Page No.
Date: / /

— Imp Qs —

Digital gates - are basically electronic components which are used for switching & manipulating binary data

Universal gate - Those gates from which we can make any gate by using them.
NAND & NOR Gates are universal.

Truth Table - Table from which we can get o/p of different gates

Diff b/w Ex-OR & Ex-Nor Gate :—
The basic difference between these too gates is the Ex-nor gate gives o/p when both the i/p is different & Ex-Nor gate gives o/p when both the input are same.

Page No.
Date : / /

What is D Morgans theorem-

D'Morgans theorem is theorems through which we can easily manipulate & reduce the given equation

1) $(A+B)' = A'B' = (\overline{A+B}) = \overline{A} \cdot \overline{B}$

2) $A'B' = (A+B)'$ $\quad \overline{A \cdot B} = \overline{A} + \overline{B}$

e.g. $(A+B+C)'$

→ Augustus De Morgan invented it used for solving boolean expressions.

* Half Adder is used for adding 2 bit data

Inputs = Two

Output = Sum, Carry

Sum = A (+) B

carry = AB

AND Gate required = 1

Types of gates required = 2 - NAND, Ex-OR

Diff b/w Half & FULL ADDER

In half Adder only 2 bits can be used but in Full Adder we can use 3 bit data

Half substractor = 2 bit data

Inputs in full adder - 3.

Outputs = Sum & Carry

Sum = A⊕B⊕C

Carry = AB + BC + AC

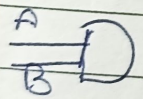
2 Half Adders are required to make a full adder.

Types of gates required = AND
EX-OR

Counter → Type of Register OR to determine sequence

Register - Group of flip-flops used to store data

AND Gate [only if all I/P are True)

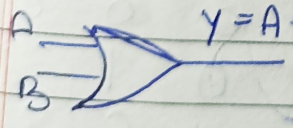


Y = AB

(multiplication)

A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

OR (sum) [True if one I/P is True]

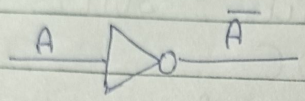


Y = A+B

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

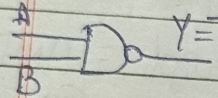
Page No.
Date : / /

NOT gate (complement) opposite



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

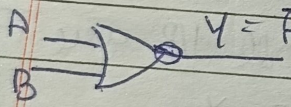
NAND



— = compliment

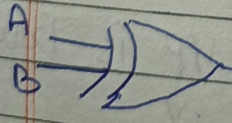
A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

NOR



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

XOR Gate ($A \oplus B$)



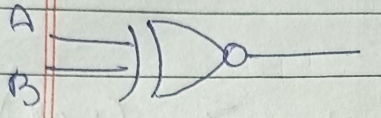
inputs are same = 0

I/p diff = 1

A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Page No.
Date : / /

XNOR



A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

— (Boolean Expressions) —

$A + A = A$ | $A \cdot A = A$ (idempotent law)

$A + \ldots + 1 = 1$

$A + \bar{A} = 1$; $A \cdot 0 = 0$ (identity law)

$A \cdot \bar{A} = 0$.

DeMORGAN'S THEOREM

$\overline{A+B} = \bar{A} \cdot \bar{B}$

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

Commutative Law

$A + B = B + A$

$A \cdot B = B \cdot A$

$A \oplus B = B \oplus A$

$\overline{A \cdot B} = \overline{B \cdot A}$

Assosiative Law

$A + (B + C) = (A + B) + C$

$A \oplus (B \oplus C) = (A \oplus B) \oplus C$

$A (B \cdot C) = (A \cdot B) \cdot C$

$A \odot (B \odot C) = (A \odot B) \odot C$

(K-MAP)
flip flops

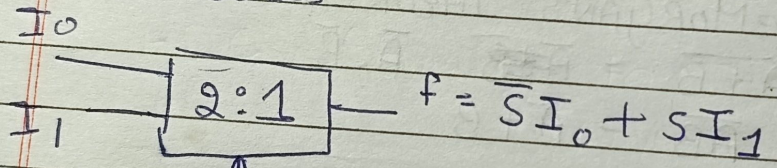
Page No.
Date: / /

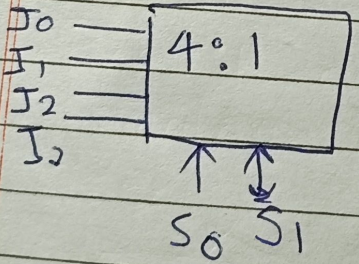
Dual form
+ → ·
1 → 0

Sum of Product Σ (SOP)
F = ABC + ĀB̄C̄ + ... (min terms = 1)

Product of Sum (POS) Π
F = (A+B+C) . (B+A+C̄)
max terms = 0

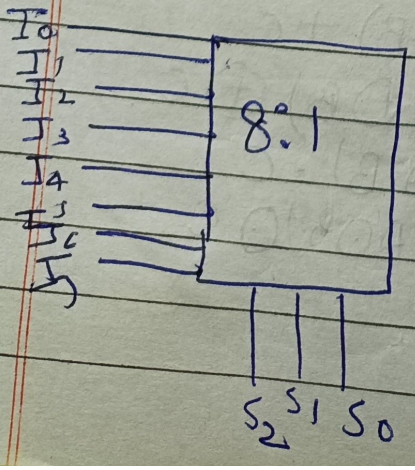
Mux (Multiplexer)
combines





4 inputs 1 o/p
2 select lines

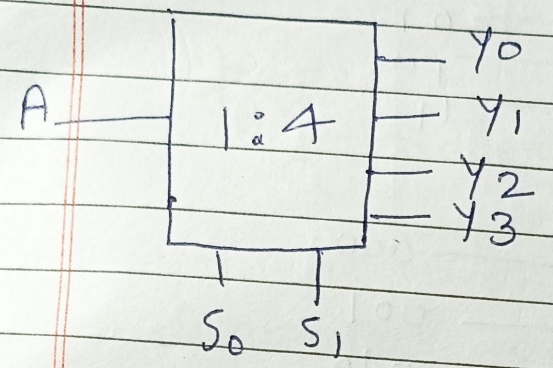
n = Select lines
2^n = inputs
1 = output



