30/09/2022

* COMBINATIONAL CIRCUITS:

Interconnection of logic gates whose outputs depend only on present inputs.

→ Adders, Subtractors, Multiplexer, Decoder, Encoder

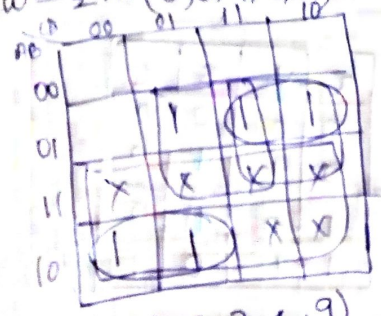
** Design Procedure:

1) I/P specifications ; o/p logic circuits
2) Truth table : relation between I/Ps & O/Ps.
3) K-Map and write Boolean function for each O/P
4) Draw logic circuit.

** Logic circuit to convert BCD to excess 3 code

I/P				O/P			
A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	1	1
0	0	0	1	0	1	0	0
0	0	1	0	0	1	0	1
0	0	1	1	0	1	1	0
0	1	0	0	0	1	1	1
0	1	0	1	1	0	0	0
0	1	1	0	1	0	0	1
0	1	1	1	1	0	1	0
1	0	0	0	1	0	1	1
1	0	0	1	1	1	0	0

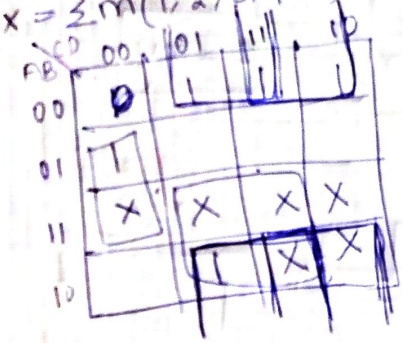
$W = \sum m(5,6,7,8,9)$



$W = A + BD + BC$

$= A + B(C+D)$

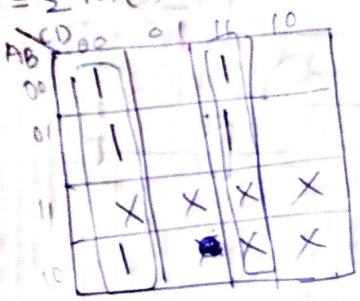
$X = \sum m(1,2,3,4,9)$



$X = BC'D' + B'D + B'C$

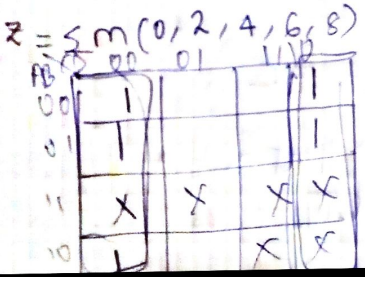
$= B'(C+D) + B(C+D)'$

$Y = \sum m(0,3,4,7,8)$



$Y = C'D' + CD$

$Z = \sum m(0,2,4,6,8)$

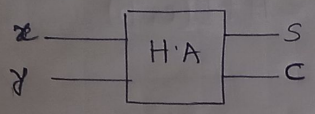


$Z = C'D' + CD'$

$Z = D'$

* → BINARY ADDER - SUBTRACTOR:

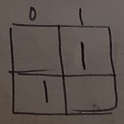
1) Half Adder:



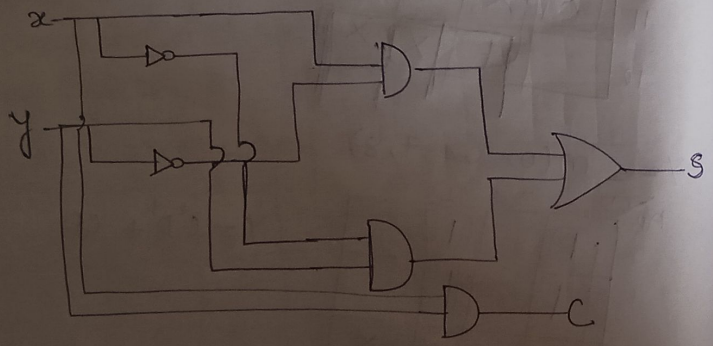
x	y	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

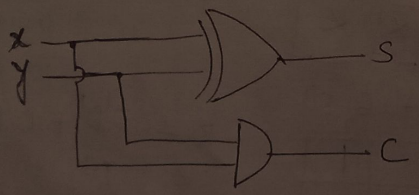
$S = \Sigma m(1,2)$

$C = \Sigma m(3)$



$S = x'y + xy'$ · $C = xy$



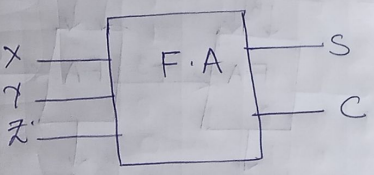


Combinational circuit that performs addition of 2 bits is half adder

2) Full Adder:

