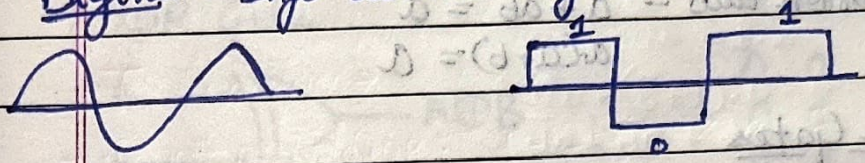
Digital Electronics

Date / /

Electronics - Electrical Engineering discipline where we work on low voltages to design electronic circuits.

Analog - Denote info in continuous value.

Digital - Info denoted by discrete value.



Advantage of Digital System:-

→ Easy to design
→ High accuracy & precision
→ Cost Effective

Boolean Algebra

Present day electronic (digital) system use just 2 discrete value & are therefore said to be binary.

Boolean Algebra was introduced by George Boole in 1847. His work laid foundation for algebra of logic which is fundamental to design of digital circuits.

Boolean Algebra Laws -

Idempotent Law - $a \cdot a = a$, $a + a = a$

Associative Law - $a \cdot (b \cdot c) = (a \cdot b) \cdot c$, $a + (b + c) = (a + b) + c$

Commutative Law - $a \cdot b = b \cdot a$, $a + b = b + a$

Distributive Law - $a \cdot (b + c) = a \cdot b + a \cdot c$

$a + (b \cdot c) = (a + b) \cdot (a + c)$

Date / /

De-Morgan's Law - $\overline{(a+b)} = \bar{a}\cdot\bar{b}$, $\overline{(a\cdot b)} = \bar{a}+\bar{b}$

Identity Law - $a+0=a$, $a\cdot 0=0$
$a+1=1$, $a\cdot 1=a$

Complementation Law - $0'=1$, $1'=0$
$a\cdot a'=0$, $a+a'=1$

Involution Law - $(a')'=a$

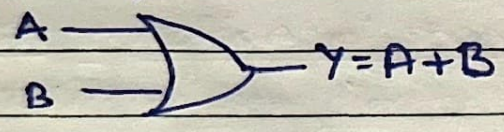
Absorption Law - $a+ab=a$
$a(a+b)=a$

Logic Gates

→ NOT Gate

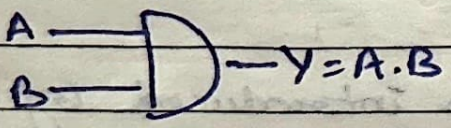
X	X'
0	1
1	0

→ OR Gate



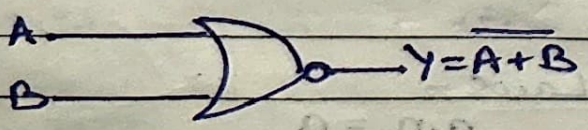
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

