# UPPSC-AE 2021

## Uttar Pradesh
## Public Service Commission

Combined State Engineering Services Examination

**Assistant Engineer**

## Electrical Engineering

## Digital Electronics

Well Illustrated **Theory** *with*
**Solved Examples** and **Practice Questions**

MADE EASY Publications

# Digital Electronics

## Contents

CHAPTER 1

# Boolean Algebra and Logic Gates

## 1.1 Introduction

- The binary operations performed by any digital circuit with the set of elements 0 and 1, are called logical operations or logic functions. The algebra used to symbolically represent the logic function is called Boolean algebra. It is a two state algebra invented by George Boole in 1854.
- Thus, a Boolean algebra is a system of mathematics logic for the analysis and designing of digital systems.
- A variable or function of variables in Boolean algebra can assume only two values, either a '0' or a '1'. Hence, (unlike another algebra) there are no fractions, no negative numbers, no square roots, no cube roots, no logarithms etc.

## 1.2 Logic Operations

- In Boolean algebra, all the algebraic functions performed is logical. These actually represent logical operations. The AND, OR and NOT are the basic operations that are performed in Boolean algebra.
- In addition to these operations, there are some derived operation such as NAND, NOR, EX-OR, EX-NOR that are also performed in Boolean algebra.

### 1.2.1 AND Operation

The AND operation in Boolean algebra is similar to the multiplication in ordinary algebra. It is a logical operation performed by AND gate.

$$A \cdot A = A$$
$$A \cdot 0 = 0 \longrightarrow \text{Null law}$$
$$A \cdot 1 = A \longrightarrow \text{Identity law}$$
$$A \cdot \bar{A} = 0$$

### 1.2.2 OR Operation

The OR operation in Boolean algebra is performed by OR-gate.

$$A + A = A$$
$$A + 0 = A \longrightarrow \text{Null law}$$
$$A + 1 = 1 \longrightarrow \text{Identity law}$$
$$A + \bar{A} = 1$$

### 1.2.3 NOT Operation

- The NOT operation in Boolean algebra is similar to the complementation or inversion in ordinary algebra. The NOT operation is indicated by a bar (—) or (') over the variable.
- $A \xrightarrow{NOT} \bar{A}$ or $A'$ (complementation law)

  and $\bar{\bar{A}} = A \Rightarrow$ double complementation law.

### 1.2.4 NAND Operation

The NAND operation in Boolean algebra is performed by AND operation with NOT operation i.e. the negation of AND gate operation is performed by the NAND gate.

### 1.2.5 NOR Operation

The NOR operation in Boolean algebra is performed by OR operation with NOT operation. i.e. the negation of OR gate operation is performed by the NOR gate.

### 1.2.6 EX-OR Operation

Unlike basic operations of logic gates, this used for special purpose and is represented by symbol '⊕' where

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

## 1.3 Laws of Boolean Algebra

The Boolean algebra is governed by certain well developed rules and laws.

### 1.3.1 Commutative Laws

- The commutative law allows change in position of AND or OR variables. There are two commutative laws.
  (*i*) $A + B = B + A$
  Thus, the order in which the variables are ORed is immaterial.
  (*ii*) $A \cdot B = B \cdot A$
  Thus, the order in which the variables are ANDed is immaterial.
- This law can be extended to any number of variables.

### 1.3.2 Associative Laws

- The associative law allows grouping of variables. There are two associative laws
  (*i*) $(A + B) + C = A + (B + C)$
  Thus, the way the variables are grouped and ORed is immaterial.
  (*ii*) $(A \cdot B) \cdot C = A \cdot (B \cdot C)$
  Thus, the way the variables are grouped and ANDed is immaterial.
- This law can be extended to any number of variables.

### 1.3.3 Distributive Laws

- The distributive law allows factoring or multiplying out of expressions. There are two distributive laws.
  (*i*) $A(B + C) = AB + AC$ (*ii*) $A + BC = (A + B)(A + C)$
- This law is applicable for single variable as well as a combination of variables.

### 1.3.4 Idempotence Laws

Idempotence means the same value. There are two Idempotence laws

(*i*) $A \cdot A = A$

i.e. ANDing of a variable with itself is equal to that variable only.

(*ii*) $A + A = A$

i.e. ORing of a variable with itself is equal to that variable only.

### 1.3.5 Absorption Laws

There are two absorption laws

(*i*) $A + AB = A(1 + B) = A$ (*ii*) $A\,(A + B) = A$

### 1.3.6 Involutionary Law

This law states that, for any variable 'A'

$$\bar{\bar{A}} = (A')' = A$$

## 1.4 Boolean Algebraic Theorems

### 1.4.1 De Morgan's Theorem

- These are very useful in simplifying expressions in which a product or sum of variables is inverted.
- De Morgan's theorem represents two of the most important rules of Boolean algebra.

