DIGITAL ELECTRONICS

Topics

1) Number System
2) Boolean Algebra
3) Logic gates
4) K-Map
5) Digital Circuits — combinational, Sequential
6) Memories
7) Logic Families, ADC, DAC.

Number System.

- BIT : Binary digit (0, 1)
- NIBBLE : Group of 4-bits.

$(xy \cdot z)_r$ — r → base (or) radix, face. (xy: magnitude)

- Base explains the no. of possible combination in given number

$r = 10$

- Base 10 decimal system, it has 10 possible combination.
- $r = 2$ (Base 2) is binary number system, it has 2 possible combinati-on the range is from (0,1)
- $r = 8$ octal number system, it has 8 possible combination, the range is from 0 to 7
- $r = 16$ Hexa decimal system, it has 16 possible combination, the range is from 0 to F

0 1 2 3 4 5 6 7 8 9 A B C D E F
A↓10, B↓11, C↓12, D↓13, E↓14, F↓15

- $r = 12$ it has 12 possible combination, the range is from 0 to B 0, 1 2 3 4 5 6 7 8 9 A B.

Decimal to others : Conversion

$(xy.z)_{10} = (\quad)_r$

→ xy is called integer, 0.z is called fractional part.
→ Divide integer part with base r and fractional part multiply with base r.

$0.z \times r$.

r | xy

Ex: 1) $(29.75)_{10} = (\quad)_2$

2 | 29
2 | 14 - 1
2 | 7 - 0
2 | 3 - 1
2 | 1 - 1
0 - 1

0.75 x 2
1.50 x 2
1.00 x 2

$(29.75)_{10} \rightarrow (11101.11)_2$.

2) $(30.25)_{10} = (\quad)_8 = (\quad)_{16}$.

8 | 30
8 | 3 - 6
0 - 3

0.25 x 8
↓2.00

$(30.25)_{10} \rightarrow (36.2)_8$

16 | 30
16 | 1 → E
0 → 1

0.25 x 16
↓4.00

$(30.25)_{10} \rightarrow (1E.4)_{16}$

3) $(26)_{10} = (\quad)_4$.

4 | 26
4 | 6 - 2
4 | 1 - 2
0 - 1

$(26)_{10} \leftrightarrow (122)_4$.

4) $(22)_{10} = (\quad)_{12}$

12 | 22
12 | 1 → 10 (A)
0 - 1

$(22)_{10} \rightarrow (1A)_{12}$

Others to Decimal : Conversion

$(pq.st)_r = (\quad)_{10}$.

$r^1\ r^0\ r^{-1}\ r^{-2}$

- The given number is multiplied with base powers

Ex: $(11101.11)_2 = (\quad)_{10}$
$2^4\ 2^3\ 2^2\ 2^1\ 2^0\ 2^{-1}\ 2^{-2}$

$\Rightarrow 2^4 \times 1 + 2^3 \times 1 + 2^2 \times 1 + 2^1 \times 0 + 2^0 \times 1 + 2^{-1} \times 1 + 2^{-2} \times 1$

$= 16 + 8 + 4 + 0 + 1 + 0.5 + 0.25 = (29.75)_{10}$

$\frac{1}{2} = 0.5$, $\frac{1}{4} = 0.25$, $\frac{1}{8} = 0.125$

2) $(36.2)_8 = (\quad)_{10}$

$\Rightarrow 3 \times 8^1 + 6 \times 8^0 + 2 \times 8^{-1}$

$\Rightarrow 24 + 6 + 0.25$

$= (30.25)_{10}$.

3) $(1E.4)_{16} = (\quad)_{10}$

$1 \times 16^1 + E \times 16^0 + 4 \times 16^{-1}$

$\Rightarrow 16 + E + 0.25$

$\Rightarrow (30.25)_{10}$

4) $(32)_4 = (\ )_{10}$.

$\Rightarrow 3\times4^1 + 2\times4^0$

$\Rightarrow (14)_{10}$.

Others to Others :

→ convert the given number system to decimal using others to decimal procedure and apply decimal to others procedure.

Ex: 1) $(32)_4 = (\ )_8$

$\Rightarrow (\ )_{10}$

$\Rightarrow (32)_4 \rightarrow (14)_{10} \rightarrow (16)_8$.

14 | 32 = (

$3\times4^1 + 2\times4^0$

$12+2 = (14)_{10}$

8 | 14

8 | 1 — 6

0 — 1

2) $(100)_x = (20)_8$.

$(1\times x^2 + 0\times x^1 \times 0\times x^0)_{10} = (8^1\times2 + 0\times8^0)_{10}$

$\Rightarrow (x^2)_{10} = (16)_{10}$

$x^2 = 16$

$x = 4$. [always the value must be +ve].

→ convert bothsides number systems to decimal number system & equate.

* EECET : $\sqrt{41} = 5$

The base of above number system is b, then the value of b is —

a) 3 b) 4 c) 5 d) 6 (or) a) 6 b) 7 c) 8 d) none

$b-1 = 5$

$b \geq 6$

$\sqrt{b^0\times1 + b^1\times4} = b^0\times5$

$\sqrt{4b+1} = 5$

$4b+1 = 25$

$4b = 24$

$b = 6$.

2) $(21)_x = (32)_5$

$x^1\times2 + x^0\times1 = 3^x5^1 + 2\times5^0$

$\therefore 2x+1 = 17$

$2x = 16$

$x = 8$

3) $64\times7 + 8\times5 + 1\times3$, the equivalent binary number system has number of 1's = ___

$64\times7 + 8\times5 + 1\times3$

$8^2\times7 + 8^1\times5 + 8^0\times3$

$(753)_8$

↓ ↓ ↓

111 101 011

$\Rightarrow (111101011)_2 \Rightarrow (7)$

No. of one's = 7

* Octal to Binary

$2^8 \rightarrow 2^3$ (possible combinations). 8 combinations.

n-bits $\Rightarrow$ possible combinations $\Rightarrow 2^n$.

$n = 3$.

	2^2	2^1	2^0
$\Rightarrow$	4	2	1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1

Hexadecimal to binary conversion

$16 \rightarrow 2^4$ $\quad n = 4$.

	2^3	2^2	2^1	2^0
	8	4	2	1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
A	1	0	1	0
B	1	0	1	1
C	1	1	0	0
D	1	1	0	1
E	1	1	1	0
F	1	1	1	1

* The equivalent binary no. system consists no. of 1's are ____

$256 \times 7 + 16 \times 9 + 1 \times 7$

$16^2 \times 7 + 16^1 \times 9 + 16^0 \times 7 \Rightarrow (797)_{16} = (0111\ 1001\ 0111)_{16}$

No. of one's (1's) are 8.

* $(ECE)_{16} \rightarrow (\ )_8$

$(ECE)_{16} \rightarrow (1110\ 1100\ 1110)_2 \rightarrow (7316)_8$.

* Addition.

→ **Decimal:**

```
  2 5 7
  9 6 7
-------
1 2 2 4
```

14 − 10 = 4 → ① carry
12 − 10 = 2 → ① carry
12 − 10 = 2 → ① carry.

→ **Octal:**

```
  1 7 5
  5 4 6
-------
  7 4 3
```

11 − 8 = 3 ①
12 − 8 = 4 ①

```
   7 3
 + 6 2
------
 1 5 5
```

→ **Hexadecimal:**

$(17A)_{16} + (992)_{16}$

```
  1 7 A
  9 9 2
-------
  B 0 C
```

```
  A 3 C
+ 2 E B
-------
  D 2 7
```

→ **Binary:**

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

* Subtraction

→ **Decimal:**

```
  9 6 5
- 7 9 6
-------
  1 6 9
```

→ **Octal:**

```
  2 5 6
- 1 7 7
-------
  0 5 7
```

```
  2 0 1
- 1 5 6
-------
  0 2 3
```

→ **Hexadecimal:**

```
  A 6 9
- 3 A A
-------
  6 B F
```

→ **Binary:**

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

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

* Multiplication : Binary Multiplication.

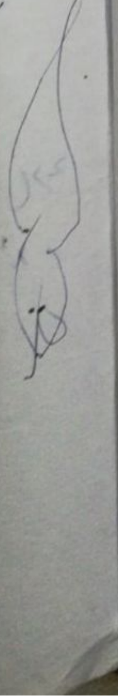
- Binary multiplication is also called as shift and Add method

→ n bits × m bits then expected result ⇒ (m+n) bits.

→ n bits × n bits ⇒ 2^n bits [max. no. of bits are]

```
      1 0 1
    × 1 1 0
-----------
      0 0 0
    1 0 1
  1 0 1
-----------
0 1 1 1 1 0
```

(3 + 3) = 6 bits



* Types of Number systems :

2 types -

1) Weighted Number system
2) Non-weighted Number system.

• If the no. system has both place and face value then it is called weighted number system.

→ with base r, the position values are $\ldots r^3, r^2, r^1, r^0$

Ex : If $r=10$, $\ldots, 10^2\ 10^1\ 10^0$ the positional values are : $\ldots 100, 10, 1$

If $r=2$, $\ldots 2^2, 2^1, 2^0$, the positional values are : $\ldots 8, 2, 1$

1) binary 2) octal 5) BCD
3) hexadecimal 4) decimal

• If the number system has only face value no place value then it is called unweighted / Non-weighted Number system.

Ex : Gray code, Excess-3 code.

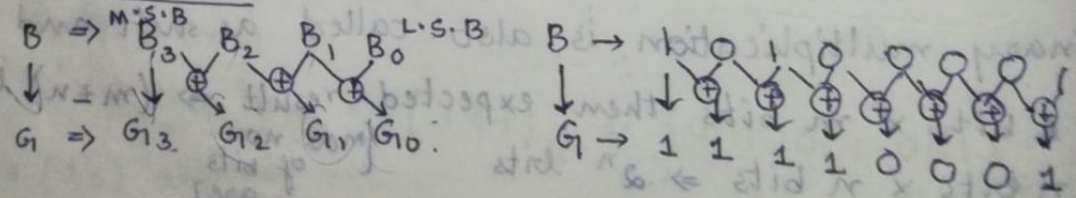
* Gray code :

• It is unweighted code.
• It is also called as unidistance code (or) cyclic code (or) reflective code.
• It has 1-bit change.
• It is used to design digital circuits i.e., in K-Map.
• X-OR operation is used to design gray code.

