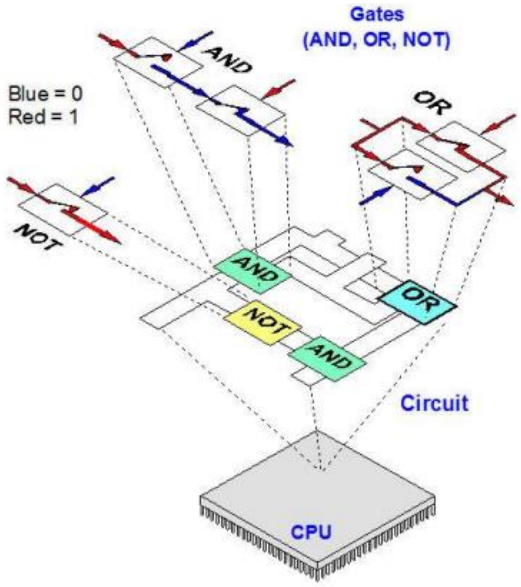
Unit 04

Logic Gates

Communication through signals is common in daily life, from ancient methods like drumming to modern examples such as railway guards using flags and horns or cars signaling when a door is open. Just as these signals guide actions and decisions, computers rely on signals through circuits to make logical decisions.

These circuits, called **logic circuits**, use binary values (0s and 1s) to process information. At the core of these circuits are **logic gates**, which perform basic logical functions like AND, OR, and NOT. Each gate takes inputs and produces outputs based on certain rules, allowing the computer to process data and make decisions.

In computers, these gates combine to form complex digital systems like the **Central Processing Unit (CPU)**, which controls almost every aspect of the machine's operation. Understanding how logic gates work is key to grasping how computers perform tasks, process information, and execute commands efficiently.



A logic gate operates by producing an output based on one or more inputs. There are various technical approaches to constructing logic gates, and their internal circuits typically include components like transistors, diodes, and resistors. Depending on how these circuits are utilized, logic gates can be categorized into two main types:

1. Basic Logic Gates
2. Combinational Logic Gates

Basic Logic Gates

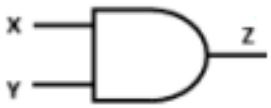
There are three types of basic logic gates. Those are,

1. AND gate
2. OR gate
3. NOT gate

1. AND Gate

Let's consider a similar example to understand AND logic.

Imagine a safe that requires two different codes to unlock it—one for the outer lock and another for the inner lock. To open the safe, you must enter both the outer and inner lock codes correctly. If only one of the codes is entered, the safe remains locked. Likewise, if neither code is entered, the safe cannot be opened.



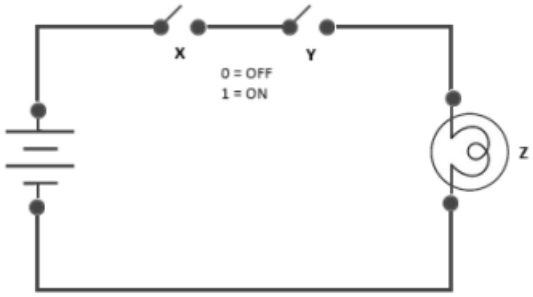
Symbol

Z = X.Y

Boolean Expression

X	Y	Z = X*Y
0	0	0
0	1	0
1	0	0
1	1	1

Truth Table



Electrical Circuit

2. OR Gate

E.g.- A student wants to borrow a book from the library. They can either check out the book using their library card **or** use their student ID. As long as they have one of these, they can borrow the book.

Y = A + B

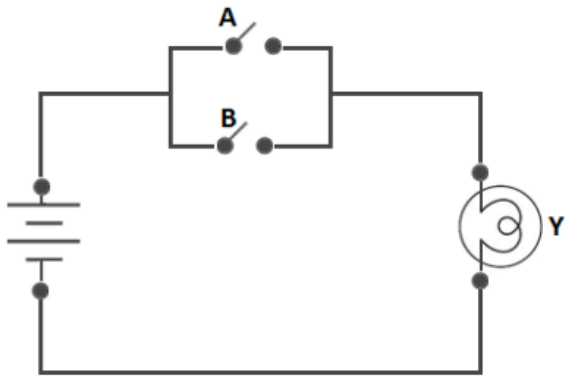
Boolean Expression



Symbol

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

Truth Table



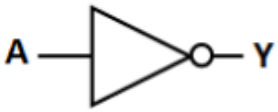
Electrical Circuit

3. NOT Gate

The task of the NOT gate is to provide the complement as the output. Thus, complement of "0" is "1" and the complement of "1" is "0".