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

  Thus, the complement of the product of variables is equal to the sum of their individual complements.

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

  Thus, the complement of a sum of variables is equal to the product of their individual complements.

- The above two laws can be extend for 'n' variables as

$$\overline{A_1 \cdot A_2 \cdot A_3 \cdots A_n} = \bar{A}_1 + \bar{A}_2 + \cdots + \bar{A}_n$$

and $$\overline{A_1 + A_2 + \cdots + A_n} = \bar{A}_1 \cdot \bar{A}_2 \cdot \bar{A}_3 \cdot \bar{A}_4 \cdots \bar{A}_n$$

### 1.4.2 Transposition Theorem

The transposition theorem states that $(AB + \bar{A}C) = (A + C)(\bar{A} + B)$

### 1.4.3 Consensus Theorem/Redundancy Theorem

- This theorem is used to eliminate redundant term.
- A variable is associated with some variable and its compliment is associated with some other variable and the next term is formed by the left over variables, then the term becomes redundant.
- It is applicable only if a Boolean function,

  (*i*) Contains 3-variables.

  (*ii*) Each variable used two times.

  (*iii*) Only one variable is in complemented or uncomplemented form.

Then, the related terms to that complemented and uncomplemented variable is the answer.

- Consensus theorem can be extended to any number of variables.

  e.g. $AB + \bar{A}C + BC = AB + \bar{A}C$

**Example - 1.1** The Boolean expression $(A + B)(\bar{B} + C)(C + A) = (A + B)(\bar{B} + C)$ can be simplified as

(a) $(A + B)(\bar{B} + C)$
(b) $(\bar{A} + B)(\bar{B} + C)$
(c) $(A + \bar{B})(\bar{B} + C)$
(d) $(A + \bar{B})(B + \bar{C})$

**Solution : (a)**

Proof:

$$\begin{aligned} \text{LHS} &= (A + B)(\bar{B} + C)(C + A) \\ &= (A\bar{B} + AC + BC)(C + A) \\ &= A\bar{B}C + AC + BC + A\bar{B} + AC + ABC \\ &= AC + BC + A\bar{B} \\ \text{RHS} &= (A + B)(\bar{B} + C) \\ &= A\bar{B} + AC + BC = \text{LHS} \end{aligned}$$

**Example - 1.2** $\overline{\bar{A}\,\bar{B}\,\bar{C}}$ is equal to

(a) $\bar{A} + \bar{B} + \bar{C}$
(b) $\overline{ABC}$
(c) $A + B + C$
(d) $A \cdot B \cdot C$ [UPPSC]

**Solution: (c)**

$$\overline{\bar{A}\,\bar{B}\,\bar{C}} = \bar{\bar{A}} + \bar{\bar{B}} + \bar{\bar{C}} = A + B + C$$

## 1.4.4 Duality Theorem

It is one of the elegant theorems proved in advance mathematics.

**"Dual expression"** is equivalent to write a negative logic of the given Boolean relation. For this we have to

(i) change each **OR** sign by an **AND** sign and vice-versa.
(ii) complement any **'0'** or **'1'** appearing in expression.
(iii) keep literals/variables as it is.

**Example - 1.3** The self dual expression of Boolean relation, $\bar{A}BC + AB\bar{C} + A\bar{B}\bar{C}$ is

(a) $(A + B + \bar{C})(A + B + \bar{C})(A + \bar{B} + \bar{C})$
(b) $(A + B + C)(A + B + C)(A + \bar{B} + \bar{C})$
(c) $(\bar{A} + B + C)(A + B + \bar{C})(A + \bar{B} + \bar{C})$
(d) $(\bar{A} + B + \bar{C})(A + \bar{B} + \bar{C})(A + \bar{B} + \bar{C})$

**Solution: (c)**

$$\bar{A}BC + AB\bar{C} + A\bar{B}\bar{C}$$

Its DUAL

$$\Downarrow$$

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

NOTE

- For any logical expression, if we take two times Dual, we get same given expression as previous.
- For 1-*time Dual*, if we get same function or expression it is called "*Self Dual Expression*".
- With *N-variables*, maximum possible *Self-Dual Function* = $(2)^{2^{n-1}} = 2^{(2^n/2)}$.
- Remember that with *N-variables, maximum possible distinct logic functions* = $2^{2^n}$.