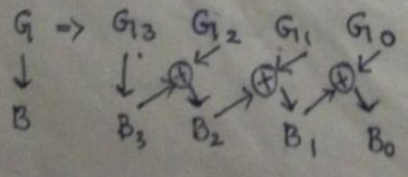
X-OR ⊕

$0 \oplus 0 = 0$
$1 \oplus 1 = 0$
$0 \oplus 1 = 1$
$1 \oplus 0 = 1$

Binary to Gray



Gray to Binary



G ⇒ 1 1 1 1 0 0 0 1
B ⇒ 1 0 1 0 0 0 0 1

Binary	Gray
0000	0000
0001	0001
0010	0011
0011	0010
0100	0110
0101	0111
0110	0101
0111	0100
1000	1100
1001	1101
1010	1111
1011	1110
1100	1010
1101	1011
1110	1001
1111	1000

$(67)_8$ ⇒ Equivalent gray code - ?

- first convert given no. system to binary

$(67)_8 \Rightarrow (110111)_2$

B → 1 1 0 1 1 1

G → 1 0 1 1 0 0

- To convert any number system into binary and then convert to graycode.

$(ABC)_{16}$ ⇒ graycode - ?

$(ABC)_{16} \Rightarrow (1010\ 1011\ 1100)_2$

B → 1 0 1 0 1 0 1 1 1 1 0 0

G → 1 1 1 1 1 1 1 0 0 0 1 0

* **BCD** : Binary coded Decimal.

⇒ It is weighted code.

- It is 4-bit code.
- It has 10 possible combinations
- It has 6 don't care combinations, those are : 1010, 1011, 1100, 1101, 1101, 1111

	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

- **Procedure to write BCD code :**

1) Convert any number system into decimal number system, then write each digit in 4-bit binary representation.

- ECET-Q :

$(EC)_{16}$ → BCD - ?, the equivalent BCD consists no. of nibbles are

Wkt $E \times 16^1 + C \times 16^0 \Rightarrow 224 + 12 = (236)_{10}$

(0010 0011 0110)

⇒ 3 Nibbles.

- ECET-Q : From the following code, which code is not used in BCD

a) 0111 b) 1000 c) 1001 d) 1010.

- $(1000)_8$: No. of nibbles in BCD

$1 \times 8^3 + 0 \times 8^2 + 0 \times 8^1 + 0 \times 8^0 \Rightarrow (512)_{10} \Rightarrow (0101\ 0001\ 0010)_2$ = 3 Nibbles

* Excess-3 Code :

- It is also called self complementing code
- It is [BCD + 0011] code
- It has 10 combinations
- 6 don't care conditions: 0000, 0001, 0010, 1101, 1110, 1111

BCD	Excess-3
0000	0011
0001	0100
0010	0101
0011	0110
0100	0111
0101	1000
0110	1001
0111	1010

0 1 2 13 14 15

	BCD
1	0 0 0 0
2	0 0 0 1
3	0 0 1 0
4	0 1 1 1
5	0 1 0 0
6	0 1 0 1
7	0 0 1 0
8	1 0 1 1
9	1 0 0

Eg : $(EC)_{16} \Rightarrow (236)_{10}$

0010 0011 0110
0011 0011 0011
0101 0110 1001

- Convert any number system into decimal then decimal to BCD then add 0011 to every four bits

	BCD	Excess-3
0	0 0 0 0	0 0 1 1
1	0 0 0 1	0 1 0 0
2	0 0 1 0	0 1 0 1
3	0 0 1 1	0 1 1 0
4	0 1 0 0	0 1 1 1
5	0 1 0 1	1 0 0 0
6	0 1 1 0	1 0 0 1
7	0 1 1 1	1 0 1 0
8	1 0 0 0	1 0 1 1
9	1 0 0 1	1 1 0 0

→ self-complementing code

* Alphanumeric code : ASCII, EBCDIC

- These codes are used to convert (A-Z), (a-z), (0 to 9), (+, -, *, /), special symbols into 0's and 1's
- It is of 2 types:
  1) ASCII - American Standard Code for Information Interchange
  2) EBCDIC -

ASCII – American standard code for Information Interchange

- It is 7-bit code
- It converts 128 characters into 0's and 1's

$2^7 \rightarrow 128 \longrightarrow 64\ 32\ 16\ 8\ 4\ 2\ 1$

Ex: A → 65 → 1000001

a → 97 → 1100001

0 → 48 → 0110000

EBCDIC – Extended Binary coded Decimal Interchange code

- It is 8-bit code ($2^8 \rightarrow 256$)
- It converts 256 characters into 0's and 1's

* Parity code:

- It is one-bit error code.
- In this code extra 1 parity bit is adds with information it is two types: i) Even parity ii) odd parity.
- Even parity: If the no of 1's in the information is even number then the even parity bit is 0. otherwise 1 [odd].
- Odd parity: If the no. of 1's in the information is odd number then the odd parity bit is 0 otherwise 1.

Data	P_e	P_o
10101110	1	0
11110000	0	1

* Hamming Code:

- It is error detecting and correcting code. → $2^K \geq n+K+1$ [No. of possible combinations]
- Data (n bits) ← HC → codded ($n+k$)

parity bits (k bits)

* Binary Data Representation

Compliments

- The number system with base r has two types of complements r's complement and $(r-1)$'s complement.
- r.

base r	(r-1)'s complement	(r)'s complement
r = 2	1's	2's
r = 8	7's	8's
r = 10	9's	10's
r = 16	F's	16's

- To find (r-1)'s complement, subtract given number from maximum possible number.
- To find r's complement, add 1 to (r-1)'s complement
- Eg :1) $(253)_{10}$

```
      999        10's complement
    - 253          746
9's  ---           + 1
complement → 746   ---
                   747
```

2) $(1010010)_2$ find 2's complement

```
  1111111
- 1010010
  0101101 → 1's complement
      + 1
  0101110 → 2's complement
```

* 1010010
→ from right-side until 1 write same and then complement other values
0101110

* 101011
010101 → 2's

* GCET-Q:
$(200)_8$ → 8's a) 577 b) 578 c) 600
d) None

```
  777
- 200
  577 → 7's
   +1
  600 → 8's
```

*) $(320)_4$ → 4's.

```
  333     4's complement → 020.
- 320
  013
+   1
  020
```

Binary Data Representation

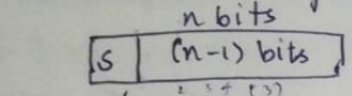
- Magnitude
  - unsigned (+ve) ✓
  - signed (+ve or -ve) × not used
- complement
  - 1's (-ve, +ve) × not used
  - 2's (+ve, -ve) ✓

* Unsigned Number System

- It is used to represent only positive numbers, range is 0 to $(2^n - 1)$ for n bits.
- if $n = 4$ then range is 0 to $(2^4 - 1)$ => 0 to 15
- if $n = 8$ then range is 0 to $(2^8 - 1)$ => 0 to 255

* Signed Representation :

- It is used to represent both +ve & -ve numbers.
- MSB bit is used to represent 'sign' of the number system.
- The number system has n-bits



+ve → 0
-ve → 1

	bits
+7	0111
-7	1111

-9 => is not possible [it requires 5 bits].

+0 => 0000
-0 => 1000
[+0 ≠ -0].

Range for signed no.s

$-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$.

$n = 4$ => -7 to +7

$n = 8$ => -127 to +127

Eg : signed representation

→ ① 00010 => 2
→ ① 10010 => -18
→ ⑥ 11000 => +24

Draw back

- It has two zeroes.

* One's Complement Representation:

- It is used to represent +ve and -ve numbers.
- +ve numbers are similar to signed representation
- -ve numbers are 1's complement of +ve numbers

1's complement

+7 => 0111
+2 => 0010
+6 => 0110
-6 => 1001
-2 => 1101
+0 => 0000
-0 => 1111

-0 ≠ 0

Range :

$-(2^{n-1} - 1)$ to $+(2^{n-1} - 1)$

- To find equivalent no. in 1's complement representation
  - → if MSB is 0, the no. is +ve number, then write equivalent decimal no. from the binary
  - → if MSB is 1, the no. is -ve number, then find 1's complement of the

number and write equivalent decimal value.

0 1001 ⇒ +9

1 1001 ⇒ ~~-9~~ -6.

0 0110 (1's as it is -ve)

Drawback: It has two zeroes.

* 2's Complement Representation:

• It is used to represent +ve and -ve numbers. +ve numbers are similar to signed representation, -ve numbers are 2's complement of signed representation. (or) +ve number.

2's

+7 → 0 111

-7 ⇒ 1 001

+17 ⇒ 010001

-17 ⇒ 101111

+0 ⇒ 0000

-0 ⇒ 0000

+0 = -0

Range:

$-(2^{n-1})$ to $(2^{n-1} - 1)$.

using 8 bits: n = 8 ⇒ -128 to +127

Binary	Unsigned	Signed	1's	2's
0000	0	+0	+0	+0
0001	1	+1	+1	+1
0010	2	+2	+2	+2
0011	3	+3	+3	+3
0100	4	+4	+4	+4
0101	5	+5	+5	+5
0110	6	+6	+6	+6
0111	7	+7	+7	+7
1000	8	-0	-7	-8
1001	9	-1	-6	-7
1010	10	-2	-5	-6
1011	11	-3	-4	-5
1100	12	-4	-3	-4
1101	13	-5	-2	-3
1110	14	-6	-1	-2
1111	15	-7	-0	-1

Boolean Algebra

Minimization of boolean expression: 1) Boolean algebra
K-Map
Tabulation Method.

