Module 3

Boolean Algebra and Logic Circuits

Syllabus

Boolean Algebra and Logic Circuits: Binary numbers, Number Base Conversion, octal &Hexa-Decimal Numbers, Complements, Basic definitions, Axiomatic Definition of Boolean Algebra, BasicTheorems 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

Digital systems

Digital systems such as computers, smartphones, calculators etc. processes information in digital form or binary form. Digital systems have only two discrete values, 0s and 1s unlike analog systems which can take a continuous range of values.

Advantages of digital systems:

- Digital signals can convey information with less noise, distortion, and interference.
- Digital signal processing is more secure because digital information can be easily encrypted and compressed.
- Digital systems are more accurate, and the probability of error occurrence can be reduced by employing error detection and correction codes, hence reliable.
- Digital signals can be easily stored on any magnetic media or optical media using semiconductor chips, can be implemented in form of ICs.

Binary numbers

Decimal number is of base 10.

Binary numbers are numbers expressed in base-2 number system and use only 2 symbols: "0" and "1". Each digit is referred to as bit.

Eg.(1): Decimal equivalent of 11010 =

$(1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) = 16 + 8 + 2 = 26$

Eg.(2): Decimal equivalent of 11010.11 =

$(1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) + (1 \times 2^{-1}) + (1 \times 2^{-2}) = 26.75$

Decimal numbers 1 to 15 can be represented in 4 bit binary as follows:

Decimal	Binary
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111

Decimal	Binary
8	1000
9	1001
10	1010
11	1011
12	1100
13	1101
14	1110
15	1111

Octal and hexadecimal systems

Number system with base (or radix) eight is called octal number system. In this system, eight symbols 0, 1, 2, 3, 4, 5, 6 and 7 are used to represent numbers.

Number system with base (or radix) sixteen is called hexadecimal number system. In this system, 16 symbols are used to represent numbers mainly 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E and F.

Number Base Conversions

Numbers can be converted from one system to the other in the following ways:

1) **Any number system to decimal**

A number expressed in base r can be converted to decimal equivalent by multiplying each coefficient with the corresponding power of r and adding. Number is separated into integer part and fractional part.

Example: **a) Binary to decimal**

Convert $(1010.011)_2$ to decimal.

$(1010.011)_2 = (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (0 \times 2^0) + (0 \times 2^{-1}) + (1 \times 2^{-2}) + (1 \times 2^{-3})$
$= 8 + 2 + 0.25 + 0.125$
$= (10.\ 375)_{10}$

b) Octal to decimal

Convert $(630.4)_8$ to decimal.

$(630.4)_8 = (6 \times 8^2) + (3 \times 8^1) + (0 \times 8^0) + (4 \times 8^{-1})$
$= 384 + 24 + 0 + 0.5$
$= (408.5)_{10}$

2) **Octal and hexadecimal to binary or vice versa**

As $2^3 = 8$ and $2^4 = 16$, each octal digit corresponds to three binary digits and each hexadecimal digit corresponds to four binary digits. Conversion from binary to octal or hexadecimal is done by partitioning binary number into groups of three or four bits starting from the binary point and proceeding to left or right.

Example:

1) Convert $(10110001101011.111100000110)_2$ to octal.

$(10\ 110\ 001\ 101\ 011.\ 111\ 100\ 000\ 110)_2 = (26153.7406)_8$

2) Convert $(10110001101011.\ 11110010)_2$ to hexadecimal.

$(10\ 1100\ 0110\ 1011.\ 1111\ 0010)_2 = (2C6B.\ F2)_{16}$

3) $(673.124)_8 = (110\ 111\ 011.\ 001\ 010\ 100)_2$

4) $(306.D)_{16} = (0011\ 0000\ 0110.\ 1101)_2$

3) Decimal to any other system

i) Decimal to binary

Convert $(13)_{10}$ to binary.

To convert to binary, divide by 2.

13 / 2, Q = 6, R = 1

6 / 2, Q = 3, R = 0

3 / 2, Q = 1, R = 1

1/ 2, Q = 0, R = 1

Stop when Q = 0.

Hence, binary representation is $(1101)_2$.

