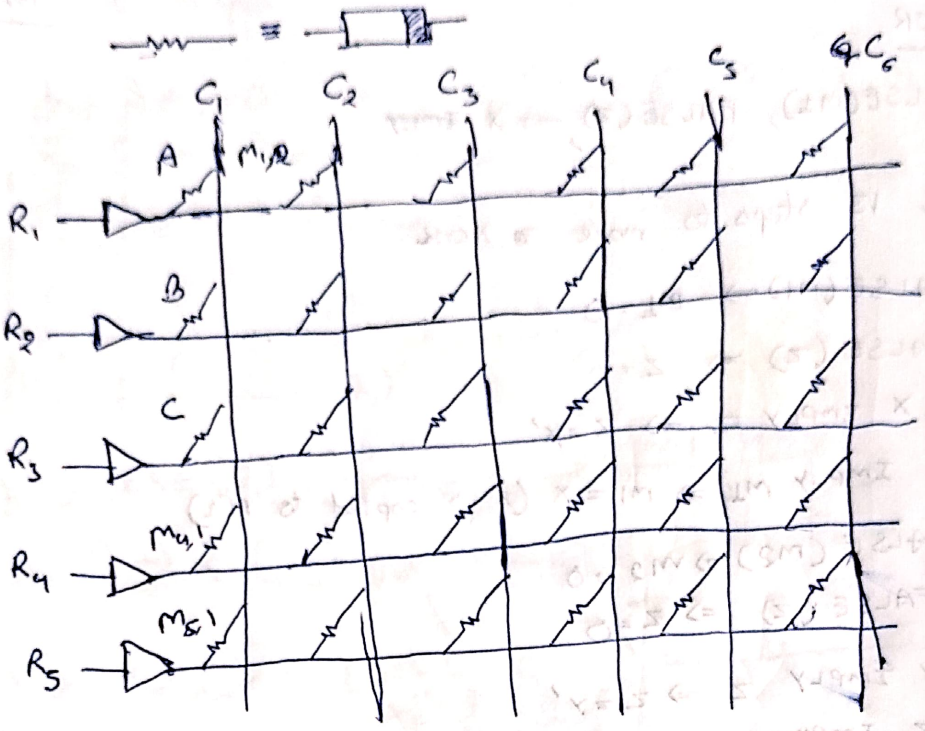
13/03/24

MAGIC





5×6



→ ~~Initially~~ Initialize A, B, C

Columns C_i should be ; R_{ON} → logic 1

$M_{B,1} \rightarrow R_{ON}$

$M_{5,1} \rightarrow R_{ON}$

▷ Step 1 : $M_{1,2}$: $\bar{A}$ ⟶ $\frac{V_0}{C_1}$ $\left(\frac{V_{GND}}{C_2}\right)$

$M_{2,2} = \bar{B}$

to negate 'A' in $M_{1,2}$

▷ Step 2 $M_{4,1} = \overline{A+B}$

$M_{4,2} = \overline{\bar{A}+\bar{B}}$

$\left[\begin{array}{l} V_0 \rightarrow R_4 \\ GND \rightarrow R_1, R_2 \end{array}\right]$

$X \rightarrow A \oplus B$

▷ Step 3 : $M_{4,3} = X$

$$X = \overline{(\overline{\bar{A}+\bar{B}}) + (\overline{A+B})}$$

NOR of $M_{4,1}$ & $M_{4,2}$

→ In this we can figure out how do various operations using NOR & NOT & the crossbar circuits.

→ Sorting can also be realised in memristor based design.

→ In several sorting techniques, parallelism can be implemented.

a_i $0 \le i$ $i \to n-1$

$a_i \geq a_{i-1}$ $a_{i+1} < a_i$

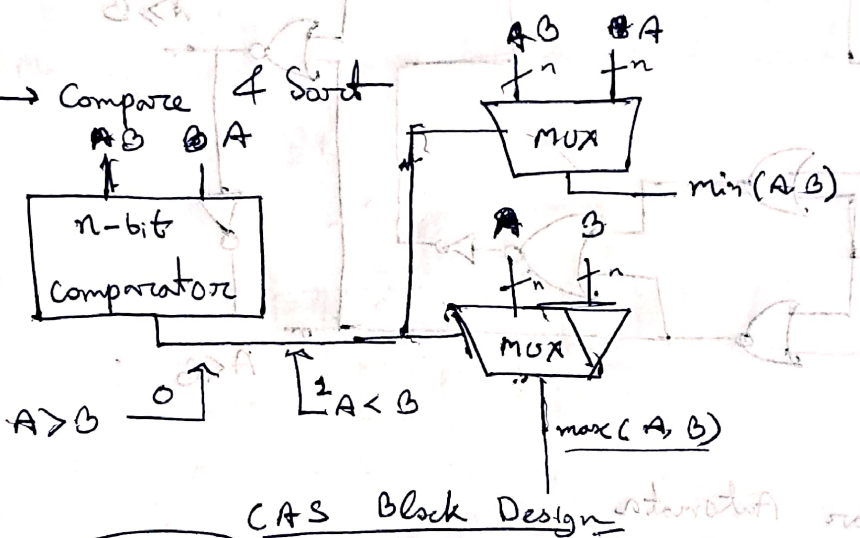
one ascending order
one descending order