n = 3
2^3 = 8 i/p
1 = o/p

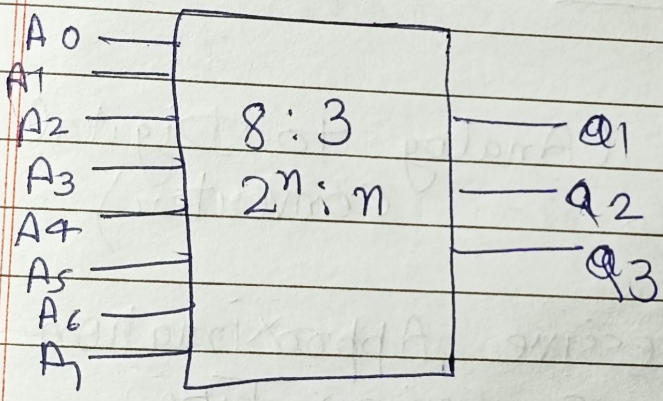
Page No.
Date : / /

DeMux



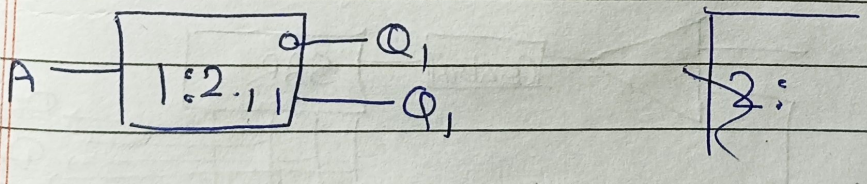
1 = input
2^n = o/p
n = Select lines

Encoder (used for producing codes)



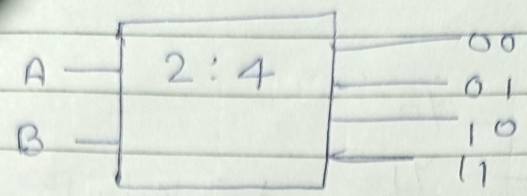
2^n = i/p
n = o/p
n_0 = select lines

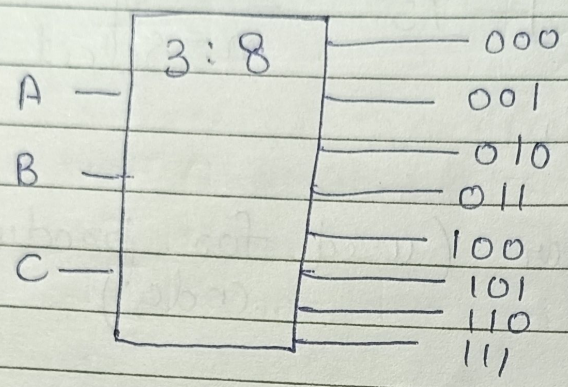
Decoder (Actual VALUE)



n i/p = 2^n = o/p

3 doesn't have select lines



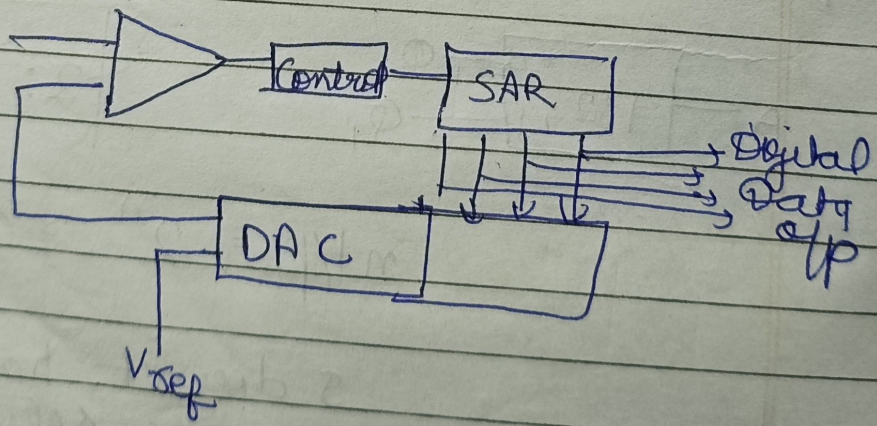


ADC (Analog to Digital Converter)

Successive Approximation ADC

→ Better Conversion Type

→ Doesn't depend on input Analog voltage.



Page No.
Date : / /

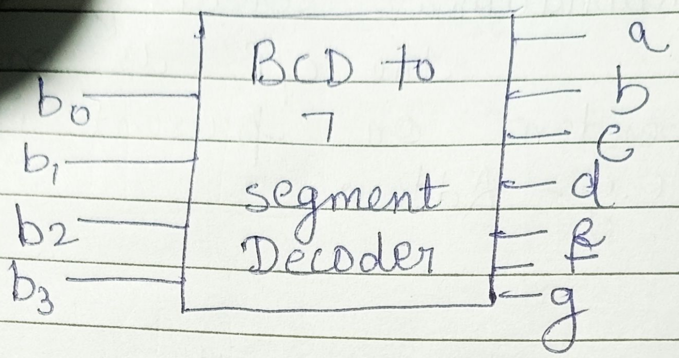
Combinational circuit)
the o/p is only dependent on present i/p
e.g - Adders

sequential circuit)
o/p depends on the present i/p as well as previous outputs.
e.g. counter.

BCD To 7-segment decoder

→ A digital or binary decoder is a digital combinational logic circuit which can convert one form of digital code into another form

Seven segment displays are widely used in digital clocks, basic calculators etc



for$_a$ 0 — for 1

f| — | b → Not needed — | b

e| — | c — | c

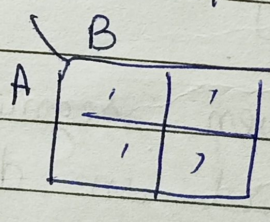
d

K-MAP

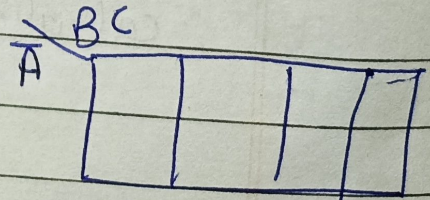
used to simplify the boolean expression in pictorial format without using any boolean theorem or exp.

formula = $2^n = n$ (no. of variables)