$A = \overline{A}$

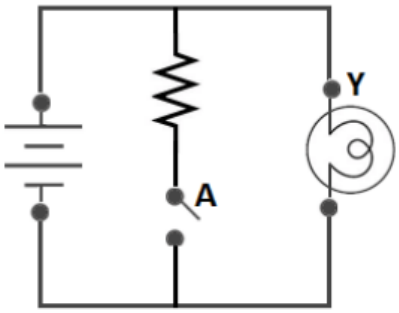
Boolean Expression



Symbol

A	Y = Ā
0	1
1	0

Truth Table

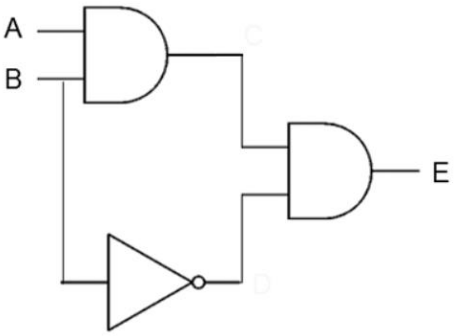


Electrical Circuit

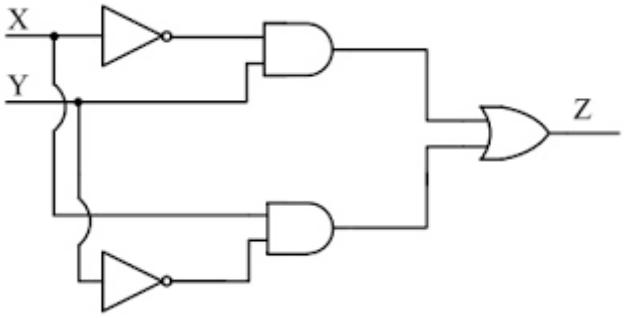
Exercise 01

Write the Boolean expressions and truth tables for the following simple logical circuits.

1.



2.



Combinational Gates

There are four types of combinational logic gates. Those are,

1. XOR gate
2. KNOR gate
3. NAND gate
4. NOR gate

1. XOR Gate

The XOR gate (pronounced as Exclusive OR gate) is a digital logic gate that results a true if and only one of the inputs to the gate is true. Otherwise, output is false.



(a) Circuit symbol

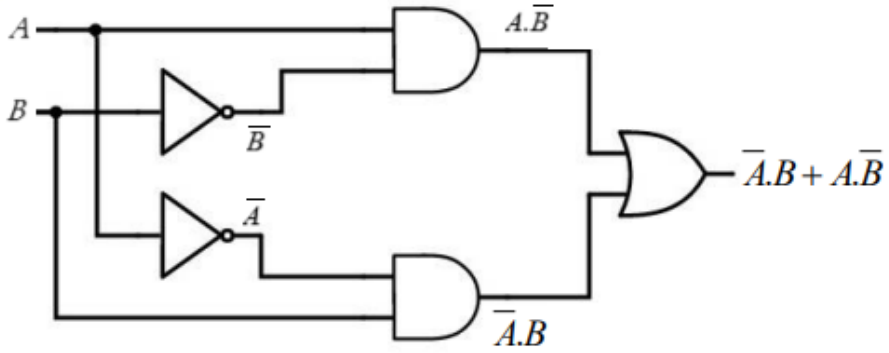
S1	S2	L
0	0	0
0	1	1
1	0	1
1	1	0

(b) Truth table

$L = S1 \oplus S2$

(c) Boolean expression

- **XOR gate using basic logic gates**



A	B	$\overline{A}$	$\overline{B}$	$A.\overline{B}$	$\overline{A}.B$	$A.\overline{B} + \overline{A}.B$
0	0	1	1	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	1
1	1	0	0	0	0	0

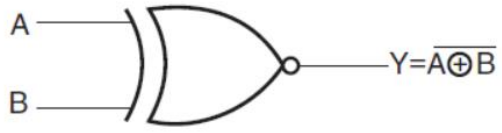
- **Truth table for three input XOR gate**

INPUTS				Final Output
S1	S2	S3	S1 ⊕ S2	S1 ⊕ S2 ⊕ S3
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	0
1	0	0	1	1
1	0	1	1	0
1	1	0	0	0
1	1	1	0	1

***** Compare S1 & S2 and if the inputs are same then output is zero (false). Otherwise, output is one. Then compare the result of S1 ⊕ S2 with S3. if the values are same then final output is 0. else final output is 1.

