4.11.19.

PAGE NO. :

Ch - 4
Number Systems

Numeral Systems : A number system is a code which uses symbols to represent quantities. Four commonly used number systems are The Decimal System, The Binary System, The Hexadecimal (HEX) system and The Octal (OCT) system.

Decimal System (also known as base-10 or Radix-10 system) uses Ten symbols
0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Binary System (also known as base-2 or Radix-2 system) uses two symbols - 0 & 1.

Octal System (also known as base-8 or Radix-8 system) uses eight symbols.
0, 1, 2, 3, 4, 5, 6, 7

Hexadecimal System (also known as base-16 or Radix-16) uses sixteen symbols:
0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

All number systems use the concept of "Place Value" in coding quantities

Basic Terminology used in Binary System:

1) Bit ⇒ Binary Digit (a single Binary number)
2) Byte ⇒ An 8-bit group of data
3) Nibble ⇒ A half-byte (A 4-bit group of data)

Comparison Table for Various Number Systems

Decimal	Octal	Hexadecimal	Binary
0	0	0	0
1	1	1	1
2	2	2	10
3	3	3	11
4	4	4	100
5	5	5	101
6	6	6	110
7	7	7	111
8	10	8	1000
9	11	9	1001
10	12	A	1010
11	13	B	1011
12	14	C	1100
13	15	D	1101
14	16	E	1110
15	17	F	1111
16	20	10	10000
31	37	1F	11111
32	40	20	100000

Procedure for converting a Decimal Number to Binary

g- Convert $(137)_{10}$ to binary

$137 \div 2 = 68 +$ Remainder of 1

$68 \div 2 = 34 +$ Remainder of 0

$34 \div 2 = 17 + R.0$

$17 \div 2 = 8 + R.1$

$8 \div 2 = 4 + R.0$

$4 \div 2 = 2 + R.0.$

$2 \div 2 = 1 + R.0$

$1 \div 2 = 0 + R.1.$

Ans = $(10001001)_2$

Conversion from Decimal to HEX

g. Convert $(47)_{10}$ to HEX

$47 \div 16 = 2 +$ Remainder is 15 (F)

$2 \div 16 = 0 +$ Remainder of 2 (2)

Ans. $(2F)_{16}$

Binary to HEX conversion.

Split the given number into groups of four.

g. $\underline{1110}\ \underline{1010}$

$(EA)_{16}$

Q. Convert binary number 110011 into its decimal equivalent.

Ans. $(110011)_2 = 2^5 + 2^4 + 0 + 0 + 2^1 + 2^0$

$2^5 \; 2^4 \; 2^3 \; 2^2 \; 2^1 \; 2^0$

$= 32 + 16 + 2 + 1$

$= (51)_{10}$

Q. Convert decimal number 39 to its binary equivalent

Ans.

$39/2 = 19 + 1$

$19/2 = 9 + 1$

$9/2 = 4 + 1$

$4/2 = 2 + 0$

$2/2 = 1 + 0$

$1/2 = 0 + 1$

$(100111)_2$

Q. Convert HEX number B6 to binary and decimal equivalents

Ans. B6 - $(10110110)_2$

$(B6)_{16} = (11 \times 16^1 + 6 \times 16^0)_{10}$

$= (182)_{10}$

Q. Convert binary number \$11111110 to HEX and decimal equivalents.

Ans. $(11111110)_2 = 2^7 + 2^6 + 2^5 + 2^4 + 2^3 + 2^2 + 2^1 + 0$

$= 128 + 64 + 32 + 16 + 8 + 4 + 2$

$= (254)_{10}$

$(11111110)_2$ = 1111 1110 $\Rightarrow$ $(FE)_{16}$
F E

8. Convert HEX number 2DB to decimal and binary equivalents.

Ans. $(2DB)_{16} = (2\times16^2 + 13\times16^1 + 11\times16^0)$
$= 2\times256 + 13\times16 + 11\times1$
$= 512 + 208 + 11$
$= (731)_{10}$

$(2DB)_{16}$ = 2 D B $\Rightarrow$ $(001011011011)_2$
0010 1101 1011

9. Convert Octal number 67 to binary

Ans. $(67)_8 = (6\times8^1 + 7\times8^0)_{10}$
$= 48 + 7 = (55)_{10}$
~~$(55)_{10} = (5\times2^1 \; 55)$~~
6 7
110 111 $\Rightarrow$ $(110111)_2$

10. Convert decimal number 498 to octal equivalent.

Ans. 498/8 = 62 + R2
62/8 = 7 + R6
7/8 = 0 + R7
$\Rightarrow (762)_8$

PAGE NO. :

Q. Convert binary number 10000 1101 into octal equivalent

Ans. 100 001 101
4 1 5 → $(415)_8$

Q. Convert decimal number 0.6875 to binary

Ans. 0.6875 × 2 = Integer 1 + Fraction 0.3750
0.3750 × 2 = Integer 0 + Fraction 0.7500
0.7500 × 2 = Integer 1 + Fraction 0.5000
0.5000 × 2 = Integer 1 + Fraction 0.0
⇒ $(0.1011)_2$
$(0.1011)_2 = (1\times2^{-1} + 0 + 1\times2^{-3} + 1\times2^{-4})_{10}$
$= \left[\frac{1}{2} + \frac{1}{8} + \frac{1}{16}\right]_{10}$

Q. Convert decimal number 117.23 to octal.

Ans. 117/8 = 14 + R5
14/8 = 1 + R6
1/8 = 0 + R1
$(165)_8 = (117)_{10}$
0.23 × 8 = Integer 1 + Fraction 0.84
0.84 × 8 = Integer 6 + Fraction 0.72
0.72 × 8 = Integer 5 + F. 0.76
0.76 × 8 = In. 6 + Frac. 0.08
Assuming it ends at 0.08
$(0.1656)_8 = (0.23)_{10}$
⇒ $(117.23)_{10} = (165.1656)_8$

DATE: / /

PAGE NO.:

Q. Convert decimal number 117.23 to its binary equivalent.

Ans.