Advantages of minimization of boolean expressions:

1. Number of logic gates are decreases
2. circuit size is decreases
3. Speed of the operation increases.
4. power dissipation & cost decreases.

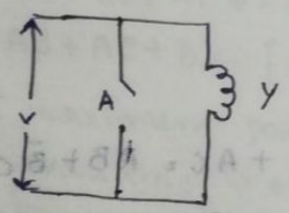
3 techniques are used to minimize boolean expression

1. Boolean Algebra: This technique is used if the output consists either 0 (or) 1. It is suitable for 0-3 variables.
2. K-Map: This technique is used if the output consists 0, 1 or X (don't care). It is suitable to 5 variables.
3. Tabulation Method: It is suitable than more than 5 variables

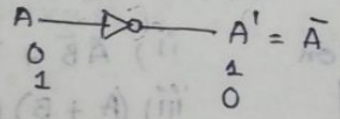
Boolean Algebra

0 => low => +ve => OFF => OPEN circuit
1 => high => -ve => ON => Short circuit

Invertor (NOT):

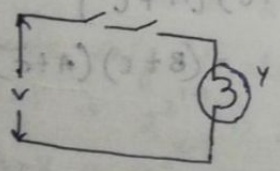


A	Y
0	1 (ON)
1	0 (OFF)

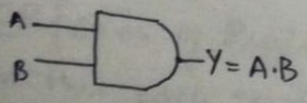


A	A' = Ā
0	1
1	0

AND:

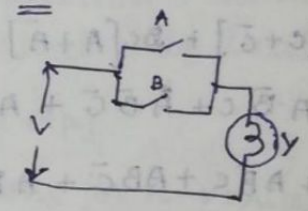


A	B	Y
0	0	0
0	1	0
1	0	0
1	1	1

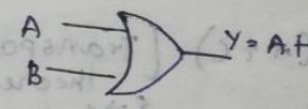


A·A = A
A·Ā = 0
A·0 = 0
A·1 = A

OR:



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



A+A = A
A+Ā = 1
A+1 = 1
A+0 = A

→ series (AND)
→ parallel (OR)
→ parallel with load (NOT)

1) $Y = A+A+A+A = A$

2) $Y = A + \underbrace{A+\bar{A}}_{1} + A = 1$

3) $Y = AB + A\bar{B}$
$= A[B+\bar{B}] = A$

4) $Y = AB + A\bar{B} + \bar{A}B$ –
$= A(B+\bar{B}) + \bar{A}B$
$= A + \bar{A}B$. adding AB

$\Rightarrow AB + A\bar{B} + \bar{A}B + AB$
$A(B+\bar{B}) + B(A+\bar{A})$
$= A + B$.

Absorption

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

except a:
$wx + \overline{wx}yz = wx + yz$.

$wx + \overline{wx}y$

Laws:

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

Canonical form.

- Each term consists all the variables it contains 3 terms.

$f = AB + BC$
$= AB[C+\bar{C}] + [A+\bar{A}]BC$
$= ABC + AB\bar{C} + ABC + \bar{A}BC$
$= ABC + AB\bar{C} + \bar{A}BC$

$f = A + BC$
$= A[B+\bar{B}][C+\bar{C}] + BC[A+\bar{A}]$
$= ABC + A\bar{B}C + A\bar{B}\bar{C} + ABC + \bar{A}BC + AB\bar{C}$
$= ABC + A\bar{B}C + AB\bar{C} + A\bar{B}\bar{C} + \bar{A}BC$.

Redundancy theorem: Each term is defined with two variables and 1 variable is complement, then the complement variable is not defined in one term is called redundant term

$f = (A+B)(A+C)$ [Transposition theorem $\Sigma \rightarrow \cdot$, $\cdot \rightarrow \Sigma$.]
$= A + BC$.

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

$f = AB + BC + \bar{A}C$. [Redundancy theorem / consenses theorem].
$= AB[C+\bar{C}] + BC[A+\bar{A}] + \bar{A}C[B+\bar{B}]$
$= ABC + AB\bar{C} + ABC + \bar{A}BC + \bar{A}BC + \bar{A}\bar{B}C$
$= ABC + \bar{A}BC + AB\bar{C} + \bar{A}\bar{B}C$
$= AB(C+\bar{C}) + \bar{A}C(B+\bar{B})$
$= AB + \bar{A}C$. [BC – Redundant term]

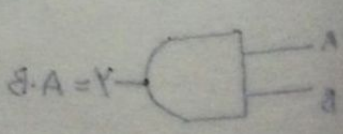
i) $AB + \bar{B}C + AC = AB + \bar{B}C$

ii) $A\bar{B} + \bar{A}C + \bar{B}\bar{C} = \bar{A}C + \bar{B}\bar{C}$

iii) $(A+B)(B+C)(A+\bar{C})$
$= (B+C)(A+\bar{C})$

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

$A = A \cdot A$
$0 = \bar{A} \cdot A$
$0 = 0 \cdot A$
$A = 1 \cdot A$



* De-Morghan's theorem

- This theorem is used to convert total compliments into individual complements

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

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

$Y \xrightarrow{\text{compliment}} \bar{Y}$

AND → OR

OR → AND

0 → 1

1 → 0

A → $\bar{A}$

$\bar{A}$ → A

i) $Y = A+\bar{B}C$

$\bar{Y} = \bar{A}\cdot[B+\bar{C}]$.

ii) $Y = AB+\bar{C}D$

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

$(A+\bar{B})(\bar{A}+B)$

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

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

$(\bar{A}+B)(A+\bar{B})$

$\bar{A}+B$

- **Duality:** It is used to interchange +ve logic and -ve logic

+ve logic	-ve logic
0 -> low	0 -> high
1 -> high	1 -> low

AND $\xrightarrow{\text{duality}}$ OR

OR → AND

0 → 1

1 → 0

A → A

$\bar{A}$ → $\bar{A}$ } literals donot change [self-duals].

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

$= A\cdot(\bar{B}+C)$.

2) $AB+BC+AC$

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

$(B+AC)(A+C)$

$AB+BC+AC+AC$

$\Rightarrow AB+AC+BC$. [self-duality's (or) self dual exp.].

* duality of one expression is same expression then it is called self dual.

The maximum possible boolean expressions for n-variables are: 2^{2^n}

=> 1-variable => M.P.exp => $2^{2^1} = 4$. [0, 1, A, $\bar{A}$].

=> n=2 => 2^{2^2} => $2^4 = 16$

=> n=3 => $2^{2^3} = 2^8 = 256$

Possible self-duals using n variables are: $2^{2^{n-1}}$

=> n=1 => $2^{2^{1-1}}$ => 2

=> n=2 => $2^{2^{2-1}}$ => 4.

=> n=3 => $2^{2^{3-1}}$ => $2^{2^2} = 2^4 = 16$. [ECET].

SOP [Sum of Product]

- In this each term is minterm (or) product term
- Σm, it explains the sum of minterms (or) product terms

1 - A (active)

0 - $\bar{A}$

=> 1 1 0 1

A B $\bar{C}$ D

POS [Product of sum]

- In this, each term is called max term (or) sum term.
- ΠM explains the POS

  0 → A (active)

  1 → $\bar{A}$

Ex: 1101

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

- Relation b/w SOP and POS

$\overline{SOP} = POS$

$\overline{POS} = SOP$

AB	SOP	POS
00	$\bar{A}\bar{B}$	$A+B$
10	$\bar{A}B$	$A+\bar{B}$
01	$A\bar{B}$	$\bar{A}+B$
11	AB	$\bar{A}+\bar{B}$

$f(A,B) = \Sigma m(1,2)$

AB	f
00	0
01	1
10	1
11	0

Expression:

SOP: $\bar{A}B + A\bar{B}$

$= \Pi M(0,3)$.

$AB + \bar{A}\bar{B}$.

SOP

0 → $\bar{A}$

1 → A

POS

0 — A

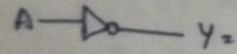
1 — $\bar{A}$

Logic Gates

- Basic Gates (AND, OR, NOT) designed from boolean expressions
- Special Gates (XOR, XNOR)
- Universal gates (NAND, NOR)

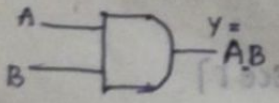
BASIC GATES

1) NOT:



A	Y
0	1
1	0

2) AND

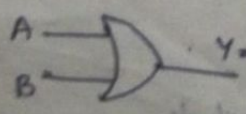


A B	Y
00	0
01	0
10	0
11	1

i/p of

- disable and gate is 0
- enable i/p of and gate is 1

3) OR

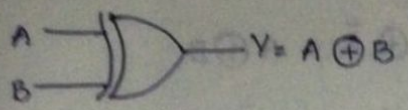


A B	Y
00	0
01	1
10	1
11	1

- disable i/p of AND gate is 1
- enable i/p of OR gate is 0

SPECIAL GATES

1. Exclusive-OR (X-OR): used to compare two inputs.



→ even/odd parity generators
→ acts as inverter/buffer.

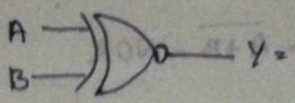
A	B	Y
0	0	0
0	0	1
0	1	1
1	1	0

$A \oplus B = \bar{A}B + A\bar{B}$ [SOP]

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

$= (A+B)\cdot(\bar{A}+\bar{B})$ [POS]

2. Exclusive-NOR (X-NOR)

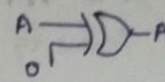


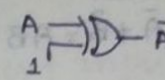
A	B	Y
0	0	1
0	1	0
1	0	0
1	1	1

$A \odot B = \bar{A}\bar{B} + AB$ (SOP)

$\Rightarrow (A+\bar{B})\cdot(\bar{A}+B)$ (POS)

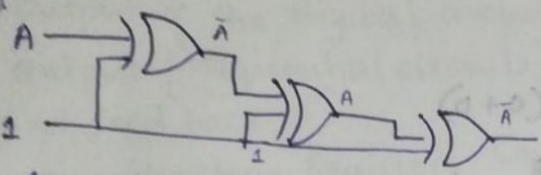
$A \oplus A = 0$.
$A \oplus \bar{A} = 1$
$A \oplus 0 = A$
$A \oplus 1 = \bar{A}$





UNIVERSAL GATES

(a)



- If n is even, output is A
- If n is odd, output is Ā

EG: $Y = A \oplus A \oplus A \oplus A \oplus A = A$.