2. XNOR Gate

The XNOR gate is a digital logic gate whose function is the logical complement of the XOR gate.



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

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

- **Truth table for three input XNOR gate**

INPUTS					Final Output
S1	S2	S3	S1 ⊕ S2	S1 ⊕ S2 ⊕ S3	$\overline{S1 \oplus S2 \oplus S3}$
0	0	0	0	0	1
0	0	1	0	1	0
0	1	0	1	1	0
0	1	1	1	0	1
1	0	0	1	1	0
1	0	1	1	0	1
1	1	0	0	0	1
1	1	1	0	1	0

Universal Gates

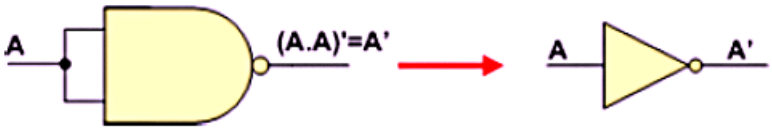
A universal logic gate is a logic gate that can be used to construct all other logic gates. The NAND gate and NOR gates can be considered as universal logic gates.

The advantage of universal gates:

NAND and NOR gates are economical and easier to fabricate and are the basic gates used in all IC digital logic families.

1. NAND Gate

- **How to implement NOT gate using NAND gates?**



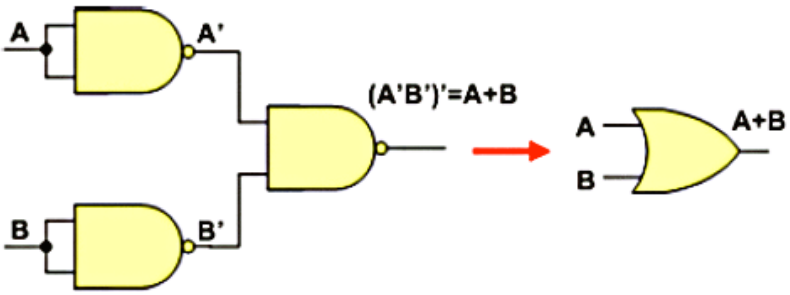
A	$\overline{A}$
0	1
1	0

- **How to implement AND gate using NAND gates?**



A	B	$\overline{A.B}$	$\overline{\overline{A.B}}$
0	0	1	0
0	1	1	0
1	0	1	0
1	1	0	1

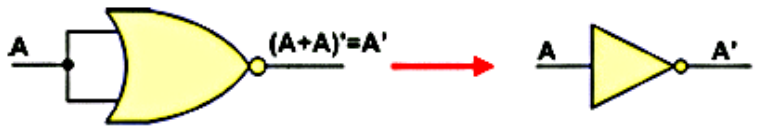
- **How to implement OR gate using NAND gates?**



A	B	$\overline{A}$	$\overline{B}$	$\overline{\overline{A} \cdot \overline{B}}$
0	0	1	1	0
0	1	1	0	1
1	0	0	1	1
1	1	0	0	1

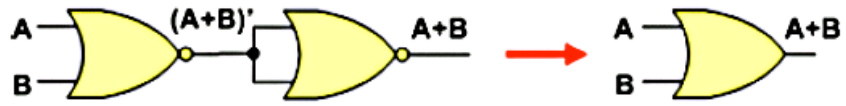
2. NOR Gate

- How to implement NOT gate using NOR gates?



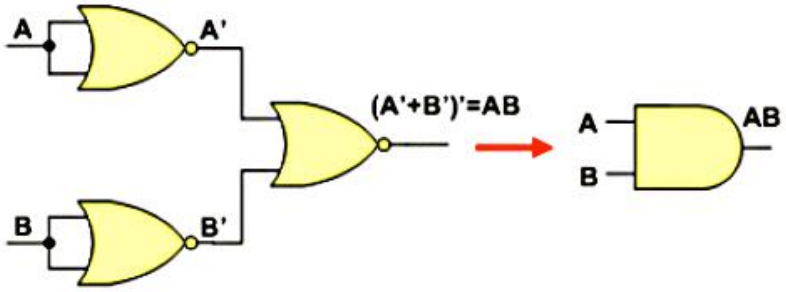
A	$\overline{A}$
0	1
1	0

- How to implement OR gate using NOR gates?