117/2 = 58 + R1
58/2 = 29 + R0
29/2 = 14 + R1
14/2 = 7 + R0
7/2 = 3 + R1
3/2 = 1 + R1
1/2 = 0 + R1

$(1110101)_2$

0.23×2 = I.0 + F.0.46
0.46×2 = I 0 + F 0.92
0.92×2 = I 1 + F 0.84
0.84×2 = I 1 + F 0,68
0.68×2 = I 1 + F. 0.36
0.36×2 = I 0 + F. 0.72
0.72×2 = I 1 + F 0.44

Addition of Two Binary Numbers

The addition of two single digit binary numbers, A and B is done according to the values contained in the Table shown below.

A	B	"Sum" Bit	"carry" Bit
0	0	0	None
0	1	1	None
1	0	1	None
1	1	0	1

Carrying - When the result of an addition exceeds the maximum possible value of a digit, the procedure is to "carry" the "excess" amount divided by Yadix to the left, adding it to the next positional value.

eg-

```
    1 1   ← Carry Bit
A = 0111
B = 0010
+ = 1001
```

```
  + + + + + + + ← Carry
  0 1 1 1 1 1 1 1  (63)
+ 0 1 1 1 1 0 1    (61)
  1 1 1 0 1 0 0    (124)
```

Subtraction of Binary Numbers : The rules for subtraction of one one-digit number B from another one-digit binary number A are given below -

A	B	"Sub" Bit	"Borrow" Bit
0	0	0	None
0	1	1	1
1	0	1	None
1	1	0	None

Borrowing - When the result of a subtraction is less than 0, the procedure is to borrow the deficit divided by Yadix from the left, subtracting it from the next positional value.

eg-

```
A = 01011
B = 00010
- = 01001
```

```
  1 1   ← Borrow
  1 1 0 1
- 0 1 1 1
  0 1 1 0
```

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

Complements of a number

1) Radix's complement → (For Binary system 2's comp.)
   → (For decimal system 10's comp.)
2) Diminished Radix's Complement (Radix - 1 Complement)
   → (For decimal system, 9's complement)
   → (For Binary system, 1's complement)

How to Calculate 9's complement of a Decimal Number

Step 1 : Count the number of digits in the given number and form a new number, called Maximum Number, by replacing each digit of the given number by radix - 1 = 9

Step 2 : Subtract the given number from the max. number produced at the end of STEP-1. This will give the required answer.

Q. Calculate 9's complement of the decimal number 546700.

Sol.
Step 1 - 999999
Step 2 - -546700

         453299

Q. Calculate 10's complement of the decimal number 546700

Ans. Add 1 to 9's complement of 546700
→ 453300

Most Significant Bit (MSB) and Least Significant Bit (LSB)

Eg - (1)101.11(1)
MSB LSB

Procedure for obtaining 1's complement of a Binary Number

Step 1 - Replace each digit in the given number by 1 (=Radix -1). This gives the Maximum no.

Step 2 - Subtract the given number from the Maximum no. to get the final number

Procedure for obtaining 2's complement of a Binary Number -

Obtain 1's complement and add 1 to LSB

Q. Calculate 1's complement of 1011000.

Ans. Step 1 : 1 1 1 1 1 1 1

Step 2 : − 1 0 1 1 0 0 0

0 1 0 0 1 1 1

Q. Obtain 2's complement of 1011000

Ans. 1011000 → 0100111

0100111 + 1

0101000

Q. Obtain 2's complement of 10000.001

Ans. Step 1 - 1's complement = 01111.110

2's complement = + 1

01111.111

Subtraction using Radix Complement

1) Let us consider the operation A-B where A and B are given numbers.
2) Take the Radix complement of B. Let us call it, say $\bar{B}$.
3) Obtain $A+\bar{B}$
4) Is CARRY present in $A+\bar{B}$?

Yes → Discard carry to get ans.

No - Take Radix comp. of $A+\bar{B}$ and attach minus sign.

Q. $A=(72532)_{10}$ $B=(03250)_{10}$ obtain A-B

Ans. 10's comp. of B is $\hat{B} = (96750)_{10}$

$A+\hat{B} = 72532 + 96750 = (1)69282$

↳ carry

Q. $A=(03250)_{10}$ $B=(72532)_{10}$

Ans. 10's comp. of B - 99999 − 72532 = 27467 + 1

$\hat{B} = (27468)_{10}$

$A+\hat{B} = 03250 + 27468 = (30718)_{10}$

→ 99999 − 30718 = 69281 + 1 = −(69282)

How to Store Sign?

(0)1011

↳ first bit is used to store sign

↳ 1 here means −ve sign

DATE : / /
PAGE NO. :

Q. $X = (1010100)_2$ $Y = (1000011)_2$
obtain $X-Y$ and $Y-X$ using 2's complement

Ans. a) $\dot{Y}$ - 01100111 → 01111100
+1
01111101

$X + \dot{Y} = 1010100$
01111101
(1)0010001 ⇒ 0010001

b) $\dot{X} = 0101100$

$Y + \dot{X} = 1000011$
+0101100
1101111

1's - 0010000
+1
-0010001

Subtraction using Diminished Radix's complement

Q. Given $X = (1010100)_2$, $Y = (1000011)_2$, calculate $X-Y$ and $Y-X$ using 1's complement

Ans. (a) $X-Y$:

Step 1 - Calculate 1's complement of Y and call it say $\bar{Y}$.