$2^2 = 4$



$2^3 = 8$



Page No.
Date : / /

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

A \ BC	00	01	11	10
0	0	1	3	2
1	4	5	7	6

A \ B	0	1
0	0	1
1	3	2

15 - 1111

OR

AB \ CD	00	01	11	10
00				
01				
11			15	
10				

Rules for Creating
No zeroes allowed.
Overlapping allowed

Page No.
Date : / /

$F(A,B) = \Sigma(0,2,3)$

A \ B	0	1
0	1	
1	1	1

$F(A,B) = A + B$

$F(A,B,C) = \Sigma(0,1,2,3,5)$

$2^n = n$

$2^3 = 8$

A \ BC	00	01	11	10
0	1	1	1	1
1		1		

$A' = 0$

$A = 1$

g_1

A	B	C
0	0	0
0	0	1
0	1	1
0	1	0

g_2

A	B	C
0	0	1
1	0	1

$A' + B'C$

$F = A' + B'C$

Page No.
Date: / /

$F(A\ B\ C\ D) = \Sigma(0, 1, 4, 5, 8, 9, 12, 13)$

$2^n = \#$

$2^4 = 16$

AB \ CD	00	01	11	10
00	1	1		
01	1	1		
11	1	1		
10	1	1		

$A' = 0$

$A = 1$

0 1 3 2

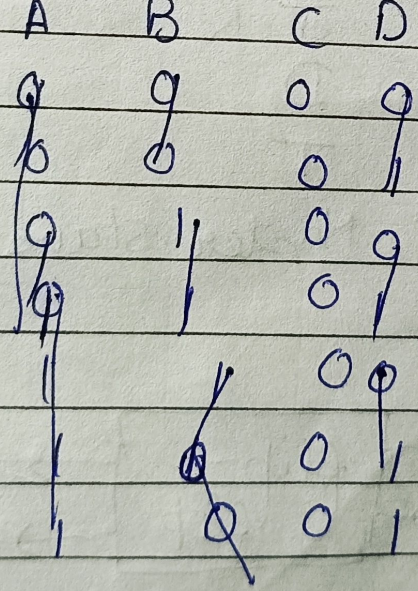
4 5 7 6

12 13 15 14

8 9 11 10

g1

A B C D



Page No.
Date : / /

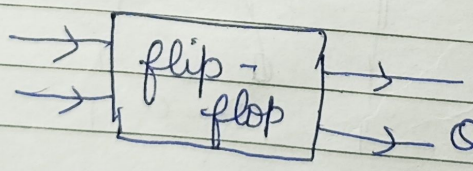
Flip-flop is a sequential circuit that is used to store 1 bit binary data as well as construct a register is called flip-flop

Two states
Set = (1) current
Clear = (0) no current

flip flop can be made by
- NOR - (RS)
- NAND - (SR)

J K
D
T

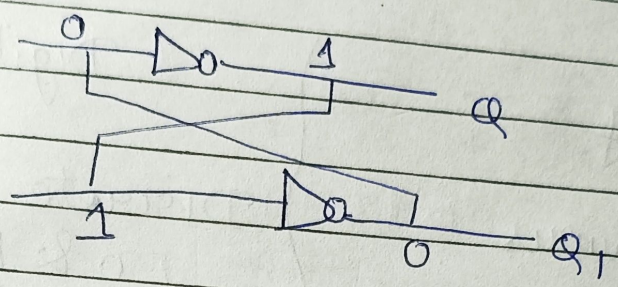
Master slave flip flop



Latches
basic building block of the sequential circuit that is used to construct flip flop.

It stores 1 bit binary data at a time

& changes its output immediately based on the new input



using not gate

ASCII
(American standard code for information interchange

Boolean Laws.

$x_1 \rightarrow x$

$x_0 \rightarrow 0$

$x\bar{x} \rightarrow 0$

$x\,x \rightarrow x$

$x+1 = 1$

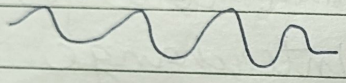
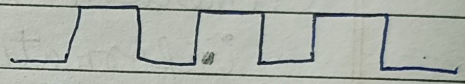
$x+0 = x$

$x+\bar{x} = 1$

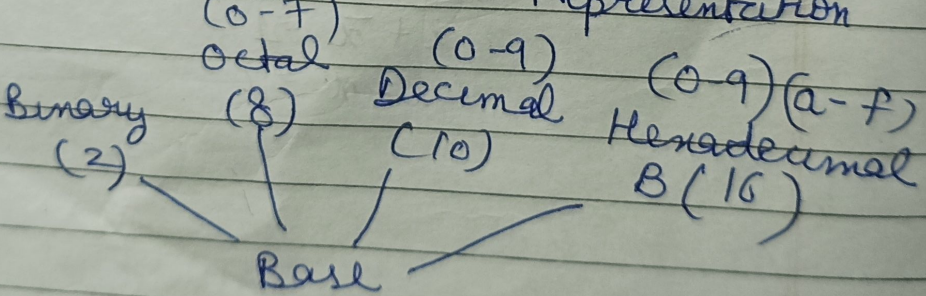
$x+x = x$

Page No.
Date : / /

Digital Electronics – Branch & Digital signals

Analog	Digital
→ continuous nature	Discrete nature [0 & 1]
→ Sine wave 	Square wave 
Analog circuits (resistors, inductors)	Digital (Transistors, logic gates, ICs)
	Sound recording

Number System – count : 02
↳ Representation



(Most) MSB
$_{b}0\ 1\ 1\ 0_{e}$ LSB (least significant bit)

Page No.
Date: / /

+2 4 2 1 0
0 0 0 ,
1 0 0 4

Hexadecimal
0, 1, 2, 3, 4, 5, 6, 7, 8, 9,
A, b, C, D, e, f
10 11 12 13 14 15.

Binary Arithmetic
+, −, ⊙, ÷, X.

0 + 0 = 0
0 + 1 = 1
1 + 0 = 1
1 + 1 = 0. [carry = 1]

```
  1
+ 1
---
  0
```

Subtraction
0 − 0 = 0
0 − 1 = 1 (borrow = 1)
1 − 0 = 1
1 − 1 = 0.

Multiply
0 × 0 = 0
0 × 1 = 0 , 1 × 0 = 0
1 × 1 = 1

Division
0 ÷ 0 = Not Defined
1 ÷ 1 = 1
0 ÷ 1 = 0

Page No.
Date : / /

1's complement (Reverse)

1 0 1 0 1 0 1

0 1 0 1 0 1 0

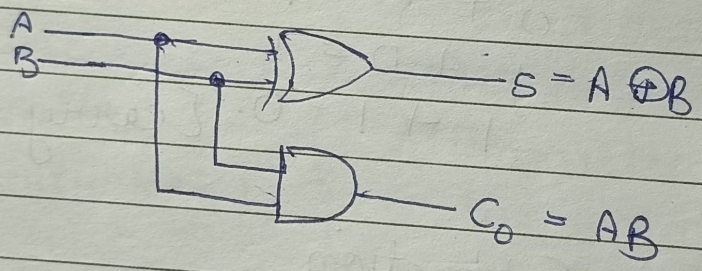
2's complement - 1 + 1's comp.