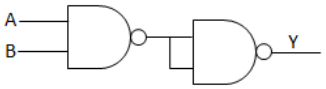
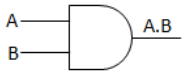
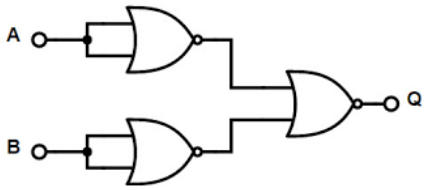
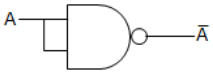
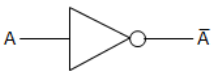
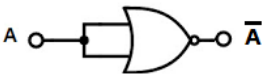
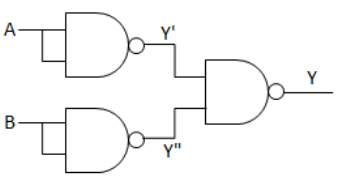
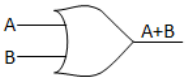
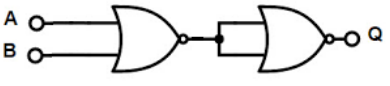


A	B	$\overline{A+B}$	$\overline{\overline{A+B}}$
0	0	1	0
0	1	0	1
1	0	0	1
1	1	0	1

- How to implement AND gate using NOR gates?



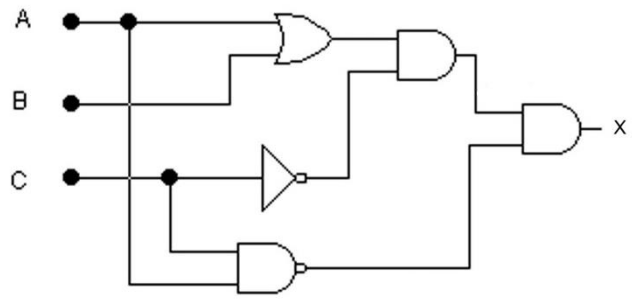
A	B	$\overline{A}$	$\overline{B}$	$\overline{\overline{A}+\overline{B}}$
0	0	1	1	0
0	1	1	0	0
1	0	0	1	0
1	1	0	0	1

Using **NAND**	Basic operator	Using **NOR**
		
		
		

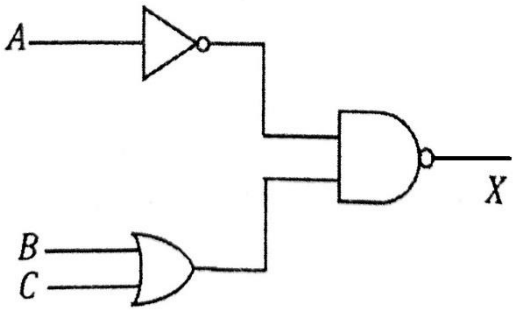
Exercise 02

Write the Boolean expressions and truth tables for the following logical circuits.

1.



2.



Draw truth tables and logical circuits for the following Boolean expressions.

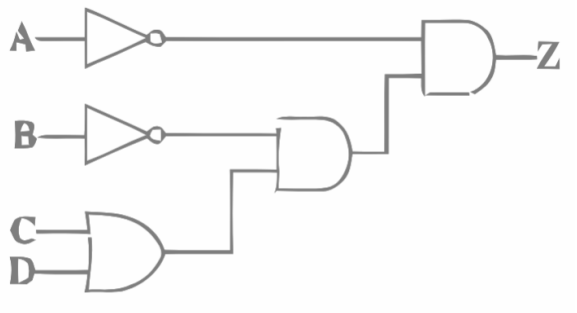
3. (A.B.C) + A.($\bar{B}$+$\bar{C}$)