$\dot{Y} = 0111100$

Step 2 - Calculate $X + \dot{Y}$

Step 3 - Is carry present in the result obtained at the end of Step 2

Yes	No
↓	↓
Add it to the answer to obtain the final answer	Obtain 1's complement and attach minus sign to obtain the final answer

Ans.

$$X = 1010100$$
$$+ Y = 0111100$$
$$\overline{\quad 10010000 \quad}$$
$$\Rightarrow \quad 0010001$$

(B) Y - X

Step 1 - X = 0101011

Step 2 - Y + X = 1101110

Step 3 - Ans ⇒ -0010001

Binary Codes

These are methods of converting a given decimal number to a binary number. Besides the representation based on successive division by 2 and noting remainders, there are several other methods to achieve the goal. These are known as Binary codes.

DATE : / /
PAGE NO. :

1) BCD 8421 Code : BCD → Binary Coded Decimal
8 (= 2^3), 4 (= 2^2), 2 (= 2^1), 1 (= 2^0) are the weights used for various "Place values".

Q. Express the Decimal Number 926 as BCD code.

Ans. Express each decimal digit using its binary representation and combine the results.

9 2 6

1001 0010 0110 ⇒ 926 - 100100100110

Q. Express BCD Code 00011000 0111001 as a decimal number.

Ans. Divide the given number into groups of 4-bits (nibbles) and express each nibble as its decimal equivalent

→ 1871

2) BCD 4221 is also used
3) BCD 84, -2, -1 is also used
4) Excess - 3 BCD Code - Also known as XS-3
In this code we add 3 to the given number and then take BCD Equivalent.

Q. Convert $(4)_{10}$ into XS-3

Ans. $(7)_{10}$ = 0111

American Standard Code for Information Interchange (ASCII)

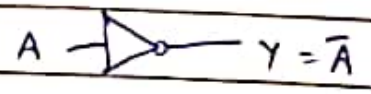
This is a 7-bit code which can be used to represent numbers, letters, punctuation mark and special characters.

Gray Code : As we go from one quantity to next, there is a change in only 1 bit in the representation.

Logic Gates -

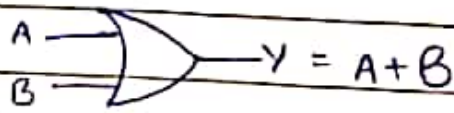
1) Basic building blocks of a digital system.
2) Output is only one, but input can be more than one.

NOT Gate



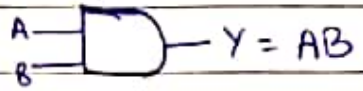
A	Y
1	0
0	1

OR Gate



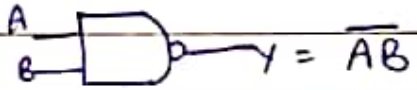
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	1

AND Gate



A	B	Y	$\bar{Y} = \bar{AB}$
0	0	0	1
0	1	0	1
1	0	0	1
1	1	1	0

NAND Gate



Boolean Algebra

Defines a set of rules (postulates + theorems) using which a binary expression (also known as logical expression) can be simplified without changing its functionality.

eg - $F(A, B, C) = \bar{A}B + BC + ABC$

Boolean Function — Boolean Expression (Binary Expression)

$= \bar{A}B + BC$ (Simplified version)

A boolean function can be described by an algebraic expression called Boolean expression.

The Boolean expression consists of binary expression (eg. AB), constants (0 and 1) and logic operation symbols (eg +, −, ., etc)

The only possible states for Boolean expression are 0 and 1.

Boolean functions can be represented using Truth Table, too.

Boolean Algebra Postulates

Postulate is a law or rule which is scientifically accepted to be true.

Postulate 1 - The Boolean structure is closed for operations '0' and '1'. That is, operations '.' and '+' will always result in 1 or 0.

Postulate 2 - Element "0" is unity element for "+" and element "1" is unity element for "." That is
a) $0 + x = x$
b) $x + 0 = x$
c) $1 \cdot x = x$
d) $x \cdot 1 = x$

Postulate 3 - The structure is commutative for "." and "+" operations. That is
$x + y = y + x$
$x \cdot y = y \cdot x$

Postulate 4 - The structure is distributive for "." and "+" operations. That is
$x \cdot (y + z) = x \cdot y + x \cdot z$
and $x + (y \cdot z) = (x + y) \cdot (x + z)$

Postulate 5 - For every element x belonging to Boolean structure there exists a complement $\bar{x}$ such that

$$x + \bar{x} = 1$$
$$x \cdot \bar{x} = 0$$

Postulate 6 - There exists at least two elements x and y in a Boolean structure such that $x \neq y$

Basic Theorems of Boolean Algebra:

Each theorem can be proved using either postulates or Truth table.

Theorem 1 : a) $x + x = x$
b) $x \cdot x = x$

Theorem 2 : a) $x + 1 = 1$
b) $x \cdot 0 = 0$

Theorem 3 : $(\bar{x}) = x$

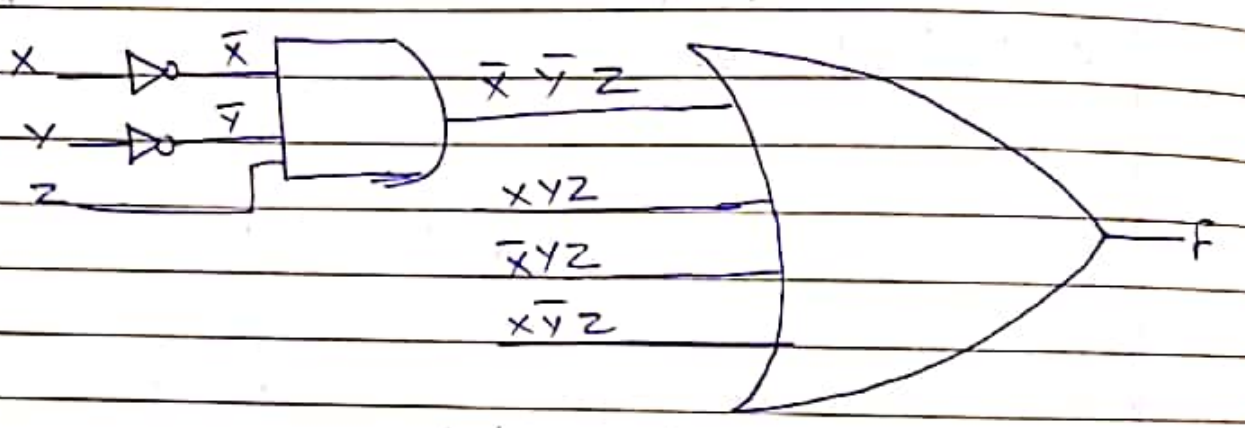
Theorem 4 : a) $x + (y + z) = (x + y) + z$
b) $x \cdot (yz) = (x \cdot y) \cdot z$
} Associative Property