→ AND Gate



A	B	A.B
0	0	0
0	1	0
1	0	0
1	1	1

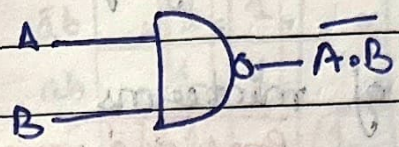
→ NOR Gate



A	B	$\overline{(A+B)}$
0	0	1
0	1	0
1	0	0
1	1	0

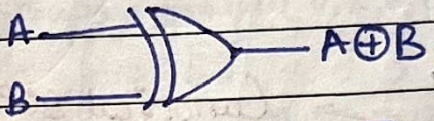
Date / /

→ NAND Gate



A	B	$\overline{A \cdot B}$
0	0	1
0	1	1
1	0	1
1	1	0

→ XOR Gate

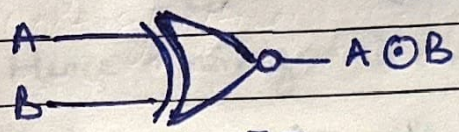


$A \oplus B = \bar{A}B + A\bar{B}$

A	B	A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

Odd 1's then 1

→ XNOR Gate



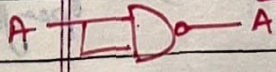
$A \odot B = \bar{A}\bar{B} + AB$

A	B	A⊙B
0	0	1
0	1	0
1	0	0
1	1	1

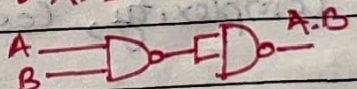
Odd 1's then 1

Note- NAND & NOR are universal gates (can get any gate from them)

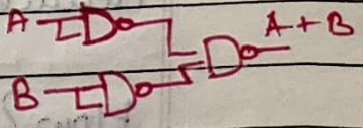
• Complement using Nand



• AND



• OR



Boolean Expressions

Mathematical expression that represent logical relationships using binary variables (0, 1) and logical operators (AND, OR, NOT)

Approaches of writing these expressions -

i. SOP - Sum of Product Ex - $a \cdot b + c \cdot d$

ii. POS - Product of Sum Ex - $(a+b) \cdot (c+d)$

Date / /

Canonical Forms → Standard way of writing Boolean expressions.

→ Canonical SOP : Sum of minterms

Product of variables where each variable occurs atleast once in complemented or uncomplemented form.

Ex →

$F(a,b) = a + a\bar{b}$

$= a.1 + a\bar{b}$

$= a.(b+\bar{b}) + a\bar{b}$

$= ab + a\bar{b} + a\bar{b}$

$= ab + a\bar{b}$ → Canonical SOP

→ Canonical POS: Product of maxterms

Ex →

$F(a,b) = (a+\bar{b}).(a+b)$

Simplification of Boolean Expressions

Reducing complexity of boolean expression while maintaining their functionality.

Why?

Reduced Complexity, Cost Efficient, Improved performance

How?

→ Using Boolean Laws

Ex: $F(a,b) = a.b + a.\bar{b}$

$= a.(b+\bar{b})$

$= a.1$

$= a$

→ Using Karnaugh-Maps (K-maps)

Visual method of simplifying boolean expression.

Date / /

ab \ cd	$\bar{c}\bar{d}$	$\bar{c}d$	cd	$c\bar{d}$
$\bar{a}\bar{b}$	1 (0)	1 (1)	(3)	1 (2)
$\bar{a}b$	(4)	1 (5)	1 (7)	(6)
ab	(12)	1 (13)	1 (15)	(14)
$a\bar{b}$	1 (8)	1 (9)	(11)	1 (10)

$F = \Sigma m(0,1,2,5,7,8,9,10,13,15)$

3 groups → Group1 (0,2,8,10)

$= \bar{b}\bar{d}$

→ Group 2 (5,7,13,15)

$= bd$

⇒ Group 3 (1,5,13,9)

$= \bar{c}d$

Hence minimized ⇒ $\bar{b}\bar{d} + bd + \bar{c}d$

Rules of grouping →

- Group 1's (SOP)
- Can overlap
- Can only be horizontal or vertical
- Should be as large as possible
- Corner grouping allo
- Should have as few grou as possible
- Element that changes in group should be removed.

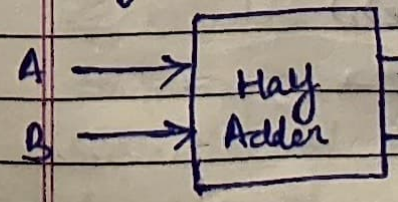
Combinational Circuits

Type of digital logic circuits wherein output is determined solely on inputs & have no memory element.

Common Combinational Circuits :-

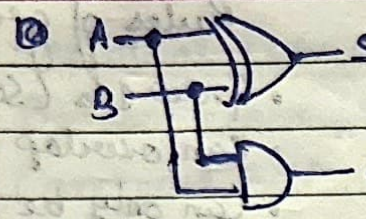
Adder → Performs addition of numbers. Used in ALUs & lots of other places.