To convert fractional part of decimal to binary, multiply by 2 and take the integer part alone.

Example: Convert $(0.65625)_{10}$ to binary.

0. 65625 x 2 = **1**.31250

0. 3125 x 2 = **0**.6250

0. 625 x 2 = **1**.2500

0. 25 x 2 = **0**. 5000

0. 5 x 2 = **1**. 0000

Now it can be stopped. Hence, binary representation is $(0.10101)_2$

ii) Decimal to octal

Convert $(153)_{10}$ to octal

To convert to octal, divide by 8.

153 / 8, Q = 19, R = 1

19 / 8, Q = 2, R = 3

2 / 8, Q = 0, R = 2

Hence, octal representation is $(231)_8$.

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

0. 513 x 8 = **4**.104

0. 104 x 8 = **0**.832

0. 832 x 8 = **6**.656

0. 656 x 8 = **5**.248

0. 248 x 8 = **1**.984

Hence, octal representation is $(0.\ 40651\ldots\ldots)_8$

Complements

Complements are used in digital computers to simplify subtraction operation and for logical manipulation. There are two types of complements for each base-r system:

- Radix complement or r's complement
- Diminished radix complement or (r-1)'s complement

For binary numbers, it is 2's complement and 1's complement. For decimal numbers, it is 10's complement and 9's complement.

Given a number N in base r having n digits,

r's complement of N is defined as $r^n - N$.

(r-1)'s complement of N is defined as $(r^n - 1) - N$.

Example, consider decimal system with base 10.

To find 10's complement of 7

N = 7, no.of digits n = 1 (7 is single digit no), r is base = 10

10's complement = $r^n - N = 10^1 - 7 = 10 - 7 = 3$

10's complement of $9 = 10^1 - 9 = 1$

10's complement of $5690 = 10^4 - 5690 = 4310$

To find 9's complement of 546700,

$(r^n - 1) - N = 10^6 - 1 - 546700 = 453299$

For binary numbers, r =2 and r-1 =1

1's complement of 1101 = $2^4 - 1 - 1101 = (16)_{10} - 1 - 1101 = (15)_{10} - 1101$

= 1111-1101 = 0010

It can be seen that 1's complement of binary number is formed by changing 1's to 0's and 0's to 1's.

2's complement of 1101 = $2^4 - 1101 = (16)_{10} - 1101 = 10000 - 1101 = 0011$

It can be seen that 2's complement of binary number is formed by adding 1 to 1's complement.

In binary subtraction, subtracting B from A is equivalent to adding A to 2's complement of B. If final carry is generated, then the result is positive, final carry is neglected. If final carry is not produced, result is negative. In that case, take its 2's complement to get final result.

Example: To subtract 1111- 1101

A= 1111, B = 1101

2's complement of B = 0011
1111 + 0011 = 10010, carry =1, neglect it, result is positive
Final result = 0010

To subtract 0111 – 1010

A= 0111, B = 1010

2's complement of B = 0110

0111 + 0110 = 1101

No carry, result is negative.

So 2's complement of 1110 = 0011

Final result is 0011 and it is negative.

Boolean Algebra

Boolean algebra is a branch of mathematics that deals with operations on logical values with binary variables. The Boolean variables are represented as binary numbers to represent truths: 1 = true and 0 = false.

Basic definitions:

- Boolean algebra

It is a set of elements, a set of operators, and a number of unproved axioms or postulates.

- Set of elements is any collection of objects, usually having a common property.

Example: A = {1, 2, 3, 4} indicates that set A has the elements of 1, 2, 3, and 4.

- A binary operator defined on a set S of elements is a rule that assigns, to each pair of elements from S, a unique element from S.

The most common postulates used to formulate various algebraic structures are:

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.
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 € S.
3. Commutative law: A binary operator * on a set S is said to be commutative whenever x * y = y * x for all x, y € 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 € S with the property that
   e * x = x * e = x for every x € S

Example: The element 0 is an identity element with respect to the binary operator + on the set of integers I = {c, -3, -2, -1, 0, 1, 2, 3, c}, since x + 0 = 0+ x = x for any x € I

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

1. 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 € S, there exists an element y € S such that x * y = e

Example: In the set of integers, I, and the operator +, with e = 0, the inverse of an element a is (-a), since a + (-a) = 0.

