MODULE-3

Boolean Algebra and Logic Circuits:

Binary Numbers.
Number Base Conversion
Octal and Hexa decimal numbers.
Complements.
Basic definitions.
Axiomatic Definition of Boolean Algebra.
Basic Theorems and properties of Boolean Algebra.
Boolean Functions.
Canonical and Standard forms.
Other Logic Operations.
Digital logic Gates.

Combinational Logic:

Introduction.
Design procedure.
Adders - Half adder, Full adder.

Boolean Algebra and Logic Circuits:

⇒ Binary Numbers :

→ The decimal number system is said to be of base or radix, 10 because it uses 10 digits and the coefficients are multiplied by powers of 10 (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)

eg : $(7392)_{10}$

$= 7 \times 10^3 + 3 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$

(↑ Co-efficients)

→ Binary number is a number expressed in the base 2 numerical system or binary numeral system.

- The coefficients has only 2 values 0 and 1
- Each coefficient ~~multiplied~~ is multiplied by 2^i

eg : $(11010.11)_2 = (\quad)_{10}$?

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

Ans = $(26.75)_{10}$

→ In general, a number expressed in base r system has co-efficients multiplied by powers of r.

$$a_n \cdot r^n + a_{n-1} r^{n-1} + \ldots a_2 r^2 + a_1 \cdot r + a_0 + a_{-1} r^{-1} + a_{-2} r^{-2}$$

$$+ \ldots a_{-m} r^{-m}.$$

→ Base 5 number uses co-efficients of 0, 1, 2, 3, 4.

$(4021.2)_5 = 4 \times 5^3 + 0 \times 5^2 + 2 \times 5^1 + 1 \times 5^0 + 2 \times 5^{-1}$

$= 375 + 10 + 1 + 0.4$ (500)

Ans = $(511.4)_{10}$

→ Base 16 number uses co-efficient of (0, 1, 2 ... 9, A, B, C, D, E, F, G)

$(B65F)_{16} = 11 \times 16^3 + 6 \times 16^2 + 5 \times 16^1 + F \times 16^0$ (15)

$= 45,056 + 1536 + 80 + 15$

$= (46687)_{10}$

Low to high → read high to low - direction in fraction it differ

⇒ Number Base Conversions:

Examples

1. Convert decimal 41 to binary

$\frac{41}{2} = 20 - 1$ (MSB)

$\frac{20}{2} = 10 - 0$

$\frac{10}{2} = 5 - 0$

$\frac{5}{2} = 2 - 1$

$\frac{2}{2} = 1 - 0$

(LSB)

$(41)_{10} \rightarrow (101001)_2$

2. Convert decimal 153 to octal

$\frac{153}{8} = 19 - 1$ (MSB)

$\frac{19}{8} = 2 - 3$

(LSB)

$(153)_{10} \rightarrow (231)_8$

3. Convert $(0.6875)_{10}$ to binary

$0.6875 \times 2 = 1.375 \rightarrow 1$ (LSB)

$0.375 \times 2 = 0.75 \rightarrow 0$

$0.75 \times 2 = 1.5 \rightarrow 1$

$0.5 \times 2 = 1$ (MSB)

$(0.6875)_{10} = (0.1011)_2$

4. Convert $(0.513)_{10}$ to octal

$0.513 \times 8 = 4.104 \rightarrow 4$
$0.104 \times 8 = 0.832 \rightarrow 0$
$0.832 \times 8 = 6.656 \rightarrow 6$
$0.656 \times 8 = 5.248 \rightarrow 5$
$0.248 \times 8 = 1.984 \rightarrow 1$

$\therefore \boxed{(0.513)_{10} = (0.40651\ldots)_8}$

⇒ Conversion of Binary and to Octal and Hexadecimal Numbers:

1. $(1011000111010111.111100000110)_2 = (\ )_8$?

$= (\underbrace{10}_{2}\ \underbrace{110}_{6}\ \underbrace{001}_{1}\ \underbrace{101}_{5}\ \underbrace{011}_{3}.\ \underbrace{111}_{7}\ \underbrace{100}_{4}\ \underbrace{000}_{0}\ \underbrace{110}_{6})_2$

$= \boxed{\text{Ans} = (26153.7406)_8}$

2. $(1011000111010111.11110010)_2 = (\ )_{16}$?

$= \underbrace{10}_{2}\ \underbrace{1100}_{C}\ \underbrace{0110}_{6}\ \underbrace{1011}_{B}.\ \underbrace{1111}_{F}\ \underbrace{0010}_{2}$

$\boxed{\text{Ans} = (2C6B.F2)_{16}}$

→ Number base conversions are essential in digital electronics (digital electronics is the study of electronic circuits that are used to process and control digital signals).

→ In digital system, the input in decimal format, but it takes as binary numbers for the computation by binary to decimal to binary conversion.

Hexadecimal numbers are used for coding in microprocessor then it converts to binary for computation and then results are in a hexadecimal format

⇒ Complements

→ Complements are used in digital computers for simplyfy-ing the subtraction operation for logical manipulations.
There are 2 types of complements for each base r system

(1) the r's complement
(2) the (r-1)'s complement

(1) The r's Complement:

Given a positive number N in base r with an integer part of n digits, the r's complement of N is defined as $r^n - N$ for $N \neq 0$ and 0 for $N = 0$.

Examples

1. The 10's complement of $(52520)_{10}$ is ?

here n = number of digits = 5
r = 10
N = decimal number = 52520

$(52520)_{10} \Rightarrow r^n - N$
$= 10^5 - 52520$
Ans = 47480

2. The 10's complement of $(0.3267)_{10}$ is

here $r = 10$
$n = 0$ (∵ no integer part)
$N = 0.3267$

$r^n - N = 10^0 - 0.3267$
$= 1 - 0.3267$

Ans - 0.6733

3. The 10's complement of $(25.639)_{10}$ is

here $r = 10$
$n = 2$
$N = 25.639$

$r^n - N$
$= 10^2 - 25.639$

Ans = 74.361

4. The 2's complement of $(101100)_2$ is

here $r = 2$
$n = 6$
$N = 101100$

$r^n - N$
$= 2^6 - 101100$
$= 64 - 101100$
$= 100000 - 101100$

Ans = 010100 ?

```
 100000
 101100
 -------
 001100
 000100
```

5. The 2's Complement of $(0.0110)_2$ is

here $r = 2$
$n = 0$ (∵ no integer part)
$N = 0.0110$

$r^n - N$
$= 2^0 - 0.0110$

Ans = 0.1010 ?

