Module - 3

Boolean Algebra and Logic Circuits

Binary numbers

Number System

The representation of numerals in a specific way is called a number system. Mathematically, the numerals are indicated by symbols, the number of symbols used for representing the numerals decides the number system.

There are 4 types

1) Binary number System
2) Octal number System
3) Decimal number System
4) Hexadecimal number System.

Decimal number System: This is the most commonly used number system. There are 10 symbols. They are '0', '1', '2', '3', '4', '5', '6', '7', '8', '9'. Therefore, the base is "ten"

Binary number System: There are only 2 symbols '0' & '1'. Any value is represented using these two symbols only. This number system is most suitable for digital computer. The base is two.

Octal number System: There are eight symbols '0', '1', '2', '3', '4', '5', '6', '7'. The base is eight.

Hexadecimal number System: There are 16 symbols '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F'. The base is "sixteen"

Decimal number System

Number	5	6	8	9
Position	Digit 3	Digit 2	Digit 1	Digit 0
Weightage	10^3	10^2	10^1	10^0

The value of digit 3 = $5 \times 10^3 = 5000$.

Octal number System

Number	2	3	5	6
Position	Digit 3	Digit 2	Digit 1	Digit 0
Weightage	8^3	8^2	8^1	8^0

The value of digit 2 $= 3 \times 8^3$

$= 3 \times 512$

$= 1024$

Binary number System

Number	1	0	1	0
Position	Digit 3	Digit 2	Digit 1	Digit 0
Weightage	2^3	2^2	2^1	2^0

The value of bit 3 $= 1 \times 2^3 = 1 \times 8 = 8$

Hexadecimal number System

Number	2	3	5	6
Position	Digit 3	Digit 2	Digit 1	Digit 0
Weightage	16^3	16^2	16^1	16^0

The value of digit 2 $= 3 \times 16^2$

$= 3 \times 256$

$= 768.$

Conversion of Number System

Case 1: Conversion from any number System to decimal

prob. 1 Convert the given octal number 235 into decimal

Position	Digit 2	Digit 1	Digit 0
Number	2	3	5
Weightage	8^2	8^1	8^0

$= 2 \times 8^2 + 3 \times 8^1 + 5 \times 8^0$

$= 2 \times 64 + 3 \times 8 + 5 \times 1$

$= 128 + 24 + 5$

$= 157$

$(235)_8 = (157)_D$

$(235)_8 = (157)_{10}$

prob2 Convert the given octal number "235.67" into decimal

Position	Digit 2	Digit 1	Digit 0		Digit (-1)	Digit (-2)
Number	2	3	5	.	6	7
Weightage	8^2	8^1	8^0	.	8^{-1}	8^{-2}
	Integral part				Fractional part	

$= 2\times8^2 + 3\times8^1 + 5\times8^0 \cdot 6\times8^{-1} + 7\times8^{-2}$

$= 2\times64 + 3\times8 + 5\times1 \cdot 6\times\frac{1}{8} + 7\times\frac{1}{8^2}$

$= 128 + 24 + 5 \cdot 0.75 + 0.1094$

$= 157.8594$

$(235.67)_8 = (157.8594)_{10}$

prob3 Convert the given hexadecimal number "235" into decimal

Position	Digit 2	Digit 1	Digit 0
Number	2	3	5
Weightage	16^2	16^1	16^0

$= 2\times16^2 + 3\times16^1 + 5\times16^0$

$= 2\times256 + 3\times16 + 5\times1$

$= 565$

$(235)_{16} = (565)_{10}$

prob4 Convert the given hexa-decimal number "235.89" into decimal.

Position	Digit 2	Digit 1	Digit 0		Digit (-1)	Digit (-2)
Number	2	3	5	.	8	9
Weightage	16^2	16^1	16^0	.	16^{-1}	16^{-2}
	Integral part				Fractional part	

$= 2\times16^2 + 3\times16^1 + 5\times16^0 \cdot 8\times16^{-1} + 9\times16^{-2}$

$= 2\times256 + 3\times16 + 5\times1 \cdot 8\times\frac{1}{16} + 9\times\frac{1}{16^2}$

$= 512 + 48 + 5 \cdot 0.5 + 0.035156$

$(235.89)_{16} = (565.535156)_{10}$

prob5: Convert the given binary number 11010 into decimal

$= 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0$

$= 16 + 8 + 0 + 2 + 0 = 26$

$(11010)_2 = (26)_{10}$

prob6 Convert the given binary number 11010.101 into decimal.

$= 1\times2^4 + 1\times2^3 + 0\times2^2 + 1\times2^1 + 0\times2^0 + 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3}$

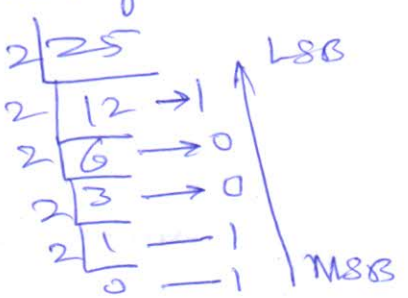
$= 16 + 8 + 0 + 2 + 0 + 0.5 + 0 + 0.125$