Combinational circuit that performs addition of 3 bits (2 significant bits; 1 previous carry) is a full adder

x	y	z	S	C
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1



$S = \Sigma m(1,2,4,7)$

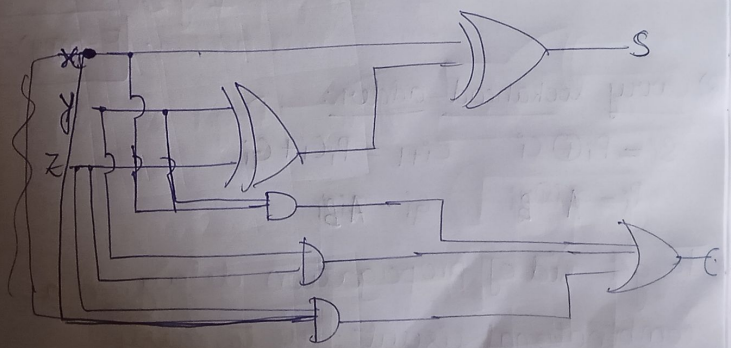
x \ yz	00	01	10	11
0		1		1
1	1		1	

$S = X \oplus Y \oplus Z$

$C = \Sigma m(3,5,6,7)$

x \ yz	00	01	11	10
0			1	
1			1	1

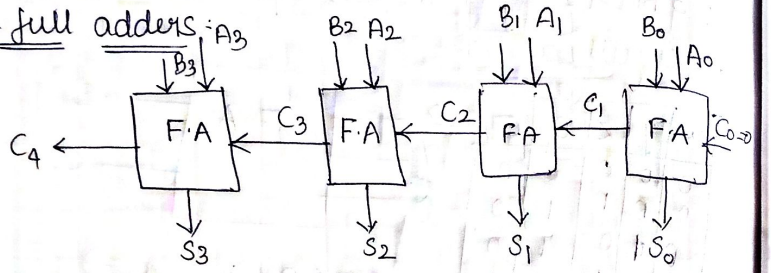
$C = xy + yz + zx$



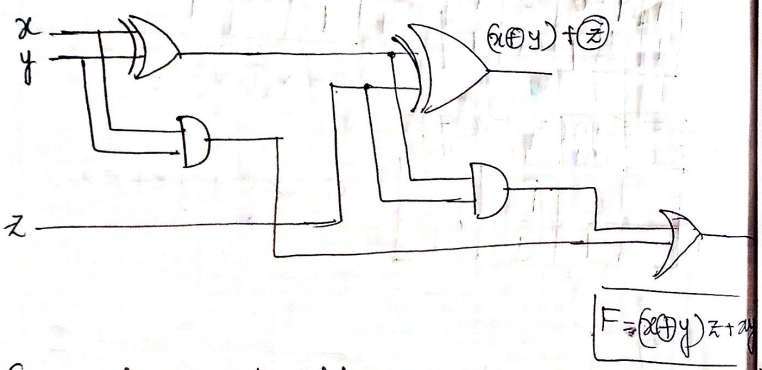
* Binary Adder:

1) Ripple carry adder:

4 full adders:



* Two half adders into a full adder

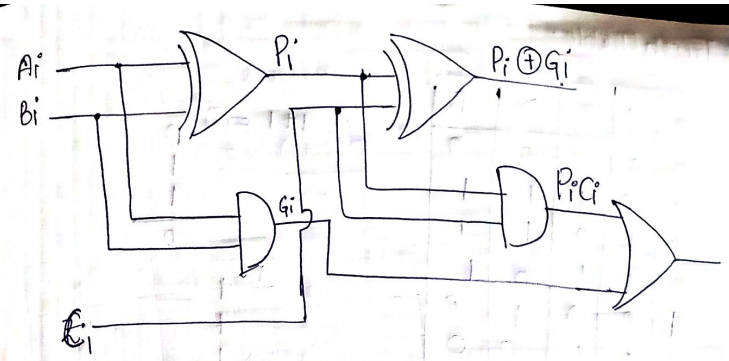


2) Carry lookahead adder:

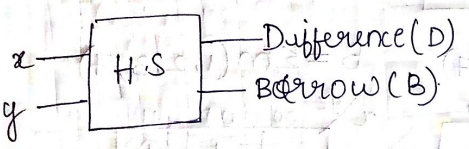
$S_i = P_i \oplus C_i \qquad C_{i+1} = P_i C_i + G_i$

$P_i = A_i \oplus B_i \qquad G_i = A_i B_i$

To get rid of propagation delay in a combinational circuit; due to carry propagation; we use carrylook ahead logic.



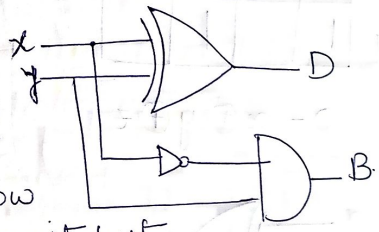
* Subtractor:



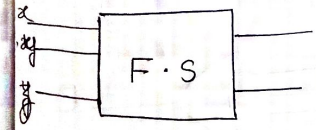
x	y	D	B
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

$D = x'y + xy'$

$B = x'y$



→ Half subtractor do not take borrow from previous circuit but which is required in real time scenario.