Given a positive number N in base r with an integer part of n digits and a fraction part of m digits, the (r-1)'s complement of N is defined as.

$r^n - r^{-m} - N$.

Examples:

1. The 9's Complement of $(52520)_{10}$ is

here $r = 10$
$n = 5, m = 0$.
$N = 52520$

$\therefore = r^n - r^{-m} - N$

$= 10^5 - 10^0 - 52520$

$= 100000 - 1 - 52520$

Ans = 47479

2. The 9's complement of $(0.3267)_{10}$ is

→ $r = 10$
$n = 1, m = 4$
$N = 0.3267$

$\therefore r^n - r^{-m} - N$

$= 10^0 - 10^{-4} - 0.3267$

$= 1 - \frac{1}{10^4} - 0.3267$

$= 1 - 0.001 - 0.3267$

Ans = 0.6732

3. The 9's complement of $(25.639)_{10}$ is.

$r = 10$
$n = 2, m = 3$.
$N = 25.639$

$\therefore 10^2 - 10^{-3} - 25.639$

Ans = 74.36

4. The i's Complement of $(101100)_2$ is.

$r = 2$

$n = 6, m = 0,$

$N = 101100$

$\therefore r^n - r^{-m} - N$

$2^6 - 2^{-0} - N$

$= 2^6 - 1 - 101100$

$=$

Ans = 010011 ?

5. The i's Complement of $(0.0110)_2$ is.

$r = 2$

$n = 0, m = 4$

$= r^n - r^{-m} - N$

$= 2^0 - 2^{-4} - 0.0110$

$= 1 - \frac{1}{2^4} - 0.0110$

0.625 0.0735

Ans = 0.1001 ?,

⇒ Subtraction with r's Complements :

The subtraction of two positive numbers $(M-N)$, both of base r, may be done as follows.

1. Add the minuend M to the r's Complement of the subtrahend N
2. Inspect the result obtained in step 1 for an end carry:
   (a) If an end carry occurs, discard it
   (b) If an end carry doesnot occur, take the r's Complement of the number obtained in step 1, and place a negative sign in front.

Examples

1. Using 10's complement, subtract 72532 - 3250

⇒ Here, M = 72532
N = 03250.

10's complement of 3250 = $r^n - N$
$= 10^5 - 3250$
$= 96750$.

Add M to r's complement of N ∴ 72532
+ 96750
→ 1 69282
Carry

If Carry exist, discard it
∴ Ans → 69282

2. Subtract $(3250 - 72532)_{10}$.

⇒ Here M = 03250
N = 72532

10's complement of N = $r^n - N$
$= 10^5 - 72532$
$= 27468$

Add M to r's Complement of N ∴ 03250
27468
30718

Since no carry, take 10's complement of 30718 and place -ve sign.

-(10's complement of 30718) = $10^5 - 30718$

Ans = -69282

3. Use 2's complement to perform M-N with given binary no.

M = 1010100

N = 1000100

2's Complement of N = $r^n - N$

$= 2^7 - 1000100$

$= 10000000 - 1000100$

$= 0111100$

Add M to the 2's Complement of N

∴ 1010100
0111100
10010000

discard Carry

∴ Ans = 0010000

4. M = 1000100

N = 1010100

2's Complement of N = $r^n - N$

$= 2^7 - 1010100$

$= 0101100$

Add M to the 2's Complement of N ∴ 1000100
0101100
1110000

Since no Carry, take 2's Complement of 1110000 and put -ve sign

∴ 2's Complement of 1110000 = $r^n - N$

$= 2^7 - 1110000$

Ans = - 10000

⇒ Subtraction of $(r-1)$'s Complement :

The subtraction of M-N, both positive numbers in base r, may be calculated in following manner.

1. Add the minuend M to the $(r-1)$'s complement of the subtrahend N.
2. Inspect the result obtained in step 1 for an end carry.
   (a) If an end carry occur, add 1 to the least significant digit (end around carry)
   (b) If an end carry doesnot occur, take $(r-1)$s complement of the number obtained in step 1 and place a -ve sign in front.

Examples

1. Use 9's Complement, subtract 72539 - 3250

M = 72532.

N = 03250.

9's Complement of 03250 $= r^n - r^m - N$

$\because r = 10$ $= 10^5 - 10^0 - 03250$

$n = 5, m = 0$

$N = 03250$ $= 96749$

Add M to 9's complement of N

```
  72539
+ 96749
-------
 169288
```

(1 → end around carry)

```
   +1
-------
 69289
```

Ans: 69289

2. M = 03250
N = 72532

⇒ 9's complement of 72532 $= r^n - r^m - N$

$= 10^5 - 10^0 - 72532$

$= 27467$

Add M to, 9's complement of N

$\therefore$ 03250
27467
30717

Since no carry, take (9's complement of 30717)

$\therefore r^n - r^m - N$

$10^5 - 10^0 - 30717$

$= 69282$

ans = −69282

3. M = 1010100
N = 1000100

⇒ 1's complement of 1000100 $= r^n - r^m - 1000100$

$= 2^7 - 1 - 1000100$

$= 10000000 - 1 - 1000100$

$= 0111011$

Add M to 1's complement of N

$\therefore$ 1010100
0111011
10001111
+1
0010000

ans = 0010000

4. 1000100 = M
1010100 = N.

⇒ 1's complement of 1010100 = $r^n - r^m - N$

$= 2^7 - 2^0 - 1010100$

$= 0101011$

```
  10000000
- 1010101
----------
  0101011
```

Add M to the 1's complement of N.

```
  1000100
+ 0101011
----------
  1101111
```

Since no carry, take 1's complement and put -ve sign.

$= r^n - r^m - N$

$= 2^7 - 2^0 - 1101111$

$= 10000$

```
  10000000
- 1101111
----------
  0010000
```

Ans = −10000

Boolean Algebra and Logic Gates:

Boolean Algebra:

It is a branch of mathematics that deals with operations on logical values with binary variables.

The postulates of a mathematical system form the basic assumptions from which it is possible to deduce the rules, theorems and properties of the system. The most common postulates used to formulate various algebraic structures are;

1. Clasure

A Set S is closed with respect to binary operator. If, for every pair of elements of S, the binary operator specifies a rule for obtaining a unique element of S.