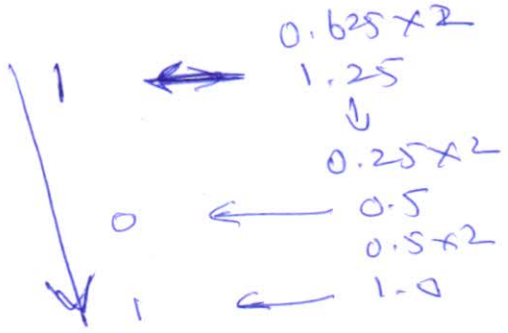
$= (26.625)_{10}$

$(11010.101)_2 = (26.625)_{10}$

Conversion of number in decimal number system to any other number

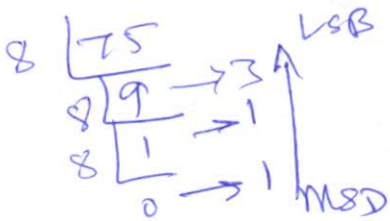
prob1 Convert the decimal number 25.625 into binary

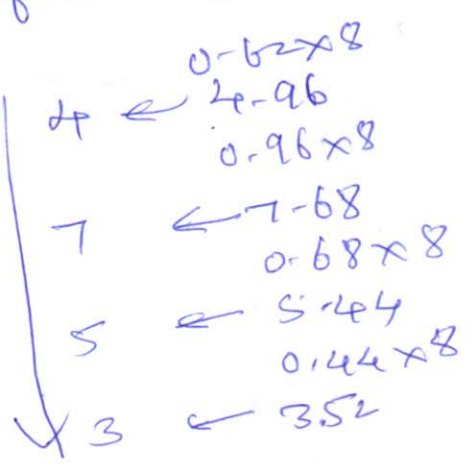




$(25.625)_{10} = (11001.101)_2$

prob2 Convert the decimal number 75.62 into octal Approximate the result to first 4 places





$(75.62)_{10} = (113.4753)_8$.

prob Convert the decimal number 75.62 into hexadecimal. Approximate the result to four places

Soln.

16 | 75
16 | 4 — 11 → B ↑ LSD
0 — 4 → 4 MSD

0.62 × 16
9 ← 9.92
0.92 × 16
E ← 14.72
0.72 × 16
B ← 11.52
0.52 × 16
8 ← 8.32

$(75.62)_{10} = (4B.9EB8)_{16}$

Numbers with Different Bases

Decimal (base 10)	Binary (base 2)	Octal (base 8)	Hexadecimal (base 16)
00	0000	00	0
01	0001	01	1
02	0010	02	2
03	0011	03	3
04	0100	04	4
05	0101	05	5
06	0110	06	6
07	0111	07	7
08	1000	10	8
09	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Conversion from binary into Octal

prob Convert the given binary number 1101101.1010111 into octal

001 101 101 . 101 011 100
1 5 5 5 3 4

$(1101101.1010111)_2 = (155.534)_8$

~~prob2~~ Conversion from binary into hexadecimal

prob2 Convert the given binary number 1101101.1010111 into hexadecimal

0110 1101 . 1010 1110
6 D A E

$(01101101.1010111)_2 = (6D.AE)_{16}$

Conversion of octal into binary

prob3 Convert the given octal number 134.67 into binary.

134 . 67
001 011 100 . 110 111

$(134.67)_8 = (001011100.110111)_2$

Conversion of hexadecimal to binary

prob4. Convert the given hexadecimal number 345.AB into binary.

3 4 5 . A B
0011 0100 0101 1010 1011

$(345.AB)_{16} = (001101000101.10101011)_2$

Complements

Complements are used in digital computers for simplifying the subtraction operation and for logical manipulation

Base-r system there are two types of complements

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

Eg: for binary numbers (r=2)
① 1's complement ie (r-1)
② 2's complements ie 'r'

for decimal numbers
① 10's complement ie (r=10) 10's
② 9's complements (r-1)= 9's

Diminished Radix complement

The 9's complement of 546700 is 999999 - 546700
= 453299

The 9's complement of 012398 is 999999 =
= 987601

The 1's complement of 1011000 is 0100111
The 1's complement of 0101101 is 1010010

The (r-1)'s complement of octal or hexadecimal numbers is obtained by subtracting each digit from 7 or F (decimal 15), respectively

Radix complement

The 10's complement of 012398 is 987602
The 10's complement of 246700 is 753300

The 2's complement of 1101100 is 0010100
The 2's complement of 0110111 is 1001001

Subtraction with complements

Eg: Using 10's complement, subtract 72532 - 3250

M = 72532
10's complement of N = +96750
Sum = 169282
Discard end carry 10^5 = 100000
Answer = 69282

Note that M has 5 digits and N has only 4 digits Both numbers must have the same number of digits; so we can write N as 03250. Taking the 10's complement of N produces a 9 in the most significant position. The occurrence of the end carry signifies that $M \geq N$ and the result is positive.

prob. Using 10's complement, subtract 3250 − 72532

M = 03250
10's complement of N = 27468
Sum 30718

there is no end carry

Answer: − (10's complement of 30718) = − 69282

Note: 3250 < 72532, the result is negative.

Since we are dealing with unsigned numbers, there is really no way to get an unsigned result for this case. When subtracting with complements, the negative answer is recognized from the absence of the end carry and the complemented result.