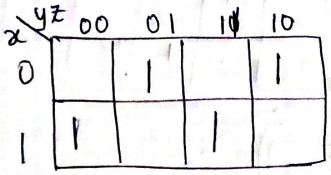
⇒ $x-(y+z)$

* Truth Table:

x	y	z	D	B
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

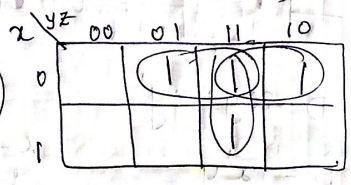
1+1 = ①

$D = \Sigma m(1,2,4,7)$

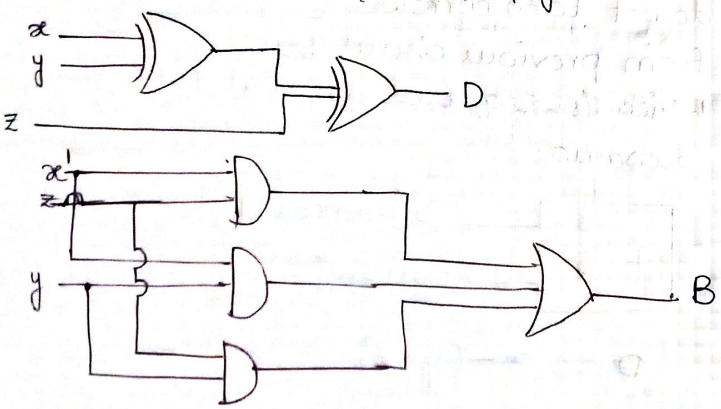


$D = x \oplus y \oplus z$

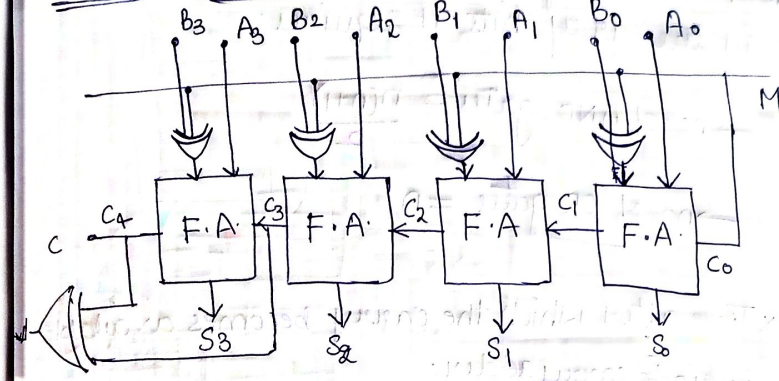
$B = \Sigma m(1,2,3,7)$



$B = x'z + x'y + yz$



* Adder - Subtractor:



→ Addition & subtraction operations can be done using single circuit by inserting an XOR gate with each full adder.

Mode M controls the operation

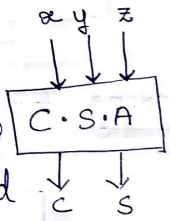
$M = 0$; $C_0 = 0$ ⟶ adder ∵ $B \oplus 0 = B$.

$M = 1$; $C_0 = 1$ ⟶ subtractor ∵ $B \oplus 1 = B'$

3) Carry - Save Adder:

x: 11011
y: 01100
z: 01001
S: 11110
C: 01001
11000

carry is saved separately and then added to sum to get the original sum value.



* for n-bit carry lookahead adder to evaluate all the carry bits it requires:

→ no. of AND gates = $\frac{n(n+1)}{2}$

→ no. of OR gates = n

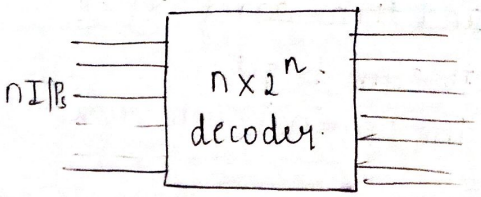
* Time after which the carry becomes available in ripple carry adder.

= (total no. of full adder used to get c_i) × carry propagation delay of full adder.

* Sum (s_x) time required =

= (total no. of full adder used to get s_i) × (carry propagation delay of full adder) + sum propagation delay of full adder.

** DECODER: for n inputs, we get 2^n outputs



x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	1	0	0	0	0	0	0	0
0	0	1	0	1	0	0	0	0	0	0
0	1	0	0	0	1	0	0	0	0	0
0	1	1	0	0	0	1	0	0	0	0
1	0	0	0	0	0	0	1	0	0	0
1	0	1	0	0	0	0	0	1	0	0
1	1	0	0	0	0	0	0	0	1	0
1	1	1	0	0	0	0	0	0	0	1

⇒ Binary to octal decoder.
(3×8 decoder).

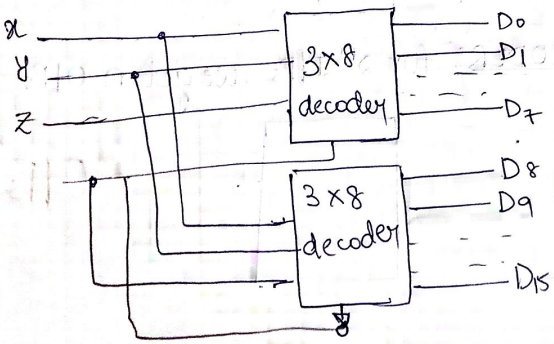
⇒ 2×4 line decoder with enable I/P:

Active when E = 0.

