Number System

Number System

- Computer treats everything in binary number.

Characteristics of any Number System are

- Base/Radix = number of digits in the system
- Four major of number systems are:
    1. Decimal number system
    2. Binary number system
    3. Octal number system
    4. Hexadecimal number system

Decimal number system

- In decimal system, base is10 and 10 symbols (0-9) are used.
- The decimal number system contain ten unique digits: **0,1,2,3,4,5,6,7,8 and 9**.
- e.g. $(1245)_{10}$
- The value attached to the symbol depends on its location with respect to the decimal point.

Binary number system

- The binary number system is a positional weighted system.
- The base or radix of this number system is 2 e.g. $\mathbf{(10101)_2}$
- The symbols used are 0 and 1.
- A single binary digit is called a bit.

Octal number system

- It is also a positional weighted system.
- Its base or radix is 8.
- It has 8 independent symbols- 0,1,2,3,4,5,6 and 7.
- Its base 8 = 2^3 (2 to the power 3) , every 3- bit group of binary can be represented by an octal digit.
- e.g.- $(14623)_8$ and $(277.13)_8$

Hexa-Decimal number system

- It is also a positional weighted system.
- Its base or radix is 16.
- It has 16 independent symbols- 0,1,2,3,4,5,6,7,9 and 10(A), 11(B),12(C),13(D),14(E),15(F). 0r 0,1,2,3,4,5,6,7,8,9 and 10- A, 11- B, 12- C, 13-D, 14- E and 15-F
- e.g.- $(14ABF)_{16}$ and $(12.CD)_{16}$

CONVERSION FROM ONE NUMBER SYSTEM TO ANOTHER

Binary number conversion to-

1. Octal
2. Decimal
3. Hexa-Decimal

OCTAL number conversion to-

1. Binary
2. Decimal
3. Hexa-Decimal

Decimal number conversion to-

1. Binary
2. Octal
3. Hexa-Decimal

Hexadecimal number conversion to-

1. Binary
2. Octal
3. Decimal

Total 12 types to convert the number system into each other

BINARY NUMBER SYSTEM: Binary to Decimal

- In this method, each binary digit of the number is multiplied by its positional weight and the product terms are added to obtain decimal number

Example: 1

(i) Convert $(10101)_2$ to decimal.

Solution :

(Positional weight) $2^4\ 2^3\ 2^2\ 2^1\ 2^0$

Binary number 10101

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

$= 16 + 0 + 4 + 0 + 1$

$= (21)_{10}$

Example-2

(ii) Convert $(111.101)_2$ to decimal.

Solution:

$$(111.101)_2 = (1 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + (1 \times 2^{-3})$$
$$= 4 + 2 + 1 + 0.5 + 0 + 0.125$$
$$= (7.625)_{10}$$

Example-3

Example 26.3. *Convert the binary number 110001 to its equivalent decimal number.*

Solution. The binary number along with its decimal values of various positions is shown.

$\therefore \quad (110001)_2 = 1 \times 2^5 + 1 \times 2^4 + 1 \times 2^0$

$= 32 + 16 + 1 = 49$

1	1	0	0	0	1
2^5	2^4	2^3	2^2	2^1	2^0

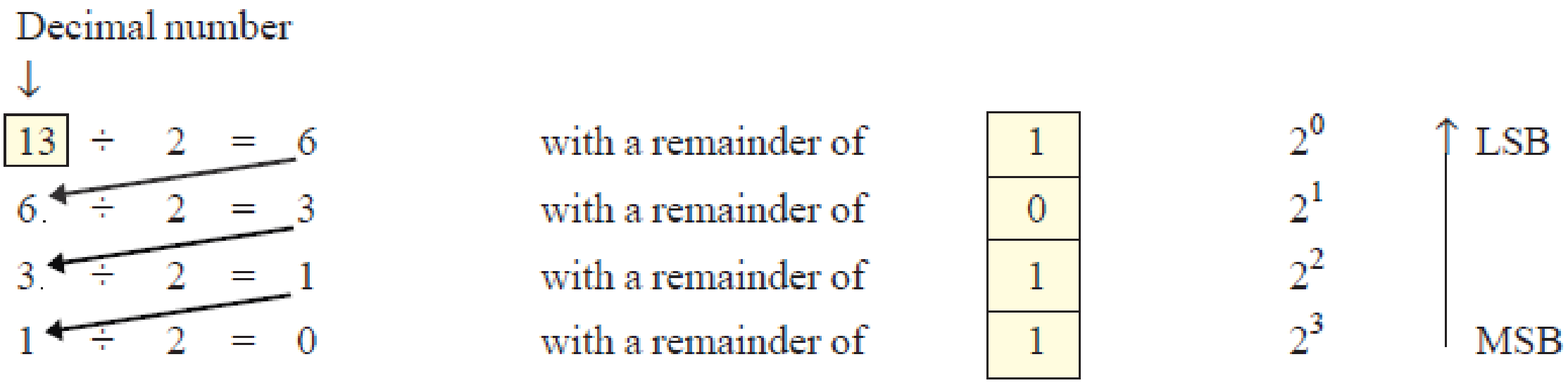
or $(110001)_2 = (49)_{10}$

Decimal to Binary Number

Divide progressively the decimal number by 2 and write down the remainder after each division. Continue this process till you get a quotient of 0 and remainder of 1, the conversion is now complete. The remainders, taken in reverse order, form the binary number [See Fig. 26.4].

Note that 13 is first divided by 2, giving a quotient of 6 with a remainder of 1. This remainder becomes the 2^0 position in the binary number. The 6 is then divided by 2, giving a quotient of 3 with a remainder of 0. This remainder becomes the 2^1 position in the binary number.

Continuing this procedure, the equivalent binary number is 1101.



Example-1

Example 26.1. *Convert the decimal number 37 to its equivalent binary number.*

Solution. Using double-dabble method, we find that the equivalent binary number is 100101. It is a usual practice to mention the base of the number system. The decimal system has a base of 10 while binary system has a base of 2.

$$\therefore \qquad (37)_{10} = (100101)_2$$

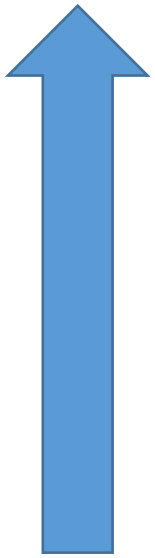
Note. This notation avoids the confusion that may arise because decimal number also involves the digits 0 and 1. Thus, $(101)_{10}$ denotes the decimal number hundred one while the binary number $(101)_2$ is equivalent to decimal number 5.

2	37
2	18 – 1
2	9 – 0
2	4 – 1
2	2 – 0
2	1 – 0
	0 – 1

Example-2

Convert $(52)_{10}$ into binary.

2	52	
2	26	— 0
2	13	— 0
2	6	— 1
2	3	— 0
2	1	— 1
	0	— 1



Result of $(52)_{10}$ is **$(110100)_2$**

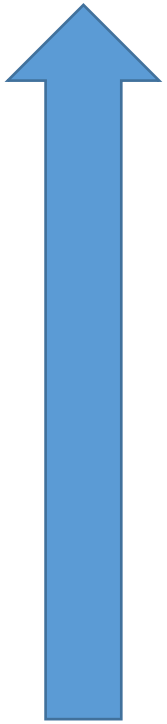
Example-3

(ii) Convert $(105.15)_{10}$ into binary.

Solution:

Integer part

2 | 105		
2 | 52	—	1
2 | 26	—	0
2 | 13	—	0
2 | 6	—	1
2 | 3	—	0
2 | 1	—	1
0	—	1



Fraction part

0.15 x 2 = 0.30

0.30 x 2 = 0.60

0.60 x 2 = 1.20

0.20 x 2 = 0.40

0.40 x 2 = 0.80

0.80 x 2 = 1.60

Result of $(105.15)_{10}$ is **$(1101001.001001)_2$**

Binary to Hexadecimal conversion:-

Hexadecimal	Binary	Hexadecimal	Binary
0	0000	8	1000
1	0001	9	1001
2	0010	A	1010
3	0011	B	1011
4	0100	C	1100
5	0101	D	1101
6	0110	E	1110
7	0111	F	1111

Example-1

(i) Convert $(1011011011)_2$ into hexadecimal.