For example, the set of Natural numbers $N = \{1, 2, 3, 4 \ldots\}$ is closed withrespect to binary operator plus (+) by the rules of arithmatic addition, since for any $a, b \in N$ we obtain a unique $C \in N$ by the operation $a + b = C$.

2. Associative Law:

A binary operator * on set S is said to be associative, whenever,

$$(x * y) * z = x * (y * z) \quad \text{for all } x, y, z \in S$$

3. Commutative Law:

A binary operator * on set S is said to be Commutative whenever

$$x * y = y * x \quad \text{for all } x, y \in S$$

4. Identity element:

A Set S is said to have an identity element with respect to binary operator * on S if there exist an element $e \in S$ with the property

$$e * x = x * e = x \quad \text{for every } x \in S.$$

5. Inverse:

A Set is having the identity element e with respect to binary operator $*$ is said to have an inverse, whenever for every $x \in S$, there exists an element $y \in S$ such that

$$x * y = e.$$

6. Distributive Law:

If $*$ and $\cdot$ are two binary operator on set S is said to be distributive over, whenever

$$x * (y \cdot z) = (x * y) \cdot (x * z).$$

Axiomatic Definition of Boolean Algebra:

Boolean Algebra is an algebraic structure defined on a set of elements B together with two binary operator $+$ and $\cdot$ provided the following postulates are satisfied.

1. (a) Closure with respect to the operator $+$.
   (b) Closure with respect to the operator $\cdot$.
2. (a) An identity element with respect to $+$, designated by 0: $x + 0 = 0 + x = x$.
   (b) An identity element with respect to $\cdot$, designated by 1: $x \cdot 1 = 1 \cdot x = x$
3. (a) Commutative with respect to $+$: $x + y = y + x$.
   (b) Commutative with respect to $\cdot$: $x \cdot y = y \cdot x$
4. (a) $\cdot$ is distributive over $+$: $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
   (b) $+$ is distributive over $\cdot$: $x + (y \cdot z) = (x + y) \cdot (x + z)$
5. For every element $x \in B$, there exist an element $x' \in B$ (called the complement of x) such that:
   (a) $x + x' = 1$ and (b) $x \cdot x' = 0$
6. There exists atleast 2 elements $x, y \in B$ such that $x \neq y$

Two valued Boolean Algebra:

A two valued boolean algebra is defined on a set of two elements $B = \{0, 1\}$ with rules for the two binary operators + and · as shown in the following operator tables

x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

x	x'
0	1
1	0

Now we must show that Huntington postulates are valid for the set $B = \{0, 1\}$ and the two binary operator defined above.

1. Closure is obvious from the tables since the result of each operation is either 1 or 0 and $1, 0 \in B$.

2. From the tables we see that,

   (a) $0 + 0 = 0 \quad 0 + 1 = 1 + 0 = 1$

   (b) $1 \cdot 1 = 1 \quad 1 \cdot 0 = 0 \cdot 1 = 0$

   which establishes two identity elements 0 for + and 1 for · as defined by postulate 2.

3. The Commutative laws are obvious from the symmetry of the binary operator tables

4. (a) The distributive law $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$ can be shown to hold true from the operator tables by performing a truth table of all possible values of x, y, z. For each combination we derive $x \cdot (y + z)$ and $(x \cdot y) + (x \cdot z)$

x	y	z	$y+z$	$x\cdot(y+z)$	$x\cdot y$	$x\cdot z$	$(x\cdot y)+(x\cdot z)$
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	1	0	1	1
1	1	0	1	1	1	0	1
1	1	1	1	1	1	1	1

(b) The distributive laws of + over · can be shown to hold true by means of truth table similar to the one above.

5. From the complement table it is easily shown that:
   (a) $x + x' = 1$, since $0 + 0' = 0 + 1 = 1$ and $1 + 1' = 1 + 0 = 1$
   (b) $x \cdot x' = 0$, since $0 \cdot 0' = 0 \cdot 1 = 0$ and $1 \cdot 1' = 1 \cdot 0 = 0$.

6. Postulate 6 is verified because the two valued Boolean algebra has two distinct elements 1 and 0 with $1 \neq 0$

Basic Theorems and Properties of Boolean Algebra:

Table lists six theorems of Boolean Algebra and four of its postulates. The theorems and postulates listed are the most basic relationships in Boolean Algebra.

Postulate 2	(a) $x+0=x$	(b) $x \cdot 1 = x$.
Postulate 5	(a) $x+x'=1$	(b) $x \cdot x' = 0$.
Theorem 1	(a) $x+x=x$	(b) $x \cdot x = x$.
Theorem 2	(a) $x+1=1$. $(x')'=x$	(b) $x \cdot 0 = 0$.
Theorem 3, involution	$(x')'=x$	
Postulate 3, Commutative	(a) $x+y=y+x$.	(b) $xy=yx$
Theorem 4, associative	(a) $x+(y+z)=(x+y)+z$	(b) $x(yz)=(xy)z$.
Postulate 4, distributive	(a) $x(y+z)=xy+xz$	(a) $x+yz=(x+y)(x+z)$.
Theorem 5, De Morgan	(a) $(x+y)'=x'y'$	(b) $(xy)'=x'+y'$
Theorem 6, absorption	(a) $x+xy=x$	(b) $x(x+y)=x$

Theorem 1(a) : $x+x=x$

$x+x=(x+x)\cdot 1$ — by postulate 2(b)

$=(x+x)(x+x')$ — 5(a)

$=x+xx'$ — 4(b).

$=x+0$ — 5(b)

$=x$ — 2(a).

Theorem 1(b) : $x \cdot x = x$

$x \cdot x = xx+0$ — by postulate 2(a)

$=xx+xx'$ — 5(b)

$=x(x+x')$ — 4(a).

$=x \cdot 1$ — 5(a)

$=x$. — 2(b).

Theorem 2(a) : $x + 1 = 1$

$$x + 1 = 1 \cdot (x + 1) \quad \text{by postulate 2(b).}$$
$$= (x + x')(x + 1) \quad \text{5(a).}$$
$$= x + x' \cdot 1 \quad \text{4(b).}$$
$$= x + x' \quad \text{2(b).}$$
$$= 1 \quad \text{5(a)}$$

Theorem 2(b) : $x \cdot 0 = 0$ by duality

Theorem 3 : $(x')' = x$

From postulate 5, we have $x + x' = 1$ and $x \cdot x' = 0$ which defines the complement of x. The complement of x' is x and is also $(x')'$. Therefore, since the complement is unique, we have that $(x')' = x$.