Theorem 5 : De Morgan's Theorem
a) $\overline{(x+y)} = \bar{x} \cdot \bar{y}$
b) $\overline{(x \cdot y)} = \bar{x} + \bar{y}$

Theorem 6 : a) $x + xy = x$ } Absorption
b) $x.(x+y) = x$ } Property

Q-1. Implement
$F = \bar{x}\bar{y}z + xyz + \bar{x}yz + x\bar{y}z$
using logic gates

Ans-1.



Q-2. Using basic theorems and postulates simplify $F = \bar{x}\bar{y}z + xyz + \bar{x}yz + x\bar{y}z$

Ans. $= z$

Q-3. Develop a Truth Table for $F = \bar{x}\bar{y}z$

Ans.

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	1	1	0

DATE: / /
PAGE NO.:

Complement of a Binary Function -

Can be obtained using De Morgan's Theorem

eg- $\overline{(A+B+C)}$ Define $B+C = x$

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

2-4. Minimize the number of logic gates required to implement $F = \bar{x}\bar{y}z + xyz + \bar{x}yz + x\bar{y}z$

Minterms and Maxterms

In a 2-variable binary system with x and y as the two variables involved, all possible combinations of x and y using the AND operation are known as MINTERMS and all possible combinations of x and y using the OR operation are known as MAXTERMS.

Minterms -

$m_0 \rightarrow \bar{x}\bar{y}$ $m_1 \rightarrow \bar{x}y$ $m_2 \rightarrow x\bar{y}$ $m_3 \rightarrow xy$

Maxterms -

$M_0 = x+y$ $M_1 = x+\bar{y}$ $M_2 = \bar{x}+y$ $M_3 = \bar{x}+\bar{y}$

Canonical Terms

A boolean expression can be expressed as sum of MINTERMS or Product of MAXTERMS

In either representation, all terms must contain all variables or their complements.

eg- $F(x,y) = xy + \bar{x}\bar{y}$

$F(x,y) = (x+y)(\bar{x}+y)$

Standard Forms

A boolean expression can also be expressed as sum of PRODUCTS (SoP) or PRODUCT of SUMS (PoS). Terms need not contain all variables or their complements.

eg- $F = x + yz$

$F = (x+z)(y+z)$

Q-1. Simplify $Y = \bar{A}\bar{B} + A\bar{B} + AB$ using K-Map.

Ans- ~~Problem~~ Procedure-

1) Start with sum of MINTERMS representation
2) For each term, mark "1" in the appropriate cell.
3) Put "0" elsewhere.
4) Create as many loops as possible, ~~and~~ containing "1" in 2, 4, 8 ... number
5) A "1" can be in more than one loop.
6) Make sure no "1" remains un-looped.
7) For each loop, write a term by removing changing variables.
8) OR all the terms obtained in 7) to get the simplified expression.

$Y = A + B$

DATE : / /

PAGE NO. :

	$\bar{B}$	B
$\bar{A}$	0	1 → L1
A	1	1

L2

Q-2. Simplify $F = A\bar{B}\bar{C} + \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + AB\bar{C}$ using K-Map.

Ans-2.

	$\bar{C}$	C
$\bar{A}\bar{B}$	1	1 → L1
$\bar{A}B$	0	0
$A\bar{B}$	1	0
AB	1	0

L2

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

Q-3. $F = A\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}CD + A\bar{B}\bar{C}D + \bar{A}BCD$

Ans-3.

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$		1	1	
$\bar{A}B$		1	1	
AB				
$A\bar{B}$	1	1		

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

Q-4. $Y = \bar{A}B\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + A\bar{B}\bar{C}\bar{D} + A\bar{B}\bar{C}D$

Ans-4.

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	1		
$\bar{A}B$				
AB				
$A\bar{B}$	1	1		

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

Q-5. $Y = \bar{A}B\bar{C}\bar{D} + A\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + A\bar{B}C\bar{D}$

Ans 5

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1			1
$\bar{A}B$				
AB				
$A\bar{B}$	1			1

$Y = \bar{B}\bar{D}$

"Don't Care" entries (Normally represented by x)

eg -

	$\bar{C}\bar{D}$	$\bar{C}D$	CD	$C\bar{D}$
$\bar{A}\bar{B}$	1	x	x	1
$\bar{A}B$	x	0	0	x
AB	0	0	0	0
$A\bar{B}$	1	x	0	0

Ans. $Y = \underbrace{\bar{A}\bar{B}\bar{C}\bar{D}}_{m_0} + \underbrace{A\bar{B}\bar{C}\bar{D}}_{m_8} + \underbrace{\bar{A}\bar{B}C\bar{D}}_{m_2} + \underbrace{A\bar{B}C\bar{D}}_{m_{10}}$

$F = \Sigma(0, 2, 8, 10)$

It can be shown that

$\bar{F} = \Sigma(1, 3, 4, 5, 6, 7, 9)$

Q - 6. Simplify the function whose K-Map is shown below:

	$\bar{C}$	C
$\bar{A}\bar{B}$	0	0
$\bar{A}B$	1	0
AB	0	1
$A\bar{B}$	0	0

Ans-6. $F = \bar{A}B\bar{C} + ABC$

DATE : / /
PAGE NO. :

Combinational Circuits

Use logic gates to implement various mathematical /logical operations in Digital Systems.
Outputs depend only on the present state of inputs. History of inputs does not matter (no memory!).