Half Adder ● :- Adds 2 single bit binary numbers

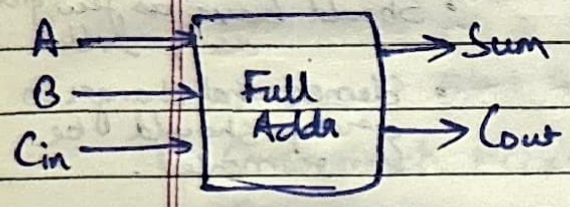


A	B	Sum	Carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

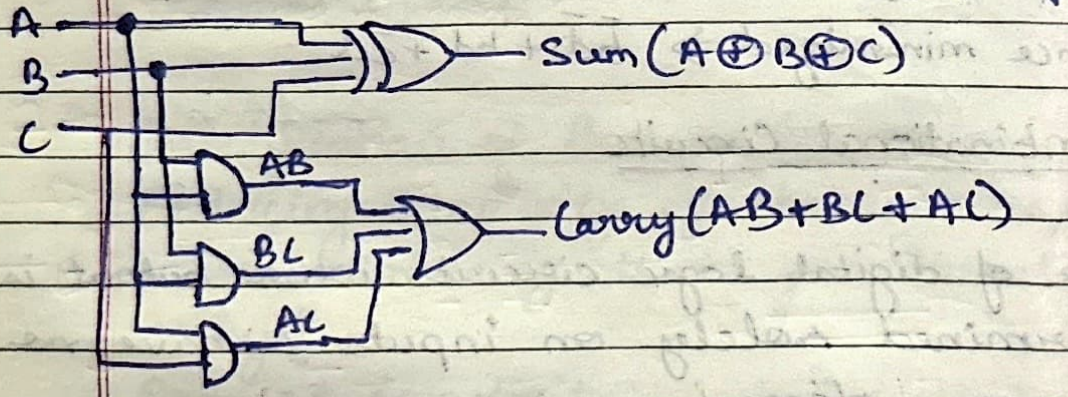
Date : / /



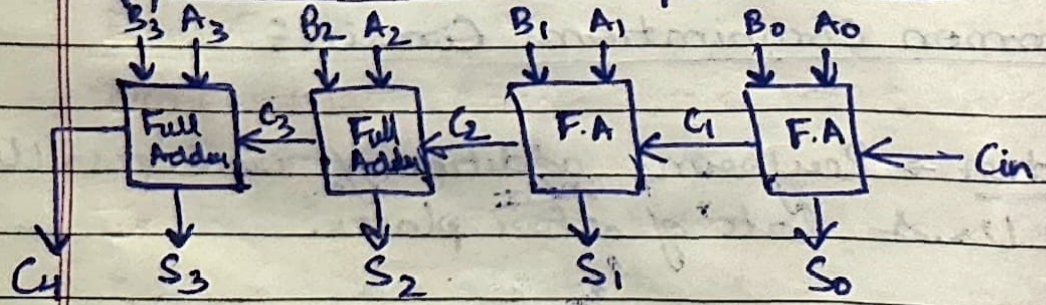
Full Adder → Includes additional input to handle carry from previous stage. Hence works on 3 bits.



A	B	C	Cout	Sum
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



Ripple Adder / Four-bit parallel adder

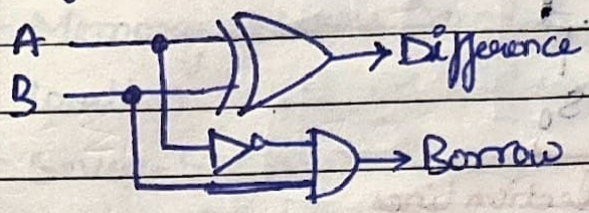


Can add 2 no.s (4 bit each)

Date / /

Subtractor - Used to subtract 2 numbers

Half Subtractor →



A	B	Diff	Borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0

2's complement → Method for representing signed integers in binary form. It is used because it simplifies design of circuits.