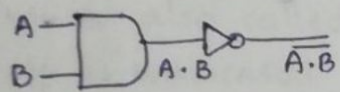
$0 \oplus A$

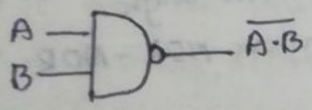
$A \oplus A$

$0 \oplus A = A$

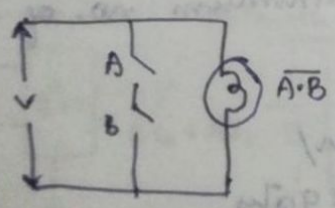
UNIVERSAL GATES

1. NAND [NOT+AND].

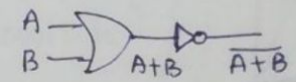


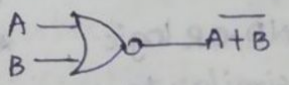


A	B	Y
0	0	1
0	1	1
1	0	1
1	1	0

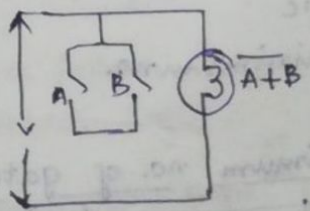


2. NOR [NOT+OR]



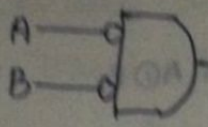


A	B	Y
0	0	1
0	1	0
1	0	0
1	1	0

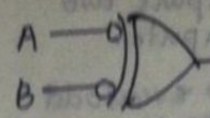


- Implementation of any boolean expression is possible using NAND and NOR [universal gates].

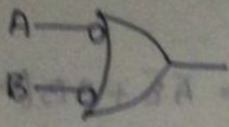
Bubbled gates



$\bar{A} \cdot \bar{B} = \overline{A+B} = NOR$



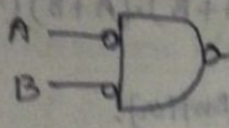
$\bar{A} \oplus \bar{B} = A \oplus B$



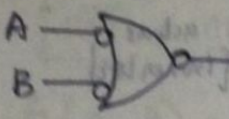
$\bar{A} + \bar{B} = \overline{A \cdot B} = NAND$

If any one input is bubbled then
X-OR → X-NOR

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

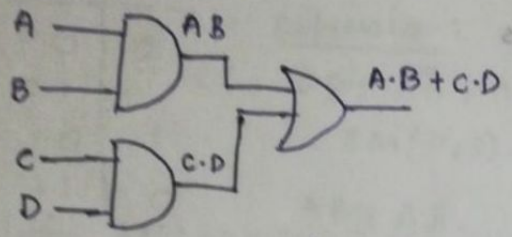


$\overline{\bar{A} \cdot \bar{B}} = A + B = OR$



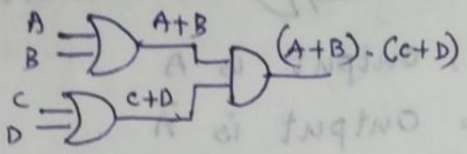
$\overline{\bar{A} + \bar{B}} = AB = AND$

$Y = AB + CD.$



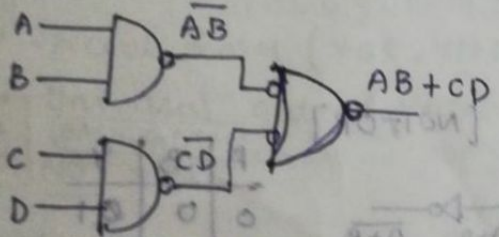
SOP : AND-OR logic

$Y = (A+B) \cdot (C+D)$

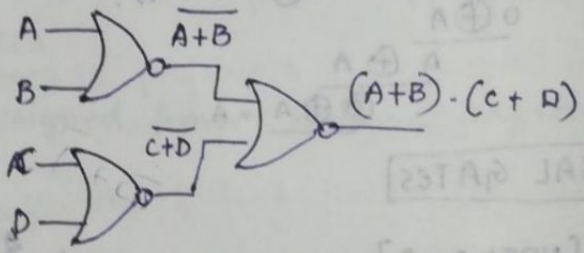


POS: OR-AND logic.

Using NAND gates:



SOP: NAND-NAND logic



POS: NOR-NOR logic

* SOP follows the AND-OR logic
* AND-OR logic is similar to NAND-NAND logic
* SOP requires minimum no. of NAND gates

* POS follows the OR-AND logic
* OR-AND logic is similar NOR-NOR logic
* POS requires minimum no. of NOR gates.

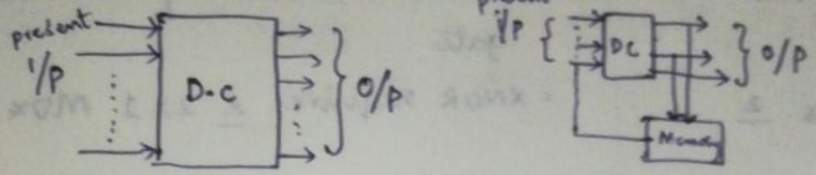
Minimum no. of gates required to design/develop/implement other gates

	NAND	NOR
NOT	1	1
AND	2	3
OR	3	2
X-OR	4	5
X-NOR	5	4

Digital Circuits

Digital circuits

Combinational circuits | Sequential circuits



- Output of the digital circuits is depends on present input [combin-ational circuits]
- Output of the digital circuits depends on present input and past output [sequential circuits]. Sequential circuits requires memory and feed back

Ex: flipflops, Registers, Counters, Serial adder

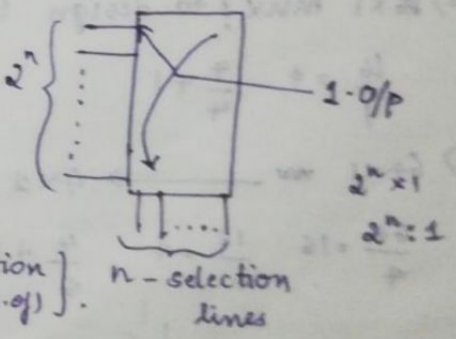
Combinational circuits:

- Procedure to design combinational circuits:

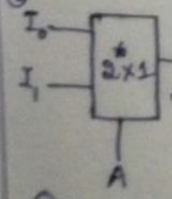
1. Identify the no. of inputs and outputs
2. Write the relation b/w inputs and ontputs using truth table
3. Design the circuit from output combinations.

Multiplexer:

- It has many inputs and one output
- It is also called as data selector switch
- It is universal logic circuit
- It acts as parallel to serial converter
- Standard representation :- $2^n : 1$ [n - selection lines (no. of)]. $2^n \times 1$

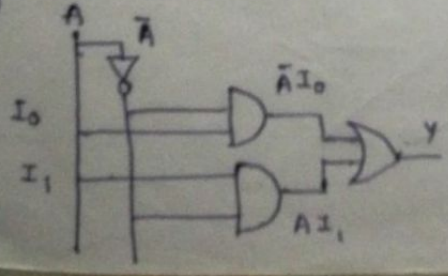


2×1 MUX:

① 

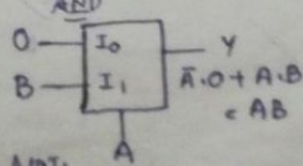
② Truth Table:

A	Y
0	I_0
1	I_1

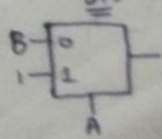
③ 

* Multiplexer follows AND-OR logic (or) NAND-NAND logic

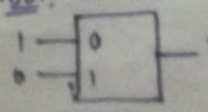
AND



OR

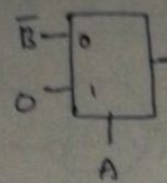


NOT:



- AND
- OR
- NOT

} operations require 1 2×1 mux

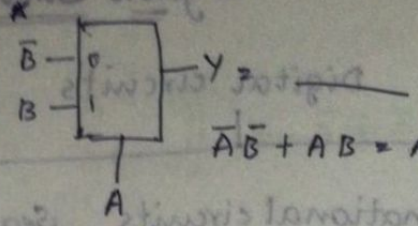
★ 

Y = NOR gate.