Solution:

Given Binary number		10	1101	1011
Group of 4 bits are		0010	1101	1011
Convert each group into hex	=	2	D	B

The result is $(\mathbf{2DB})_{16}$

Example-2

(ii) Convert $(01011111011.011111)_2$ into hexadecimal.

Solution:

Given Binary number		010	1111	1011	.	0111	11
Group of 3 bits are	=	0010	1111	1011	.	0111	1100
Convert each group into octal	=	2	F	B	.	7	C

The result is $(\mathbf{2FB.7C})_{16}$

Decimal to hexadecimal conversion:-

The decimal to hexadecimal conversion is same as octal.

Example-1

(i) Convert $(2598.675)_{10}$ into hexadecimal.

Solution:

	Remainder Decimal	Hex		Hex
16 | 2598			0.675 x 16 = 10.8	A
16 | 162	— 6	6	0.800 x 16 = 12.8	C
16 | 10	— 2	2	0.800 x 16 = 12.8	C
0	— 10	A	0.800 x 16 = 12.8	C

Result of $(2598.675)_{10}$ is **$(A26.ACCC)_{16}$**

HEXADECIMAL NUMBER SYSTEM :-

Hexadecimal to binary conversion:-

For conversion of hexadecimal to binary, replace hexadecimal digit by its 4 bit binary group.

Example-1

Convert $(3A9E.B0D)_{16}$ into binary.

Solution:

Given Hexadecimal number is		3	A	9	E	.	B	0	D
Convert each hexadecimal digit to 4 bit binary	=	0011	1010	1001	1110	.	1011	0000	1101

Result of $(3A9E.B0D)_8$ is **$(0011101010011110.101100001101)_2$**

Hexadecimal to decimal conversion:-

For conversion of hexadecimal to decimal, multiply each digit in the hexadecimal number by its position weight and add all those product terms.

Example-1

Convert $(A0F9.0EB)_{16}$ to decimal

Solution:

$$(A0F9.0EB)_{16} = (10 \times 16^3)+(0 \times 16^2)+(15 \times 16^1)+(9 \times 16^0)+(0 \times 16^{-1})+(14 \times 16^{-2})+(11 \times 16^{-3})$$
$$= 40960 + 0 + 240 + 9 + 0 + 0.0546 + 0.0026$$
$$= (41209.0572)_{10}$$

Result is **$(41209.0572)_{10}$**

Practice Questions on Number System

Example 26.4. *Convert the following decimal numbers to octal equivalent.*

(i) 76 (ii) 255 (iii) 372

Example 26.5. *Convert octal number $(24.6)_8$ to the equivalent decimal number.*

Solution.

$$(24.6)_8 = (2 \times 8^1) + (4 \times 8^0) + (6 \times 8^{-1})$$

$$= 16 + 4 + 0.75 = 20.75$$

$$\therefore \quad (24.6)_8 = (20.75)_{10}$$

Example 26.6. *Convert $(177)_{10}$ to its 8-bit binary equivalent by first converting to octal.*

Example 26.7. *Convert decimal number 541 to hexadecimal.*

Solution.

	Division	*Remainder*	
	$541 \div 16 = 33$	13	(LSB)
	$33 \div 16 = 2$	1	
	$2 \div 16 = 0$	2	(MSB)

$\therefore \quad (541)_{10} = \mathbf{(21D)_{16}}$

Example 26.8. *Convert decimal number 378 to a 16-bit number by first converting to hexadecimal.*

Solution.

	Division	*Remainder*	
	$378 \div 16 = 23$	10	(LSB)
	$23 \div 16 = 1$	7	
	$1 \div 16 = 0$	1	(MSB)

Thus $(378)_{10} = (17A)_{16}$. We can easily convert this hex number to binary 000101111010. Therefore, we can express $(378)_{10}$ as a 16-bit binary number by adding four leading 0s.

$$(378)_{10} = \mathbf{(0000000101111010)_2}$$

Example 26.9. *Convert $(B2F)_{16}$ to octal.*

Solution. It is easier to first convert hex to binary and then to octal.

$(B2F)_{16} = 1011 \quad 0010 \quad 1111$ .... conversion to binary

$= 101 \quad 100 \quad 101 \quad 111$ .... 3-bit groupings

$= 5 \quad 4 \quad 5 \quad 7$

$\therefore \quad (B2F)_{16} = \mathbf{(5457)_8}$

Practice Questions

Convert the following numbers into corresponding number system

1. $(A28.1F)_{16}$ = ($)_{10}$
2. $(378.10)_{10}$ = ($)_{8}$
3. $(431.6110)_{8}$ = ($)_{16}$
4. $(1011000111)_{2}$= ($)_{GRAY\ CODE}$
5. $(B65.6A)_{16}$ = ($)_{8}$
6. $(101011.1001)_{2}$ = ($)_{16}$
7. $(306.12)_{10}$ = ($)_{16}$
8. $(101011.11)_{8}$= ($)_{2}$
9. $(63)_{10} = (?)_{16}$
10. $(0010111000010100)_{2} = (?)_{16}$
11. $(14)_{8} = (?)_{2}$
12. $(F4)_{16} = (?)_{2}$

BINARY ARITHEMATIC OPERATION

1. BINARY ADDITION:-

The binary addition rules are as follows

0 + 0 = 0 ; 0 + 1 = 1 ; 1 + 0 = 1 ; 1 + 1 = 10 , i.e 0 with a carry of 1

Example-1

Add $(100101)_2$ and $(1101111)_2$.

Solution :-

$$\begin{array}{rr} & 1\,0\,0\,1\,0\,1 \\ + & 1\,1\,0\,1\,1\,1\,1 \\ \hline & 1\,0\,0\,1\,0\,1\,0\,0 \end{array}$$

Result is **$(10010100)_2$**

2. BINARY SUBTRACTION:-

The binary subtraction rules are as follows

0 - 0 = 0 ; 1 - 1 = 0 ; 1 - 0 = 1 ; 0 - 1 = 1 , with a borrow of 1

Example-1

Substract $(111.111)_2$ from $(1010.01)_2$.

Solution :-

$$\begin{array}{rr} & 1010.010 \\ - & \underline{111.111} \\ & \underline{0010.011} \end{array}$$

Result is **$(0010.011)_2$**

3. BINARY MULTIPLICATION:-

The binary multiplication rules are as follows

0 x 0 = 0 ; 1 x 1 = 1 ; 1 x 0 = 0 ; 0 x 1 = 0

Example-1

Multiply $(1101)_2$ by $(110)_2$.

Solution :-

$$
\begin{array}{rr}
 & 1\,1\,0\,1 \\
\times & 1\,1\,0 \\
\hline
 & 0\,0\,0\,0 \\
 & 1\,1\,0\,1 \\
+ & 1\,1\,0\,1 \\
\hline
 & 1\,0\,0\,1\,1\,1\,0
\end{array}
$$

Result is $(1001110)_2$

4. BINARY DIVISION:-

The binary division is very simple and similar to decimal number system. The division by '0' is meaningless. So we have only 2 rules

$0 \div 1 = 0$

$1 \div 1 = 1$

Example-1

Divide $(10110)_2$ by $(110)_2$.

Solution :-

```
110 ) 101101 ( 111.1
    - 110
      1010
      110
      1001
      110
      110
      110
      000
```

Result is $(111.1)_2$

1's COMPLEMENT REPRESENTATION :-

The 1's complement of a binary number is obtained by changing each 0 to 1 and each 1 to 0.