### 1.4.5 Complementary Theorem

For obtaining complement expression we have to

(i) change each **OR** sign by **AND** sign and vice-versa.

(ii) complement any **'0'** or **'1'** appearing in expression.

(iii) complement the individual literals/variables.

**Example - 1.4** The compliment of the function $f = A\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C}$ is equal to

(a) $(\bar{A}+B+\bar{C})(A+\bar{B}+C)(\bar{A}+B+C)$ (b) $(\bar{A}+\bar{B}+C)(A+\bar{B}+\bar{C})(\bar{A}+B+C)$

(c) $(\bar{A}+\bar{B}+C)(A+\bar{B}+C)(\bar{A}+\bar{B}+C)$ (d) $(\bar{A}+B+\bar{C})(A+\bar{B}+C)(\bar{A}+\bar{B}+\bar{C})$

**Solution : (a)**

$$\bar{f} = \text{Complement of f}$$

$$\bar{f} = (\bar{A}+B+\bar{C})(A+\bar{B}+C)(\bar{A}+B+C)$$

**Example - 1.5** The Boolean expression $A(A + B)$ is equal to

(a) 1 (b) B

(c) A (d) A + B [UPPSC]

**Solution: (c)**

$$\begin{aligned} Y &= A(A+B) \\ &= A\cdot A + A\cdot B \\ &= A + AB \\ &= A(1+B) \\ &= A \end{aligned}$$

**Example - 1.6** The Boolean function $(x + y)(\bar{x} + z)(y + z)$ is equal to which one of the following expressions?

(a) $(x + y)(y + z)$ (b) $(\bar{x} + z)(y + z)$

(c) $(x + y)(\bar{x} + z)$ (d) $(x + y)(x + \bar{z})$

**Solution: (c)**

By using consensus theorem,

$$(x+y)(\bar{x}+z)(y+z) = (x+y)(\bar{x}+z)$$

# Student's Assignments

**Q.1** The logic expression $F = (\bar{A} + C)(B + C)(A + B)$ is equal to

(a) $(\bar{A} + B)(A + C)$ (b) $(A + B)(\bar{A} + C)$

(c) $(A + \bar{B})(\bar{A} + C)$ (d) $(A + \bar{B})(\bar{A} + \bar{C})$

**Q.2** Which of the following statements is **not** correct?

(a) Disjunction of a variable with logic '0' results in the same variable.

(b) NOR function can be implemented by inverting the two inputs to an AND function.

(c) Conjunction of a variable with logic '1' results in the complement of variable.

(d) NAND function is commutative but not associative.

**Q.3** The *K*-map of a function is given below:

| $X_1X_4 \backslash X_2X_3$ | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 1 | | x | x |
| 01 | | 1 | x | 1 |
| 11 | | x | 1 | |
| 10 | 1 | x | | x |

If "X" indicates don't care condition, then the minimized expression for the function is

(a) $X_3\bar{X}_1 + \bar{X}_1\bar{X}_3 + X_3X_2$

(b) $X_3X_2 + X_4\bar{X}_3 + X_2\bar{X}_1$

(c) $\bar{X}_4\bar{X}_3 + X_4X_3 + X_2\bar{X}_1$

(d) $\bar{X}_4\bar{X}_3 + X_4X_3 + X_3\bar{X}_1$

**Q.4** In negative logic system

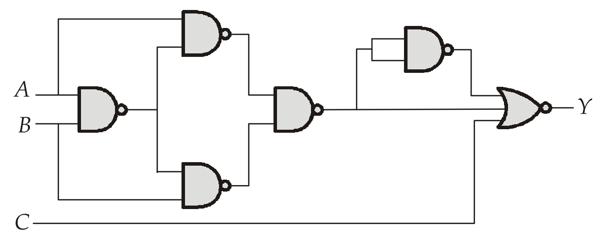
(a) The more negative of the two logic levels represents a logic '1' state.

(b) The more negative of the two logic levels represents a logic '0' state.

(c) All input and output voltage levels are negative.

(d) None of the above

**Q.5** Consider the following circuit:




If the value of *C* is stuck at '0', then the output *Y* will be

(a) $A\bar{B}$ (b) 0

(c) 1 (d) $\bar{A}$

**Q.6** The Boolean expression

$$X(P, Q, R) = \Pi(0, 5)$$

is to be realized using only two 2-input gates. Which are these gates?