How?

Step 1 → 1's complement (Invert all bits)

Step 2 → Add 1

5 → 0101

→ 1010

→ +1

1011

This is -5

To verify →

- MSB indicates sign (1 = negative, 0 = positive)

$-(1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0)$

$= -8 + 0 + 2 + 1$

$= -5$

This way we can use adders to do subtraction too.

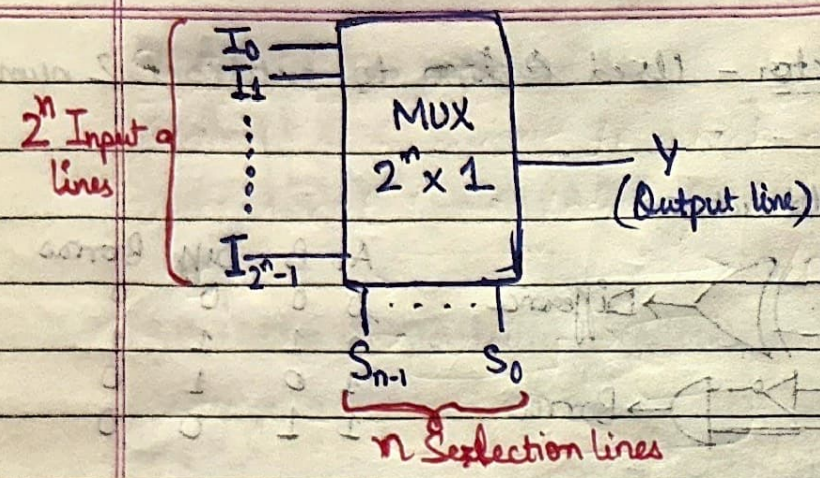
$3 - 1 \Rightarrow 3 + (-1)$

Multiplexer (MUX)

Device that combines multiple input signals & selects info from one of them & directs to single output line.

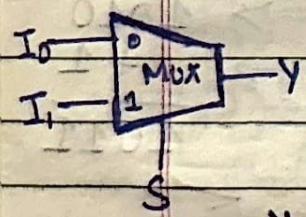
Selection of particular input line is controlled by set of selection lines.

Date / /



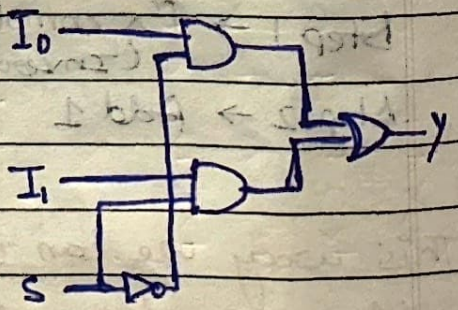
Application →
- Parallel data to serial data conversion
- Communication system
- Data Routing
- Computer memory

2 to 1 MUX



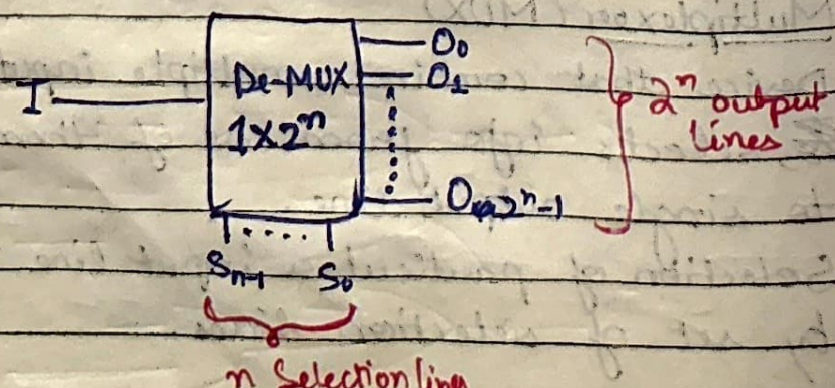
S	Y
0	I_0
1	I_1

$Y = \bar{S}_0 I_0 + S_0 I_1$



Demultiplexer (DeMUX)