2. Distributive law. If * and • are two binary operators on a set S, * is said to be distributive over • whenever, x * (y • z) = (x * y) • (x * z).

Field:

A field is an example of an algebraic structure.

• The field of real numbers is the basis for arithmetic and ordinary algebra.

– The binary operator + defines addition.

– The additive identity is 0.

– The additive inverse defines subtraction.

– The binary operator • defines multiplication.

– The multiplicative identity is 1.

– For a ≠ 0, the multiplicative inverse of a = 1/a defines division (i.e., a •1/a = 1).

– The only distributive law applicable is that of • over + is: a • (b + c) = (a • b) + (a • c).

Axiomatic Definition of Boolean Algebra:

1854: George Boole developed an algebraic system called Boolean algebra.

1904: E. V. Huntington formulated a set of postulates that formally define the Boolean algebra.

1938: C. E. Shannon introduced a two-valued Boolean algebra called switching algebra that represented the properties of bistable electrical switching circuits.

Duality property:

All binary operations remain valid when following two steps are performed:

1) Interchange OR and AND operators.

2) Replace all 1s by 0s and 0s by 1s.

Huntington postulates:

1. (a) The structure is closed with respect to the operator +.

(b) The structure is closed with respect to the operator •.

2. (a) The element 0 is an identity element with respect to +; that is, x + 0 = 0 + x = x.

(b) The element 1 is an identity element with respect to •; that is, x • 1 = 1 • x = x.

3. (a) The structure is commutative with respect to +; that is, x + y = y + x.

(b) The structure is commutative with respect to • ; that is, x • y = y • x.

4. (a) The operator • is distributive over +; that is, x • (y + z) = (x • y) + (x • z).

(b) The operator + is distributive over •; that is, x + (y • z) = (x + y) • (x + z).

5. For every element x € B, there exists an element x € B (called the complement of x)such that (a) x + x' = 1 and (b) x • x' = 0.

6. There exist at least two elements x, y € B such that x ≠ y.

• Comparing Boolean algebra with arithmetic and ordinary algebra

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

2. The distributive law of + over • (i.e., x + (y • z) = (x + y) • (x + z)) is valid for Booleanalgebra, but not for ordinary algebra.

3. Boolean algebra does not have additive or multiplicative inverses; therefore, there areno subtraction or division operations.

4. Operator called complement is not available in ordinaryalgebra.

5. Ordinary algebra deals with the real numbers, which constitute an infinite set ofelements. Boolean algebra is defined as a set with only two elements 0 and 1.

Two-valued Boolean Algebra

Two valued Boolean algebra is a set of two elements with operations:

+ : OR operation; · : AND operation, complement operator: NOT operation

Example: Binary logic is a two-valued Boolean algebra also called "switching algebra" by engineers

• B = {0,1}

• Operations

AND

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

OR

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

NOT

A	Y
0	1
1	0

- Closure: the result of each operation is either 1 or 0 and 1, 0 € B.
- Identity elements: 0 for + and 1 for ·
- Commutative laws are obvious from truth tables.
- Distributive laws:
  x·(y + z) = (x·y) + (x·z)
  x+ (y·z) = (x+y)·(x+z)
- Complement
  x+x'=1: 0+0'=0+1=1; 1+1'=1+0=1
  x·x'=0: 0·0'=0·1=0; 1·1'=1·0=0

Basic theorems and properties of Boolean Algebra

In Duality,

Binary operators are interchanged; AND is replaced by OR.

Identity elements are interchanged; 1 is replaced by 0.