Eg: Given the two binary numbers X = 1010100 and Y = 1000011, perform the subtraction a) X − Y and b) Y − X using 2's complements.

a) X = 1010100
2's complement of Y = 0111101
Sum 10010001
Discard end carry 2^7 = 10000000
Answer: X − Y 0010001

b) Y = 1000011
2's complement of X = + 0101100
Sum 1101111

There is no end carry

Answer: Y − X = − (2's complement of 1101111) = − 0010001

Eg: Given the two binary numbers X = 1010100 and Y = 1000011 perform the subtraction a) X − Y and b) Y − X using 1's complements

a) $X - Y = 1010100 - 1000011$

$X =$ 1010100

1's complement of $Y =$ + 0111100

1001000

→ +1

Answer: $X - Y =$ 0010001

b) $Y - X = 1000011 - 1010100$

$Y =$ 1000011

1's complement of $X =$ + 0101011

Sum = 1101110

There is no end carry

Answer: $Y - X = -$ (1's complement of 1101110)

$= -0010001$

Boolean Algebra

Basic definitions

1) **closure**: A set S is closed with respect to a binary operator if for every pair of elements of S, the binary operator specifies a rule for obtaining a unique element of S

eg $N = \{1, 2, 3, 4, \ldots\}$ is closed w.r.t the binary operator plus (+) by the rules of arithmetic addition

$\therefore a, b \in N$ we obtain a unique $c \in N$ by the operation $a + b = c$

The set of natural numbers is not closed with respect to the binary operator minus (−) by the rules of arithmetic subtraction because $2 - 3 = -1$ and $2, 3 \in N$ while $(-1) \notin N$.

2) **Associative law** A binary operator $*$ on a set S is said to be associative whenever

$(x * y) * z = x * (y * z)$ 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 a binary operation $*$ on S if there exists an element $e \in S$ with the property

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

eg: The element 0 is an identity element with respect to operation + on the set of integers $I = \{\ldots, -3, -2, -1, 0, 1, 2, 3 \ldots\}$ since

$$x + 0 = 0 + x = x \quad \text{for any } x \in I$$

The set of natural numbers N has no identity element since 0 is excluded from the set

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

$$x * y = e$$

eg: In the set of integers I with $e = 0$ the inverse of an element a is $(-a)$ since $a + (-a) = 0$

6) Distributive law

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

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

Axiomatic definition of Boolean Algebra

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 ·; $x \cdot y = y \cdot x$

4 a) · is distributive over +: $x \cdot (y+z) = (x \cdot y) + (x \cdot z)$

b) + is distributive over ·: $x + (y \cdot z) = (x+y) \cdot (x+z)$

5. For every element $x \in B$, there exists an element $x' \in B$ (called complement of x) such that a) $x + x' = 1$ and b) $x \cdot x' = 0$

6. There exists at least two elements $x, y \in B$ such that $x \neq y$.

Comparison between Boolean algebra and ordinary algebra

1) Huntington postulates do not include the associative law. However, this law holds for Boolean algebra & can be derived (for both operators) from the other postulates

2) The distributive law of + and ·, ie $x + (y \cdot z) = (x+y) \cdot (x+z)$ is valid for Boolean algebra, but not for ordinary algebra.

3) Boolean algebra does not have additive or multiplicative inverses; therefore there are no subtraction or division operations

4) Postulate 5 defines an operator called complement that is not available in ordinary algebra

5) Ordinary algebra with the real numbers, which constitute an infinite set of elements. Boolean algebra deals with the ~~as~~ ~~to~~ set B with only two elements, 0 and 1.

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 operator + and ·

x	y	x.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

These rules are exactly the same as the AND, OR and NOT operations, respectively.

Huntington postulates are valid for the set $B = \{0, 1\}$ and the two binary operators defined before

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 the 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 ~~forth~~ forming a truth table of all possible values of x, y and z. For each combination we derive $x \cdot (y+z)$ and show that the value is the same as $(x \cdot y) + (x \cdot z)$

x	y	z	y+z	x.(y+z)	x.y	(x.y)+(x.z)
0	0	0	0	0	0	0
0	0	1	1	0	0	0
0	1	0	1	0	0	0
0	1	1	1	0	0	0
1	0	0	1	0	0	0
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

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

5. From the complement table it is easily shown that

a) $x + x' = 1$ since $0 + 0' = 0 + 1 = 1$ & $1 + 1' = 1 + 0 = 1$

b) $x \cdot x' = 0$ since $0 \cdot 0' = 0 \cdot 1 = 0$ and $1 \cdot 1' = 1 \cdot 0 = 0$

which verifies postulate 5

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

Basic theorems and properties of Boolean algebra

Duality

The Huntington postulates have been listed in pairs and designated by part (a) and part (b). One part may be obtained from the other if the binary operators and the identity elements are interchanged. This important property of Boolean algebra is called the duality principle.

Basic Theorems

Table below lists six theorems of Boolean algebra and four of its postulates

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) $x + 1 = 1$	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) $x(y + z) = xy + xz$	b) $x + yz = (x + y)(x + z)$
Theorem 5	Demorgan's	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 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 $(x')' = x$.

The theorems involving two or three variables may be proven algebraically from the postulates and the theorems that have already been proven.