$\bar{A}\bar{B} + A0$

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

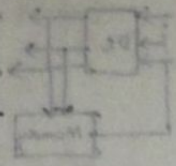
- The above circuit acts as NOR gate
- NOR gate requires 2 2×1 MUX

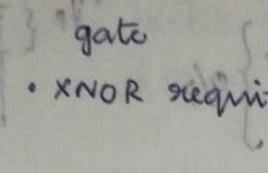


Y =

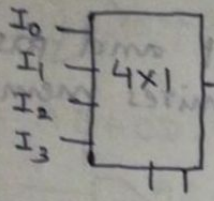
$\bar{A}\bar{B} + AB = A \oplus B$

- The above circuit represents as X-NOR gate
- XNOR requires 2 2×1 MUX

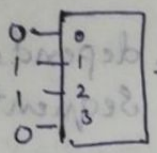




4×1 MUX

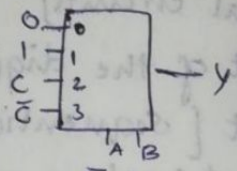


$\bar{A}\bar{B}I_0 + \bar{A}BI_1 + A\bar{B}I_2 + ABI_3$



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

$A \oplus B$



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

A	B	Y
0	0	I_0
0	1	I_1
1	0	I_2
1	1	I_3

Canonical form : $\bar{A}BC + \bar{A}B\bar{C} + A\bar{B}C + AB\bar{C}$

$\Rightarrow \Sigma m(2,3,5,6)$

1) No. of 2×1 multiplexers are required to design 4×1

$\Rightarrow \frac{4}{2} = 2 \quad \frac{2}{2} = 1 \quad 2+1=3$

2) 4×1 mux? to design 16×1 = 4+1 = 5

$\frac{16}{4} = 4 \quad \frac{4}{4} + 1$

3) 64×1 ⟶ 4×1 = 21

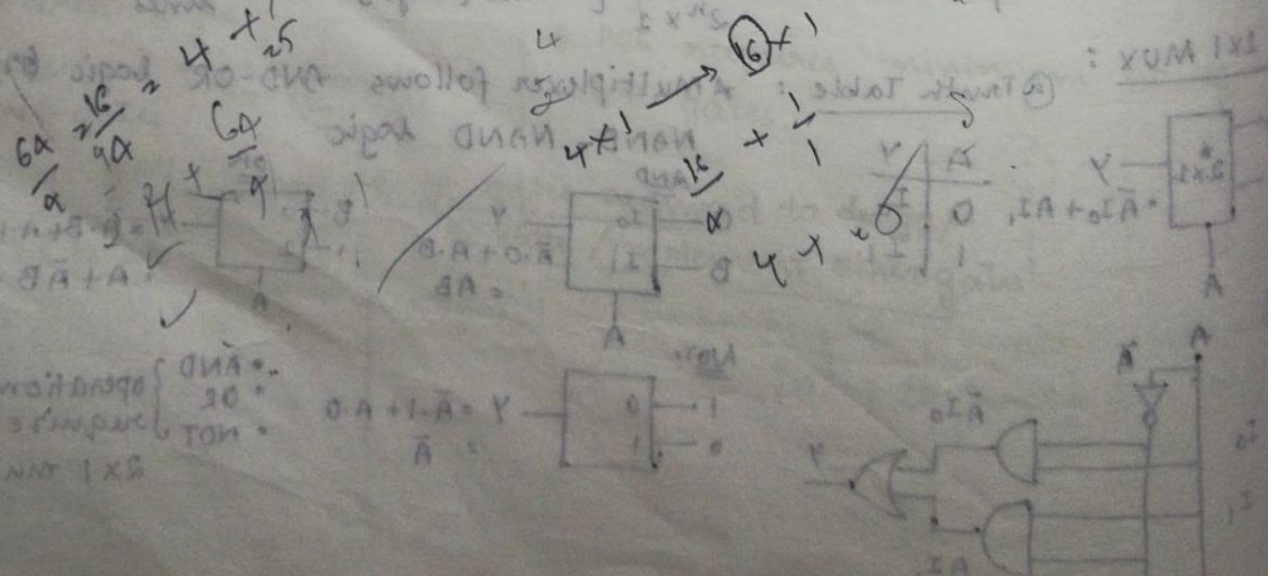
$\frac{64}{4} = 16 \quad \frac{16}{4} = 4 \quad \frac{4}{4} + 1$

4) 64×1 ⟶ 8×1 = 9

$\frac{64}{8} = 8 \quad \frac{8}{8} = 1 \quad = (9)$

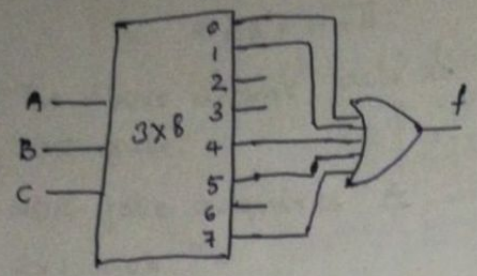
5) 8×1 ⟶ 2×1 = 4+2+1=7

$\frac{8}{2} = 4 \quad \frac{4}{2} = 2 \quad \frac{2}{2} = 1$



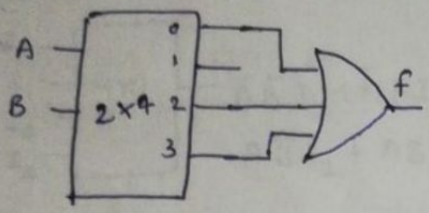
1) $f = \Sigma m(0, 1, 4, 5, 7)$ → 111

3-variable



AND-OR

2 →



$f = \Sigma m(0, 2, 3)$.

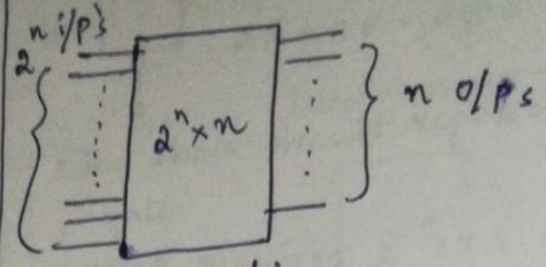
$\bar{A}\bar{B} + A\bar{B} + AB$.

- Decoder - n inputs then outputs - 2^n.

- digital to analog - Decoder

- → MUX follows AND-OR logic
- → Demux follows AND logic
- → Decoder follows AND logic
- → Encoder follows OR logic
- → Multiplexer is a universal logic gate circuit

Encoder

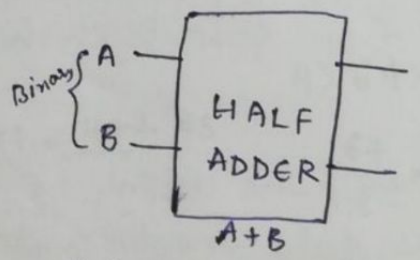


decimal → binary.

- It has 2^n inputs and n outputs.
- It is used to convert decimal to binary, octal to binary, hexadecimal to binary.
- It acts as a analog to digital converter
- It follows the OR-Logic

Binary Adder :

i) Half Adder :



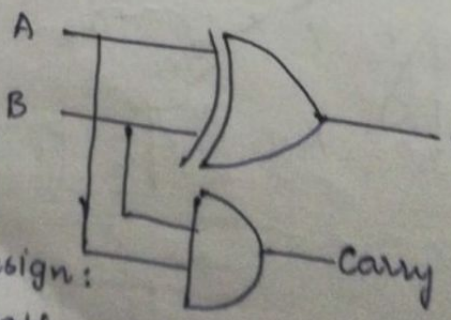
$\bar{A}B + A\bar{B} = (A+B)\cdot(\overline{AB})$

sum = $\Sigma m(1, 2) = \Pi M(0,3)$

$= A \oplus B$.

carry = AB.

A	B	sum	carry
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

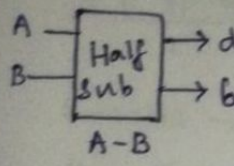


To design:

- Half adder requires 1 X-OR gate an

- Half adder requires 5 NAND gates.
- Half adder requires 5 NOR gates.
- Half adder requires 3 2×1 multiplexers to design using mux
- Half adder requires 1 2×4 decoders

Half subtractor:



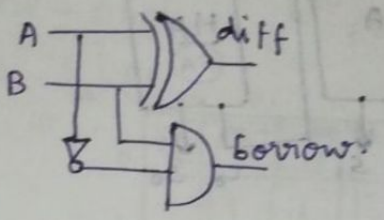
diff $\Rightarrow \bar{A}B + A\bar{B}$

$\Rightarrow A \oplus B$

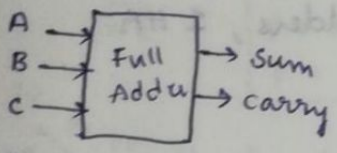
borrow $\Rightarrow \bar{A}B$

- The above points are same to design half subtractor [i.e., same as Half adder]

A	B	diff	borrow
0	0	0	0
0	1	1	1
1	0	1	0
1	1	0	0



Full Adder:



A	B	C	sum	carry
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

carry $= \Sigma m(3,5,6,7) = \pi M(0,1,2,4)$

$= \bar{A}BC + A\bar{B}C + AB\bar{C} + ABC$ [SOP]

$= ABC + (AB\bar{C}) + (A\bar{B}C) + (\bar{A}BC)$

$= C(\bar{A}B + A\bar{B}) + AB(\bar{C}+C)$

$= AB + [A \oplus B] \cdot C$

$= AB + AC + BC.$

Sum $= \Sigma m(1,2,4,7) = \pi M(0,3,5,6)$

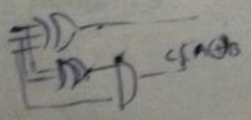
$\Rightarrow \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$

$\Rightarrow \bar{A}[B \oplus C] + A[\overline{B \oplus C}]$.

$\Rightarrow A \oplus B \oplus C$

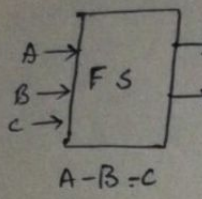
Sum $= A \oplus B \oplus C$

carry $= AB + AC + BC$ (or) $AB + [A \oplus B]C$



- Full adder requires 2 Half adders 1 OR
- Full adder requires 2 X-OR gates 2 AND gates 1 OR gates
- Full adder requires 9 NAND gates
- full adder requires 9 NOR gates
- full adder requires 1 3×8 Decoder
- sum of full adder is $A \oplus B \oplus C$
- carry - $AB + C(A \oplus B)$

Full Subtractor

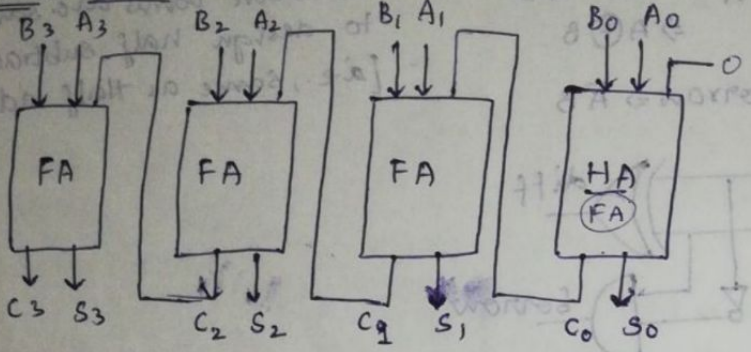


difference = $A \oplus B \oplus C$

borrow = $\bar{A}B + BC + \bar{A}C$.