0 1 0 1 0 1 0

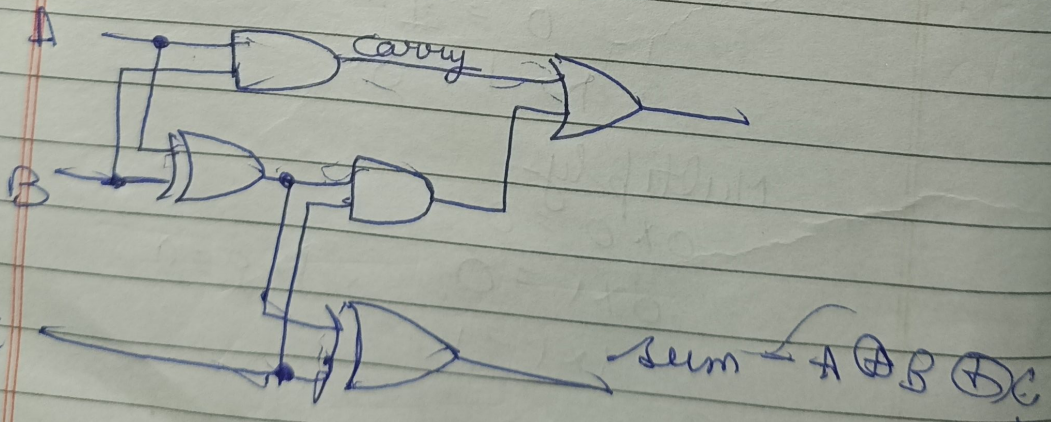
+ 1

0 1 0 1 0 1 1

Half Adder



Full Adder.



complementation law.

$\overline{T} = F$ $\quad \overline{0} = 1$

$\overline{F} = T$ $\quad \overline{1} = 0$

$x \cdot \overline{x} = 0$ $\quad x + \overline{x} = 1$

Logic gates → 1 or more I/p
but
only 1 o/p

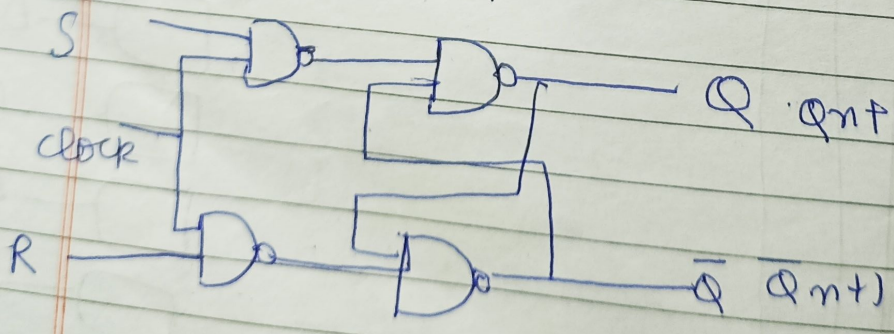
Combinational circuits → o/p depends on current i/p

→ Half Adder / Full Adder
Multiplexer, Decoder

Sequential Circuit — depends on ~~present~~ ~~each~~ i/p & previous o/p

flip-flops
counters

SR flip-flop



K-MAP

Page No.
Date: / /

DeMorgan's law

$\overline{A \cdot B} = \bar{A} + \bar{B}$

$\overline{A + B} = \bar{A} \cdot \bar{B}$

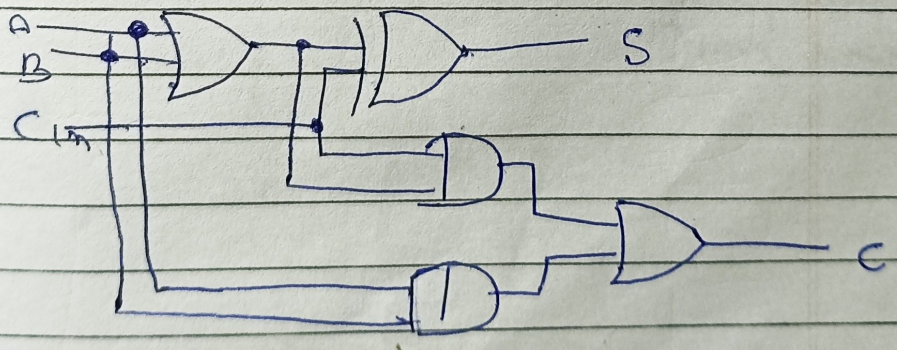
Gray Code - Unit distance Code
(One bit at a time)

ASCII - American Standard Code for information Interchange

1's compliment
↳ Reverse the digit
101 = 010

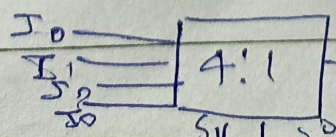
2's compliment = 1's comp + 1
010 + 1
011

Full Adder.
↳ Two XOR Gates, two And & One OR



$S = A \oplus B \oplus C$

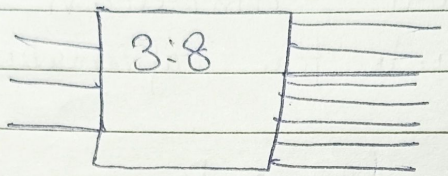
$C = AB + BC + AC$



(data Selector

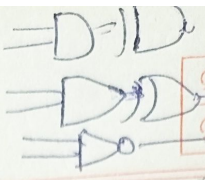
Page No.
Date: / /

Race condition of JK flip flop
J=1, K=1, the pulse duration is too long, the output toggles multiple times uncontrollably



3 i/p

SISO - Serial in
Serial Out
PIPO - Parallel IN
Parallel Out



Page No.
Date: D /

Excess -3 Code →

Gray code :- In one bit ~~stores~~

Parity - ~~extra bit~~ Used in error correction

weighted & Non-weighted code - Bits doesn't have positional value

Typical X3

TTT, CMOS, ECL, MOS.

RTL, DTL, TTL

Complement → MOSFET

Resistor Transistor Logic

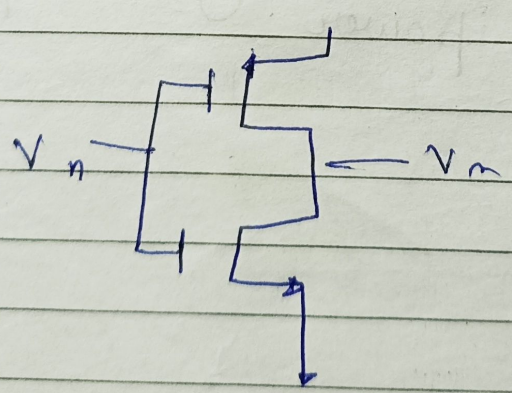
Diode Transistor logics

Transistor

Transistor logic

& integrated circuits

CMOS



(12 Gates)
SSI → Small Scale Integration
(12-100) MSI → Medium Scale
(100-1000) LSI → Large scale
10000+ VLSI → Very large Scale