1) Adder and Subtractors (Full / Half)
2) Encoders and Decoding
3) Multipliers and Demultiplier

Half - Adder -

Adds two single bit, x and y and produces two outputs, sum (Σ) and carry (C_o).

x	y	Σ	C_o
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

K-Map for Σ

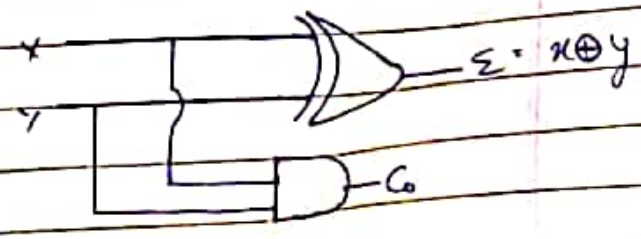
	$\bar{y}$	y
$\bar{x}$	0	1
x	1	0

$\Sigma = x\bar{y} + \bar{x}y = x \oplus y$

K-Map for C_o -

	$\bar{y}$	y
$\bar{x}$	0	0
x	0	1

$C_o = xy$



PAGE NO. :

Full Adder Truth Table

C_{in}	x	y	C_o	Σ
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

K-Map for C_o

	$\bar{C}_{in}$	C_{in}
$\bar{x}\bar{y}$		
$\bar{x}y$		1
xy	1	1
$x\bar{y}$		1

$C_o = xy + yC_{in} + xC_{in}$

$C_o = C_{in}(x \oplus y) + xy$

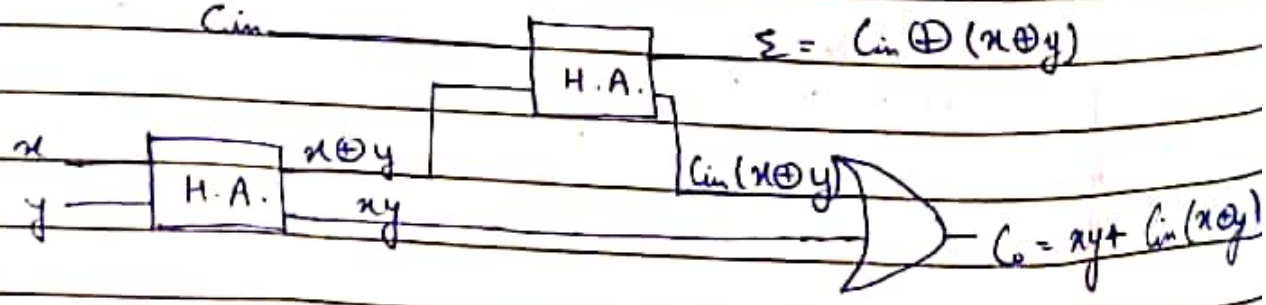
K-Map for Σ

	$\bar{C}_{in}$	C_{in}
$\bar{x}\bar{y}$	0	(1)
$\bar{x}y$	(1)	0
xy	0	(1)
$x\bar{y}$	(1)	0

$\Sigma = C_{in}\bar{x}\bar{y} + \bar{C}_{in}\bar{x}y + C_{in}xy + \bar{C}_{in}x\bar{y}$

$= C_{in} \oplus (x \oplus y)$

Full Adder Implementation using Two Half Adders.



Half Subtractor

x	y	D_o	B_o
0	0	0	0
0	0	1	1
1	0	1	0
1	1	0	0

K Map for D_o

	$\bar{y}$	y
$\bar{x}$	0	1
x	1	0

$D_o = x\bar{y} + \bar{x}y$

$= x \oplus y$

K-Map for B_o

	$\bar{y}$	y
$\bar{x}$	0	1
$\bar{x}$	0	0

$B_o = \bar{x}y$

Full Subtractor

x	y	B_{in}	D_o	B_o
0	0	0	0	0
0	0	1	1	1
0	1	0	1	1
0	1	1	0	1
1	0	0	1	0
1	0	1	0	0
1	1	0	0	0
1	1	1	1	1

K-Map for D_o

	$\bar{B}_{in}$	B_{in}
$\bar{x}\bar{y}$		1
$\bar{x}y$	1	
xy		1
$x\bar{y}$		1

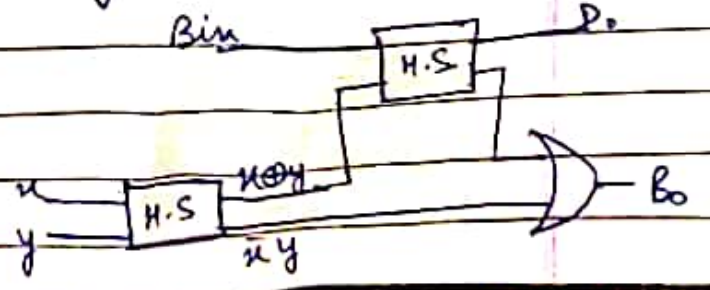
$D_o = \bar{B}_{in}x + \bar{x}yB_{in} + \bar{x}\bar{y}B_{in}$

$= x \oplus y \oplus B_{in}$

K-Map for B_o

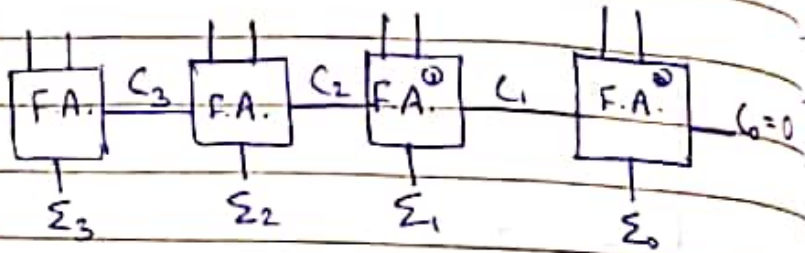
	$\bar{B}_{in}$	B_{in}
$\bar{x}\bar{y}$		1
$\bar{x}y$	1	1
xy		1
$x\bar{y}$		