Example:

Find $(1100)_2$ 1's complement.

Solution :-

Given	1	1	0	0
1's complement is	0	0	1	1

Result is **$(0011)_2$**

Example 1.13 Subtract $(1010)_2$ from $(1111)_2$ using the 1's complement method. Also subtract using direct method and compare.

To subtract a smaller number from a larger number, the 1's complement method is as follows:

(i) Determine the 1's complement of the smaller number.

(ii) Add this to the larger number.

(iii) Remove the carry and add it to the result. This carry is called *end-around-carry*.

1's Complement method

		1	1	1	1 (+)
1's Complement	→	0	1	0	1
Carry	→	1	0	1	0	0
Add Carry	→				1
		0	1	0	1

Direct subtraction

1	1	1	1	
−1	0	1	0	
0	1	0	1	

Example 1.14 Subtract $(1010)_2$ from $(1000)_2$ using the 1's complement method. Also subtract by direct method and compare.

Subtraction of a larger number from a smaller one by the 1's complement method involves the following steps:

(i) Determine the 1's complement of the larger number.

(ii) Add this to the smaller number.

(iii) The answer is the 1's complement of the true result and is opposite in sign. There is no carry.

1's Complement method		Direct subtraction
	1 0 0 0 (+)	1 0 0 0
1's Complement →	0 1 0 1	−1 0 1 0
	1 1 0 1	−0 0 1 0

No carry is obtained. The answer is the 1's complement of 1101 and is opposite in sign, i.e. –0010.

2's COMPLEMENT REPRESENTATION :-

2's complement = 1's complement + 1

Example: **Find $(1010)_2$ 2's complement.**

Solution :-

Given		1	0	1	0
1's complement is		0	1	0	1
	+				1
2's complement		0	1	1	0

Result is $(0110)_2$

Example 1.15 Subtract $(1010)_2$ from $(1111)_2$ using the 2's complement method. Subtract by direct method also and compare.

Subtraction of a smaller number from a larger one by the 2's complement method involves following steps:

(i) Determine the 2's complement of the smaller number.

(ii) Add this to the larger number.

(iii) Omit the carry (there is always a carry in this case).

2's Complement method

		1	1	1	1 (+)
2's Complement →		0	1	1	0
Carry →	1	0	1	0	1

The carry is discarded. Thus, the answer is $(0101)_2$.

Direct subtraction

1	1	1	1
−1	0	1	0
0	1	0	1

Example 1.16 Subtract $(1010)_2$ from $(1000)_2$ using 2's complement method. Subtract by direct method also and compare.

The 2's complement method for subtraction of a larger number from a smaller one is as follows:

(i) Determine the 2's complement of the larger number.

(ii) Add the 2's complement to the smaller number.

(iii) There is no carry. The result is in 2's complement form and is negative.

(iv) To get an answer in true form, take the 2's complement and change the sign.

2's Complement method

		1 0 0 0 (+)
2's Complement	→	0 1 1 0
No carry	→	1 1 1 0

Direct subtraction

1 0 0 0	
−1 0 1 0	
0 0 1 0	

No carry is obtained. Thus, the difference is negative and the true answer is the 2's complement of $(1110)_2$, i.e. $(0010)_2$.

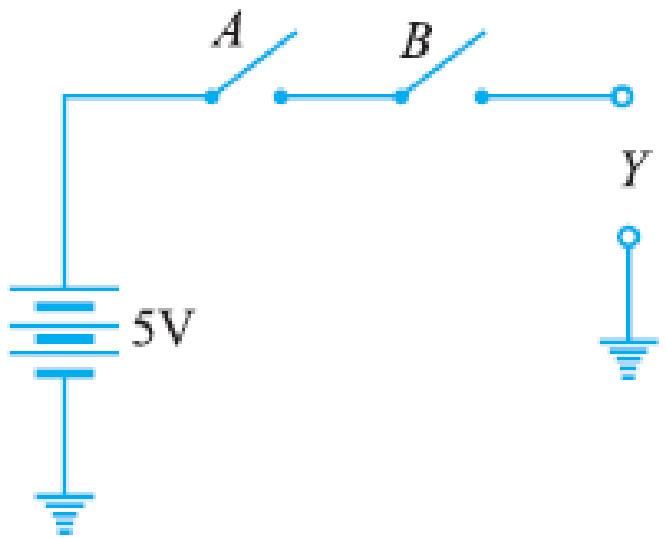
LOGIC GATES

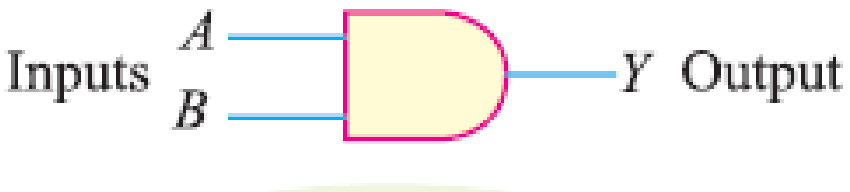
- Logic gates are the fundamental building blocks of digital systems.
- There are 3 basic types of gates AND, OR and NOT.

- Logic gates are electronic circuits because they are made up of a number of electronic devices and components.

- Inputs and outputs of logic gates can occur only in 2 levels. These two levels are termed HIGH and LOW