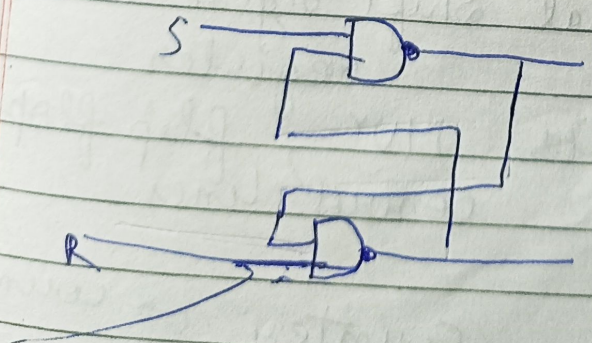
Propogation delay.

fan-Out - The max number of digital inputs that the o/p of a single logic gate can feed

D Type flip flop

D=0 $Q_n = 0$
D = 1 $Q_{next} = 1$
Captures the value of the D-input at a specific portion

S R Latch Logic



Semiconductor Memories — small in size, occupies less space

RAM / ROM — EPROM, EEPROM

4-Bit Ripple counter
An asynchronous counter where the o/p of one flip flop acts as the clock for next

Convertors
- Analog To digital
- Digital To analog

Binary Resistor Network
R-2R Ladder

Shift register - Shifts data
Left Shift
Right Shift
SISO
PIPO

Universal Shift register - bidirectional register
uses MUXs, flip flops
control lines

Counter → counts inputs
MODE → States

Page No.
Date : / /

Asynchronous

Synchronous (Parallel) clocked by same signal making them faster

Ring counter - A shift register where the o/p of last flip flop is fed back to the first

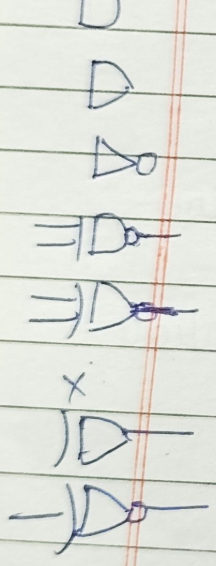
2^n upward & downward

Latch - like swinging door

Synchronous - sequential

Page No.
Date : / /

Race around conditions
async Vs sync
Gates
4 bit adder
Mux
3 to 8 Decoder
D flip flop
Shift Registers
JK flip flop Counters
A/D converters
K-Map & logic gate formation
Sync decade counter
Encoder →
ROM



PROM → Programmable

flip flop has 2 stables states

weighted & non weighted code

Mux

$(67)_{10}$ =

$\bar{A}B\bar{C} + \bar{A}\bar{B}C + \bar{A}BC + \bar{A}\bar{B}\bar{C}$

SISO Shift Register

Register converters

Page No.
Date: / /

Static VS Dynamic Memory

PIPO shift Register

Shift Registers
used for arithmetic operations

Flip-flop → binary cell capable of storing one bit of data.
→ sequential circuit
→ has memory element

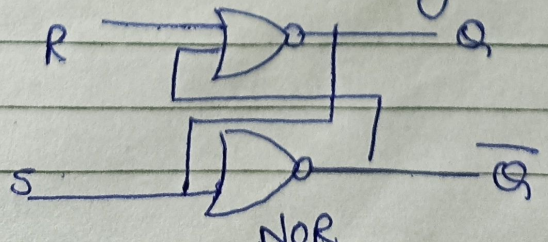
Two outputs - Normal Q
Complement $\bar{Q}$

ff

SR, Jk, Dff (Delay), T (Toggle) ff

SR Flip Flop

The set Reset flip flop is designed with the help of two NOR gates & also two NAND gates



Page No.
Date : / /

Truth Table

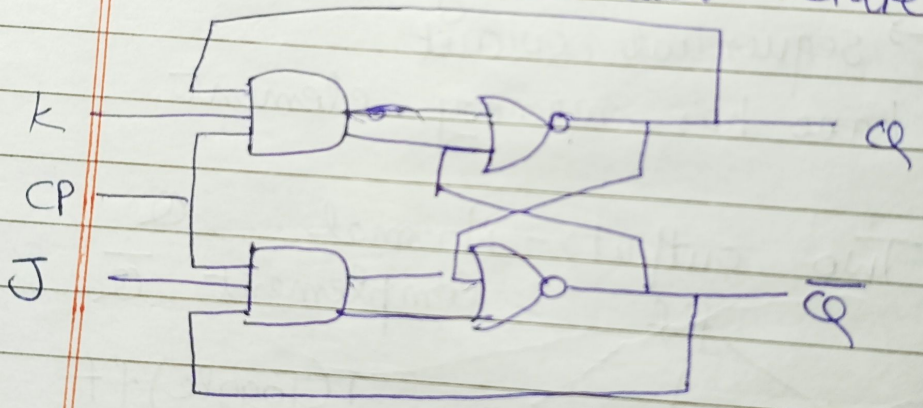
S	R	Q N+1
0	0	Q(N) previous o/p
0	1	0
1	0	1
1	1	Invalid

o/p depends

Characteristics eqs. : $Q(N+1) = S + \bar{R} Q(N)$

[J.K flip flop]

Rifinement of SR flip flop
To avoid invalid state



J	k	Q(N+1)	
0	0	Q(N)	Hold
0	1	0	clear to 0
1	0	1	Set to 1
1	1	$\bar{Q}(N)$	Toggle

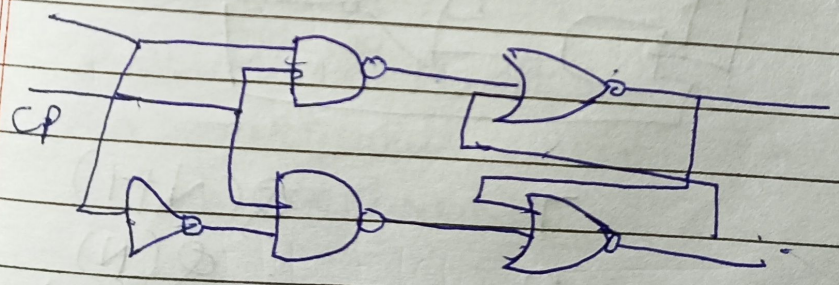
Page No.
Date : / /

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

eqn = $Q(N+1) = J\overline{Q}(N) + KQ(N)$

Delay flip flop (D flip flop)
Transparent latch because in this flip input & output are equal.
SR → D ff by applying inverter between S AND R