Theorem 6(a) : $x + xy = x$

$$x + xy = x \cdot 1 + xy \quad \text{by postulate 2(b).}$$
$$= x(1 + y) \quad \text{4(a).}$$
$$= x(y + 1) \quad \text{3(a).}$$
$$= x \cdot 1 \quad \text{2(a).}$$
$$= x.$$

Theorem 6(b) : $x(x + y) = x$ by duality

The following truth table verifies the first absorption theorem.

x	y	xy	$x + xy$
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

$=$ verified

Truth table for De Morgan's theorem is shown below.

$(x+y)' = x'y'$

x	y	x+y	(x+y)'	x'	y'	x'y'
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

↑ ——— = ——— ↑

Operator Precedence:

The operator precedence for evaluating expressions is

(1) Parentheses.

(2) NOT

(3) AND

(4) OR

Eg: Consider the Demorgan Theorem evaluated above which follows the order of precedence.

Venn Diagrams:

A helpfull illustration that may be used to visualize the relationship among the variables of a Boolean expressions in the Venn Diagrams.

- The diagram consist of a rectangle as shown in the figure inside of which are drawn overlapping circles, one for each variable

- We designate all points inside a circle as belonging to the named variable and all points outside a circle as not belonging to the variable.
- For example,

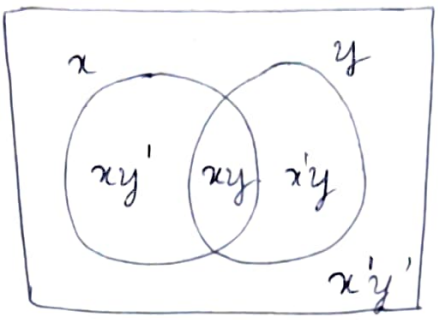


Figure: Venn diagram for two variables.

- For example, the circle labelled x. If we are inside the circle, we say that $x = 1$; when outside we say $x = 0$. Now with 2 overlapping circles, there are four distinct areas inside the rectangle; the area not belonging to either x or y $(x'y')$. the area inside circle y but outside x $(x'y)$, the area inside circle x but outside y (xy') and the area inside both circles (xy)
- Venn diagrams are used to illustrate the postulates of Boolean Algebra or to show the identity of theorems.
- For example, the area belonging to xy is inside the circle xy and therefore $x + xy = x$.

Boolean Functions:

- A Boolean Function is an expression formed with binary variables, (A binary variable can take the value 0 and 1), two binary operators OR and AND, the unary operator NOT, parenthesis and equal sign.

- For a given value of variables, the function can be either 0 or 1. Consider the example, the boolean function.

$$F_1 = xyz'$$

- A Boolean function may be represented in a truth table. Here we need 2^n combinations of 1's and 0's where n is no of variables. Hence there are 8 possible combinations.

x	y	z	F_1	F_2	F_3	F_4
0	0	0	0	0	0	0
0	0	1	0	1	1	1
0	1	0	0	0	0	0
0	1	1	0	0	1	1
1	0	0	0	1	1	1
1	0	1	0	1	1	1
1	1	0	1	1	0	0
1	1	1	0	1	0	0

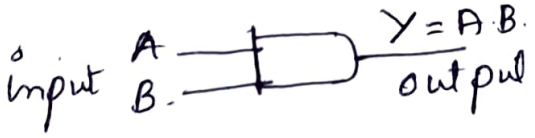
here

$$F_2 = x + y'z$$
$$F_3 = x'y'z + x'yz + xy'$$
$$F_4 = xy' + x'z$$

- A Boolean function may be transformed from an algebraic expression into a logic diagram composed of AND, OR and NOT gates.
- From the diagrams (below), we can say that implementation of F_4 requires fewer gates than F_3 hence it is more economical to implement F_4 than F_3

Basic Gates: All digital systems can be constructed by only using three basic gates. They are AND, OR and NOT gates

2 input AND gates:

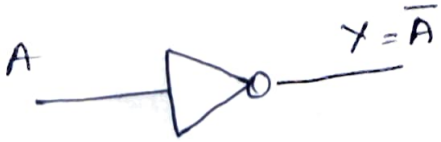


A	B	Y = A.B
0	0	0
0	1	0
1	0	0
1	1	1

2 input OR gate:

A	B	Y = A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT gate



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

Algebraic Manipulation:

- When a Boolean function is implemented with logic gates, each literal (variable) in the function designates an input to a gate; and each term is implemented with a gate. The minimization of the number of literals and the number of terms result in circuit with less equipment.
- The no. of literals in Boolean function can be minimized by algebraic manipulation.

Examples:

1. Simplify the Boolean functions to a minimum number of literals:

(i) $x + x'y$

$= (x + x')(x + y)$ $[\because x = x.x]$

$= 1.(x + y)$ $[\because x + x' = 1]$

$= x + y$

(ii) $x(x' + y) = xx' + xy$

$= 0 + xy$

$= xy$

(iii) $x'y'z + x'yz + xy'$

$= x'z(y + y') + xy'$

$= x'z + xy'$ $[\because y + y' = 1]$.

(iv) $xy + x'z + yz = xy + x'z + yz(x + x')$ $[\because x + x' = 1]$

$= xy + x'z + yzx + yzx'$

$= xy(1 + z) + x'z(1 + y)$

$= xy + x'z$ $[\because 1 + z = 1 \text{ \& } 1 + y = 1]$

(v) $(x + y)(x' + z)(y + z)$

$= (x + y)(x' + z)$ by duality from function 4

Complement of a Function:

- The Complement of a function F is F' and is obtained from an interchange of 0's for 1's and 1's for 0's in the value of F.
- The Complement of a function may be derived algebraically through De Morgan's theorem.
- De Morgan's Theorem for 2 variable:

  $(x + y)' = x'y'$
- De Morgan's theorem can be extended to three or more variables and can be derived as follows

$(A + B + C)' = (A + X)'$ let $B + C = X$

$= A'X'$ by De Morgan

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

$= A'.(B'C')$

$= A'B'C'$

- It can be derived for number of variables and can derived by successive substitutions. These theorems can be generalized as follows:

$$(A+B+C+D+\quad\quad+F') = A'B'C'D'\ldots F'$$

$$(ABCD\ldots F)' = A'+B'+C'+D'+\quad\quad F'$$

- The generalized form ~~can be derived by~~ of De Morgan Theorem states that complement of a function is obtained by interchanging AND & OR operators and complementing each literal.