$B_o = \bar{x}B_{in} + \bar{x}y + yB_{in}$



4-bit Binary Adder

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



Decoder : Maps "n" inputs to "2^n" (or less than "2^n") outputs.

eg-1 - 3 to 8 Decoder Truth Table

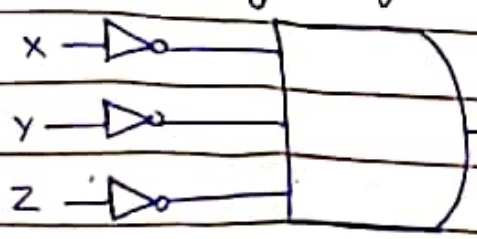
Decimal No.	x	y	z	D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7
0	0	0	0	1	0	0	0	0	0	0	0
1	0	0	1	0	1	0	0	0	0	0	0
2	0	1	0	0	0	1	0	0	0	0	0
3	0	1	1	0	0	0	1	0	0	0	0
4	1	0	0	0	0	0	0	1	0	0	0
5	1	0	1	0	0	0	0	0	1	0	0
6	1	1	0	0	0	0	0	0	0	1	0
7	1	1	1	0	0	0	0	0	0	0	1

x, y, z ⇒ Inputs

$D_0, D_1, \ldots D_7$ ⇒ Outputs (Light Emitting Diode (LED's))

$D_0 = \bar{x}\,\bar{y}\,\bar{z}$

and similarly for D_1, D_2, D_3, — etc.



1 ≡ HIGH

0 ≡ LOW

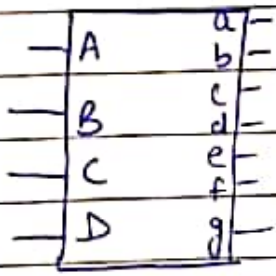
Positive Logic

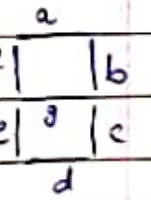
DATE : / /
PAGE NO. :

e.g. 4 to 7 Decoder : Used for driving 7-segment displays in calculator (BCD to 7 Decoder)

D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	1	1	1	1	1	1	0
0	0	0	1	0	1	1	0	0	0	0
0	0	1	0	1	1	0	1	1	0	1
0	0	1	1	1	1	1	1	0	0	1
0	1	0	0	0	1	1	0	0	1	1
0	1	0	1	1	0	1	1	0	1	1
0	1	1	0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	0	0	0	0
1	0	0	0	1	1	1	1	1	1	1
1	0	0	1	1	1	1	1	0	1	1

1 ≡ HIGH
0 ≡ LOW





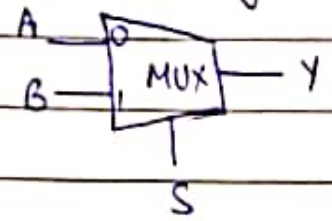
Encoders - do the opposite of what decoders do

Example - 8 to 3 Encoder - The truth table in Example 1 would apply, but roles of inputs and outputs will be reversed.

Priority Encoders - If more than one inputs become simultaneously active, only the input with the highest priority will be processed.

Multiplexer (MUX) -

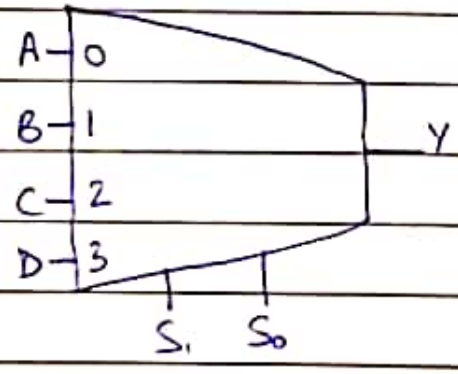
Routes the info from one of the inputs lines to the output line. The choice of input line is made by the "S" line.



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

Generate K-Map for Y and derive the minimized expression for Y and then implement

4 to 1 Multiplexer



$S_1 = 0, S = 0 \quad Y = A$

$S_1 = 0, S_0 = 1 \quad Y = B$

$S_1 = 1, S_0 = 0 \quad Y = C$

$S_1 = 1, S_1 = 1 \quad Y = D$

Demultiplexer (DEMUX)-

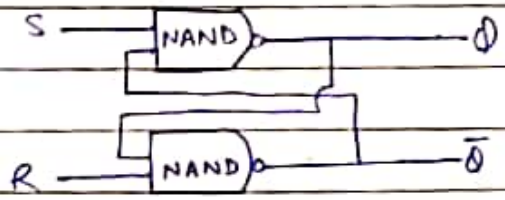
Its functionality is opposite of MUX's functionality

DATE : / /
PAGE NO. :

Sequential Circuits

1) Have "memory" or "storage capacity".
2) Outputs depend not only on the current state of Inputs but also on history of inputs.
3) Feedback paths exist between outputs and inputs.