T.T

D	Q(N+1)
0	0
1	1

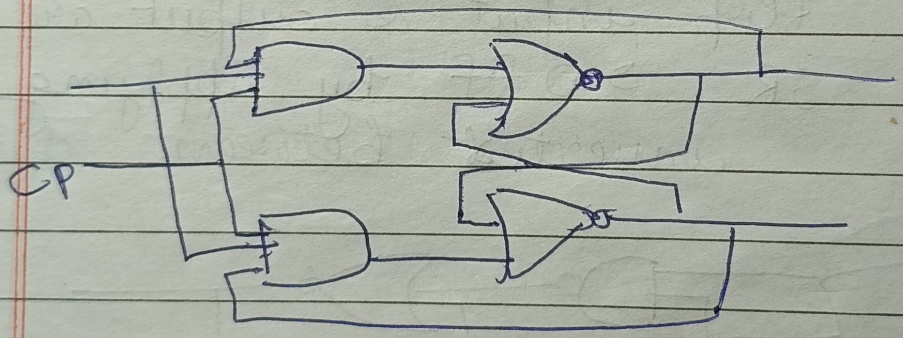
Characteristics

D	Q(N)	Q(N+1)
0	0	0
0	1	0
1	0	1
1	1	1

$Q(N+1) = D$

T (Toggle) —
obtained from JK.
In JK flip flop when inputs J & K connected to provide a single input by T

$J = K = T$



T	Q(N+1)
0	Q(N)
1	$\overline{Q}(N)$

Characteristics

	Q(N)	Q(N+1)
0	0	0
0	1	1
1	0	1
1	1	0

$Q(N+1) = T \oplus Q(N) + \overline{Q}\overline{N}$

Page No.
Date : / /

Flip-flop has a clock signal

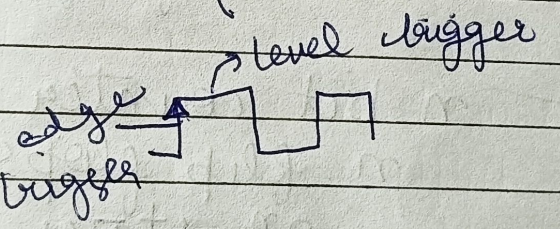
→ Checks the input but changes the output only at times defined by the clock

→ edge triggered device

Latch doesn't

electronic device, changes o/p immediately based on the applied signal

No classification level triggered



— Counter —

Device that stores no. of times a particular event or process has occured.

Page No.
Date : / /

constructed using no. of flip flops connected in cascade
(edge trigger)
provides clock pulse

Up counter ⇒ ascending order
0, 1, 2, 3.

Down - descenting
Up/down - both

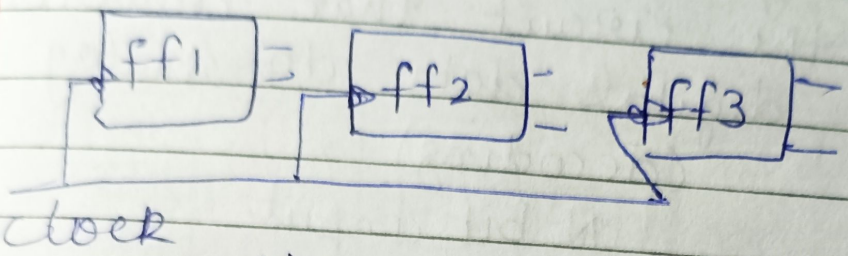
uses → Digital watches,
To create time delays
frequency divider.

— n bit counter ——
n flip flops
2^n states
divides input frequency by 2^n

Mod-4 counter
↳ States
No. of "

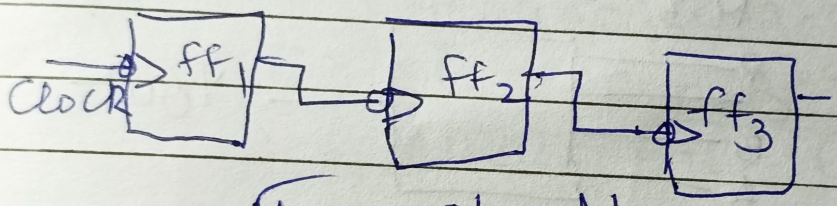
Page No.
Date : / /

Synchronous



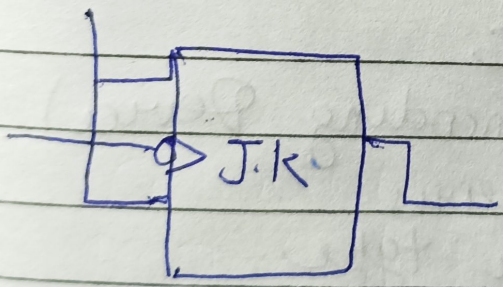
(~~low~~ High speed)

Asynch



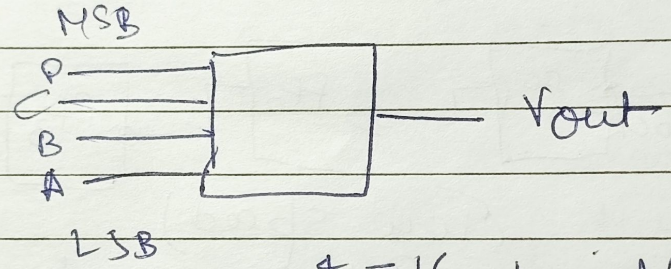
(low speed)

Ripple counters & simple



Page No.
Date : / /

Digital to Analog Converter
the circuit that converts digital data to Analog data (decoding)
N-bit input
Two method-
weighted D/A
R-2R ladder

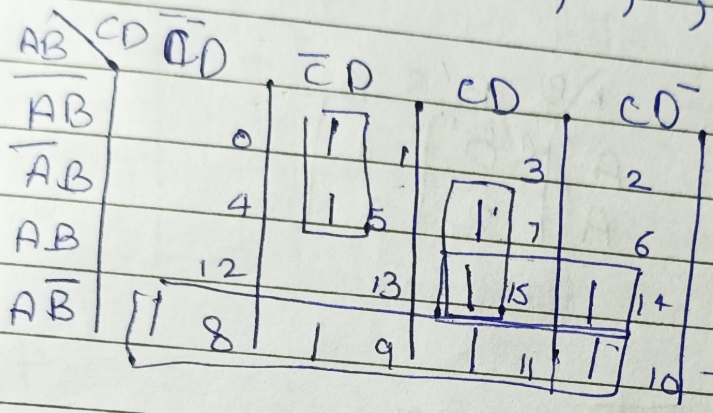


$2^4 = 16$ possible combos
2^n = states

ADC
(Incoding Device)
Simultaneous
conter type
Successive approximation
Dual slope

Page No.
Date : / /

$f(ABCD) = \Sigma m(1,5,7,8,9,10,11,14,15)$



$A\bar{B}$, $A\bar{C}\bar{D}$, $BC\bar{D}$ + AC

$F = A\bar{B} + AC + \bar{A}\bar{C}D + BCD$

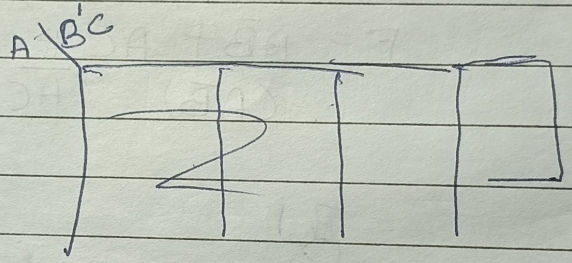
$\overline{(A\bar{B})} \cdot \overline{(AC)} \cdot \overline{(\bar{A}\bar{C}D)} \cdot \overline{(BCD)}$