→ Bitonic sequence

4, 5, 4, 3, 2, 1, 1, 2, 3

1, 2, 3, 4, 5, 4, 3, 2, 1 → bitonic sequence

CAS → Compare & Sort



CAS Block Design

now > $n = 2^l$ → to realize bitonic sort

Date: 19/03/234

Cost in computation,

$C_p = P \times T_p$ → Time

Processor

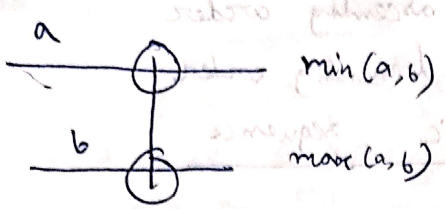
$S = \langle a_0, a_1, \ldots, a_{n-1} \rangle$ be a bitonic sequence such that, $a_0 \le a_1 \ldots \le a_{n/2-1} \not\le a_{n/2} \ge a_{n/2+1} \ldots \ge a_{n-1}$

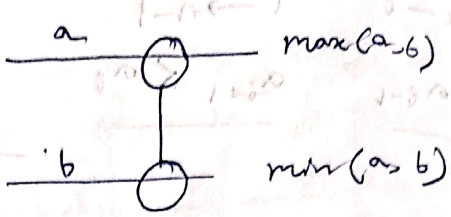
Consider subsequences,

$S1 = \langle \min(a_0, a_{n/2}), \min(a_1, a_{n/2-1}) \ldots \rangle$

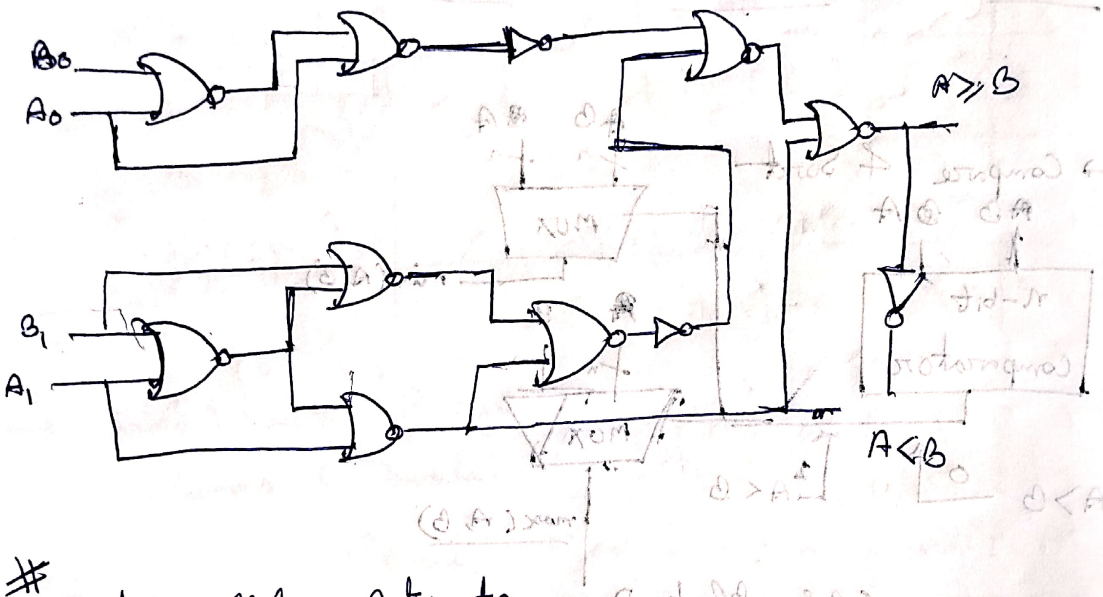
$S2 = \langle \max(a_0, a_{n/2}), \max(a_1, a_{n/2-1}) \ldots \rangle$

Read Bitonic Sort from GFG & see diagram.