- Requirement is same as full adder but half adder replaced with half subtractor

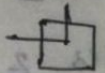
Parallel Adder



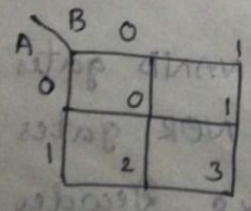
* parallel adder requires :
→ (n-1) full adders and 1 half adder
→ If n = 4, then parallel adder requires 3 full adders, 1 HA
→ (2n-1) half adders and (n-1) OR gates.
Eg : 4 bit, then 7 HA and 3 OR gates
→ n full adders

K-Map : Karnaugh Maps

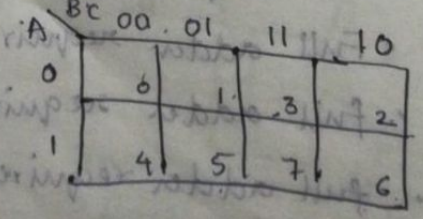
- It is used to minimize the boolean expression. if the O/P is 0, 1 (or) don't care
- It is a graphical method of representation.
- Graycode is used in K-Map.
- Using n-variables, it requires 2^n cells. cell is a rectangular or square which is a combination of row and column



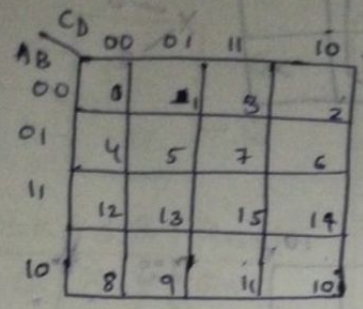
n = 2



n = 3



n = 4



Mapping.

2^n → highest priority

2^{n-1} → ...

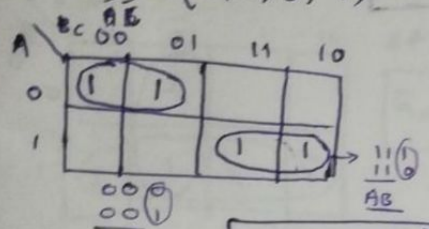
2^{n-2}

2^0 → lowest priority.

8-cells → octet 2-cells → pairs

4-cells → quad 1-cell – single.

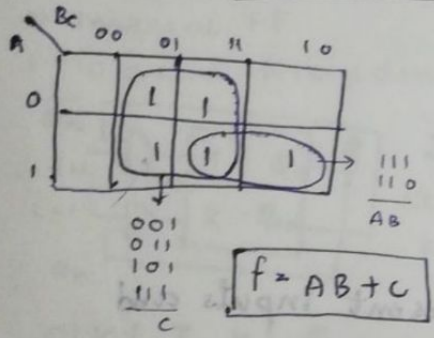
1) $f = \sum m(0,1,6,7)$



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

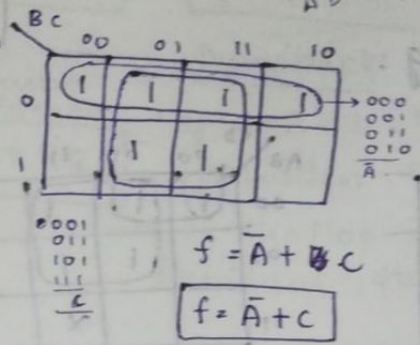
$f = A \odot B$

2)



$f = AB + C$

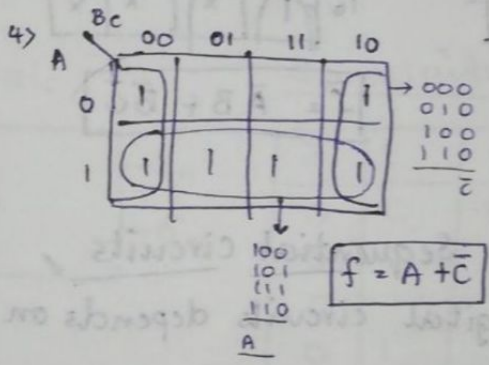
3)



$f = \bar{A} + BC$

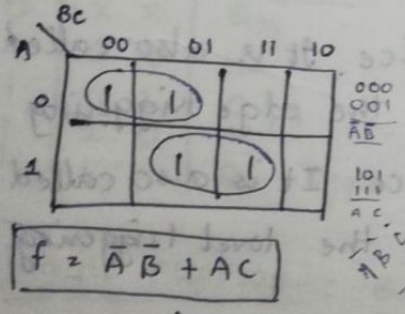
$f = \bar{A} + C$

4)



$f = A + \bar{C}$

5)



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

(or)

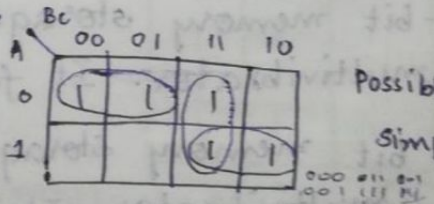
$f = \bar{A}B + AC + \bar{B}C$

Redundant

max Possible pairs : 3

min: 2

6)

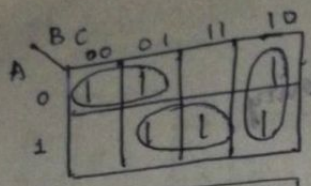


Possible pairs max : 4

Simplified exp : 3

$f = \bar{A}\bar{B} + BC + AB$ (or)

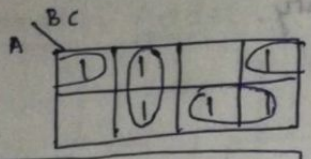
$f = \bar{A}\bar{B} + AB + \bar{A}C$



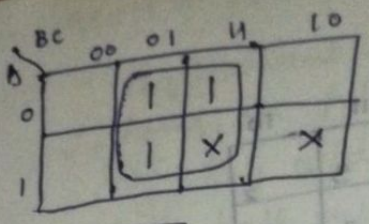
Possible: 6
simplified: 3

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

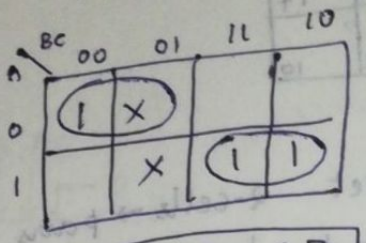
(or)



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



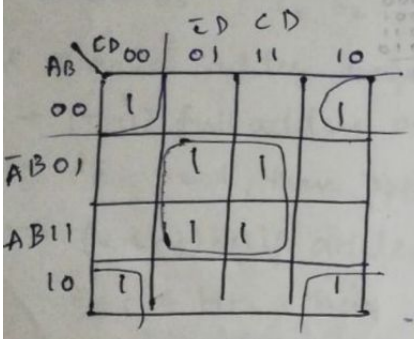
$f = C$



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

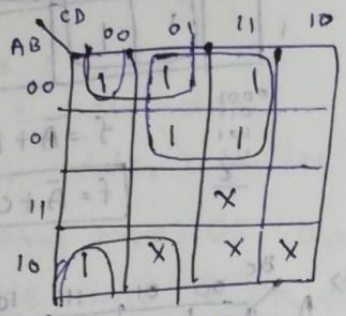
$\bar{f} = A \odot B$

4-variable Mapping:



$\bar{f} = BD + \bar{B}\bar{D}$

$f = B \odot D$

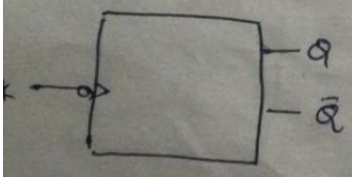


$\bar{f} = \bar{A}B + B\bar{C}$

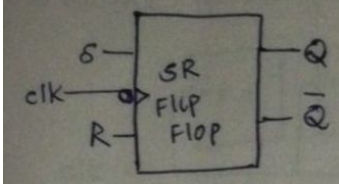
Sequential circuits

- Output of the digital circuits depends on present inputs and past outputs.
- Flipflop is a 1-bit memory storage device. It is also called as bistable multivibrator. It follows the edge triggering
- Latch is a 1-bit memory storage device. It is also called as bistable multivibrator. It follows the level triggering

FlipFlop



1) SR-FF : Basic flipflop

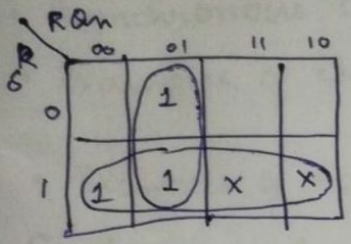


CLK	S	R	Q_{n+1}
0	X	X	Q_n
1	0	0	Q_n
1	0	1	0
1	1	0	1
1	1	1	X

- $S = R = 0$
  $\Rightarrow Q_{n+} = Q_n$
- $S \neq R$
  $\Rightarrow$ then $Q_{n+1} = S$
- $S = R = 1$
  $\Rightarrow Q_{n+1} = X$
  (forbidden/indetimiate)

characteristic Table :

CLK	S	R	Q_n	Q_{n+1}	
0	0	0	0	0	} Q_n No change
0	0	1	1	1	
0	1	0	0	0	
0	1	1	1	0	
1	0	0	0	1	
1	0	1	1	1	
1	1	0	0	X	
1	1	1	1	X	



** characteristic equation

$$Q_{n+1} = S + \bar{R}Q_n$$

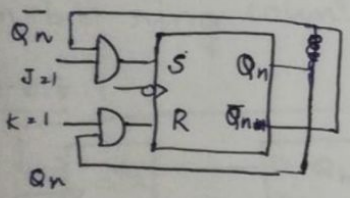
Draw back

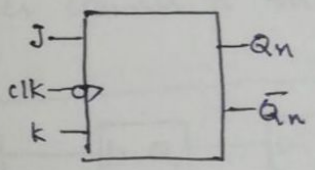
- If $S = R = 1$, the FF enters into forbidden state.