1. AND Gate

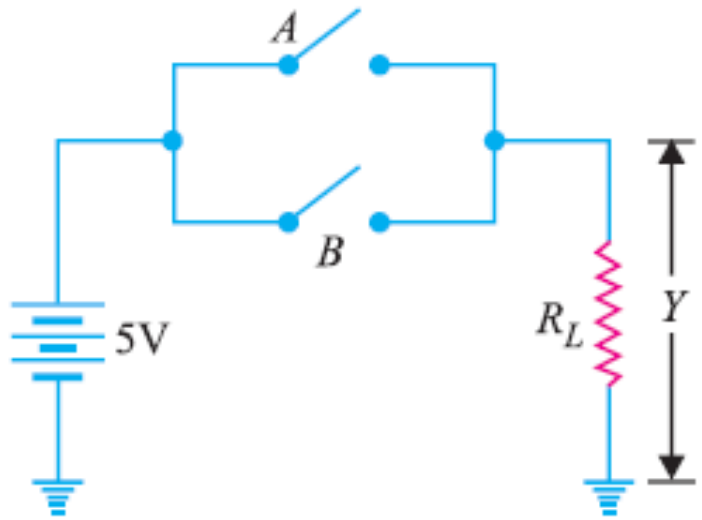


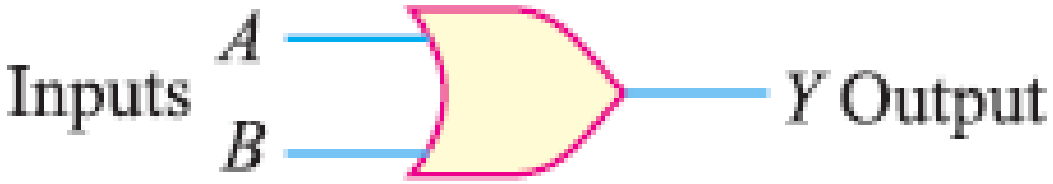


Truth Table

		OUTPUT
A	B	Q=A . B
0	0	0
0	1	0
1	0	0
1	1	1

2. OR Gate

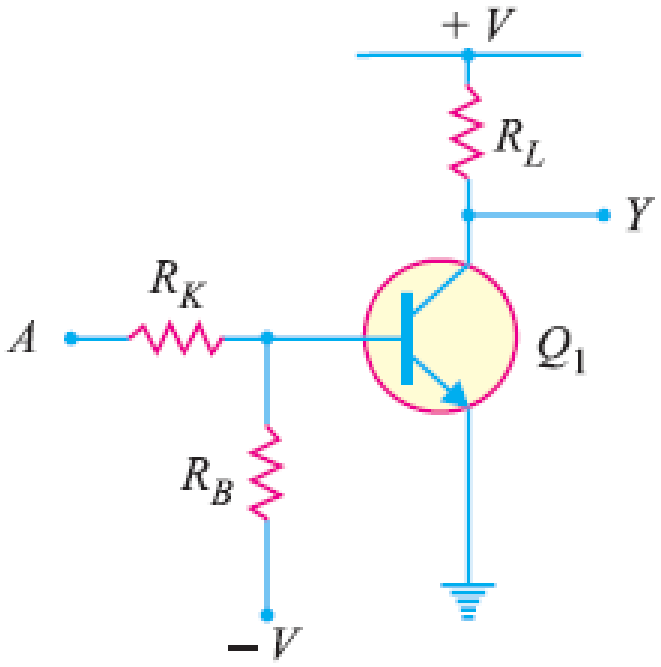




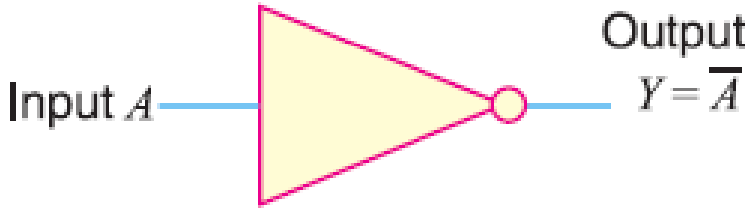
Truth Table

INPUT		OUTPUT
A	B	Q=A + B
0	0	0
0	1	1
1	0	1
1	1	1

3. NOT Gate



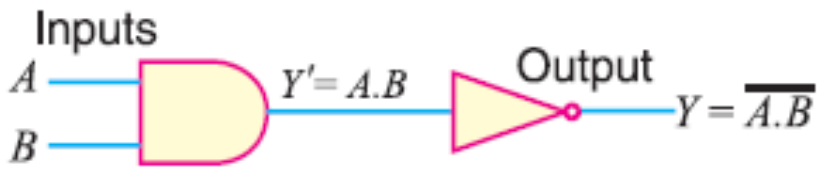
A	Y
0	1
1	0



UNIVERSAL LOGIC GATES
(NAND and NOR Logic Gates)

1. NAND Logic Gate

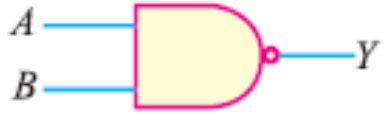
first the inputs must be ANDed and then the inversion is performed. *Note that output from a NAND gate is always 1 except when all of the inputs are 1.* Fig. 26.11 (*iii*) shows the logic symbols for a NAND gate. The little bubble (small circle) on the right end of the symbol means to invert the AND.



(*i*)

Inputs		Output	
A	*B*	AND (Y')	NAND (Y)
0	0	0	1
1	0	0	1
0	1	0	1
1	1	1	0

(*ii*)

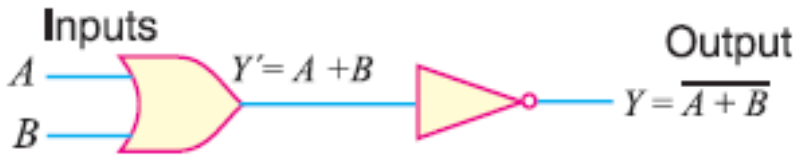


(*iii*)

Fig. 26.11

2. NOR Logic Gate

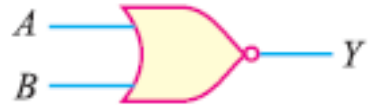
(*ii*) NOR gate. It is a combination of OR gate and NOT gate. In other words, output of OR gate is connected to the input of a NOT gate as shown in Fig. 26.12 (*i*). Note that output of OR gate is inverted to form NOR gate. This is illustrated in the truth table for NOR gate. It is clear that truth table for NOR gate is developed by *inverting the outputs* of the OR gate.



(*i*)

Inputs		Output	
A	*B*	OR (*Y*′)	NOR (*Y*)
0	0	0	1
1	0	1	0
0	1	1	0
1	1	1	0

(*ii*)



(*iii*)

Fig. 26.12

The Boolean expression for NOR function is

$$Y = \overline{A + B}$$

EXCLUSIVE – OR (X-OR) GATE:-

- An X-OR gate is a two input, one output logic circuit.
- The output is logic 1 when one and only one of its two inputs is logic 1. When both the inputs is logic 0 or when both the inputs is logic 1, the output is logic 0.

IC No.:- 7486

Logic Symbol

INPUTS are **A** and **B**

OUTPUT is **Q** = $A \oplus B$

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

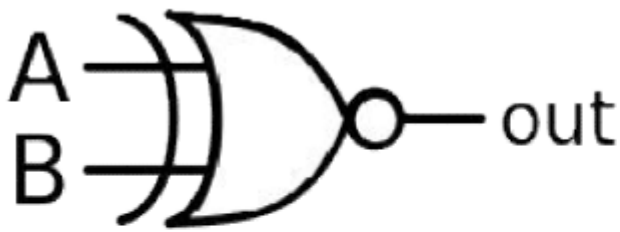
Truth Table

INPUT		OUTPUT
A	**B**	**Q = A ⊕ B**
0	0	0
0	1	1
1	0	1
1	1	0

EXCLUSIVE – NOR (X-NOR) GATE:-

- An X-NOR gate is the combination of an X-OR gate and a NOT gate.
- An X-NOR gate is a two input, one output logic circuit.
- The output is logic 1 only when both the inputs are logic 0 or when both the inputs is 1.
- The output is logic 0 when one of the inputs is logic 0 and other is 1.

Logic Symbol



OUT =A B + $\overline{A}$ $\overline{B}$

= A XNOR B

INPUT		OUTPUT
A	B	OUT =A XNOR B
0	0	1
0	1	0
1	0	0
1	1	1

- Practice Questions on LOGIC GATES

  Design NOT, AND and OR Logic Gates using NAND Universal Logic Gate

  Design NOT, AND and OR Logic Gates using NOR Universal Logic Gate

  Design Ex-OR and Ex- NOR Logic Gates using (i) NAND (ii) NOR Logic Gates

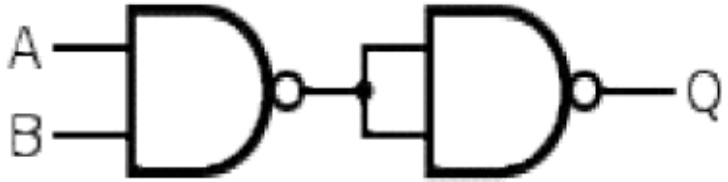
SOLUTION: Design NOT, AND and OR Logic Gates using NAND Universal Logic Gate