4. A($\overline{B + C}$) + B$\bar{C}$ + B$\bar{A}$

5. $\overline{\bar{C}.(A + BC)}$

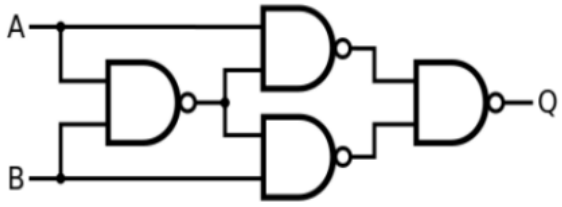
6. $\overline{AB}$. B+$\bar{C}$

7. What is the output Z of the following logic circuit?



(1) $AB(C + D)$ (2) $ABCD$ (3) $(\overline{A + B})(C + D)$ (4) $\bar{A}B(C + D)$ (5) $\bar{A}\bar{B}(\bar{C} + D)$

8. Consider the following logic circuit.



Which of the following Boolean expression is equivalent to the output Q of the logic circuit given above?

(1) $\bar{A}B + A\bar{B}$ (2) $AB + \bar{A}\bar{B}$ (3) $AB + \bar{A}$ (4) $A + \bar{B}$ (5) A + B

Introduction to Boolean laws

- What is the requirement of Boolean algebra simplification?
  - Simplification of Boolean expressions reduces the number of operations and number of logic gates for the implementation.

Most laws exist in two forms. One is multiplicative form and the other is additive form. At the multiplication, variables in the Boolean expression are multiplied by each other and at the addition, variables in the Boolean expression are added to each other.

1. Idempotent Law

Law and its representation from a truth table

$A.A=A$

A	A	A.A
0	0	0
1	1	1

$A+A=A$

A	A	A+A
0	0	0
1	1	1

$\overline{A}.\overline{A} = \overline{A}$

A	$\overline{A}$	$\overline{A}$	$\overline{A}.A$
0	1	1	1
1	0	0	0

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

A	$\overline{A}$	$\overline{A}$	$\overline{A}+\overline{A}$
0	1	1	1
1	0	0	0

Above highlighted columns are identical.

2. Identity Law

$1.A=A$

1	A	1.A
1	0	0
1	1	1

$0+A=A$

0	A	0+A
0	0	0
0	1	1

$0.A = 0$

0	A	0.A
0	0	0
0	1	0

$1+A = 1$

1	A	1+A
1	1	1
1	0	1

Above highlighted columns are identical.

3. Inverse/Complement Law

Multiplicative form

$A.\overline{A}=0$

A	$\overline{A}$	$A.\overline{A}$
0	1	0
1	0	0

Additive form

$A+\overline{A}=1$

A	$\overline{A}$	$A+\overline{A}$
0	1	1
1	0	1

4. De Morgan's Law

A mathematician named De Morgan developed a pair of important laws regarding group complementation in Boolean algebra.

Multiplicative form

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

A	B	AB	$\overline{A.B}$	$\overline{A}$	$\overline{B}$	$\overline{A}+\overline{B}$
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

Additive form

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

A	B	A+B	$\overline{A+B}$	$\overline{A}$	$\overline{B}$	$\overline{A}.\overline{B}$
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

Above highlighted columns are identical.

5. Double Complement Law

$A = \overline{\overline{A}}$

A	$\overline{A}$	$\overline{\overline{A}}$
0	1	0
0	1	0

Above highlighted columns are identical.

6. Commutative Law

Multiplicative form

$AB = BA$

A	B	AB	BA
0	0	0	0
0	1	0	0
1	0	0	0
1	1	1	1

Additive form

$A + B = B + A$

A	B	A+B	B+A
0	0	0	0
0	1	1	1
1	0	1	1
1	1	1	1

Above highlighted columns are identical.

7. Associative Law

Multiplicative form

A(BC)=(AB)C

A	B	C	BC	AB	A(BC)	(AB)C
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	0
0	1	1	1	0	0	0
1	0	0	0	0	0	0
1	0	1	0	0	0	0
1	1	0	0	1	0	0
1	1	1	1	1	1	1

Above highlighted columns are identical.

Additive form.

A + (B + C) = (A + B) + C

A	B	C	B+C	A+B	A+(B+C)	(A+B)+C
0	0	0	0	0	0	0
0	0	1	1	0	1	1
0	1	0	1	1	1	1
0	1	1	1	1	1	1
1	0	0	0	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Above highlighted columns are identical.

8. Distributive Law

A (B+C) = AB+AC

A	B	C	B+C	AB	AC	A (B+C)	AB+AC
0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0
0	1	0	1	0	0	0	0
0	1	1	1	0	0	0	0
1	0	0	0	0	0	0	0
1	0	1	1	0	1	1	1
1	1	0	1	1	0	1	1
1	1	1	1	1	1	1	1

Above highlighted columns are identical.