E	x	y	D_0	D_1	D_2	D_3
1	x	x	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

A decoder works (or) operates as demultiplexer

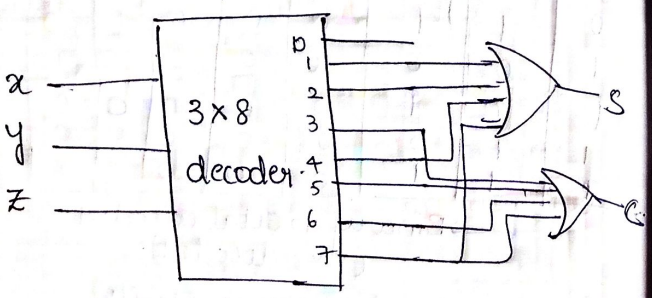
* 4×16 decoder using 3×8 decoder:



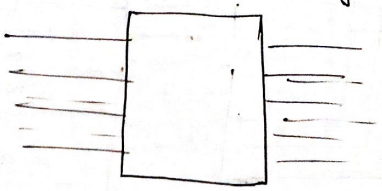
* Full adder using 3x8 decoder.

$S = \Sigma m(1,2,4,7)$

$C = \Sigma m(3,5,6,7)$



* Purpose of decoder is generate 2^n or fewer O/Ps for n I/Ps such that each combination of I/P will assert a unique O/P.
* Decoders include one or more enable I/Ps to control the circuit operation.
* A decoder with enable input can function as de-multiplexer.
* ENCODER: for 2^n I/Ps, we get n O/Ps



* Octal to binary encoder.

D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	x	y	z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

$x = D_4 + D_5 + D_6 + D_7$ $\quad z = D_1 + D_3 + D_5 + D_7$

$y = D_2 + D_3 + D_6 + D_7$

→ When more than 1 I/P is high; unexpected O/P will be resulted. Therefore; we use priority encoder.

* Priority Encoder:

D_0	D_1	D_2	D_3	x	y	V
0	0	0	0	X	X	0
1	0	0	0	0	0	1
X	1	0	0	0	1	1
X	X	1	0	1	0	1
X	X	X	1	1	1	1

$x = D_2 + D_3$

$y = D_1 + D_3$

→ An encoder has 2^n or fewer I/Ps & generate n O/P lines.

→ Priority Encoder is an encoder which includes a priority function.

The operation of priority encoder is such that if 2 or more I/Ps are high at the same time, the I/P having highest priority will take precedence.

* Multiplexer (MUX)

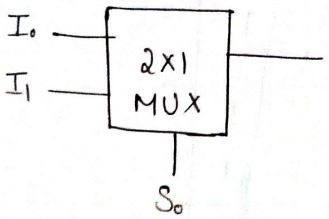
for n I/P lines, we get only a single O/P.

→ selection of particular I/P line is done using selection lines.

for 2^n I/Ps, there are n selection lines.

→ Multiplexer is called data selector.

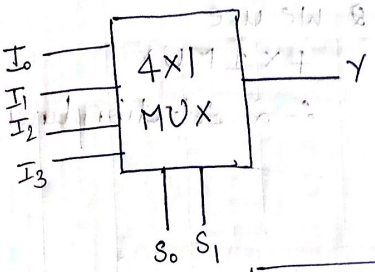
→ 2 x 1 MUX:



S_0	Y
0	I_0
1	I_1

$Y = S_0'I_0 + S_0I_1$

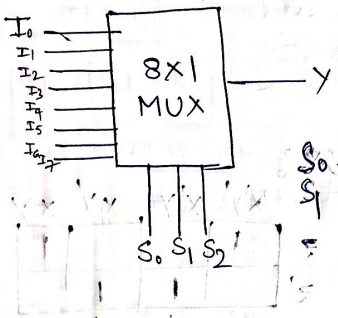
→ 4x1 MUX:



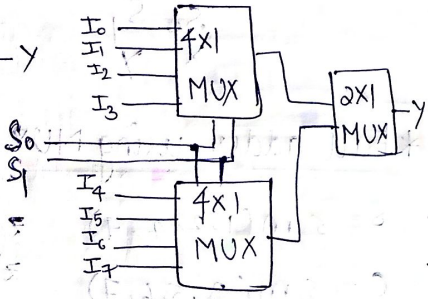
S_0	S_1	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

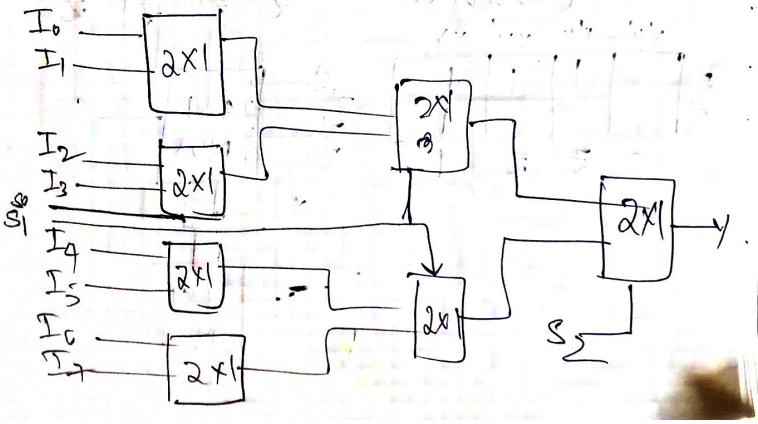
$Y = S_0'S_1'I_0 + S_0'S_1I_1 + S_0S_1'I_2 + S_0S_1I_3$

→ 8x1 MUX:



using 4 x 1



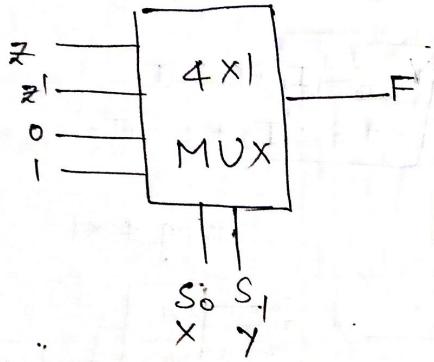


* $F = \Sigma(1,2,6,7)$

	$x'y'$	$x'y$	xy'	xy
z	1			1
z'		1		1
	z	z'	0	1

3 variable function

@ We use
4 × 1 MUX
∴ 2 selection lines.