a) **Inverter from NAND gate**

Input = **A**
Output **Q** = $\overline{A}$

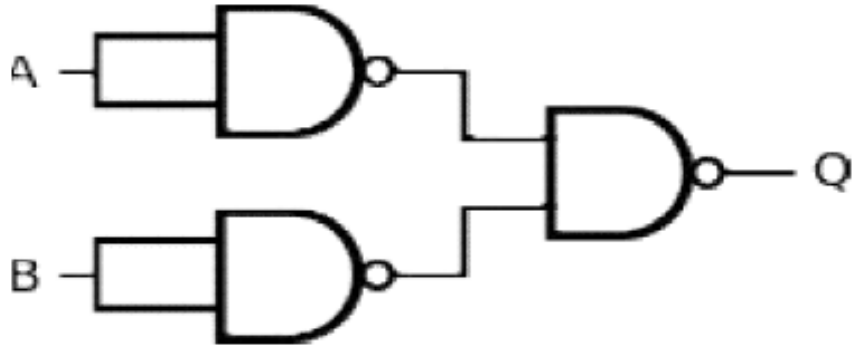
b) **AND gate from NAND gate**

Input s are **A** and **B**
Output **Q = A.B**



c) **OR gate from NAND gate**

Inputs are **A** and **B**
Output **Q = A+B**



SOLUTION: Design NOT, AND and OR Logic Gates using NOR Universal Logic Gate

a) Inverter from NOR gate

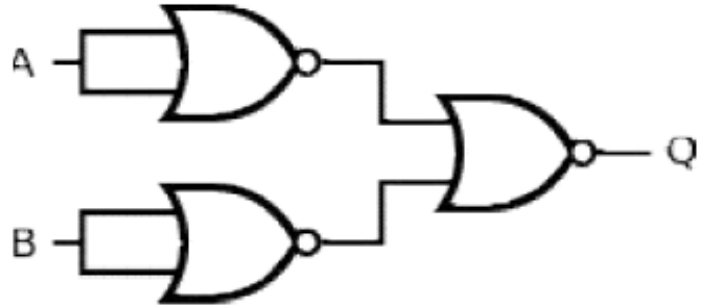
Input = A

Output $Q = \overline{A}$

b) AND gate from NOR gate

Input s are **A** and **B**

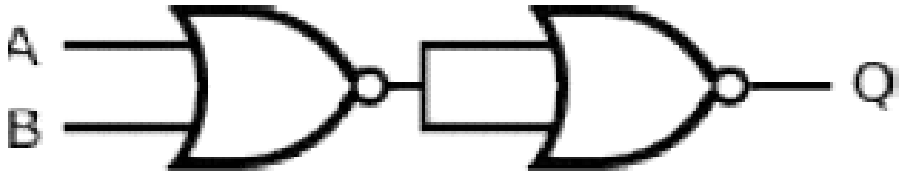
Output **Q = A.B**



c) OR gate from NOR gate

Inputs are **A** and **B**

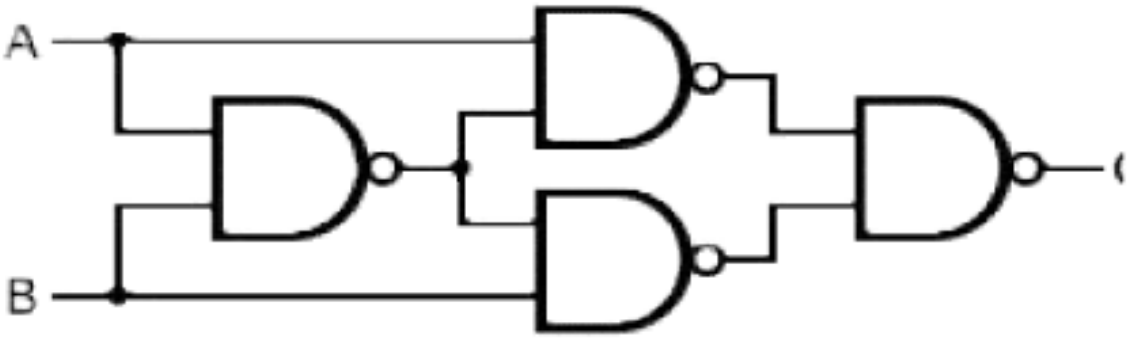
Output **Q = A+B**



SOLUTION: Design Ex-OR and Ex- NOR Logic Gates using (i) NAND Logic Gate

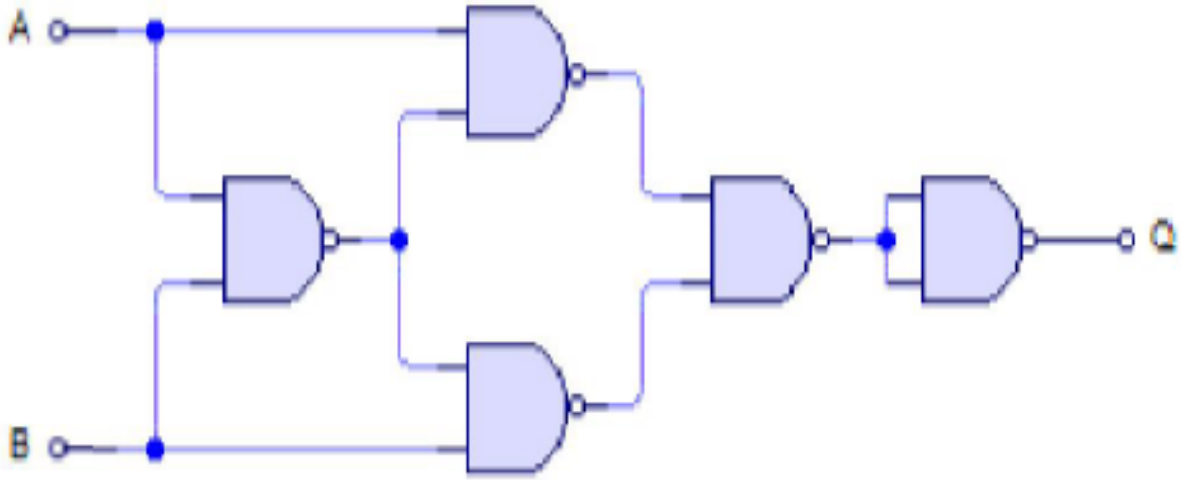
EX-OR gate from NAND gate

Inputs are **A** and **B**
Output **Q = A $\overline{B}$ + $\overline{A}$B**



EX-NOR gate From NAND gate

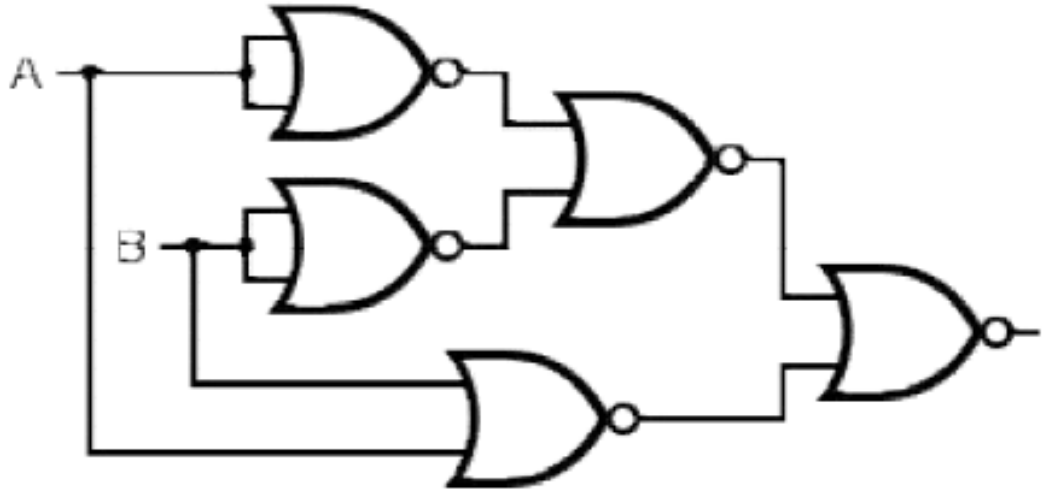
Inputs are **A** and **B**
Output **Q = A B + $\overline{A}$ $\overline{B}$**