Page No.
Date : / /

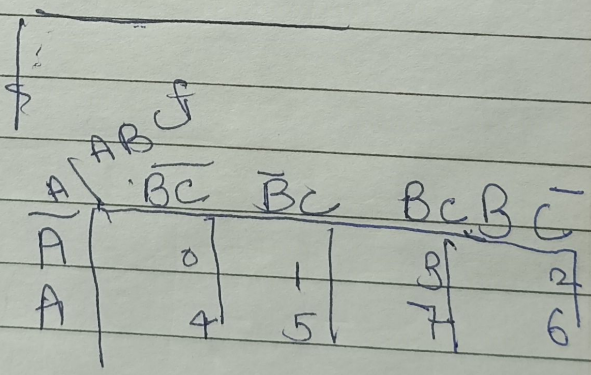
K-MAP

A \ B	B'	B
A'	$A'B'$ 0	1
A	2	3

Three var -



Three var

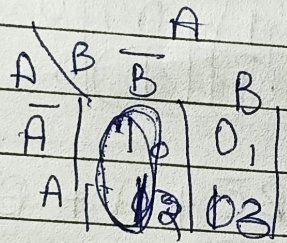


Page No.
Date : / /

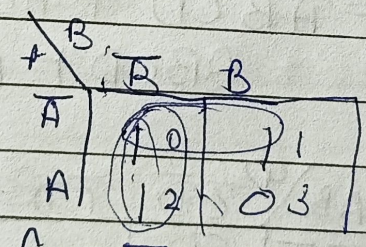
	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

$f = \Sigma(0,2)$

$f = A'B' + AB'$

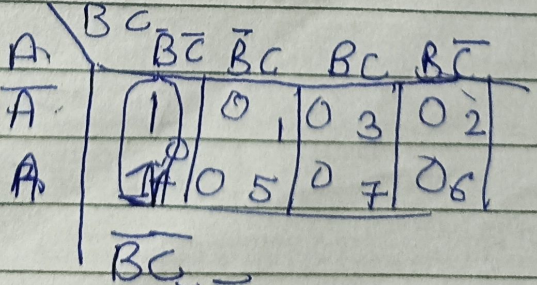


$f = \bar{B}$



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

$f = A'B'C' + AB'C'$



$f = \bar{B}\bar{C}$

$f(ABC) = \Sigma[0, 2, 4, 5, 6]$

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	1 0	0 1	1 3	0 2
A	1 4	1 5	0 7	1 6

A \ BC	$\bar{B}\bar{C}$	$\bar{B}C$	BC	$B\bar{C}$
$\bar{A}$	1 0	0 1	0 3	1 2
A	1 4	1 5	0 7	1 6

$\bar{B}\bar{C} + A + B\bar{C}$

$f = A'BC'D + ABC'D + A'BCD + ABCD + A'B'C'D'$

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1 0	0 1	0 3	0 2
$\bar{A}B$	0 4	1 5	1 7	0 6
AB	0 12	1 13	1 15	0 14
$A\bar{B}$	0 8	0 9	0 11	0 10

$f = A'B'C'D' + DB$

$\Rightarrow AB'C'D' + DB$

Page No.
Date : / /

$Y = \Sigma m [0, 1, 3, 5, 7, 9, 11, 13, 15)$

AB \ CD	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1 0	1	1	0 2
$\bar{A}B$	0 4	1 5	1 7	0 6
AB	0 12	1 13	1 15	0 14
$A\bar{B}$	0 8	1 9	1 11	0 10