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

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

$= x(1 + y)$ 4(a)

$= x(y + 1)$ 3(a)

$= x \cdot 1$ 2(a)

$= x$ 2(b)

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

The theorems of Boolean algebra can be shown to hold true by means of truth table. In truth tables, both sides of relation are checked to yield identical results for all possible combinations of variables involved. The following truth table verifies the first absorption theorem, $x + xy = x$

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

The algebraic proofs of the absorptive law and DeMorgan's theorem are long and will not be shown here. However, their validity is easily shown with truth tables for example, the truth table for the first DeMorgan's theorem $(x+y)' = x'y'$ is shown below

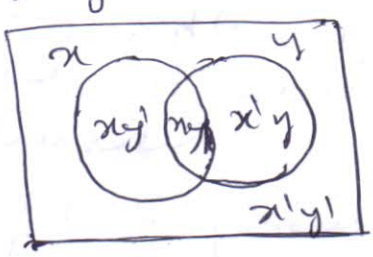
x	y	$x+y$	$(x+y)'$	x'	y'	$x'y'$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

Operator Precedence

The operator precedence for evaluating Boolean expressions is (1) parentheses (2) NOT (3) AND and (4) OR. In other words, the expression inside the parentheses must be evaluated before all other operations. The next operation that holds precedence is the complement, then follows the AND and finally the OR

Venn Diagram

A helpful illustration that may be used to visualize the relationship among the variables of a Boolean expression is the Venn diagram. This diagram consists of a rectangle such as shown below, inside of which are drawn overlapping circles, one for each variable. Each circle is labeled by a 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.

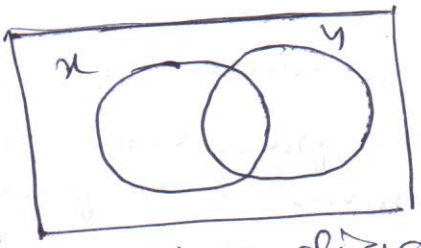


Venn diagram for two variables

Take, for example, the circle labeled x, if we are inside the circle, we say that x=1 when outside, we say x=0. Now, with two overlapping circles, there are four distinct areas inside the rectangle:

① the area not belonging to either x or y
② the area inside circle y but outside x (xy')
③ the area inside circle x but outside y (xy')
and ④ the area inside both circles (xy)

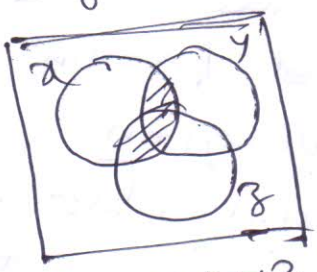
Venn diagrams may be used to illustrate the postulates of Boolean algebra or to show the validity of theorems.



Venn diagram ; illustration $x = xy + x$

The area belonging to the xy is inside the circle x and therefore $x + xy = x$.

$x(y+z)$



$xy + yz$

Venn diagram illustration of the distributive law.

Above fig illustrates the distributive law $x(y+z) = xy + xz$. For this diagram, we have three overlapping circles, one for each of the variables x, y, and z. It is possible to distinguish eight distinct areas in a three-variable Venn diagram. For this particular example, the distributive law is demonstrated by noting that the area intersecting the circle x with the area enclosing y or z is the same area belonging to xy or xz.

2.4. Boolean functions

A binary variable can take the value of 0 or 1. A Boolean function is an expression formed with binary variables, the two binary operators OR and AND and unary operator NOT, parentheses and an equal sign. For a given value of the variables, the function can be either 0 or 1

Consider the Boolean function

$$F_1 = xyz'$$

The function F_1 is equal to 1 if $x=1$ and $y=1$ and $z'=1$; otherwise $F_1=0$. The above is an example of a Boolean function represented as an algebraic expression.

A Boolean function may also be represented in a truth table. To represent a function in a truth table, we need a list of the 2^n combinations of 1's & 0's of the n binary variables & a column showing the combinations for which the function is equal to 1 or 0

Truth table for $F_1 = xyz'$, $F_2 = x + y'z$

$F_3 = x'y'z + xy'z + xy'$ & $F_4 = xy' + xz$

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	0	1	0	0
1	1	1	1	1	0	0

$F_2 = x + y'z$

$F_2 = 1$ if $x=1$ or if $y=0$ while $z=1$

In table 2. $x=1$ in the last four rows & $yz=01$ in rows 001 & 101. The latter combination applies also for $x=1$. Therefore, there are five combinations that make $F_2=1$

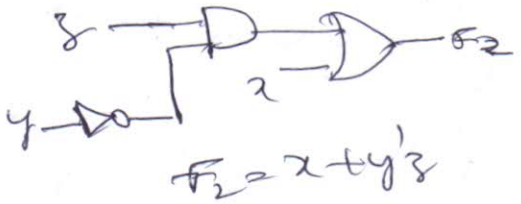
$F_3 = x'y'z + x'yz + xy'$

This is shown in table with four 1's and four 0's. F_4 is the same as F_3 and is considered below.

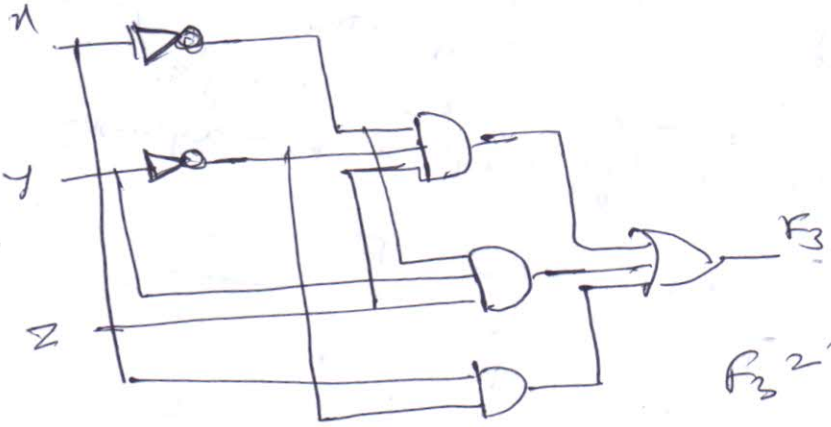
Algebraic Manipulation

When a Boolean function is implemented with logic gates, each literal 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 a circuit with less equipment. It is not always possible to minimize both simultaneously; usually, further criteria must be available.