SOLUTION: Design Ex-OR and Ex- NOR Logic Gates using (ii) NOR Logic Gate

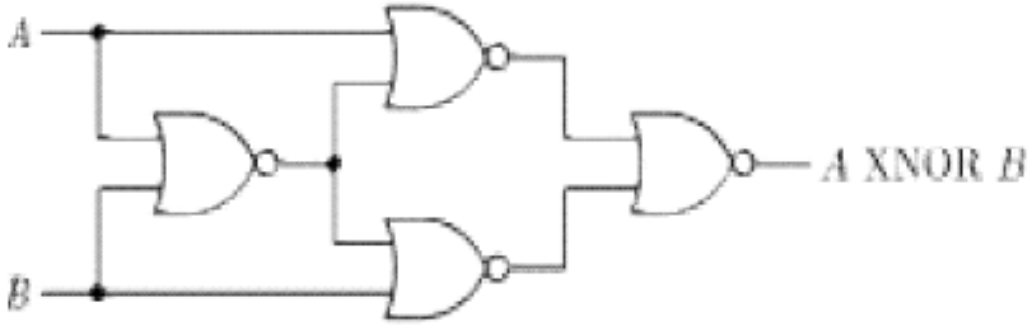
EX-OR gate from NOR gate

Inputs are **A** and **B**
Output **Q = A $\overline{B}$ + $\overline{A}$B**



EX-NOR gate From NOR gate

Inputs are **A** and **B**
Output **Q = A B + $\overline{A}$ $\overline{B}$**



Gray codes The *Gray code* belongs to a class of codes called *minimum-change codes*, in which only one bit in the code group changes when moving from one step to the next. The Gray code is a *non-weighted code*. Therefore, it is not suitable for arithmetic operations but finds applications in input/output devices and in some types of analog-to-digital converters. The Gray code is a reflective digital code which has a special property of containing two adjacent code numbers that differ by only one bit. Therefore, it is also called a *unit-distance code*.

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

Decimal numbers	Binary code	Gray code
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

Conversion of a binary number to gray code A binary number can be converted to its Gray code when

(i) the first bit (MSB) of the Gray code is the same as the first bit of the binary number;

(ii) the second bit of the Gray code equals the exclusive-OR, of the first and second bits of the binary number, i.e. it will be 1 if these binary code bits are different and 0 if they are the same;

(iii) the third Gray code bit equals the exclusive-OR of the second and third bits of the binary number, and so on.

Conversion from gray code to binary Conversion of a Gray code into its binary form involves the reverse of the procedure given above:

(i) The first binary bit (MSB) is the same as that of the first Gray code bit.

(ii) If the second Gray bit is 0, the second binary bit is the same as that of the first binary; if the second Gray bit is 1, the second binary bit is the inverse of its first binary bit.

(iii) Step 2 is repeated for each successive bit.

BOOLEAN ALGEBRA

- Switching circuits are also called logic circuits, gates circuits and digital circuits.
- Switching algebra is also called Boolean algebra.
- Boolean algebra is a system of mathematical logic. It is an algebraic system consisting of the set of elements (0,1), two binary operators called OR and AND and unary operator called NOT.
- It is the basic mathematical tool in the analysis and synthesis of switching circuits.
- It is a way to express logic functions algebraically.
- Any complex logic can be expressed by a Boolean function.
- The Boolean algebra is governed by certain well developed rules and laws.

Rules for Minimization of Boolean Expression

Theorem 1 :	$A + 0 = A$
Theorem 2 :	$A \cdot 1 = A$
Theorem 3 :	$A + \overline{A} = 1$
Theorem 4 :	$A \cdot \overline{A} = 0$
Theorem 5 :	$A + A = A$
Theorem 6 :	$A \cdot A = A$
Theorem 7 :	$A + 1 = 1$
Theorem 8 :	$A \cdot 0 = 0$
Theorem 9 :	$\overline{\overline{A}} = A$

Continue..

Theorem 10	:	$A + B = B + A$	*Commutative Law*
Theorem 11	:	$A \cdot B = B \cdot A$	
Theorem 12	:	$A + (B + C) = (A + B) + C$	*Associative Law*
Theorem 13	:	$A \cdot (B \cdot C) = (A \cdot B) \cdot C$	
Theorem 14	:	$A \cdot (B + C) = A \cdot B + A \cdot C$	*Distributive Law*
Theorem 15	:	$(A + B) \cdot (C + D) = A \cdot C + B \cdot C + A \cdot D + B \cdot D$	
Theorem 16	:	$A + A \cdot B = A$	
Theorem 17	:	$\overline{(A + B)} = \overline{A} \cdot \overline{B}$	*De Morgan's Theorems*
Theorem 18	:	$\overline{(A \cdot B)} = \overline{A} + \overline{B}$	

Examples for Boolean Expression Minimization

Example 26.13. *Using Boolean algebraic techniques, simplify the following expression :*

$$Y = A \cdot B \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot C \cdot \overline{D} + A \cdot B \cdot C \cdot \overline{D}$$

Solution.

$$Y = A \cdot B \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot \overline{C} \cdot \overline{D} + \overline{A} \cdot B \cdot C \cdot \overline{D} + A \cdot B \cdot C \cdot \overline{D} \quad \ldots(i)$$

Step 1 : Take out the common factors as below :

$$Y = B\,\overline{C}\,\overline{D}\,(A + \overline{A}) + B\,C\,\overline{D}\,(A + \overline{A})$$

Step 2 : Apply Theorem 3 $(A + \overline{A} = 1)$:

$$Y = B\,\overline{C}\,\overline{D} + B\,C\,\overline{D}$$

Step 3 : Again factorise :

$$Y = B\,\overline{D}\,(C + \overline{C})$$

Step 4 : Apply Theorem 3 $(C + \overline{C} = 1)$:

$$Y = B\,\overline{D} \cdot 1 = B\overline{D}$$

Example 26.14. *Using Boolean techniques, simplify the following expression :*

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

Solution.

$$Y = AB + A(B + C) + B(B + C) \quad \ldots(i)$$

Step 1 : Apply Theorem 14 (distributive law) to second and third terms:

$$Y = AB + AB + AC + BB + BC$$

Step 2 : Apply Theorem 6 $(B \cdot B = B)$:

$$Y = AB + AB + AC + B + BC$$

Step 3 : Apply Theorem 5 $(AB + AB = AB)$:

$$Y = AB + AC + B + BC$$

Step 4 : Factor B out of last 2 terms :

$$Y = AB + AC + B(1 + C)$$

Step 5 : Apply commutative law and Theorem 7 $(1 + C = C + 1 = 1)$:

$$Y = AB + AC + B \cdot 1$$

Step 6 : Apply Theorem 2 $(B \cdot 1 = B)$:

$$Y = AB + AC + B$$

Step 7 : Factor B out of first and third terms :

$$Y = B(A + 1) + AC$$

Step 8 : Apply Theorem 7 $(A + 1 = 1)$:

$$Y = B \cdot 1 + AC$$

Step 9 : Apply Theorem 2 $(B \cdot 1 = B)$:

$$Y = B + AC$$

Example 26.15. *Simplify the following Boolean expressions to a minimum number of literals :*

(*i*) $Y = A + \overline{A}B$ (*ii*) $Y = AB + \overline{A}C + BC$

Solution. (*i*)

$$Y = A + \overline{A}B$$

$$= A + AB + \overline{A}B \qquad [\because A = A + AB \text{ from Theorem 16}]$$

$$= A + B(A + \overline{A})$$

$$= A + B \qquad [\because A + \overline{A} = 1 \text{ from Theorem 3}]$$

$\therefore$ $Y = A + B$

(*ii*)

$$Y = AB + \overline{A}C + BC$$

$$= AB + \overline{A}C + BC \cdot (A + \overline{A})$$

$$= AB + \overline{A}C + ABC + \overline{A}BC$$

$$= AB(1 + C) + \overline{A}C\ (1 + B)$$

$$= AB + \overline{A}C$$

$\therefore$ $Y = AB + \overline{A}C$

Example 26.16. *Determine output expression for the circuit shown below and simplify it using De Morgan's theorem.*