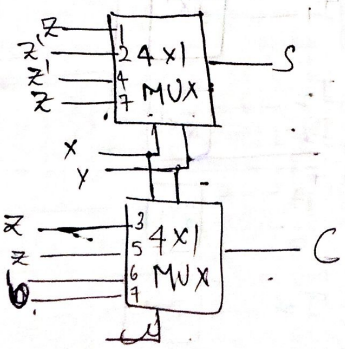
* Full adder using MUX:

$S = \Sigma m(1,2,4,7)$

$C = \Sigma m(3,5,6,7)$

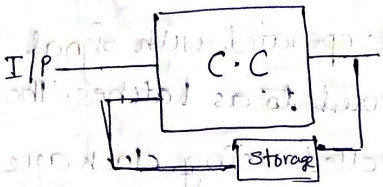
	$x'y'$	$x'y$	xy'	xy
z	1		1	
z'		1		1
	z'	z	z'	z

	$x'y'$	$x'y$	xy'	xy
z		1	1	1
z'				1
	0	z	z	1



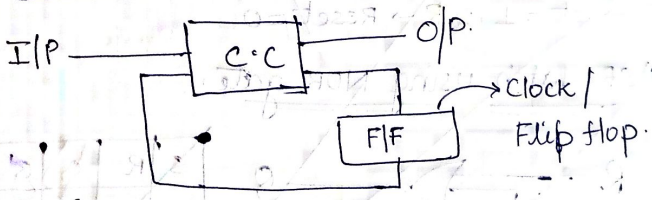
* SEQUENTIAL CIRCUITS:

→ The sequential circuit is a logic circuit or a combinational circuit where the output depends on both present input and previous input.



→ Sequential Circuits → Synchronous
→ Asynchronous.

Synchronous sequential circuits:



Asynchronous circuits use Latches

Both flipflops and latches are memory elements.

→ A synchronous sequential circuit is a system whose behaviour can be defined

from its input signals at discrete instances of time.

→ Synchronisation of a circuit is done using clock generator which provides periodic input signals.

→ Storage elements operated with signal levels are referred to as Latches. Those memories controlled using clock are called Flip-flops.

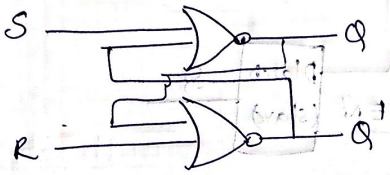
Latches → Level-sensitive.

Flip-flops → Edge-sensitive

* LATCHES:

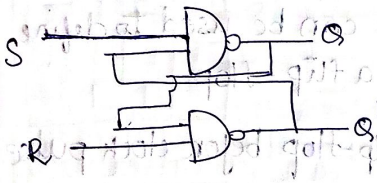
S → set = 1 ; R → Reset = 0

1) SR latch using NOR gate:



S	R	Q	Q'
0	0	Memory	
0	1	0	1
1	0	1	0
1	1	Invalid	

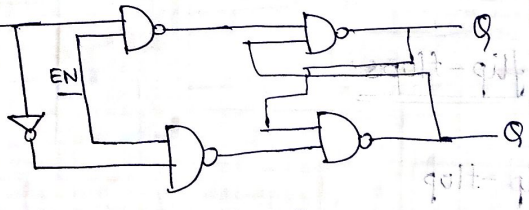
2) SR latch using NAND gate:



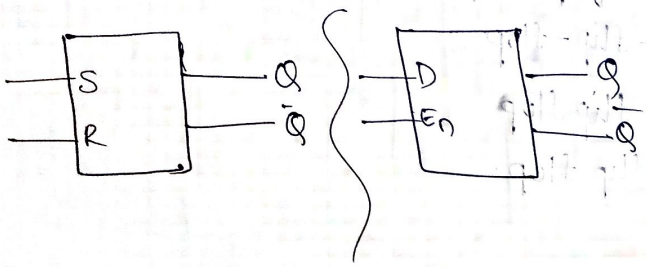
S	R	Q	Q'
0	0	Invalid	
0	1	1	0
1	0	0	1
1	1	Memory	

3) D-latch:

1)



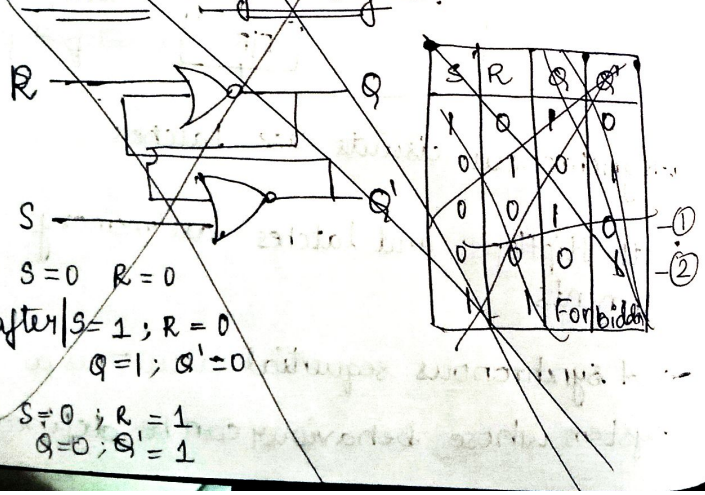
EN	D	Q_{n+1}
0	X	Memory
1	1	1
1	0	0



* LATCHES:

S → set = 1 ; R → Reset = 0

1) SR latch using NOR gate:



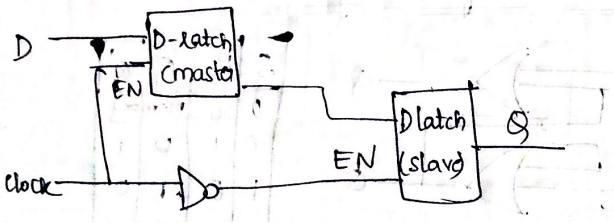
at S = 0 R = 0

① after S = 1 ; R = 0
Q = 1 ; Q' = 0

② S = 0 ; R = 1
Q = 0 ; Q' = 1

* Flip-Flops:

Edge-Triggered:



Master-slave D-flip flop

* A characteristic table can be used to define logical properties of a flip-flop

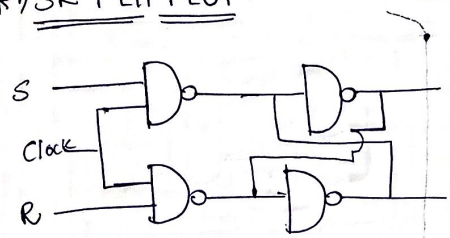
Q(t) → state of flip-flop before clock pulse.

Q(t+1) → state after the clock pulse.

* Types of flip-flops:

1) 'SR' flip-flop
2) D-flip-flop
3) JK flip flop
4) T-flip flop

*1) SR FLIP FLOP:

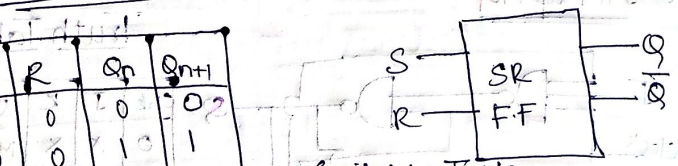


Truth Table

Clock	S	R	Q	Q̄
0	X	X	Memory	
1	0	0	Memory	
1	0	1	0	1
1	1	0	1	0
1	1	1	Invalid	

Characteristic Table:

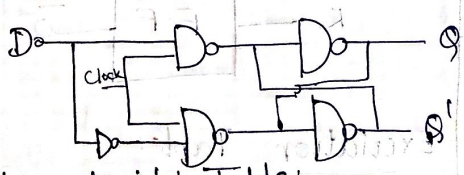
S	R	Qn	Qn+1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	Invalid
1	1	1	Invalid



Excitation Table

Qn	Qn+1	S	R
0	0	0	X
0	1	1	0
1	0	0	1
1	1	X	0

2) D FLIP FLOP:



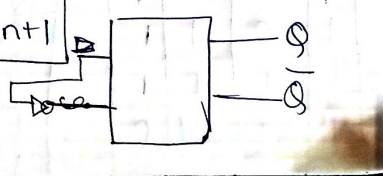
Truth Table

C	D	Q	Q̄
0	X	Memory	
1	0	0	1
1	1	1	0

Characteristic Table:

D	Qn	Qn+1
0	0	0
0	1	0
1	0	1
1	1	1

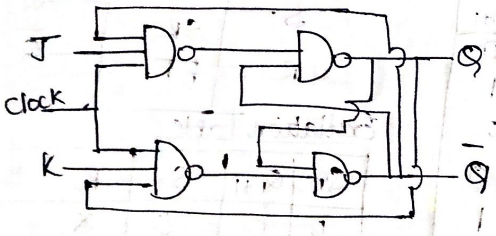
D = Qn+1



* Excitation Table:

Q_n	Q_{n+1}	D
0	0	0
0	1	1
1	0	0
1	1	1

3) JK Flip Flop:



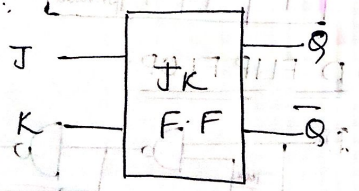
Truth Table:

Clock	J	K	Q	$\bar{Q}$
0	x	x	Memory	
1	0	0	Memory	
1	0	1	0	1
1	1	0	1	0
1	1	1	Toggling	

In a certain: if the values are changing continuously i.e. called Racing

Characteristic Table:

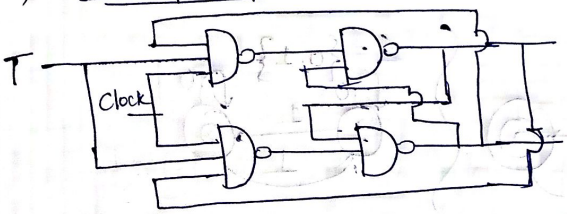
J	K	Q_n	Q_{n+1}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0



Excitation Table:

Q_n	Q_{n+1}	J	K
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

4) T- Flip Flop:



Truth Table:

Clock	T	Q	$\bar{Q}$
0	x	Memory	
1	0	Memory	
1	1	Toggling	

Characteristic Table:

T	Q_n	Q_{n+1}
0	0	0
0	1	1
1	0	1
1	1	0

Excitation Table:

Q_n	Q_{n+1}	T
0	0	0
0	1	1
1	0	1
1	1	0

Two types of Sequential Circuits:

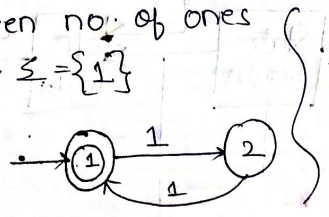
1) Mealy Model: Present state; I/P & O/P

2) Moore Model: Present state & O/P

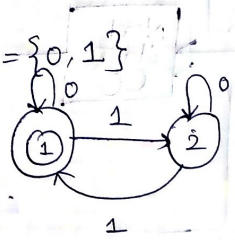
○ ⟶ Initial / ON

◎ ⟶ Final / OFF

1) Even no. of ones with $\Sigma = \{1\}$

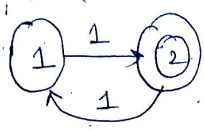


$\Sigma = \{0, 1\}$

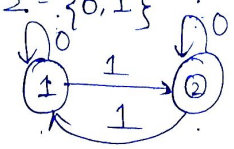


2) Odd no of ones:

$\Sigma = \{1\}$



$\Sigma = \{0,1\}$

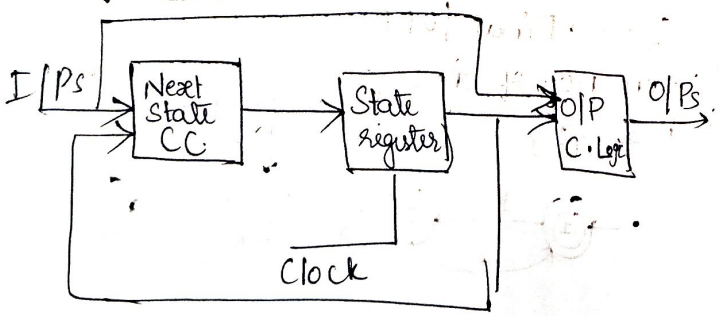


* Procedure to analyse state diagram to Sequential circuit:

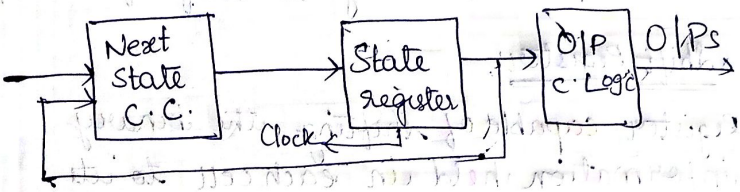
1) State diagram
2) Label the states in binary function → State reduction
3) State Table Preparation
4) Characteristic Table & Excitable
5) Boolean function of Inputs of Flip flops
6) Sequential Circuit

* Block Diagram of

1) Mealy Machine

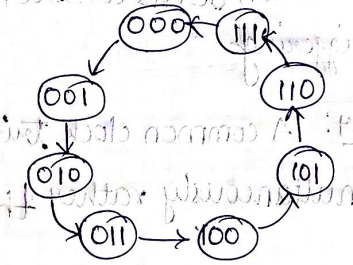


2) Moore Machine:

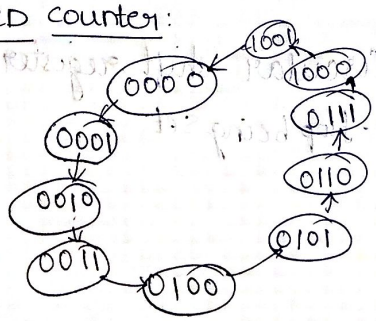


* Two states are said to be equivalent, if each member of the set of I/Ps they give exactly the same O/P and send the circuit either to same or equivalent state.

* Binary Counter:



* BCD counter:



* → Register is a group of flipflops each of which and is capable of storing 1 bit of information

→ Shift Register:

Register capable of shifting the binary information held in each cell to its neighbouring cell in specified direction

* → Counter is essentially a register that goes through a pre-determined sequence of a binary state.

Ripple Counter: Transistion serves as source of triggering

Synchronous Counter: A common clock triggers all flip-flops simultaneously rather than 1 at a time

* → Ring counter: Circular shift register with only one flip-flop being set.

* PLA, PAL, ROM:

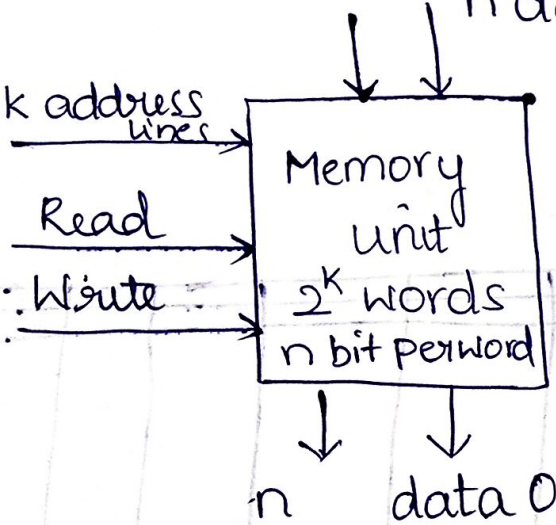
→ Programmable Logic Devices:

1) ROM → Read only Memory
2) PLA → Programmable Logic Array
3) PAL → Programmable Array Logic.