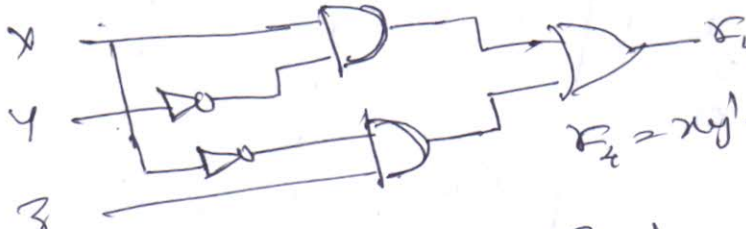
$F_1 = xyz'$



$F_2 = x + y'z$



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



$F_4 = xy' + x'z$

Simplify the following Boolean functions to a minimum number of literals.

1) $x + x'y = (x + x')(x + y) = 1 \cdot (x + y) = x + y$

2) $x(x + y) = xx' + xy = 0 + xy = xy$

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

$= x'z + xy'$

4) $xy + x'z + yz = xy + x'z + yz(x+x')$

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

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

$= xy + x'z$

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

Complement of a function

The complement of a function F is F' and is obtained from a 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 DeMorgan's theorem

Three variable DeMorgan's theorem

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

$= A'X'$ 5(a)

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

$= A'.(B'C')$ 5(a) 4(b)

$= A'B'C'$

De Morgan's theorems for any number of variables resemble in the form the two variable case

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

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

DeMorgan's theorem: states that the complement of a function is obtained by interchanging AND and OR operators and complementing each literal.

Eg: Find the complement of the functions $F_1 = x'y'z' + x'y'z$ and $F_2 = x(y'z' + yz)$ by applying DeMorgan's theorem as many times as necessary

Soln. $F_1' = (x'y'z' + x'y'z) = (x'y'z')'(x'y'z)'$

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

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

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

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

Find the complement of the functions F_1 and F_2 by taking their duals & complementing each literal

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

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

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

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

The dual of F_2 is $x + (y' + z')(y + z)$

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

Canonical and Standard forms

Minterms and Maxterms

A binary variable may appear either in its normal form (x) or in its complement form (x'). Now consider two binary variable x and y combined with an AND operation. Because each variable may appear in either form, there are four combinations: $x'y'$, $x'y$, xy', xy

Each of these four AND terms represents one of the distinct areas in the Venn diagram & is called a minterm or a standard product.

n variables forming an OR term with each variable being primed or unprimed, provide 2^n possible combination, called maxterms or standard sums

Minterms and Maxterms for 3 binary variables

			Minterms		Maxterms	
			Term	Designation	Term	Designation
x	y	z				
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

001 100 111

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

$f_2 = x'yz + xy'z + xyz' + xyz = m_3 + m_5 + m_6 + m_7$

Functions of three variables

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	1	1
1	1	1		

$f_1' = x'y'z' + xyz' + x'yz + xy'z + xyz$

Minterms of f_1' are the complement of f_1

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

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

Maxterms

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

$= M_0 M_1 M_2 M_4$

Eg: Express the Boolean function $F = A + B'C$ in a sum of minterms. The function has three variables A, B and C. The first term A is missing two variables; therefore

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

This is still missing one variable;

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

The second term $B'C'$ is missing one variable

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

Combining all terms, we have

$$F = A + B'C$$
$$= ABC + ABC' + AB'C + AB'C' + AB'C + A'B'C$$

But $AB'C$ appears twice, and according to theorem $((x + x) = x)$ it is possible to remove one of them

Rearranging the minterms in ascending order

$$F = A'B'C + AB'C' + AB'C + ABC' + ABC$$
$$m_1 + m_4 + m_5 + m_6 + m_7$$

In short notation

$$F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$$

$\Sigma \Rightarrow$ ORing

An alternate procedure for deriving the minterms of a Boolean function is to obtain the truth table of the function directly from the algebraic expression and then read the minterms from the truth table

Truth table for $F = A + B'C$

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

Eg: Express the Boolean function $F = xy + x'z$ in a product of maxterms form. First convert the function into OR terms using the distributive laws;

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

The function has three variables; x, y and z. Each OR term is missing one variable;

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

combining all the terms and removing those that appear more than once, we finally obtain

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

A convenient way to express this function is as follows:

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

$\Pi \Rightarrow$ ANDing of maxterms

Conversion between canonical forms

$F(A, B, C) = \Sigma(1, 4, 5, 6, 7)$

This has a complement that can be expressed as

$$F'(A, B, C) = \Sigma(0, 2, 3) = m_0 + m_2 + m_3$$

Now, if we take the complement of F' by DeMorgan's theorem, we obtain F in a different form;

$$F = (m_0 + m_2 + m_3)' = m_0' \cdot m_2' \cdot m_3' = M_0 M_2 M_3 = \Pi(0, 2, 3)$$

From the table it is clear that the following relation holds true

$$m_j' = M_j$$

$F = xy + x'z$

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

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

Since there are a total of eight minterms or maxterms in a function of three variable. missing terms to be 0, 2, 4, and 5