Device that takes single input line & routes it to one of several digital output lines.
It is also called data distributor



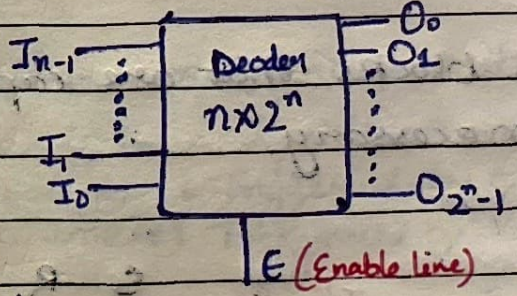
Date / /

Decoder

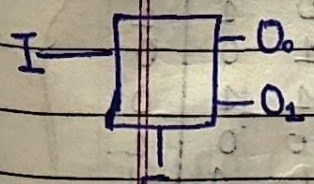
Decodes binary info from n input lins to 2^n o/p

Applications :-

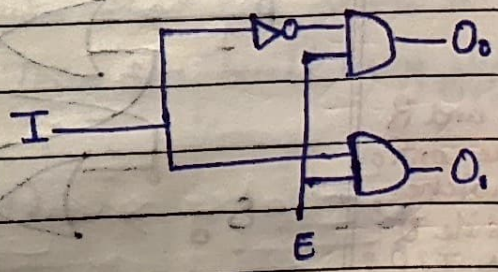
- Memory Address Decoding : Select specific memory location based on address provided.
- Seven-Segment displays
- Instruction decoding



1 to 2 Decoder

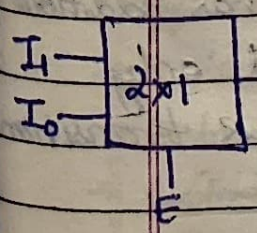


I	O
0	O_0
1	O_1

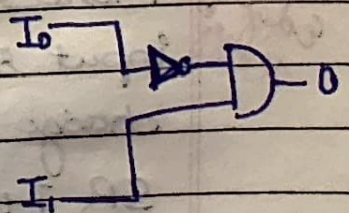


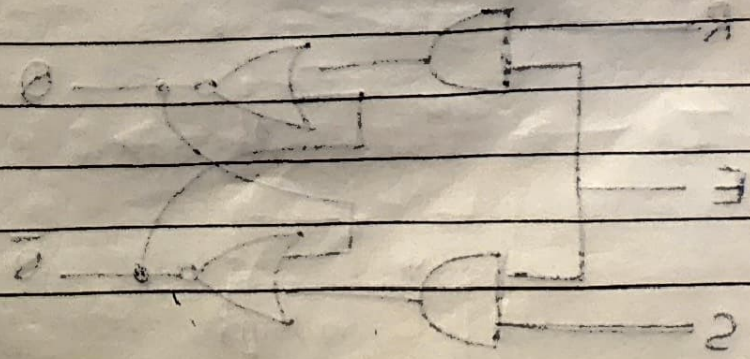
Encoder

Encodes binary info from one of 2^n input lines & encode it to n output lines.



I_1	I_0	O
0	1	0
1	0	1

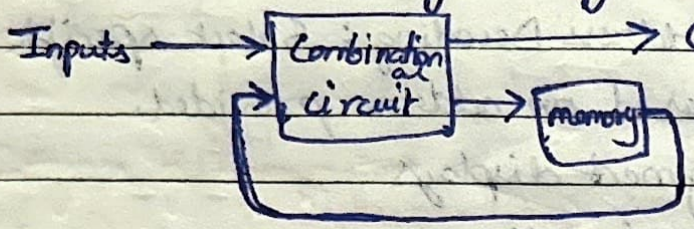




Date / /

Sequential Circuits

Combinational circuit to which memory elements are connected to form feedback path.