Memory: → ROM (& Read)
→ RAM (volatile)

* Block Diagram of Memory unit:

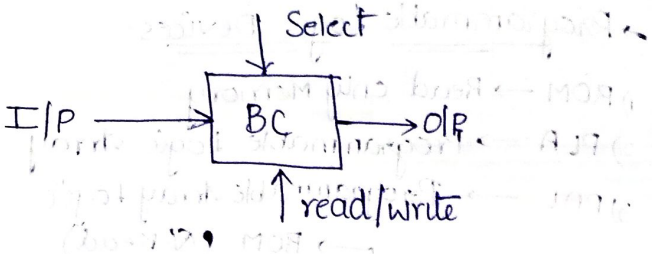
1 byte = 8 bits
1 KB = 1024 bits
1 MB = 2^{20}
1 GB = 2^{30}



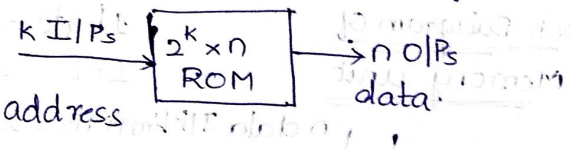
→ Control I/Ps to memory chip:

Memory Enable	Read/Write	Operation
0	X	None
1	0	Write
1	1	Read.

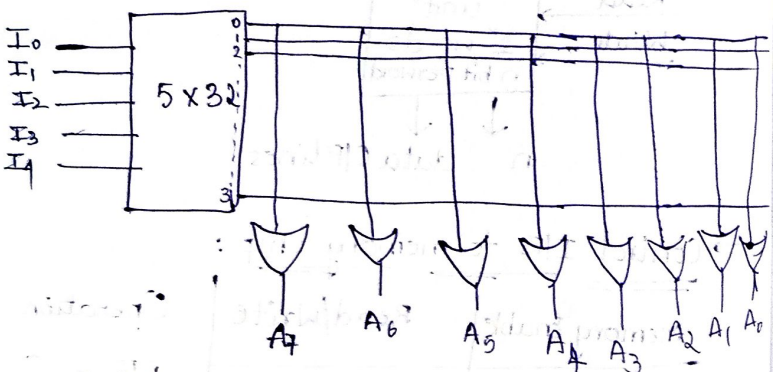
* Binary Storage Cell:



* Block diagram of ROM: Fixed AND Programmable OR.

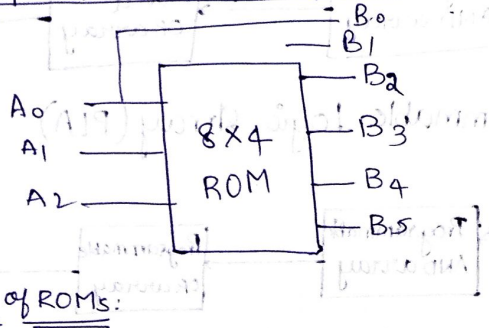


→ 32 × 8 ROM



* 3 bit number to Square of I/P no. *

A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	Decimal
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49



* Types of ROMs:

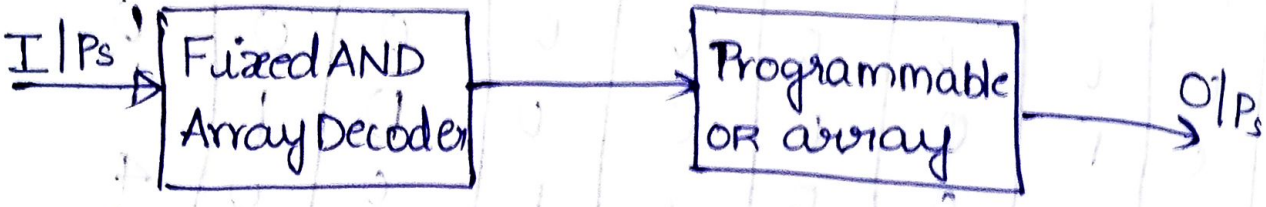
PROM → Programmable ROM

EPROM → Erasable Programmable ROM

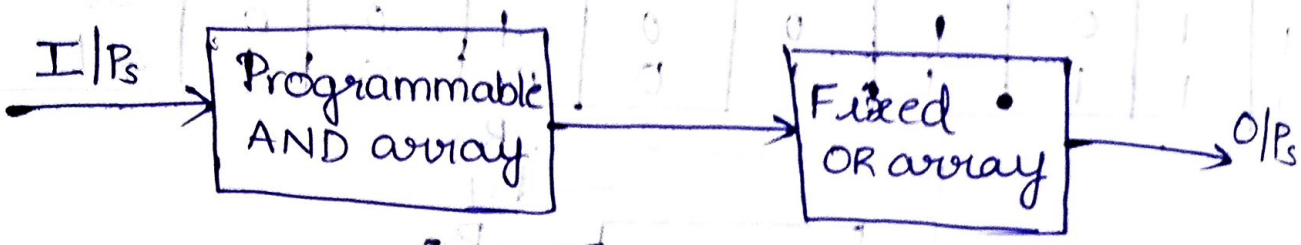
EEPROM → Electrical Erasable Programmable ROM

Flash Memory

* Programmable Read-only Memory [PROM]



* Programmable Array Logic (PAL)



* Programmable Logic Array (PLA)