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

Stand forms

In this configuration, the terms that form the function may contain one, two or any number of literals

Sum of products

$F_1 = y' + xy + x'y'z'$

Product of sums

$F_2 = x(y' + z)(x' + y + z' + w)$

non standard form

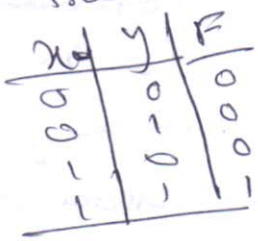
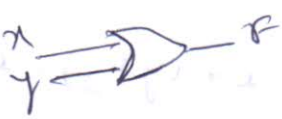
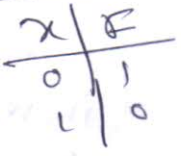
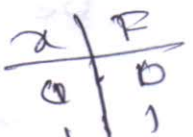
$F_3 = (AB + CD)(A'B' + C'D')$

It is neither sum of products nor in product of sums It can be changed to a standard form by using the distributive law to remove the parentheses $F_3 = A'B'CD + ABC'D'$

Boolean expressions for the 16 functions of two variables

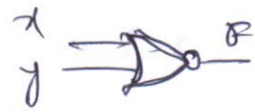
Boolean Functions	Operator Symbol	Name	Comments
$F_0 = 0$		Null	Binary constant 0
$F_1 = xy$	$x \cdot y$	AND	x and y
$F_2 = xy'$	x/y	Inhibition	x but not y
$F_3 = x$		Transfer	x
$F_4 = x'y$	y/x	Inhibition	y but not x
$F_5 = y$		Transfer	
$F_6 = xy' + x'y$	$x \oplus y$	Exclusive-OR	x or y but not both
$F_7 = x + y$	$x + y$	OR	x or y
$F_8 = (x+y)'$	$x \downarrow y$	NOR	Not OR
$F_9 = xy + x'y'$	$x \odot y$	Equivalence	x equals y
$F_{10} = y'$	y'	Complement	Not y
$F_{11} = x + y'$	$x \subset y$	Implication	If y then x
$F_{12} = x'$	x'	Complement	Not x
$F_{13} = x' + y$	$x \supset y$	Implication	If x then y
$F_{14} = (xy)'$	$x \uparrow y$	NAND	Not AND
$F_{15} = 1$		Identity	Binary constant 1

Digital logic Gates

Name	Graphic Symbol	Algebraic function	Truth table
AND		$F = xy$	
OR		$F = x + y$	
Inverter		$F = x'$	
Buffer		$F = x$	

NAND  $F = (xy)'$

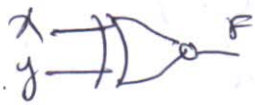
x	y	F
0	0	1
0	1	1
1	0	1
1	1	0

NOR  $F = (x+y)'$

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

Exclusive-OR (XOR)  $F = xy' + x'y$
$= x \oplus y$

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

Exclusive-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

Extension of Multiple Inputs

Except for the inverter and buffer, can be extended to have more than two inputs. A gate can be extended to have multiple inputs if the binary operation is represented is commutative and associative The AND and OR operation defined in Boolean algebra possess these two properties

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

and $(x+y)+z = x+(y+z) = x+y+z$ associative

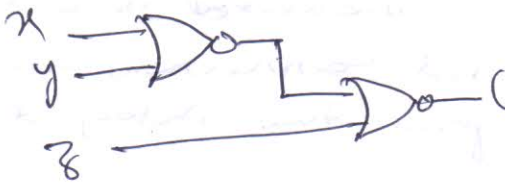
which indicates that the gate inputs can be interchanged and that the OR function can be extended to three or more variables.

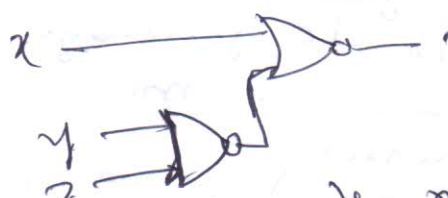
The NAND and NOR functions are commutative and their gates can be extended to have more than two inputs, provided the definition of the

operation is slightly modified. The difficulty is that the NAND and NOR operators are not associative ie, $(x \downarrow y) \downarrow z \neq x \downarrow (y \downarrow z)$

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

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





Demonstrating the nonassociativity of the NOR operator.

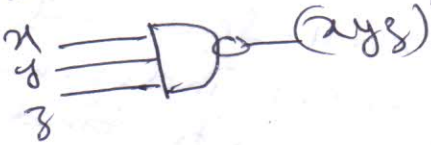
To overcome this difficulty, we define the multiple NOR (or NAND) gate as a complemented OR (or AND) gate as