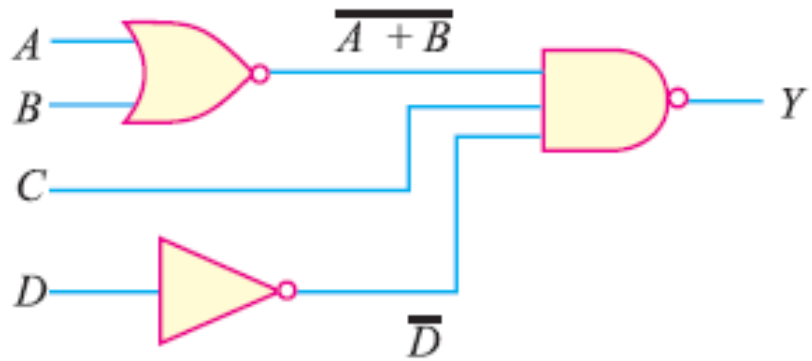


Fig. 26.32

Solution. The output expression for the circuit shown above is :

$$Y = \overline{[\overline{(A+B)} \cdot C \cdot \overline{D}]}$$

Using De Morgan's theorem :

$$Y = (A+B) + \overline{C} + D$$

$$\therefore \quad Y = A + B + \overline{C} + D$$

Example 26.18. *Simplify the following Boolean expressions :*

(*i*) $Y = (A + B + C) \cdot (A + B)$

(*ii*) $Y = AB + ABC + A\overline{BC}$

(*iii*) $Y = 1 + A(B \cdot \overline{C} + BC + \overline{B}\,\overline{C}) + A\overline{B}C + AC$

(*iv*) $Y = \overline{(A + \overline{B} + C) + (B + \overline{C})}$

Example 26.19. *Simplify the following Boolean expression :*

$$Y = A\overline{B}D + A\overline{B}\,\overline{D}$$

Solution.

$$Y = A\overline{B}D + A\overline{B}\,\overline{D}$$

Factoring out the common variables $A\overline{B}$ (using Theorem 14), we get ,

$$Y = A\overline{B}(D + \overline{D})$$

Using Theorem 3, $D + \overline{D} = 1$.

$\therefore$

$$Y = A\overline{B} \cdot 1$$

Using Theorem 2, we get,

$$Y = A\overline{B}$$

Example 26.20. *Simplify the following Boolean expression :*

$$Y = (\overline{A} + B)(A + B)$$

De-Morgan's Theorem

De Morgan's theorems are extremely useful in simplifying expressions in which a product or sum of variables is inverted. The two theorems are :

(*i*) $(\overline{A+B}) = \overline{A} \cdot \overline{B}$

(*ii*) $(\overline{A.B}) = \overline{A} + \overline{B}$

(*i*) The first De Morgan's theorem may be stated as under :

When the OR sum of two variables is inverted, this is equal to inverting each variable individually and then ANDing these inverted variables i.e.,

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

In this expression, A and B are the two variables. The L.H.S. is the complement of the OR sum of the two variables. The R.H.S. is the AND product of individual inverted variables.

(*ii*) The second De Morgan's theorem may be stated as under :

When the AND product of two variables is inverted, this is equal to inverting each variable individually and then ORing them i.e.,

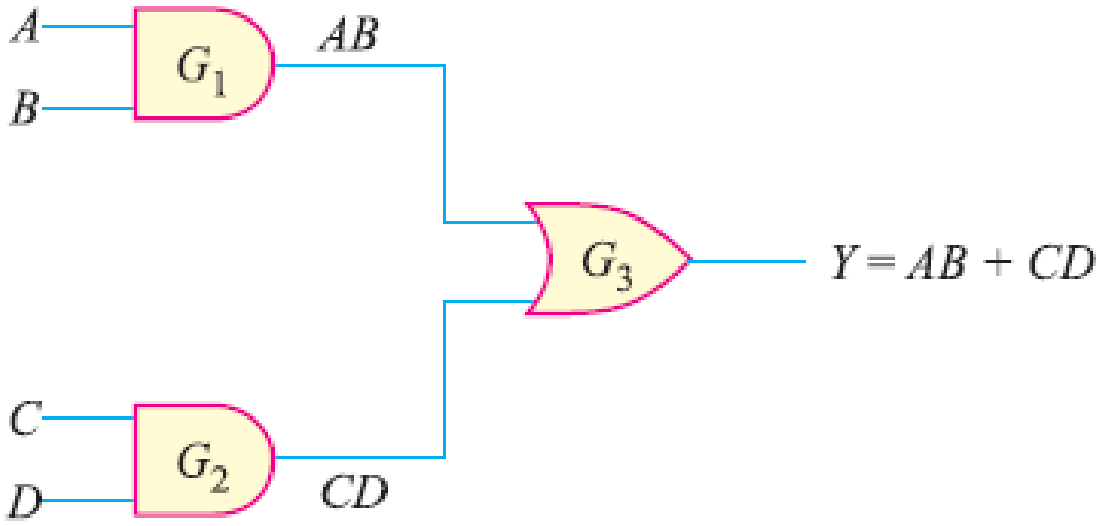
$$(\overline{A \cdot B}) = \overline{A} + \overline{B}$$

In this expression, A and B are the two variables. The L.H.S. is the complement of the AND product of the two variables. The R.H.S. is the OR sum of the individual inverted variables.

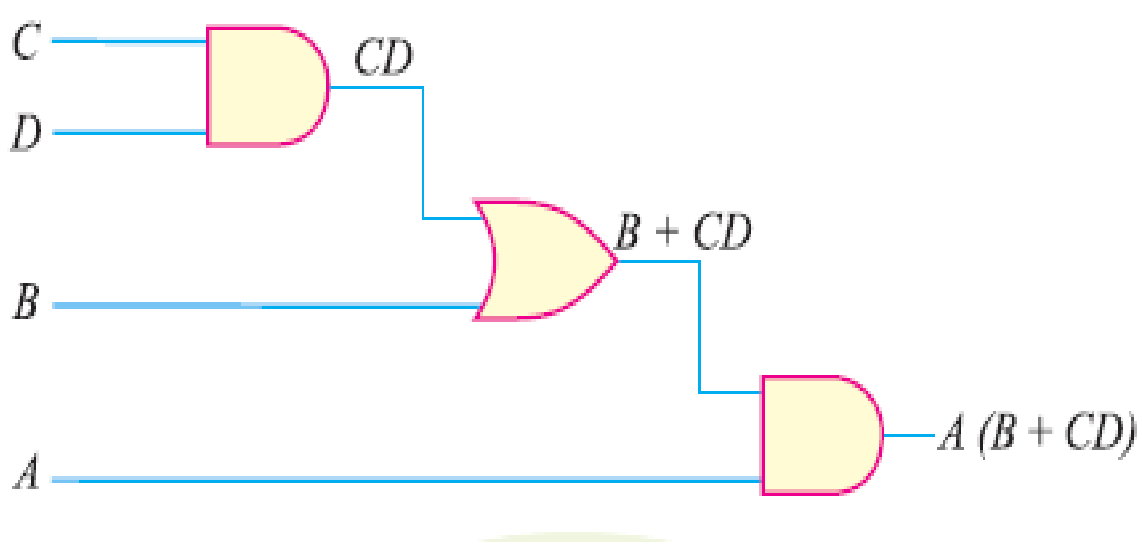
Realisation of Boolean Expression using Logic Gates

Q-1 Realise the Boolean expression using logic gates.

(i) Y=AB+CD



(ii) Y= A(B+CD)



Unsolved Questions

1. Convert decimal number 23 into equivalent binary number. $[(10111)_2]$
2. Simplify the expression $Y = A\,C\,D + \overline{A}\,B\,C\,D$. $[Y = A\overline{C} + \overline{B}D]$
3. Simplify the expression $Y = \overline{(\overline{A} + C) \cdot (B + \overline{D})}$ to one having only single variables inverted.