Postulates of Boolean Algebra:

1a) x + 0 = x 1b) x . 1 = x

2a) x + x' = 1 2b) x. x' = 0 2c) x + 1 =1 2d) x.0 =0

Commutative:

3a) x + y = y + x 3b) xy = yx

Distributive:

4a) x (y + z) = xy + xz4b) x + yz = (x + y). (x + z)

Associative:

5a) x + (y + z) = (x+ y) = z 5b) x (yz) = (xy) z

Theorems:

Theorem 1(a): x + x = x

Proof:

x+x = (x+x) •1	by postulate:	1b
= (x+x) (x+x')		2a
= x+xx'		4b
= x+0		2b
= x		1a

Theorem 1(b): x • x = x

Proof:

x • x = xx + 0	by postulate:	1a
= xx + xx'		2b
= x (x + x')		4a
= x • 1		2a
= x		1b

Theorem 1b) is the dual of theorem 1a).

Theorem 2(a): x + 1 = 1

Proof:

x + 1 = 1 • (x + 1)	by postulate:	1b
= (x + x')(x + 1)		2a
= x + x' • 1		4b
= x + x'		1b
= 1		2a

Theorem 2(b): x • 0 = 0 by duality

Theorem 3: (x')' = x

Complement of x' is x, so (x')' = x

Theorem 4a: x + xy = x

x + xy = x • 1 +xy	by postulate:	1b
= x (1 +y)		4a
= x • 1		2c
= x		1b

Theorem 4(b): x (x + y) = x by duality

From truth table,

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

Theorem 5: DeMorgan's theorems

5a: (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

Column 4 = Column 7

5b: (x y)' = x' + y'

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

Column 4 = Column 7

Operator precedence for evaluating Boolean expressions is

a) Parenthesis b) not c) and d) or.

Boolean functions:

Boolean function is an algebraic expression consisting of binary variables, binary operators OR and AND, unary operator NOT and parentheses. It expresses the logical relationship between binary variables and is evaluated by determining the binary value of the expression for all possible values of the variables.

• Examples

1) **F1= x + y z'**

It means F1 = 1 if x = 1 or if y = 0 and z = 1, otherwise F1 = 0.

2) **F2 = x' y' z + x' y z + x y'**

It means F2 = 1 if (x = 0, y = 0, z = 1) or (x = 0, y = 1, z = 1) or (x = 1, y = 0), otherwise F2 = 0.

Truth Table

• Boolean function can be represented in a truth table.

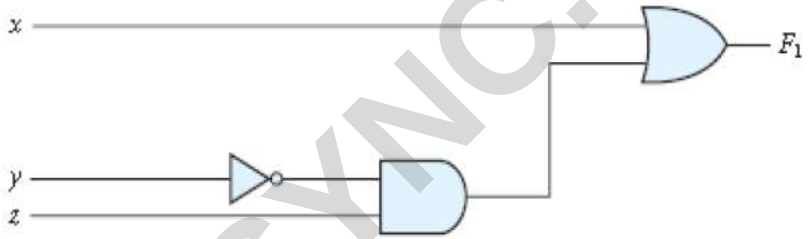
• Truth table has 2^n rows where n is the number of variables in the function.

• The binary combinations for the truth table are obtained from the binary numbersby counting from 0 through 2^n- 1.

Example: Truth table for F1 and F2 can be written as

x	y	z	F1	F2
0	0	0	0	0
0	0	1	1	1
0	1	0	0	0
0	1	1	0	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	0
1	1	1	1	0

Implementation of F1 with logic gates



F1= x + y z'

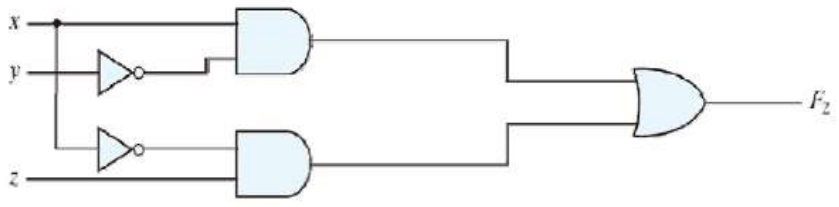
F2 can be simplified using the rules of Boolean Algebra.

F2 = x'y'z + x'yz + xy'

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

= x'z + xy'

Implementation of F2 with logic gates



So Boolean expressions can be simplified using the rules of Boolean Algebra.

Literal: a complemented or un-complemented variable (an input to a gate).

Minimization of the number of literals results in a simple circuit with less number of gtes.