SR (Set-Reset) Latch



Q_{t-1} & $\bar{Q}_{t-1}$ → previous outputs

Q_t & $\bar{Q}_t$ → current outputs

S	R	Q_{t-1}	$\bar{Q}_{t-1}$	Q_t	$\bar{Q}_t$	
0	1	0	1	1	0	
0	1	1	0	1	0	
1	0	0	1	0	1	
1	0	1	0	0	1	
1	1	0	1	0	1	→ Hold State
1	1	1	0	1	0	
0	0	← FORBIDDEN →				

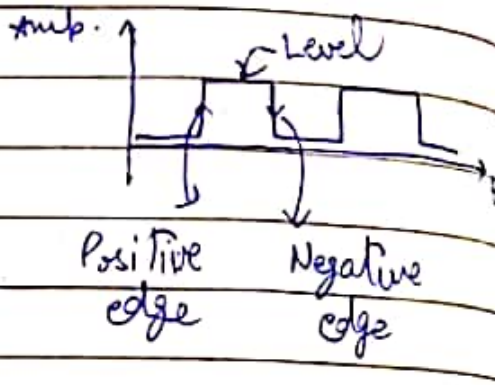
Problems with SR latch

1) S = 1 does not always produce Q = 1
2) Inputs S and R may change in an "uncontrolled" fashion there by making the output unpredictable, especially if inputs are changing fast.

3) Forbidden states of inputs.

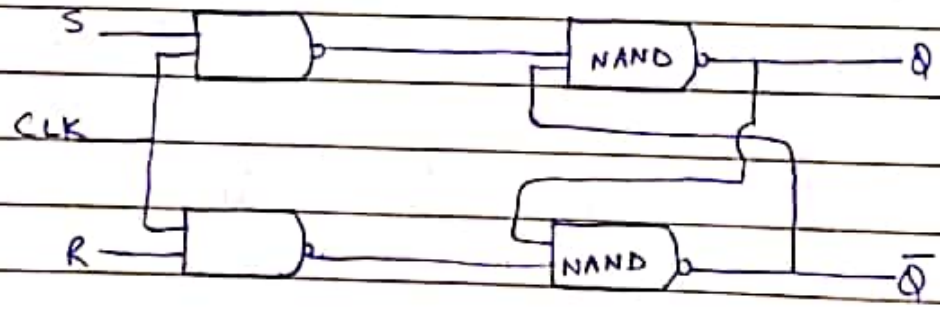
Clock Signal (CLK) -

There are 2 possibilities for triggering -



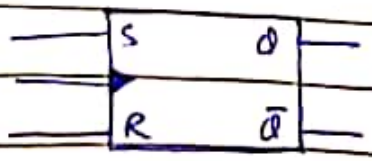
1) Edge - Triggered Clock
2) Level - Triggered Clock

SR Flip flop



CLK	S	R	Q_t	$Q_t\bar{}$
0	x	x	Unchanged	
1	0	0	Unchanged	
1	0	1	0	1
1	1	0	1	0
1	1	1	← FORBIDDEN →	

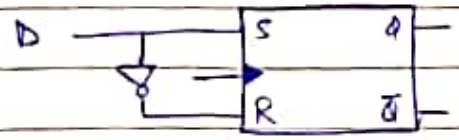
Symbol for SR flip flop



→ Positive Edge Triggered Clk
→○ Negative Edge Triggered Clk

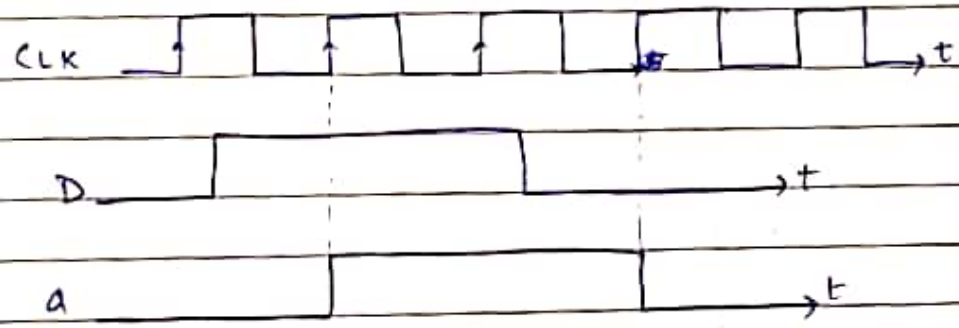
DATE : / /
PAGE NO. :

Delayed (D) Flip Flop -



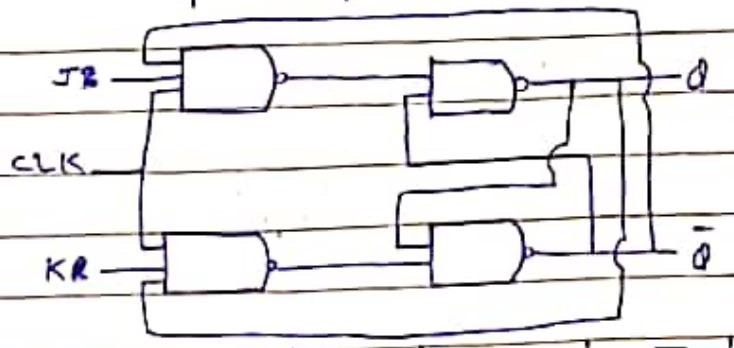
CLK	D	Q_t	$\bar{Q}_t$
0	X	Unchanged	
1	0	0	1
1	1	1	0

Timing Diagram



Output is a delayed version of the input

J-K Flip Flop

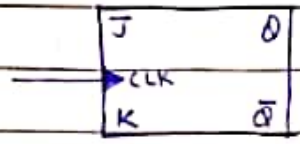


CLK	J	K	Q_t	$\bar{Q}_t$
0	X	X	Unchanged	
1	0	0	Q_{t-1}	$\bar{Q}_{t-1}$
1	1	0	1	0
1	0	1	0	1
1	1	1	$\bar{Q}_{t-1}$	Q_{t-1}

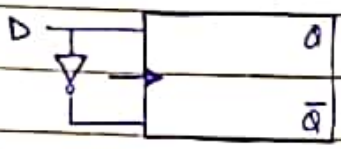
Output toggle every time S=1, R=1

PAGE NO. :

Symbol for JK Flip Flop.