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

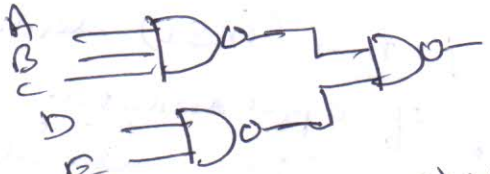
$$x \uparrow y \uparrow z = (xyz)'$$

3-input NOR

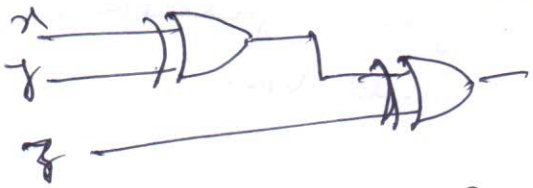
3-input NAND gate



$$F = [(ABC)'(DE)']' = ABC + DE$$



a) using 2-input gate

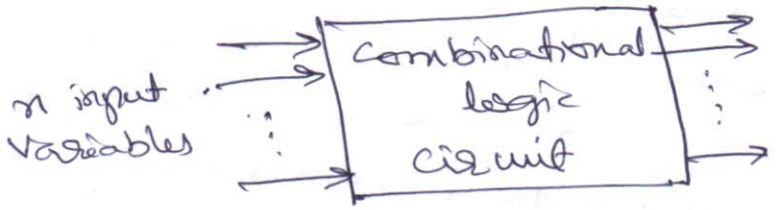


b) 3-input gate

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

Introduction

A combinational circuit consists of input variables, logic gates and output variables. The logic gates accept signals from the inputs and generate signals to the outputs. This process transforms binary information from the given input data to the required output data. Obviously, both input and output data are represented by binary signals i.e, there exists in two possible value, one representing logic-1 and the other logic-0. A block diagram of a combinational circuit is shown below.



The n input binary variables come from an external source; the m output variables go to an external destination. In many applications, the source and/or destination are storage registers located either in the vicinity of the combinational circuit or in a remote external device. By definition, an external register does not influence the behavior of the combinational circuit because, if it does, the total system becomes a sequential circuit

For n-input variables, there are 2^n possible combinations of binary input values. For each possible input combination, there is one and only one possible output combination. A combinational circuit can be described by m Boolean functions, one for each output variable Each output function is expressed in terms of the n input variables.

The procedure 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 outputs is derived.
5. The simplified Boolean function for each output is obtained
6. The logic diagram is drawn.

A truth table for a combinational circuit consists of input columns and output columns. The 1's and 0's in the input columns are obtained from the 2^n binary combination available for n input variables. The binary values for the output are determined from examination of the stated problem An output can be equal to either 0 or 1 for every valid input combination. However the specification may indicate that some input combination will not occur. These combinations become don't care conditions.

The output Boolean functions from the truth table are simplified by any available method, such as algebraic manipulation, the map method, or the tabulation procedure.

A practical design method would have to consider such constraints as ① minimum number of gates ② minimum number of inputs to a gate ③ minimum propagation time of the signal through the circuit 4) minimum number of interconnections & ⑤ limitations of the driving capabilities of each gate.

Adders

Digital computers perform a variety of information-processing tasks. Among the basic function encountered are the various arithmetic operations. The most basic arithmetic operation, no doubt, is the addition of two binary digits. This simple addition consists of four possible elementary operations $0+0=0$, $0+1=1$, $1+0=1$, $1+1=10$. The first three operations produce a sum whose length is one digit, but when both augend & addend bits are equal to 1 the binary sum consists of two digits. The higher significant bit of this result is called a carry. When the augend and addend numbers contain more significant digits, the carry obtained from the addition of two bits is added to the next higher-order pair of significant bits. A combinational circuit that performs the addition of two bits is called a half adder. One that performs the addition of three bits (two significant & a previous carry) is a full adder

Half Adder

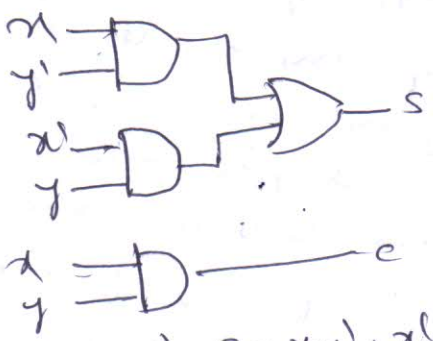
A half adder circuit needs two binary inputs and two binary outputs. The input variables designate the augend and addend bits; the output variables produce the sum and carry. It is necessary to specify two output variables because the result may consist of two binary digits.

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

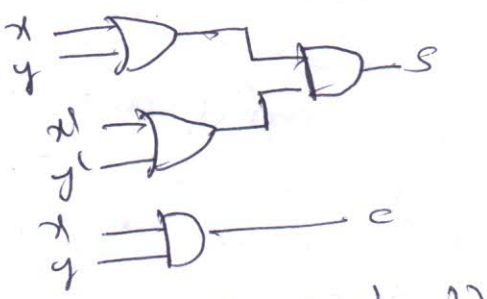
The carry output is 0 unless both inputs are 1. The S output represents the least significant bit of the sum.
The simplified Boolean functions for the two outputs can be obtained directly from the truth table. The simplified sum of products expressions are

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

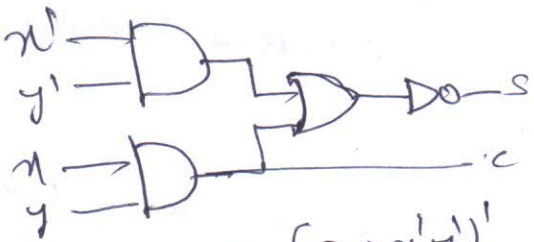
Four possible implementation are as follows