→ JK Flip Flop is used to avoid forbidden state.

2) JK-FF :

- Universal FF
- To avoid forbidden state





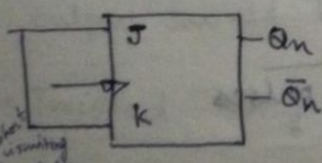
CLK	J	K	Q_{n+1}
0	X	X	Q_n
1	0	0	Q_n
1	0	1	0 } J
1	1	0	1
1	1	1	$\bar{Q}_n$ Toggle

characteristic table

J	K	Q_n	Q_{n+1}	
0	0	0	0	} Q_n
0	0	1	1	
0	1	0	0	
0	1	1	0	
1	0	0	1	
1	0	1	1	
1	1	0	1	} $\bar{Q}_n$
1	1	1	0	

** characteristic equation $Q_{n+1} = J\bar{Q}_n + \bar{K}Q_n$

3) Toggle FlipFlop :



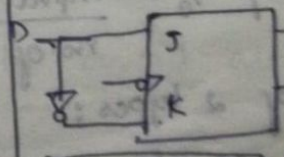
T	Q_{n+1}
0	Q_n
1	$\bar{Q}_n$ toggle

characteristic Table

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

** $Q_{n+1} = \bar{T}Q_n + Q_n\bar{T}$
$= T \oplus Q_n$

4) Delay FF (D-FF)



D	Q_{n+1}
0	0
1	1

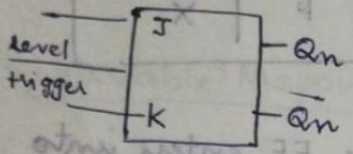
* characteristic eq

$$Q_{n+1} = D$$

Excitation Table :

Q_n	Q_{n+1}	J	K	S_n	R_n	T	D
0	0	0	x	0	x	0	0
0	1	1	x	1	0	1	1
1	0	x	1	0	1	1	0
1	1	x	0	x	0	0	1

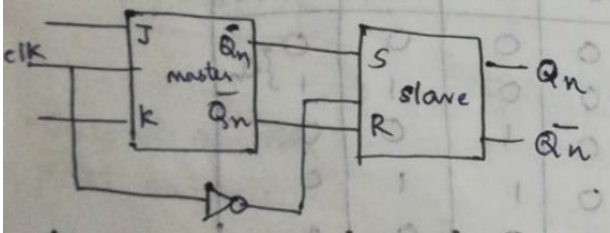
Race around Condition : In J-K flipflop, $J = K = 1$ and level trigger is connected then the output changes many times. This condition is also called as Race around condition.



$Q_{n+1} = \bar{Q}_n$

- To avoid race around condition we can use
1) Edge triggered flipflop
2) $t_w < t_{pd} < T$ [Master slave JK FF].

here t_w - width of clock pulse T - Time period of the clock pulse
t_{pd} - propagation delay of FF



Applications of flipflops :

1) Flipflops acts as frequency dividers, registers, counters

Counters :

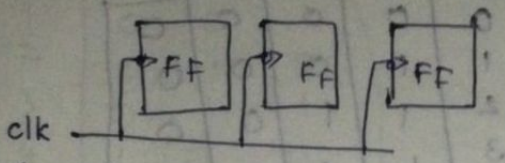
These are used to count different states or clock pulses. It is a combination of more than one flipflop.

Output frequency $f_0 = \frac{\text{input frequency}}{\text{no. of clock pulses}} = \frac{f_i}{\text{no. of clk pulses}}$

- Counters are of 2 types :
1) Synchronous counters
2) Asynchronous counters

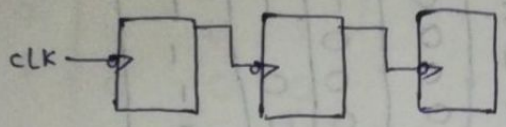
• Synchronous counters

→ same clock pulse is connected to all the Flipflops



• Asynchronous counters

→ different clock pulses are connected to the flipflop



• Synchronous counters are fast counters.

• Asynchronous counters are slow counters.

• Examples of Synchronous counter : Ring counter, twisted ring counter

• Examples of Asynchronous Counter : Ripple counters.

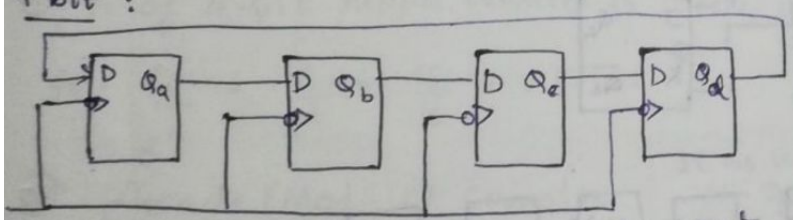
Synchronous Counter

is Ring counter

n-bit ring counter requires n no. of FFs. It is also called as Mod n counter.

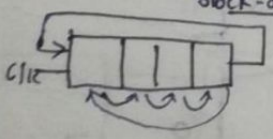
n-bit ring counter counts n no. of states.

4-bit :



block-diagram

Mod 4 counter



clk	Q_A	Q_B	Q_C	Q_D
0	1	0	0	0
1	0	1	0	0
2	0	0	1	0
3	0	0	0	1
4	1	0	0	0

∴ It counts 4 clk pulses

$$\Rightarrow f_n = \frac{f_i}{4} = \frac{f_i}{n} \quad \text{mod-}N$$

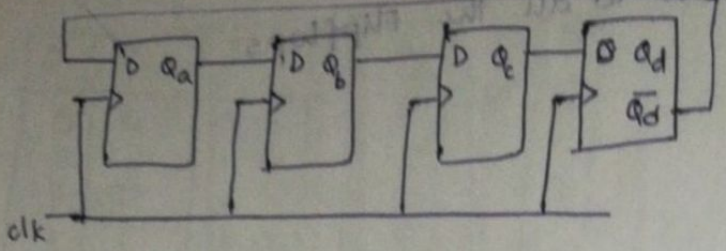
→ initial state 1000.

Drawback :

• Starting problem, [any one flipflop output should be 1]

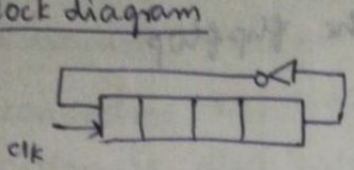
→ Twisted ring counter is used to avoid above drawback

Twisted/Johnson Ring Counter



clk	Qa	Qb	Qc	Qd
0	0	0	0	0
1	1	0	0	0
2	1	1	0	0
3	1	1	1	0
4	1	1	1	1
5	0	1	1	1
6	0	0	1	1
7	0	0	0	1
8	0	0	0	0

Block diagram

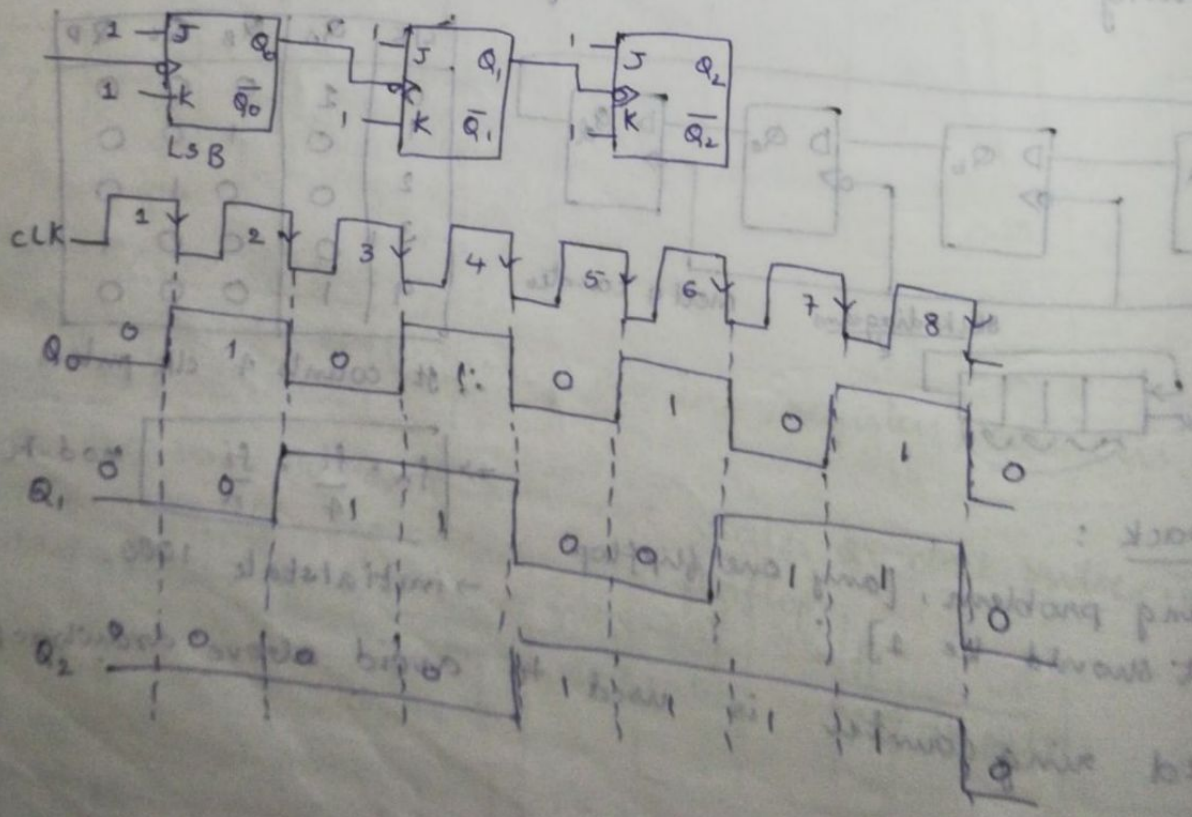


- n-bit Johnson counter requires n flip flops
- n-bit Johnson counter counts 2n clk pulses