Example:

1. Find the complement of the function $F_1 = x'yz' + x'y'z$ and $F_2 = x(y'z' + yz)$.

(i) $F_1' = (x'yz' + x'y'z)'$

$= (x'yz')'(x'y'z)'$ $\quad [\because (A+B)' = A' \cdot B']$

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

(ii) $F_2' = [x(y'z' + yz)]'$

$= x' + (y'z' + yz)'$

$= x' + (y'z')' \cdot (yz)'$

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

2. Find the complement of the functions F_1 and F_2 by taking their dual and complementing each literal.

1. $F_1 = x'y'z' + x'y'z$

dual of F_1 is $(x'+y+z')(x'+y'+z)$.

Complement each literal : $(x+y'+z)(x+y+z') = F_1'$

2. $F_2 = x(y'z' + yz)$

dual of F_1 : $x + (y'+z')(y+z)$

Complement each literal : $x' + (y+z)(y'+z') = F_2'$

Canonical and Standard Forms :

1. **Literal :** It is a boolean expression variable or its complement.

eg : Let x be the literal & $\bar{x}$ is the complement

2. **Product term :** A product term is a literal or logical AND of multiple literals

eg : AB, $\bar{A}BC$

3. **Sum term :** A sum term is a literal or logical OR operation of multiple literals

eg : $A+B$, $\bar{A}+B+C$

4. Standard forms:

POS (Product of Sum) SOP (Sum of Product)

* Product of Sum (POS): A POS is logical AND and of multiple sum terms

eg: $(A+B+C)(\bar{A}+B+\bar{C})$

* Sum of product (SOP): A SOP is logical OR and of multiple product terms

eg: $AB+BA$

5. Min term: Min term is a product term. It contains all the input variables which makes up a boolean expression specified by Σ

eg: $\bar{a}\bar{b}\bar{c}$

6. Max term: Max term is a sum term. It contains all the input variables which makes up a boolean expression specified by Π

eg: $a+b+c$

	Min terms		Max terms	
x y z	Term	Designation	Term	Designation
0 0 0	$x'y'z'$	m_0	$(x+y+z)$	M_0
0 0 1	$x'y'z$	m_1	$x+y+z'$	M_1
0 1 0	$x'yz'$	m_2	$x+y'+z$	M_2
0 1 1	$x'yz$	m_3	$x+y'+z'$	M_3
1 0 0	$xy'z'$	m_4	$x'+y+z$	M_4
1 0 1	$xy'z$	m_5	$x'+y+z'$	M_5
1 1 0	xyz'	m_6	$x'+y'+z$	M_6
1 1 1	xyz	m_7	$x'+y'+z'$	M_7

Fig: Min terms and Max terms for three binary variables

Canonical SOP and POS forms:

- A truth table consists of a set of inputs and outputs. If there are n input variables, then there will be 2^n possible combinations with 0's & 1's. So the value of each output variable depends on combination of input variables. So each output variable will have '1' for some combination of input variable and '0' for some other combination.
- Therefore, we can express each output variable in following two ways:
  (i) Canonical SOP form.
  (ii) Canonical POS form.

(i) Canonical SOP form:

- Canonical SOP form means Canonical Sum of Products form. In this form each product term contain all literals. So these product terms are nothing but the min terms. Hence Canonical SOP form is also called as sum of min terms form.

- First, identify the min terms for which, the output variable is one and then do the logical OR of those min terms in order to get Boolean expression function corresponding to that output variable. This Boolean function will be in the form of sum of min terms.

x	y	z	function f_1	function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	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

- A Boolean function may be expressed algebraically from a given truth table by forming a minterm for each combination which produces 1, and then taking the OR of these terms. For example,

$$f_1 = x'y'z + xy'z' + xyz$$
$$= m_1 + m_4 + m_7$$

which can be algebraically expressed as.

$$f_1 = \Sigma m(1, 4, 7)$$

- Similarly for $f_2 = x'yz + xy'z + xyz' + xyz$

$$f_2 = m_3 + m_5 + m_6 + m_7$$

$$f_2 = \Sigma m(3, 5, 6, 7)$$

(ii) Canonical POS form:

- Canonical POS form means Canonical Product of Sums form, each sum term contain all literals. So these sum terms are nothing but Max terms. Hence canonical POS form is also called as product of Maxterms
- First, identify the Maxterms for which, the output variable is zero. and then do the logical AND of those Maxterms in order to get the Boolean expression function. This Boolean function will be in the form of product of Max terms.
- For example,

$$f_1 = (x+y+z)(x+y'+z)(x'+y+z)(x'+y+z')(x'+y'+z)$$

$$f_1 = M_0 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6$$

$$f_1 = \Pi M(0, 2, 3, 5, 6)$$

Similarly $f_2 = (x+y+z)(x+y+z')(x+y'+z)(x'+y+z)$

$$f_2 = M_0 \cdot M_1 \cdot M_2 \cdot M_4$$

$$f_2 = \Pi M(0, 1, 2, 4)$$

1. Express the Boolean function $F = A + B'C$ in a sum of minterms.

- Here the function has 3 variables A, B, C. The first term A is missing two variables; therefore

$$A = A(B + B') = AB + AB'$$

it is still missing one variable:

$$\therefore A = AB(C + C') + AB'(C + C')$$
$$= ABC + ABC' + AB'C + AB'C'$$

- Second term $B'C$ missing one variable;

$$\therefore B'C = B'C(A + A')$$
$$= B'CA + B'CA'$$

- Combining all terms, we get

$$A + B'C = ABC + \underline{ABC'} + \underline{AB'C} + AB'C' + \underline{AB'C} + A'BC$$
$$= ABC + ABC' + AB'C + AB'C' + A'BC'$$
$$= m_7 + m_6 + m_5 + m_4 + m_1$$

Rearranging, $= m_1 + m_4 + m_5 + m_6 + m_7$.

$$\therefore \boxed{F(A,B,C) = \Sigma m(1,4,5,6,7)}$$

2. Express the Boolean Function $f = xy + x'z$ in a product of maxterm form.

- First, convert the function into OR terms using distributive law:

$$F = xy + x'z = (xy + x')(xy + z)$$
$$= (x + x')(y + x')(x + z)(y + z)$$
$$= (x' + y)(x + z)(y + z) \quad [\because x + x' = 1]$$