F2 = x'y'z + x'yz + xy' has 3 terms with 8 literals. It can be simplified as:

F2 = x'z (y' + y) + xy' = x'z + xy' .

Now the simplified function has 2 terms and 4 literals only.

Minterm and maxterm:

Minterm (standard product): an AND term consisting of all literals in their normal form or in their complement form

• For example, two binary variables x and y, has 4 minterms: xy, xy', x'y, x'y'

• n variables can be combined to form 2^n minterms.

Maxterm (standard sum): an OR term consisting of all literals in their normal form or in their complement form.

•Each maxterm is the complement of its corresponding minterm, and vice versa.

			Minterms		Maxterms	
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

Any Boolean function can be expressed as sum of products or minterms (SOP form) or product of sums or maxterms (POS form).

SOP is expressed in the form of ∑ m.

POS is expressed in the form of π M.

Example: Y 1= A'B'C + ABC' + ABC is an SOP expression.

Y = (A+ B+ C). (A'+B'+C') is a POS expression.

Conversion between SOP and POS

Let SOP form be F(A,B,C) = ∑ (1,4,5,6,7) = m1 + m4 + m5 + m6 + m7

POS form is complement of SOP form.

F'(A,B,C) = π M (0,2,3) = M0. M2. M3.

Canonical and Standard forms

Canonical form is a method of representing Boolean outputs of digital circuits using Boolean Algebra in such a way that each term contains all the literals.

Standard form is a method of representing Boolean outputs of digital circuits using Boolean Algebra in such a way that there exists at least one term that does not contain all the literals.

Example:

Y = A'B'C + ABC' + AB'C is in canonical form.

Y = A + BC is in standard form.

Other Logic Operations

For n binary variables, there can be 2^n functions.

Example for four variables, 16 functions are possible.

16 functions can be subdivided into various categories:

a) Two functions that produce a constant 0 or 1.

b) Four functions with unary operations: complement and transfer

c) Ten functions with binary operators that define eight different operations: AND, OR, NAND, NOR, XNOR, equivalence, inhibition and implication.

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	y
$F_6 = xy' + x'y$	$x \oplus y$	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
$F_9 = xy + x'y'$	$(x \oplus y)'$	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

- Complement function produces the complement of each binary variable.
- Function that is equal to input variable has been given the name transfer.
- Equivalence is a function that is 1 when both binary variables are equal (XNOR).

Digital Logic Gates:

The basic digital electronic circuit that has one or more inputs and single output is known as **Logic gate**. Hence, the Logic gates are the building blocks of any digital system. Logic gates are classified into three categories: Basic Gates (AND, OR, INVERTER), Universal Gates (NAND, NOR) and Special Gates (Ex-OR, Rx-NOR).

Name	Distinctive-Shape Graphics Symbol	Algebraic Equation	Truth Table
AND		$F = XY$	X Y | F 0 0 | 0 0 1 | 0 1 0 | 0 1 1 | 1
OR		$F = X + Y$	X Y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 1
NOT (inverter)		$F = \overline{X}$	X | F 0 | 1 1 | 0
NAND		$F = \overline{X \cdot Y}$	X Y | F 0 0 | 1 0 1 | 1 1 0 | 1 1 1 | 0
NOR		$F = \overline{X + Y}$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 0
Exclusive-OR (XOR)		$F = X\overline{Y} + \overline{X}Y$ $= X \oplus Y$	X Y | F 0 0 | 0 0 1 | 1 1 0 | 1 1 1 | 0
Exclusive-NOR (XNOR)		$F = XY + \overline{X}\overline{Y}$ $= X \oplus Y$	X Y | F 0 0 | 1 0 1 | 0 1 0 | 0 1 1 | 1