9. Redundancy Law

Form 1

A + AB =A

A	B	AB	A+AB
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

Form 2

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

A	B	A'	A'B	A+A'B	A+B
0	0	1	0	0	0
0	1	1	1	1	1
1	0	0	0	1	1
1	1	0	0	1	1

Above highlighted columns are identical.

Identity Name	AND Form	OR Form
Identity Law	$1x = x$	$0+x = x$
Null (or Dominance) Law	$0x = 0$	$1+x = 1$
Idempotent Law	$xx = x$	$x+x = x$
Inverse Law	$x\bar{x} = 0$	$x+\bar{x} = 1$
Commutative Law	$xy = yx$	$x+y = y+x$
Associative Law	$(xy)z = x(yz)$	$(x+y)+z = x+(y+z)$
Distributive Law	$x+yz = (x+y)(x+z)$	$x(y+z) = xy+xz$
Absorption Law	$x(x+y) = x$	$x+xy = x$
DeMorgan's Law	$(\overline{xy}) = \bar{x}+\bar{y}$	$(\overline{x+y}) = \bar{x}\bar{y}$
Double Complement Law	$\bar{\bar{x}} = x$	

Examples

1.

$A + B(A + C) + AC$

↓ Distributing terms

$A + AB + BC + AC$

↓ Applying rule $A + AB = A$ to 1st and 2nd terms

$A + BC + AC$

↓ Applying rule $A + AB = A$ to 1st and 3rd terms

$A + BC$

2.

$AB + BC(B + C)$

↓ Distributing terms

$AB + BBC + BCC$

↓ Applying identity $AA = A$ to 2nd and 3rd terms

$AB + BC + BC$

↓ Applying identity $A + A = A$ to 2nd and 3rd terms

$AB + BC$

↓ Factoring B out of terms

$B(A + C)$

3.

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

↓ Factoring BC out of 1st and 4th terms

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

↓ Applying identity $A + \bar{A} = 1$

$BC(1) + A\bar{B}C + AB\bar{C}$

↓ Applying identity $1A = A$

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

↓ Factoring B out of 1st and 3rd terms

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

↓ Applying rule $A + \bar{A}B = A + B$ to the $C + A\bar{C}$ term

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

↓ Distributing terms

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

↓ Factoring A out of 2nd and 3rd terms

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

↓ Applying rule $A + \bar{A}B = A + B$ to the $B + B\bar{C}$ term

$BC + A(B + C)$

↓ Distributing terms

$BC + AB + AC$

or

$AB + BC + AC$ Simplified result

4.

$\overline{\overline{A + BC} + \overline{A\bar{B}}}$

↓ Breaking longest bar

$\overline{\overline{(A + BC)}}\ \overline{\overline{(A\bar{B})}}$

↓ Applying identity $\bar{\bar{A}} = A$ wherever double bars of equal length are found

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

↓ Distributive property

$AA\bar{B} + BCA\bar{B}$

↓ Applying identity $AA = A$ to left term; applying identity $A\bar{A} = 0$ to B and $\bar{B}$ in right term

$A\bar{B} + 0$

↓ Applying identity $A + 0 = A$

$A\bar{B}$

Exercise 03

Simplify the following Boolean expressions using Boolean laws.

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

2. F = (A+B) (A+C)

3. F=C+ (BC)’

4. X(X+Y)

5. ABC+$\bar{A}$BC+AB$\bar{C}$+A$\bar{B}\bar{C}$

6. AB+$\bar{A}$ (B+C)

Minterms and Maxterms

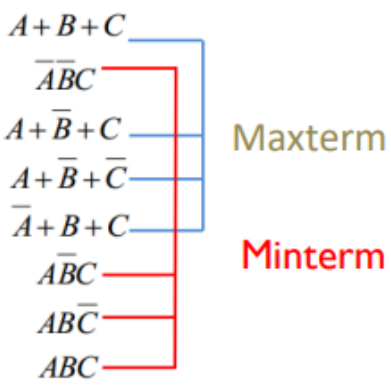
Minterm

Each row of a truth table can be associated with a minterm, which is a product (AND) of all variables in the function, in direct or complemented form. A minterm has the property that it is equal to 1 on exactly one row of the truth table.