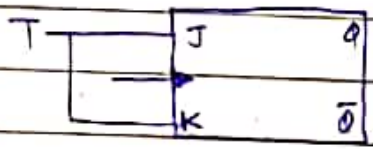


D Flip Flop can be made using JK Flip Flop:



CLK	J	K	Q_t	$\bar{Q}_t$
0	X	X	Unchanged	
1	1	0	1	0
1	0	1	0	1

T (Toggle) Flip Flop



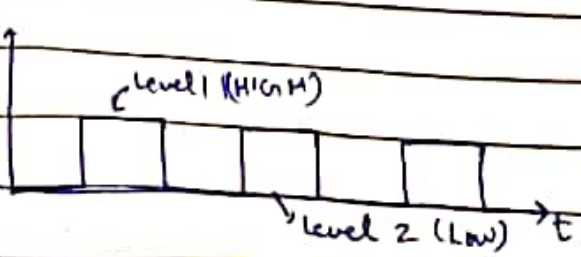
CLK	J	K	Q_t	$\bar{Q}_t$
0	X	X	HOLD	
1	0	0	HOLD	
1	1	1	$\bar{Q}_{t-1}$	Q_{t-1}

Flip-Flops

Set-Reset (SR) — JK — Delayed (D) — Toggle (T)

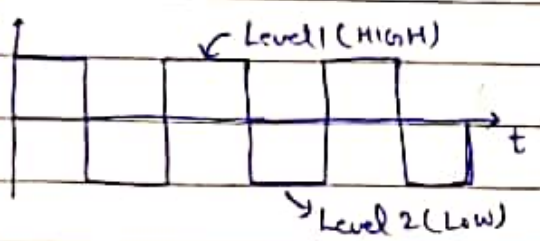
Clock Signals

① Unipolar Pulse



DATE : / /
PAGE NO. :

⑥ Bipolar Pulse



Positive Edge or Rising Edge
LOW to HIGH Transition

Negative Edge or Falling Edge
HIGH to LOW Transition

Leading Edge or Front Edge
(First Edge of the Pulse

Trading Edge or Back Edge
(Second Edge of the Pulse)

Level Triggering
Allows inputs to be processed when the clock pulse is on a particular level.

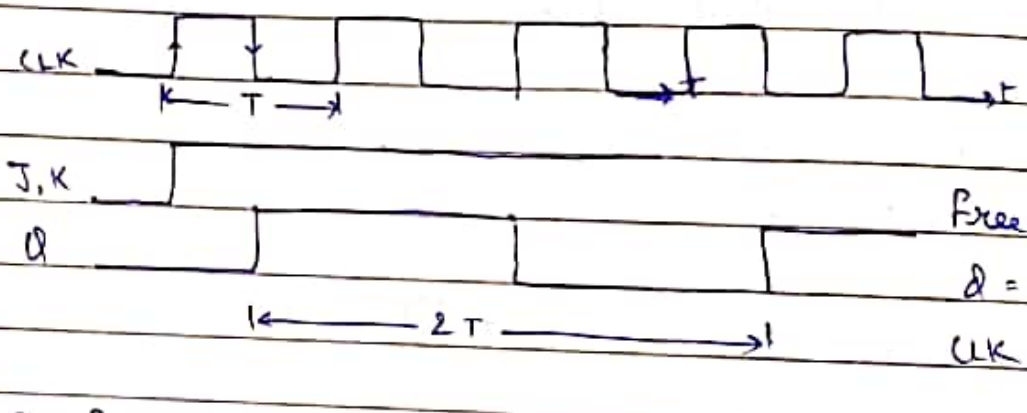
Edge Triggering
Allows inputs to be processed at the positive edge or the negative edge of the clock signal

PAGE NO. :

JK Flip Flop

CLK	J	K	Q_t	$\overline{Q_t}$
0	x	x	Unchanged	
↓	0	0	Q_{t-1}	$\overline{Q_{t-1}}$
↓	1	0	1	0
↓	0	1	0	1
↓	1	1	$\overline{Q_{t-1}}$	Q_{t-1}

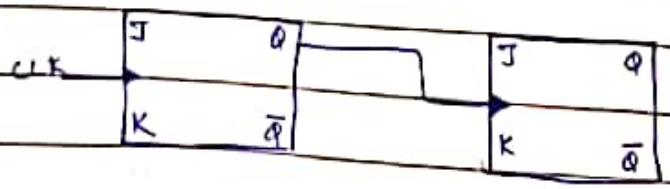
J = K = 1 Input State is useful in designing frequency divider circuits



A Single JK FF gives Frequency Division by Two

Cascading two JKFF s will give freq. division by FOUR, Cascading three JKFF will give freq. division by EIGHT and so on.

Cascading Two JKFF's

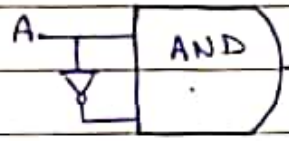


The final Q output freq. = 1/4 Input CLK Freq.

DATE : / /
PAGE NO. :

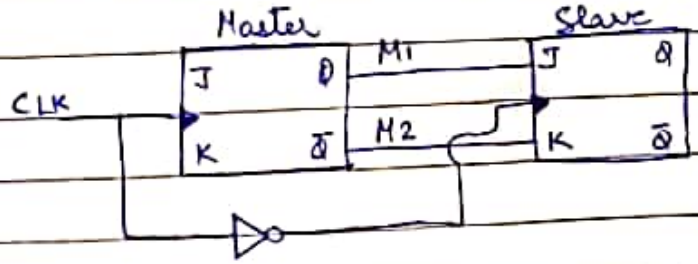
Racing

Two or more INPUT signals "racing" to have their effect on output.

eg. 

Due to racing, output becomes unpredictable and oscillations are seen.

Master - Slave Configuration using two JKFF's



How racing is avoided -
We have sufficient time allotted between two "transfers" of data. The complete transfer occurs only after the clock has gone through a positive-gain edge and the fall during negative-going edge