Fig: Commonly used Logic Gates

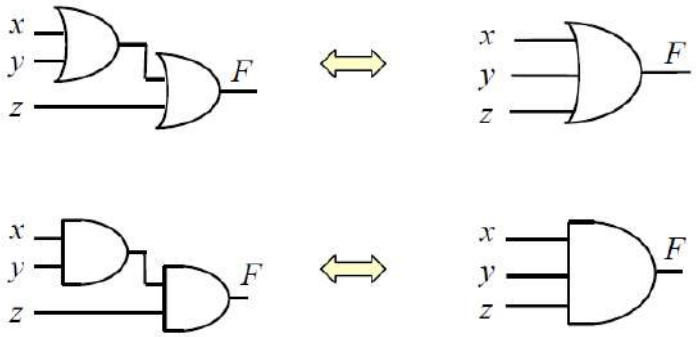
Multiple input gates

AND and OR are commutative and associative.

x + y = y + x and x. y = y . x

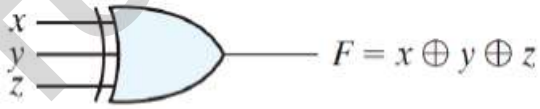
(x + y) + z = x + (y + z) and (x.y).z = x. (y . z)

So 3 input AND and OR gates are as shown:



3-input NAND and NOR gate:

3-input XOR gate is



Combinational and sequential logic

Digital logic circuits are classified into two: Combinational and Sequential

Combinational logic circuits are memoryless logic circuits in which output at any instant of time depends only on the present inputs without regard to previous inputs.

Sequential circuits are circuits with memory, whose outputs depend not only on the present combination of inputs but also on past inputs.

Combinational Logic Circuit

A combinational circuit consists of input variables, logic gates and output variables.

Logic gates accept signals from inputs and generate output signals.

Both input and output data are represented by binary signals. For n input variables, there are 2^n possible combinations of binary input values. For each possible input combination, there is only one possible output combination. Each output function is expressed in terms of n input variables.

Each input variable to a combinational circuit may have one or two wires. Variable is represented either in normal form (unprimed) or in complemented form (primed). To implement primed variables, an inverter is required.

Design procedure of combinational circuits

Design 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) Truth table that defines the relationship between input and output variables is determined.
5) Simplified Boolean function for each output is obtained.
6) Logic diagram is drawn.

Truth table for a combinational circuit consists of input and output columns. 1s and 0s in input columns are obtained from 2^n binary combinations available for input variables. Output may be either 0 or 1 for every valid input combination. Output Boolean functions from truth table are simplified using laws of Boolean algebra, K-map method or tabular method.

Practically, design method will have to consider constraints such as:

a) Minimum number of gates
b) Minimum number of inputs to a gate
c) Minimum propagation time of signal through the circuit.
d) Minimum number of interconnections
e) Limitations of driving capabilities of each gate.

Logic diagram is helpful in visualising gate implementation of expressions.

Adders:

Digital systems like computers perform all arithmetic operations like addition, subtraction, multiplication and division. The most basic operation is binary addition.

Half adder

Half adder is a combinational circuit that performs the addition of two bits and produces sum S and carry C as the outputs.

- Addition of 0 and 0 produces sum = 0, carry =0.
- Addition of 0 and 1 produces sum = 1, carry =0.
- Addition of 1 and 1 produces sum = 1, carry =1

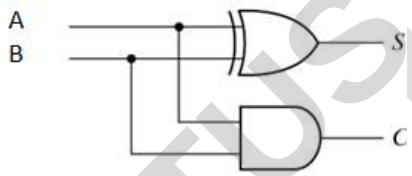
Truth table for Half adder:

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

From truth table, $S = A'B + AB' = A \text{ xor } B$

$C = AB$

Logic circuit for Half adder:



Full adder:

Full adder is a combinational circuit that performs the addition of three bit: A, B and Cin and produces sum S and carry Cout as the outputs.

Truth table for Full adder:

A	B	Cin	S	Cout
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

S = A'B'Cin+ A'BCin' + AB'Cin' + ABCin
= Cin(AB + A'B') + Cin' (A'B + AB')
= Cin (A xnor B) + Cin' (A xor B)
= A xor B xorCin

Cout = A'BCin + AB'Cin + ABCin' + ABCin
= AB (Cin + Cin') + Cin (A xor B)
= AB.1 + Cin (A xor B) = AB + Cin(Axor B)

S = A xor B xorCin
Cout = AB + Cin(Axor B)

Logic circuit for Full adder:

Implementation of Full adder with two half adders and an OR gate.