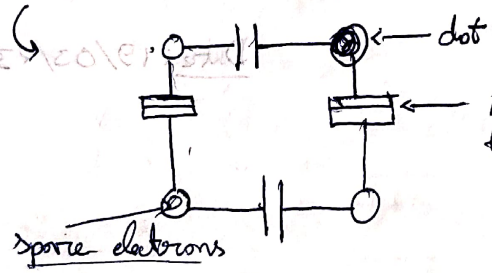
NOR-NOT Based 2-bit comparator:



Quantum cellular Automata

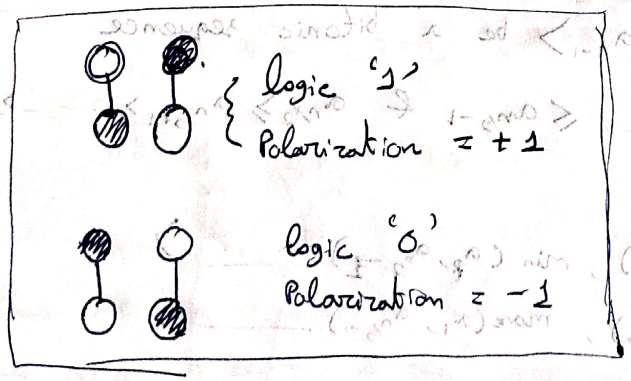
QCA

↳ QCA cell



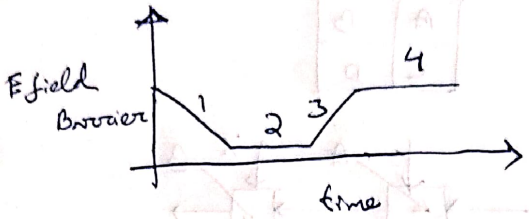
↳ Aluminium islands on SiO_2 called dots.

→ Polarization decides logic level.



Signal propagation :

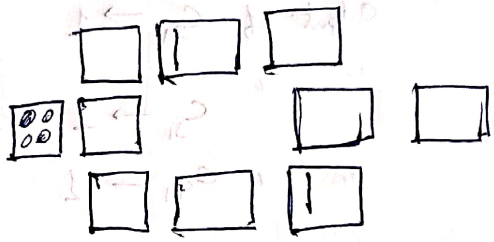
→ Release → Relax → Switch → Hold



20/03/24

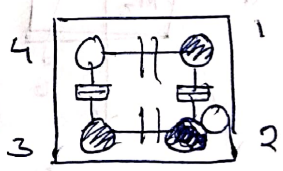
#

QCA Majority Gate + QCA inverter = Universal gate



NAND NOR INVERTER (NNI)

$$P = \frac{(P1 + P3) - (P2 + P4)}{P1 + P2 + P3 + P4}$$



→ Polarization : P measures the extent to which the electronic charge is distributed among four dots.

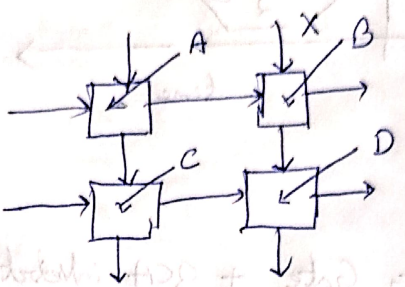
#

$f_1 = \Sigma(xy + yz + zx)$

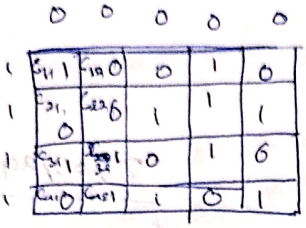
$f_2 = xz' + yz$

Akers Array

A	B
C	D



#



$xz' + yz$

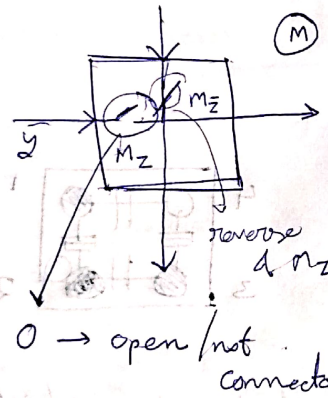
Output of $C_{11} \rightarrow 1$

Output of $C_{21} \rightarrow 1$

$C_{31} \rightarrow 1$

$C_{41} \rightarrow 1$

#



0 → open / not connected

1 → closed / connected

A = 1010

B = 0110

HA