Maxterm

Each row of a truth table is also associated with a maxterm, which is a sum (OR) of all the variables in the function, in direct or complemented form. A maxterm has the property that it is equal to 0 on exactly one row of the truth table.

A	B	C	Z
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1



Standard forms in Boolean expressions

There are two standard forms of Boolean expressions.

-SOP (Sum of Product)

-POS (Product of Sum)

What is Sum of Product?

SOP form is a set of products(AND) terms that are summed(OR) together. When an expression or term is represented in a sum of binary terms known as minterms and sum of products.

Examples: AB + ABC ABC +CDE + $\bar{B}C\bar{D}$ $\bar{A}$B +$\bar{A}$B$\bar{C}$+ AC A+ $\bar{A}\bar{B}$C + BC$\bar{D}$

- In an SOP form, a single overbar cannot extend over more than one variable; however, more than one variable in a term can have an overbar:

  Example: $\bar{A}\bar{B}\bar{C}$ is correct But $\overline{ABC}$ is not correct

- A standard SOP expression is one in which all the variables in the domain appear in each product term in the expression.

  Example: A$\bar{B}$CD + $\bar{A}\bar{B}$C$\bar{D}$ + AB$\bar{C}\bar{D}$

  Standard SOP expressions are important in:
  - Constructing truth tables
  - The Karnaugh map simplification method

What is Product of Sum ?

It is a technique of explaining a Boolean expression through a set of max terms or sum terms, is known as POS.

Example:

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

- In a POS form, a single overbar cannot extend over more than one variable; however, more than one variable in a term can have an overbar:

  Example: $\bar{A}+\bar{B}+\bar{C}$ is correct. However, $\overline{A+B+C}$ is not correct

Transforming SOP into POS and vice versa

SOP TO POS	
$F = A.\overline{B} + B.\overline{C} + \overline{A}.C$	*Obtain the compliment of whole equation by adding an over bar to the whole*
$= \overline{A.\overline{B} + B.\overline{C} + \overline{A}C}$	
$= \overline{A.\overline{B}} + \overline{B.\overline{C}} + \overline{\overline{A}C}$	De Morgan's Law
$= (\overline{A} + \overline{\overline{B}}).(\overline{B} + \overline{\overline{C}}) + (\overline{\overline{A}} + \overline{C})$	De Morgan's Law
$= \underline{(\overline{A} + B).(\overline{B} + C).(A + \overline{C})}$	Double Inverse

POS TO SOP	
$F = (\overline{A} + B).(\overline{B} + C).(A + \overline{C})$	*Obtain the compliment of whole equation by adding an over bar to the whole*
$= \overline{(\overline{A} + B).(\overline{B} + C).(A + \overline{C})}$	
$= (\overline{\overline{A} + B}) + (\overline{\overline{B} + C}) + (\overline{A + \overline{C}})$	De Morgan's Law
$= (\overline{\overline{A}.B}) + (\overline{\overline{B}.C}) + (\overline{A}.\overline{\overline{C}})$	De Morgan's Law
$= \underline{A.\overline{B} + B.\overline{C} + \overline{A}.C}$	Double Inverse

Exercise 04 - Convert the following SOP to POS and POS to SOP.

1. ABC + $\bar{A}$BC+ A$\bar{B}$C +A$\bar{B}\bar{C}$

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

Karnaugh Maps

Karnaugh maps provide an alternative way of simplifying logic circuits. Instead of using Boolean algebra simplification techniques, you can transfer logic values from a Boolean statement or a truth table into a Karnaugh map. The arrangement of 0's and 1's within the map helps you to visualize the logic relationships between the variables and leads directly to a simplified Boolean statement.

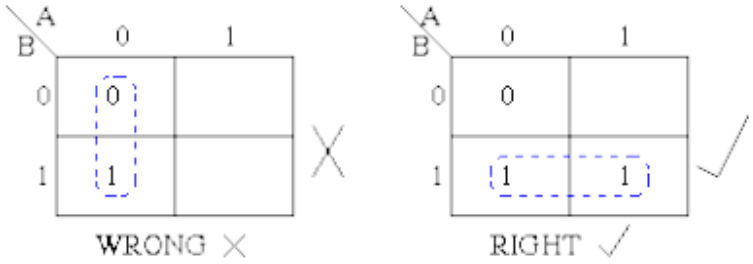
No. of cells = 2^n Where n is the number of input variables