- Function F has three variables: x, y, z. Each OR term missing one variable, therefore

$$x'+y = x'+y+zz' = (x'+y+z)(x'+y+z')$$
$$x+z = x+z+yy' = (x+y+z)(x+y'+z).$$
$$y+z = y+z+xx' = (x+y+z)(x'+y+z)$$

by combining

$$F = (x'+y+z)(x'+y+z')(x+y+z)(x+y'+z)$$
$$= M_4 M_5 M_0 M_2.$$
$$= M_0 M_2 M_4 M_5$$

$$\boxed{F(x,y,z) = \Pi(0,2,4,5)}$$

Conversion between Canonical Forms:

- Consider the function,
$$F(A,B,C) = \Sigma m(1,4,5,6,7).$$
Complement of this can be expressed as
$$F'(A,B,C) = \Sigma m(0,2,3) = m_0 + m_2 + m_3.$$
By taking the complement of F, by De Morgan Theorem, we obtain
$$F(A,B,C) = (m_0+m_2+m_3)' = m_0' m_2' m_3' = M_0 M_2 M_3.$$
$$= \Pi(0,2,3)$$

- Hence, now we can state a general conversion procedure to convert from one Canonical form to another, interchange the symbols Σ and Π, and list those numbers missing from the original form.

- For example,

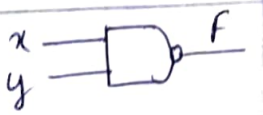
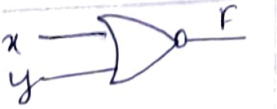
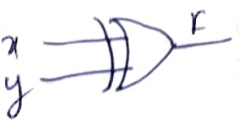
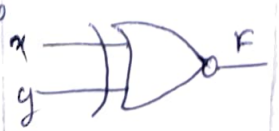
$$F(x,y,z) = \Pi(0,2,4,5)$$

$$\downarrow$$

$$\rightarrow F(x,y,z) = \Sigma m(1,3,6,7).$$

- Note: In order to find the missing terms, one must realize that the total number of minterms or maxterms is 2^n, where n is the number of binary variables in the function.

Design Logic Gates:

	Name	Graphic symbol	Algebraic function	Truth Table
1.	AND		$F = xy$	x y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
2.	OR		$F = x + y$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
3.	Inverter (NOT)		$F = x'$	x | F 0 | 1 1 | 0
4.	Buffer		$F = x$	x | f 0 | 0 1 | 1
5.	NAND		$F = (xy)' = \overline{xy}$	x y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
6.	NOR		$F = (x+y)' = \overline{x+y}$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
7.	Exclusive-OR XOR.		$F = x \oplus y = xy' + y'x$	x y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
8.	Exclusive-~~OR~~ NOR or equivalence		$F = xy + x'y' = x \odot y$	x y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

- Except for the inverter and buffer, it can be extended to have more than two inputs if the binary operation it represents is commutative and associative.
- The AND and OR operations in Boolean Algebra, posses these two properties.

$$x + y = y + x \quad \text{Commutative}$$

$$(x + y) + z = x + (y + z) = x + y + z \quad \text{associative}$$

- But NAND & NOR operators are not associative.

i.e $(x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z)$

$$(x \downarrow y) \downarrow z = [(x + y)' + z]' = (x + y)z' = xz' + yz'$$

$$x \downarrow (y \downarrow z) = [x + (y + z)'] = x'(y + z) = x'y + x'z$$

- To overcome this difficulty, multiple NAND & NOR gate as a complemented OR (or AND) gate, Thus.