$\bar{A}\bar{B}\bar{C} + D$

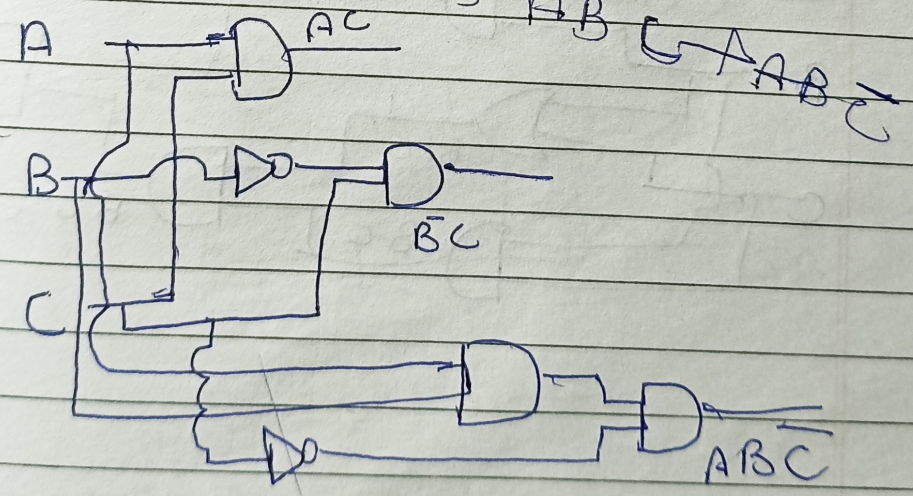
$Y = D + \bar{A}\bar{B}\bar{C}$

Double negation ⇒ $Y = \overline{\overline{D + A'B'C'}}$

demorgan's law ⇒ $\overline{\overline{D} \cdot \overline{ABC}}$

$Y = AC + \bar{B}C + AB\bar{C}$

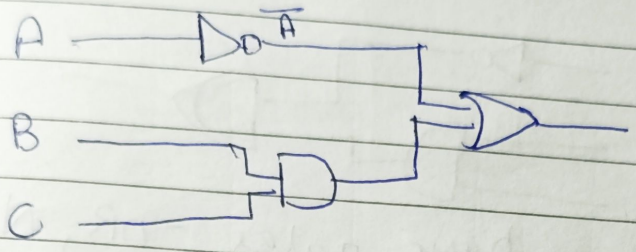
↳ AB ↳ ABC



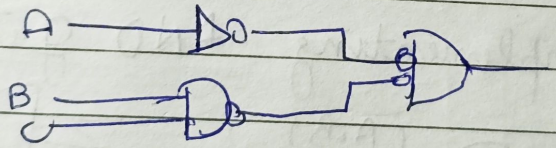
Page No.
Date: / /

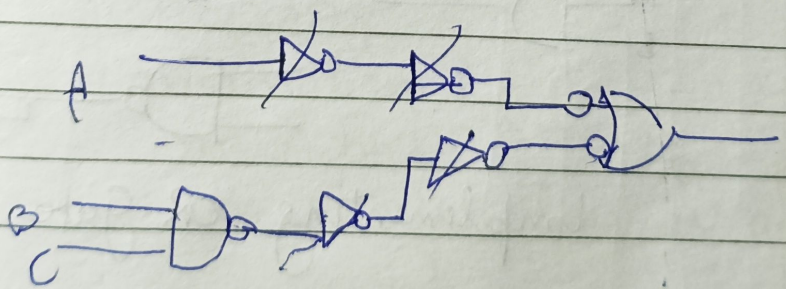
NAND Implementation

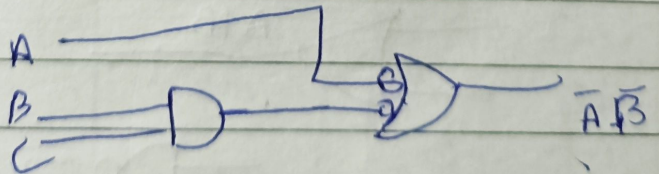
$f = \bar{A} + B.C$

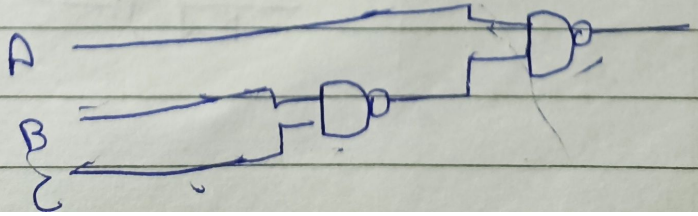


OR gates - input bubble
AND gate - o/p
Not gate at bubble
double inversion = cancel
place equivalent NAND gate.









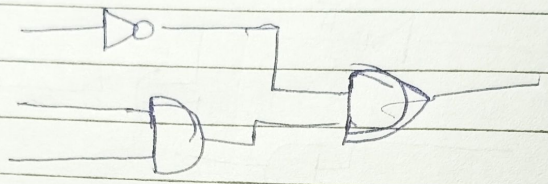
$\overline{A \cdot B} = \overline{A} + \overline{B}$

$\overline{A + B} = \overline{A} \cdot \overline{B}$

Page No.
Date : / /

Universal Gates — NAND
NOR

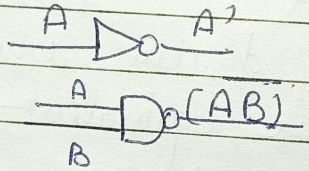
$f = \overline{A} + BC$

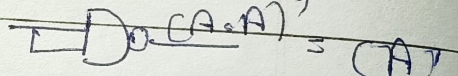


basic gates - OR, NOT, AND

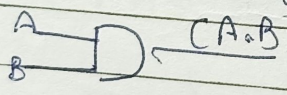
Steps

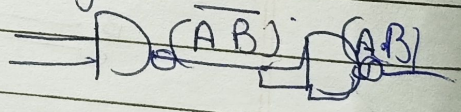
Implementing inverter

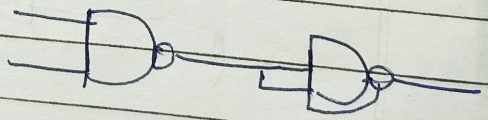




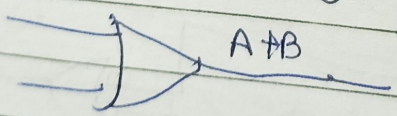
Implementing AND gate

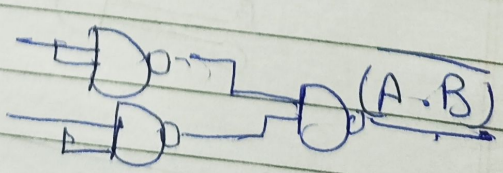






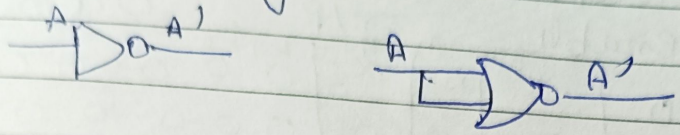
Implementing OR Gate



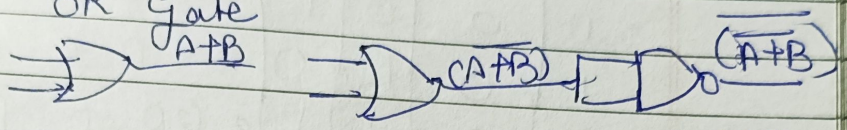


using NOR Gate

Not gate

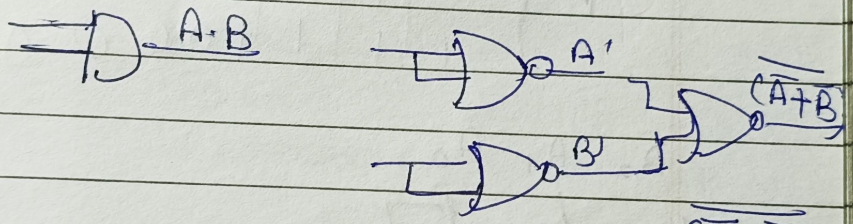


OR Gate



$= A+B$

Implementing AND Gate



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

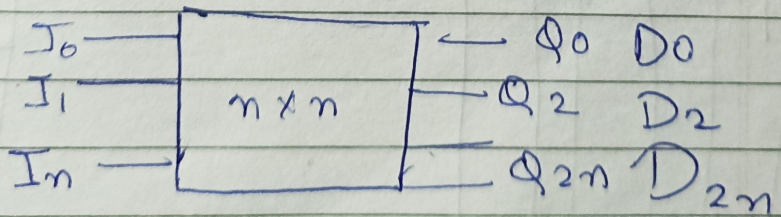
$= A \cdot B$

Decoder

Combinational

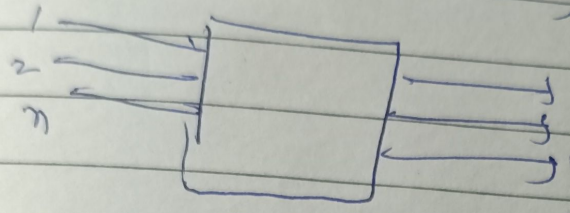
n input 2^n o/p

one o/p line is active



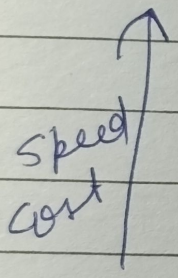
Page No.
Date : / /

Encoder (CLC)



One I/P is active

Memory Hierarchy



CPU Registers
|
Cache
main
secondary