Ripple Counter / Asynchronous:

- n-bit ripple counter requires n no. of flipflops
- n-bit ripple counter counts 2^n clock pulses
- Output frequency $f_o = \frac{f_i}{2^n}$
- It is also called as binary sequency counter

3-bit



clk	Q_2	Q_1	Q_0
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

MOD-8

n-bit Ripple $\Rightarrow$ MOD -2^n

$f_o = \frac{f_i}{2^n}$

Decade (MOD-10)

(1) Input frequency of 4-bit ring counter is 16 KHz. then the output frequency is ___

$f_o = \frac{f_i}{4} = \frac{16}{4} = 4\,KHz.$

[no. of clk pulses]

(2) Input frequency of 4-bit twisted ring counter is 16 KHz. Then the f_o is ___

$f_o = \frac{f_i}{(2n)} = \frac{16}{8} = 2\,kHz.$

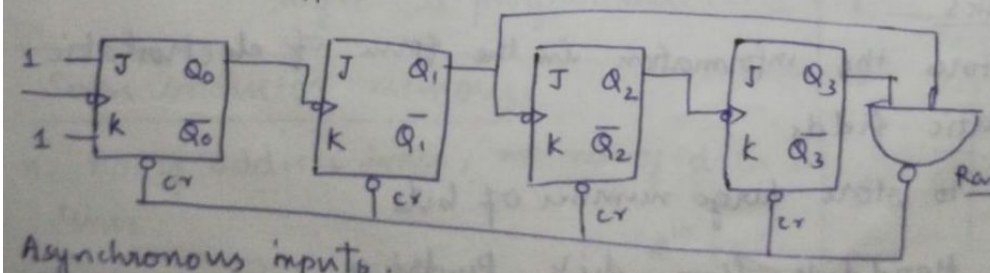
(3). f_i of 4-bit ripple counter is 16 KHz, f_o = ___

$f_o = \frac{f_i}{2^n} = \frac{16}{2^4} = \frac{16}{16} = 1\,KHz.$

(4) **Decade (Mod-10) Counter** (Ripple) : It is used to count 10 no. of clock pulses. The range is from 0-9. (10 becomes again 0)

$f_o = \frac{f_i}{10}$

10 → $Q_3 Q_2 Q_1 Q_0$ = 1010



Range: 0000, 0001, 0010, 0011, 0100, 0101, 0110, 0111, 1000, 1001, 0000

Asynchronous inputs:

- These inputs are used to set or reset the flipflops.
- These inputs doesn't depends on the clock pulses
- Those are clear and preset (Cr, Pr)

if $P_r = 0 \Rightarrow Q_{n+1} = 1$

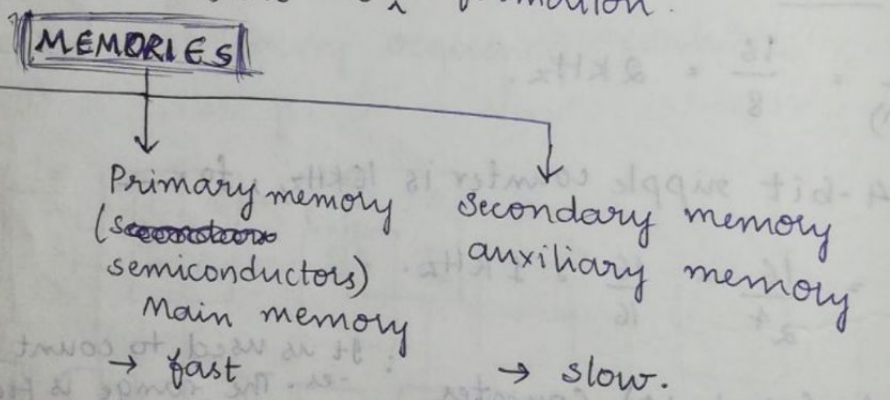
if $C_r = 0 \Rightarrow Q_{n+1} = 0$.

- the default status of clear and preset is 1, 1

	Name of the counter	No. of FFs	No. of clk pulses	output frequency f_o
1)	n-bit ring counter	n	n	$\frac{f_i}{n}$
2)	n-bit twisted ring counter/ Johnson counter	n	2n	$f_i/2n$
3)	n-bit Ripple counter	n	2^n	$f_i/2^n$
4)	Decade counter	4	10	$\frac{f_i}{10}$

Memories

- Memories are used to store the digital information.



Secondary Memories

- These memories stores the information in the form of electrostatic and magnetostatic fields
- These are used to store large number of bits
- Ex: CD, DVD, Harddisk, floppy disk, Pendrives.

Primary / Semiconductor Memories

Are of 2 types: RAM, ROM.

RAM - Random Access Memory

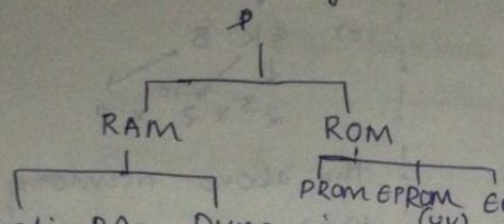
RAM is used to perform read & write operations

ROM - read only memory

→ ROM performs only read operations

→ RAM is volatile memory → ROM is non volatile

→ RAM acts as data memory → ROM acts as a program memory



→ PROM - programmable Read only Memory [erasing not possible]

→ EPROM - erasable PROM [can erase using UV signals]

→ EEPROM [electrically Erasable PROM] [can erase using electrical signal

→ SRAM is designed using FF's & bipolar devices

→ SRAM occupies more space

→ SRAM are fastest than DRAM

→ Power consumption is very high in SRAM

→ To store 1-bit 6 transistors are required

→ DRAM is designed using FET and capacitors

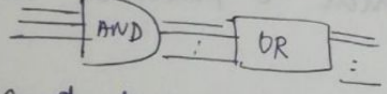
→ DRAM occupies less space.

→ DRAM is slow compared to SRAM.

→ In dynamic RAM power consumption less

→ DRAM requires 2 transistors.

PLD - programmable logic devices



PLA : And logic is programmable
OR logic is programmable

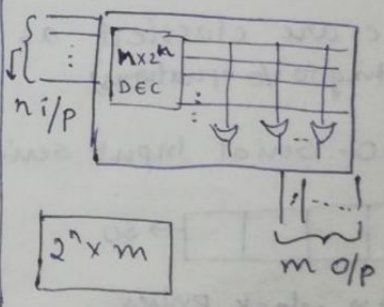
PAL : And logic is programmable
OR logic is fixed

PROM : And logic is fixed. OR logic is programmable.

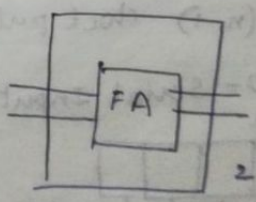
Standard PROM structure

n i/p's and m o/p's

PROM structure is $2^n \times m$



Ex:



n → 3 i/p 2 → o/p.

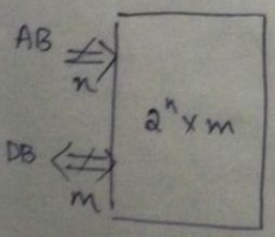
=> 3 = 2

→ 3x8 decoder

→ 2 or gates required.

Semiconductor memories

n. no. of address lines, m no. of data lines



$2^{10} \rightarrow$ KB.

$2^{20} \rightarrow$ MB

1) 1 static RAM is connected with 11 address lines and 8 data lines then memory capacity-starting address and ending address —

n = 11 and data lines m = 8

$2^n \times 8$

$2^1 \times 2^{10} \times 8$

memory capacity is $2K \times 8 = 2KB$

starting address => 000|000 0|0000

ending address => 111|111 1|1111

[address should be represent using hexa-decimal notation]

starting address : 0000 H

ending address : 7FF H

2) No. of address lines and data lines of 8KB SRAM is ____

=> 8K B

$2^3 \times 2^{10} \times 8$

the above memory requires 10+3 = 13 address lines and 8 data lines

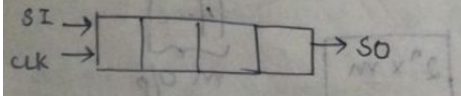
starting address : 0|000|000|0000 = 000H

ending address : 1|1111|1111|1111| = 1FFFH

Registers :

- n-bit register requires n no. of flipflops
- These are used to convert to serial to parallel and parallel to serial.
- These acts as shift registers
- These are classied as : 1) SISO 3) PISO (according to i/o operations) 2) SIPO 4) PIPO

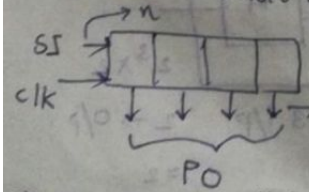
1) SISO - Serial Input serial O/P



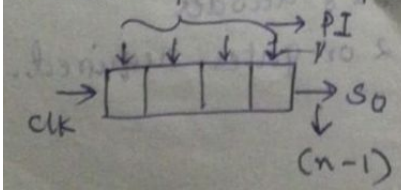
SI => n -clock pulses

SO => (n-1) clock pulses

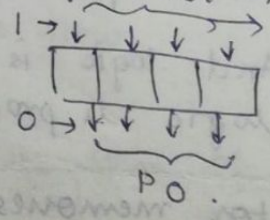
2) SIPO - Serial Input Parallel O/P



3) PISO - parallel Input serial O/P



4) PIPO - parallel I/P Parallel O/P



LOGIC FAMILIES.