(a) AND and OR (b) NAND and OR

(c) AND and XOR (d) OR and XOR

**Q.7** In Boolean algebra if $F = (A + B)(\bar{A} + C)$, then

(a) $F = AB + \bar{A}C$ (b) $F = AB + \overline{AB}$

(c) $F = AC + \bar{A}B$ (d) $F = AA + \bar{A}B$

**Q.8** The POS form of expression is suitable for circuit using:

(a) NOR (b) AND

(c) XOR (d) NAND

**Q.9** Negative logic in a logic circuit is one in which

(a) logic 0 and 1 are represented by the negative and zero voltages respectively.

(b) logic 0 and 1 are represented by zero and positive voltages respectively.

(c) logic 0 voltage level is lower than logic 1 voltage level.

(d) logic 0 voltage level is higher than logic 1 voltage level.

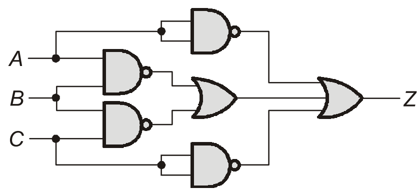
**Q.10** The reduced form of Boolean expression

$$A\left[(B+C)\overline{(AB+AC)}\right] \text{ is}$$

(a) $\bar{A}B$ (b) $A\bar{B}$

(c) 0 (d) $AB + B\bar{C}$

**Q.11** In the given combinational circuit, the output $Z$ is



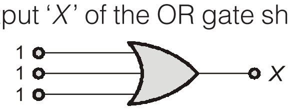

(a) $\overline{A}+\overline{B}+\overline{C}$ (b) $\overline{ABC}$
(c) $\overline{AB}+\overline{BC}+\overline{AC}$ (d) All of the above

**Q.12** The number of terms in a logic function of three variables $X$, $Y$, $Z$ are
(a) 3 (b) 4
(c) 8 (d) 16 [UPPSC]

**Q.13** How many different Boolean expressions can be constructed by using 2 variables?
(a) 18 (b) 16
(c) 32 (d) 4

**Q.14** With 4 Boolean variables, how many Boolean expressions can be formed?
(a) 16 (b) 256
(c) 1024 (1K) (d) 64 K(64 × 1024)

**Q.15** The output '$X$' of the OR gate shown in figure is




(a) 1 (b) 2
(c) 0 (d) 3 [UPPSC]

**Q.16** The most suitable gate for comparing two bit is
(a) AND gate (b) OR gate
(c) NAND gate (d) EX-OR gate
[UPPSC]

**Q.17** The minimum number of NOR gates required to implement $A(A + B)(A + B + C)$ is equal to
(a) 0 (b) 3
(c) 4 (d) 7 [UPPSC]

## ANSWER KEY — STUDENT'S ASSIGNMENTS

| | | | | |
|---|---|---|---|---|
| **1.** (b) | **2.** (c) | **3.** (c) | **4.** (a) | **5.** (b) |
| **6.** (d) | **7.** (c) | **8.** (a) | **9.** (d) | **10.** (c) |
| **11.** (d) | **12.** (c) | **13.** (b) | **14.** (d) | **15.** (a) |
| **16.** (d) | **17.** (a) | | | |

## HINTS & SOLUTIONS — STUDENT'S ASSIGNMENTS

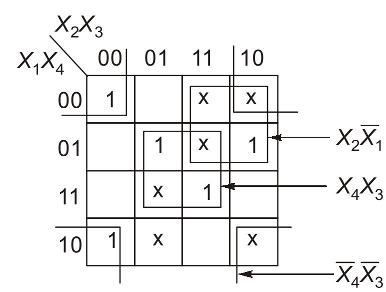
**1. (b)**

$$F = (\overline{A}+C)(B+C)(A+B)$$
$$= (\overline{A}+C)(A+B)$$
(By consensus theorem)

**2. (c)**

Conjunction of a variable with logic '1' results in the same variable.

**3. (c)**

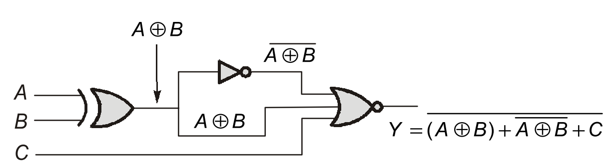



**4. (a)**

In negative logic system, the more negative of the two logic levels represents a logic '1' state.

**5. (b)**

The circuit can be redrawn as




$$Y = \overline{(A \oplus B) + \overline{A \oplus B} + C} = \overline{1 + C} = 0$$

**6. (d)**

The Boolean expression
$$X(P, Q, R) = \pi(0, 5)$$
$$= (P+Q+R)(\overline{P}+Q+\overline{R})$$
$$= Q + (\overline{P}R + P\overline{R})$$

**8. (a)**

POS form (Product of Sum form) of expression is suitable for circuit using NOR gates.