   $[Y = A\overline{C} + \overline{B}D]$
4. Find the complement function of $Y = \overline{A}\,B\,\overline{C} + \overline{A}\,\overline{B}\,C$. $[(A + \overline{B} + C)\,(A + B + \overline{C})]$
5. Simplify the expression $Y = A \cdot B + A \cdot \overline{B}$. $[Y = A]$
6. Simplify the expression $Y = A \cdot B \cdot C + B \cdot C$. $[Y = B \cdot C]$

Unsolved Questions

7. Simplify the following Boolean expression to a minimum number of literals :

$$Y = A(\overline{A} + B) \qquad [Y = AB]$$

8. Simplify the following Boolean function to a minimum number of literals :

$$Y = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,C + A\,\overline{B} \qquad [Y = \overline{A}\,C + A\overline{B}\,]$$

9. Simplify the expression : $Y = (A + B)(\overline{A} + C)(B + C)$ $[Y = (A + B)(\overline{A} + C)]$

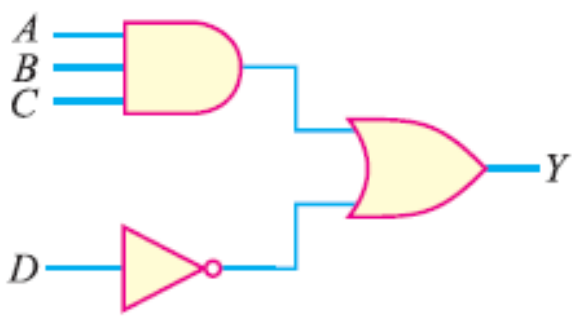
10. Find the complement of the function :

$$Y = A(\overline{B}\,\overline{C} + B\,C)\,H \qquad [\overline{Y} = \overline{A} + (B + C)(\overline{B} + \overline{C})]$$

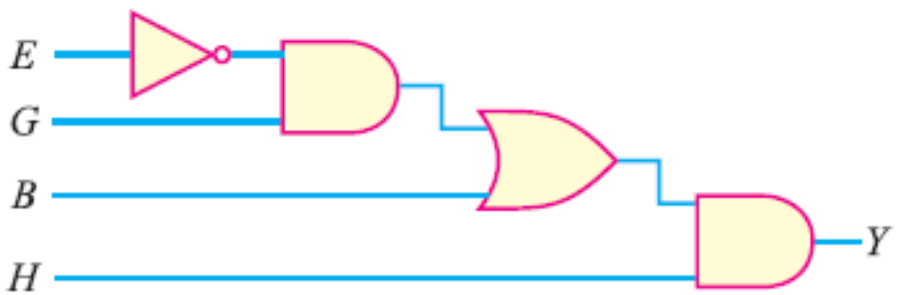
11. Draw the logic circuit for the following Boolean expressions :

(*i*) $Y = ABC + \overline{D}$ (*ii*) $Y = (\overline{E}G + B)\,H$

Ans.



(*i*)



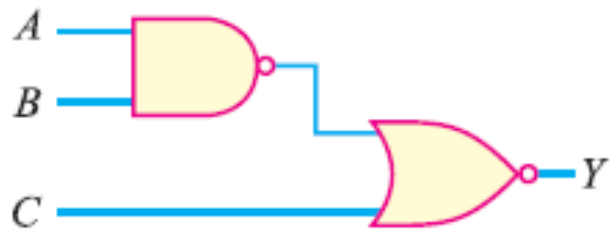
(*ii*)

Unsolved Questions

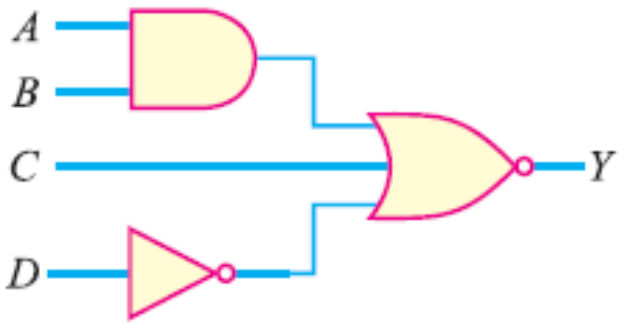
12. A logic circuit is given by the Boolean escperssion : $Y = \overline{\overline{AB} + C}$. Draw the logic circuit for this expression.

13. Write the Boolean expressions for the logic circuits shown in Fig. 26.61.

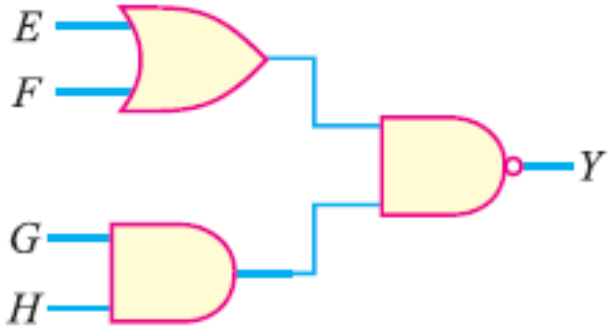
Ans.



(iii)



(i)



(ii)

Fig. 26.61

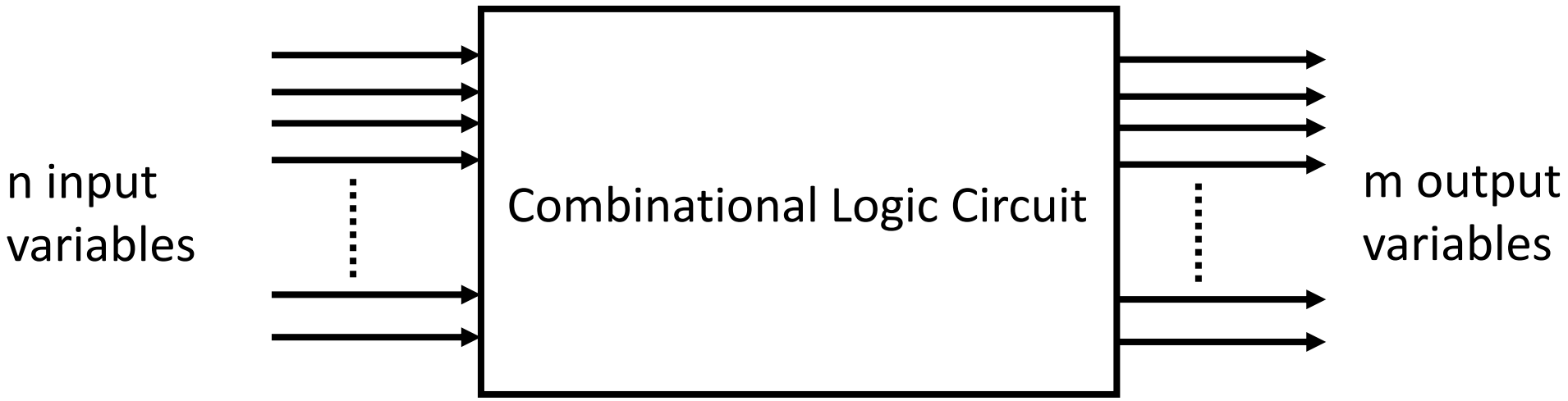
[Ans. :- (i) $Y = \overline{AB + C + \overline{D}}$ (ii) $Y = \overline{(E + F)(GH)}$

Combinational Logic Circuit

- Logic Circuit for the Digital System may be Combinational or Sequential.
- Output of the combinational Circuits depends only on the present input.
- A combination Circuits consist of input variable, logic gates, and output variables.

Block Diagram

- For n input variable 2^n possible combination of the inputs.
- For each input combination, there is only one possible output combination.



Design Procedure

- Problem Statement
- Number of available input variables and required output variable is determined.
- The input and output variable are assigned letter symbols.
- The truth table that defines the required relationship between inputs and output is derived.
- The simplified Boolean function for each output is obtained.
- The logic diagram is drawn.

Adders

- Half Adder
  - A half adder is combinational circuit with two binary inputs and two binary outputs.
  - It adds two input bits and produced the sum and carry bits.
- Full Adder
  - A full adder circuit adds two bits and carry and outputs and a sum bit and a carry bit.