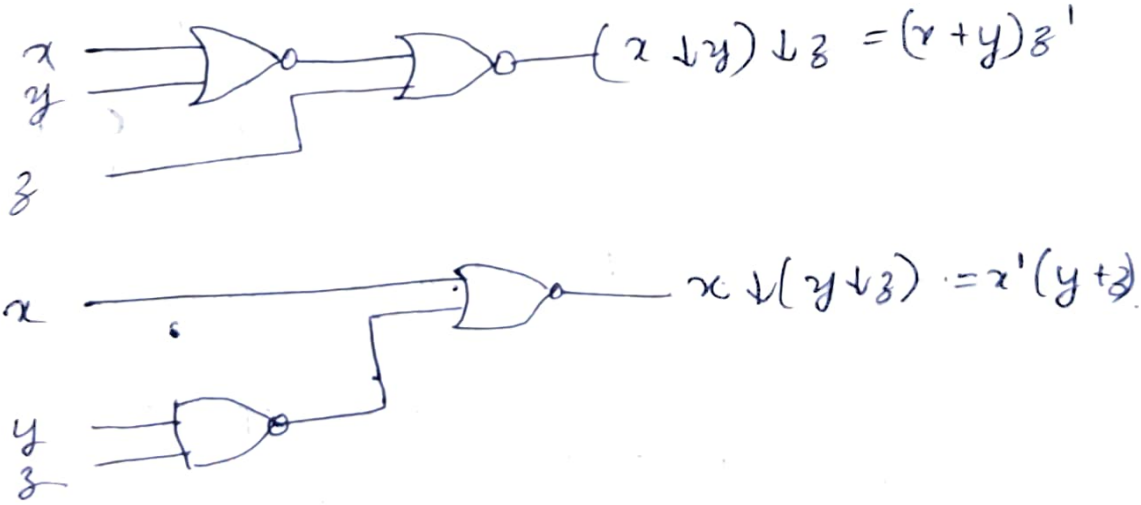


Fig: Demonstrating the nonassociativity of the NOR operator

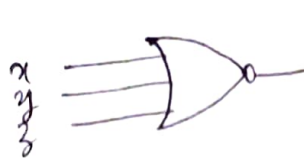


Fig (a) Three input NOR gate

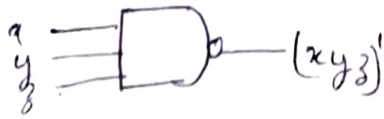


Fig (b) Three input Nand gate

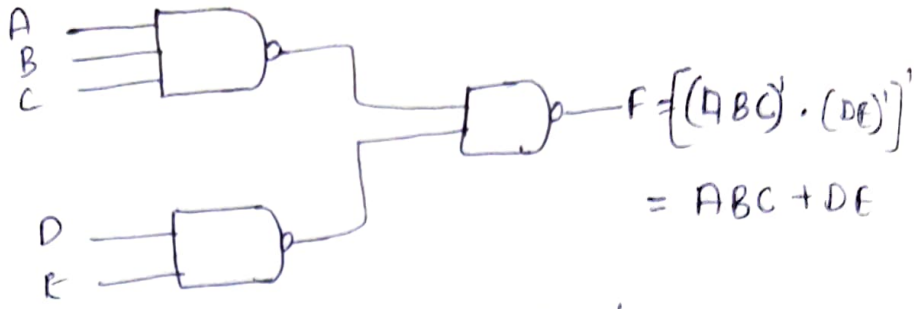


Fig (c) Cascaded Nand gates

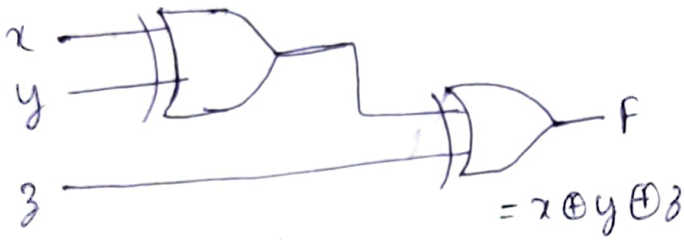
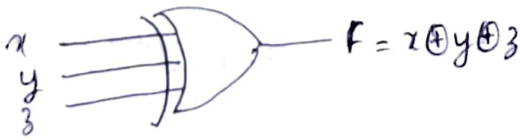


Fig (a) using two input gates.



x	y	z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

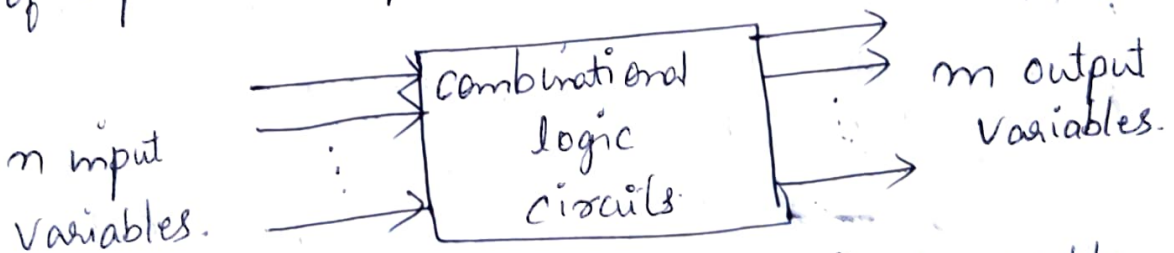
(a) Truth table

Fig : Three-input exclusive-OR gate

Combinational Logic → Introduction.
→ Design Procedure.
→ Adders

Introduction:

- Logic circuits for digital systems may be combinational or sequential.
- A combinational circuit consist of logic gates whose outputs at any time are determined directly from the present combination of inputs without regard to previous inputs. A combinational circuit performs a specific function fully specified logically by a set of boolean equations.
- Sequential circuit employs memory elements in addition to logic gates. These outputs are function of inputs and previous inputs



- A combinational circuit consist of input variables, logic gates and output variables. The n input binary variable come from an external source, the m output variable go to external destination

Design Procedure:

Design of combinational circuits involves the following steps

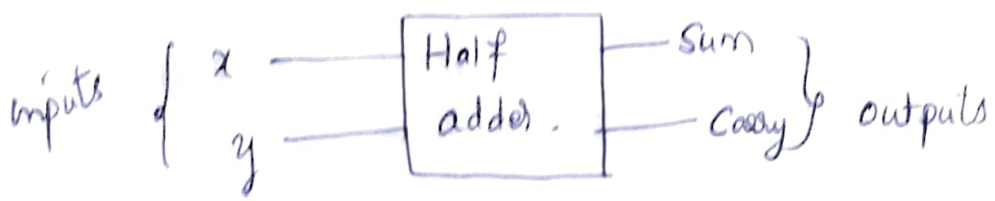
1. The problem is stated
2. The number of available input variables and required output variables is determined.
3. The input and output variables are assigned letter symbols
4. The truth table that defines the required relationship between inputs and output is derived
5. The simplified boolean function for each output is obtained
6. The logic diagram is drawn

Adders:

- Digital computers perform a variety of information processing tasks. Among the basic functions encountered are the various arithmetic operations. The most basic arithmetic operation is the addition of two binary digits.

Half adders:

- A combinational circuit that performs the addition of 2 bits is called half adder.



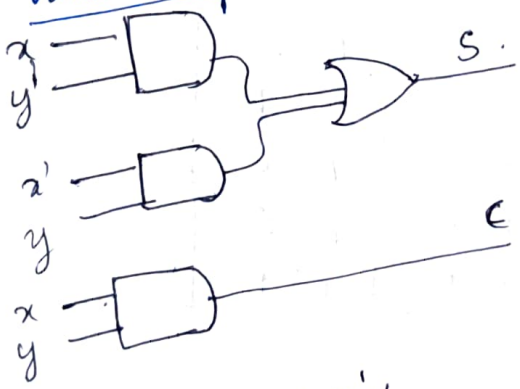
Truth table

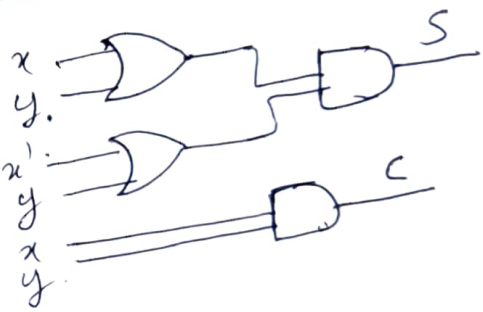
x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

- the outputs are denoted by S and C. and the related equations are.

$$S = x'y + xy'$$
$$C = xy$$

Various implementations of half adder:



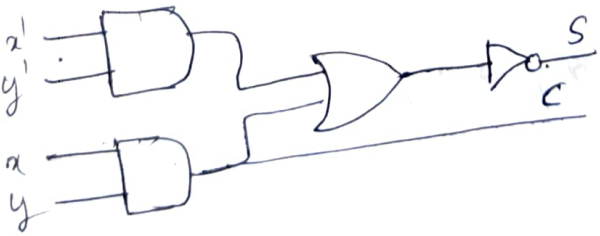


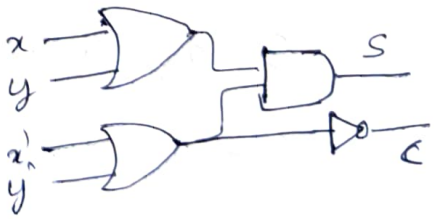
(a) $S = xy + x'y$

$C = xy$

(b) $S = (x+y)(x'+y')$

$C = xy$



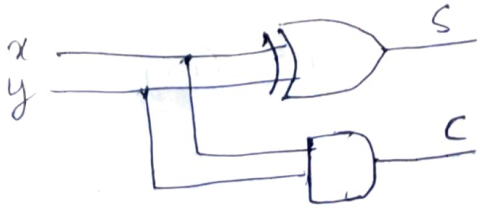


(c) $S = (C + x'y)'$

$C = xy$

(d) $S = (x+y)(x'+y')$

$C = (x'+y')'$



(e) $S = x \oplus y$

$C = xy$

Full Adder :

- A Full Adder is a Combinational circuit that forms the arithmatic sum of three input bits. It consist of three inputs and two outputs. Two of the input variables denoted by x and y. The third input, z, Represents the Carry from the previous lower significant position

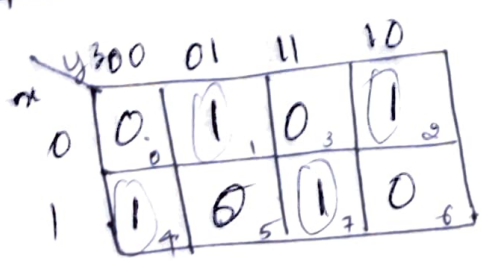
Truth table :

x	y	z	C	S	
0	0	0	0	0	0
0	0	1	0	1	1
0	1	0	0	1	2
0	1	1	1	0	3
1	0	0	0	1	4
1	0	1	1	0	5
1	1	0	1	0	6
1	1	1	1	1	7

$$S = x'y'z + x'yz' + xy'z' + xyz$$

$$C = x'yz + xy'z + xyz' + xyz$$

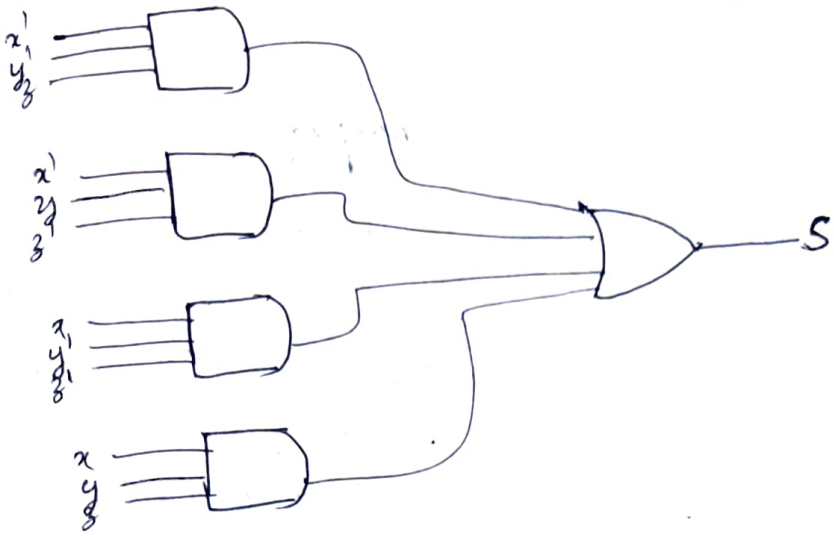
K maps for full adder :

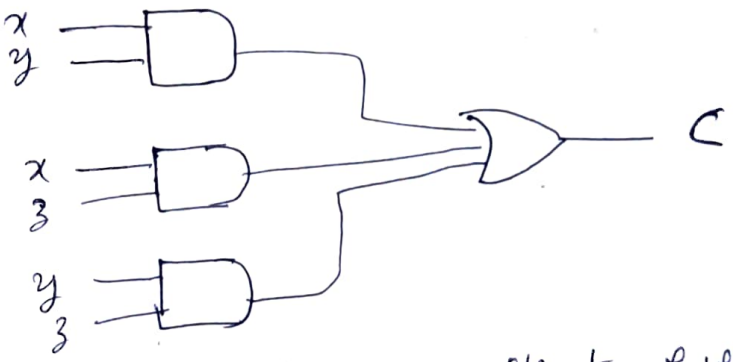


For sum.

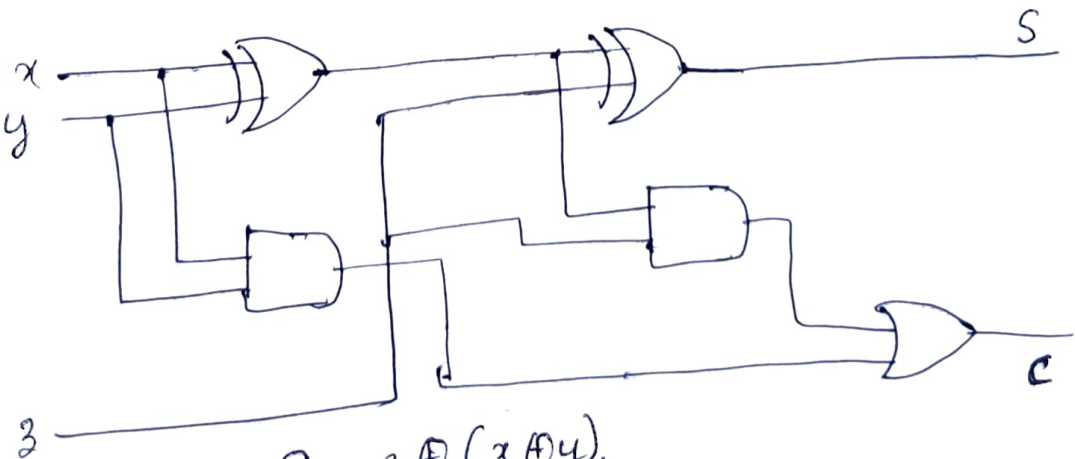
For Carry

implementation of full adder using basic gates :





Implementation of a full adder with two half and OR gate :



$$S = z \oplus (x \oplus y).$$
$$= z'(xy' + x'y) + z(xy' + x'y)'$$
$$= z'(xy' + x'y + z(xy + x'y').$$
$$= xy'z' + x'yz' + xyz + x'y'z$$

$$C = z(xy' + x'y) + xy = xyz' + x'yz + xy$$