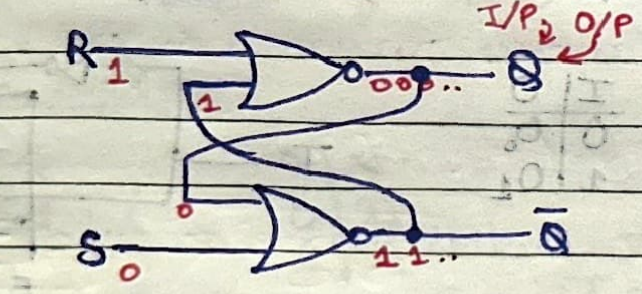


Latches

level triggered

Basic building blocks that are capable of holding 1 bit until necessary.

NOR latch →



S and R are control switches, while Q is I/P & O/P.

$\bar{Q}$ exist to check race condition

S	R	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	X
1	1	1	X

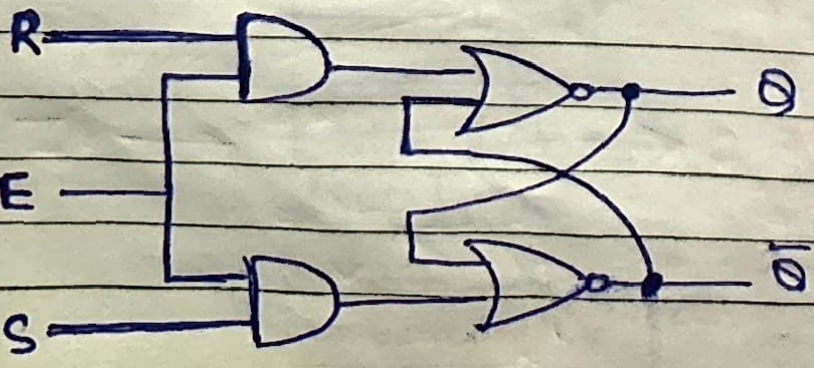
When S & R are 0, it is working purely as memory element, jo bhi value Q_{n+1} pr thi vhi Q_n pr thegi

Flip Flops

Edge triggered

More refined version of latch, it has additional input signal ~~that determines~~ and only change state at moment of clock transition

SR Flip Flop →



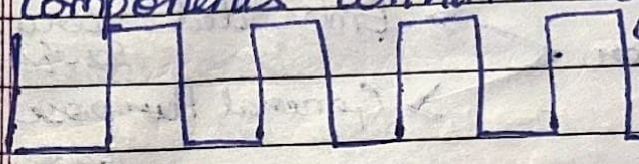
When E=0 then will behave as storage unit

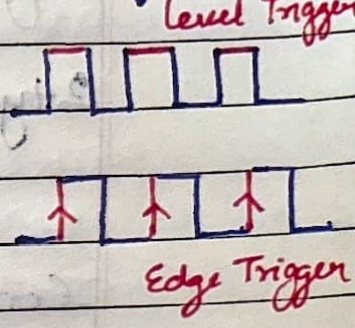
Date / /

Other types of Flip Flops - D, JK, T, etc

Clocks

Signal used to synchronize various ~~operations~~ components within a system.





Counters → Sequential circuits that can count in a specific sequence.
Async counter: Triggered by o/p of previous stage
Sync. counter: Triggered by common clock

Registers → Sequential circuits that can store a fixed no. of bits of data. They are built using flip-flops

Types:-
- → PIPO (Parallel-in, Parallel-out)
- → SIPO
- → SISO
- → PISO

Applications
- → Data Storage
- → Data Transfer
- → Control logic.