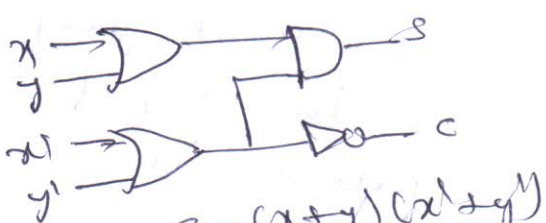
a) $S = xy' + x'y$
$C = xy$



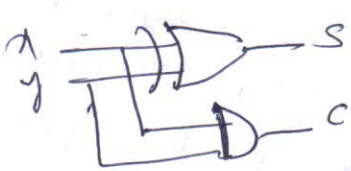
$S = (x+y)(x'+y')$
$C = xy$



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



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



$S = x \oplus y$
$C = xy$

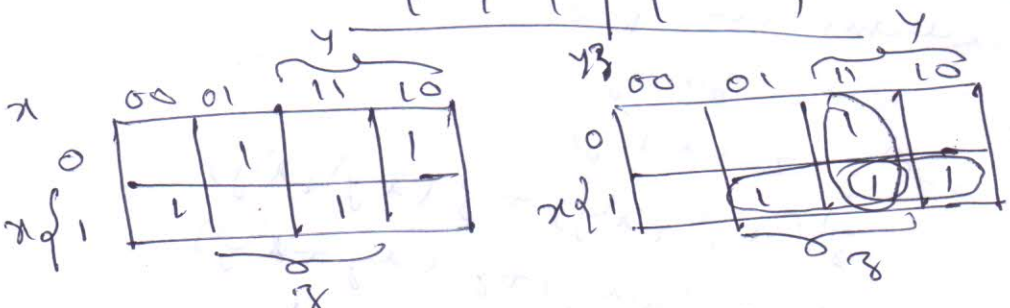
They all achieve the same result as far as the input-output behaviour is concerned.

Full Adder

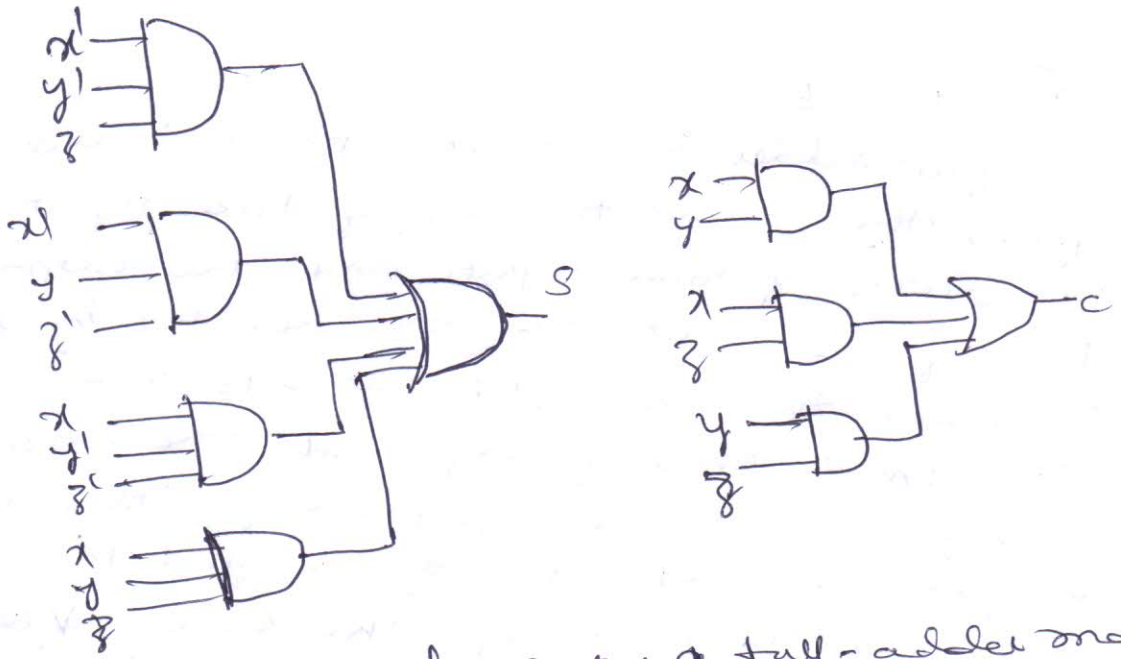
A full-adder is a combinational circuit that forms the arithmetic sum of three input bits. It consists of three inputs and two outputs. Two of the input variables, denoted by x and y, represent the two significant bits to be added. The third input z, represents the carry from the previous lower significant position. The two outputs are designated by the symbols S for Sum and C for carry. The binary variable S gives the value of the least significant bit of the sum. The binary variable C gives the output carry.

The truth table of the full adder is

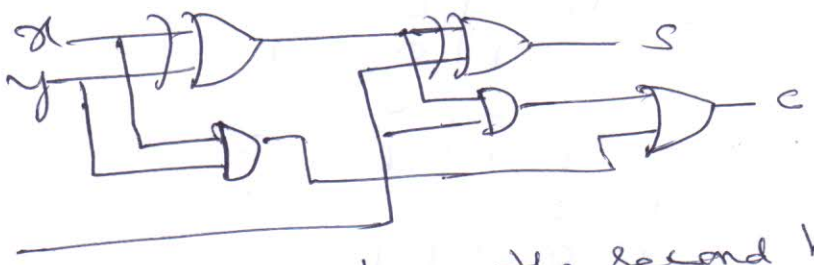
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1



$S = x'y'z + x'yz' + xy'z' + xyz$ $\quad C = xy + xz + yz$



Other configurations for a full-adder may be developed. The product of sum implementation requires the same number of gates as above with the number of AND and OR gates interchanged. A full-adder can be implemented with two half-adders and one OR gate as shown below.



The S output from the second half-adder is the exclusive-OR of z and the output of the first half-adder, giving

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

and the carry output is

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