Rules of K- map simplification

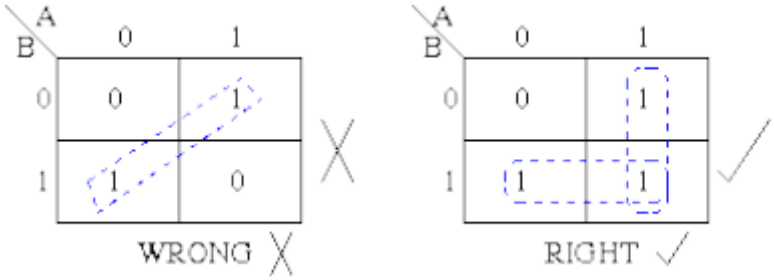
1. No zeros allowed. 2. No diagonals. 3. Only power of 2 number of cells in each group. 4. Groups should be as large as possible.	5. Every "one" must be in at least one group. 6. Overlapping allowed. 7. Wrap around allowed. 8. Fewest number of groups possible.

The Karnaugh map uses the following rules for the simplification of expressions by grouping together adjacent cells containing ones.

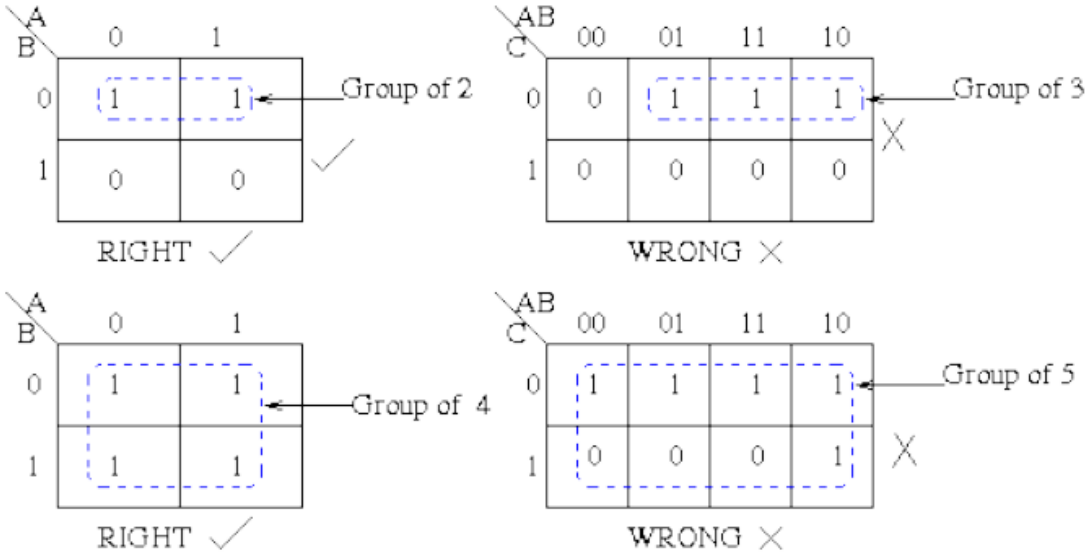
1. Groups may not include any cell containing a zero.



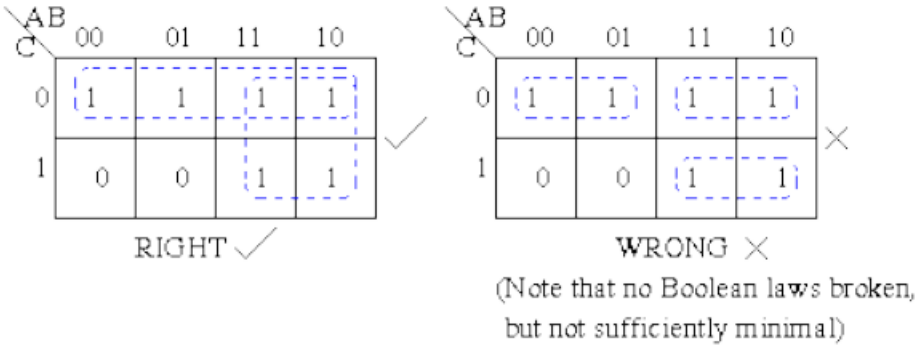
2. Groups may be horizontal or vertical, but not diagonal.



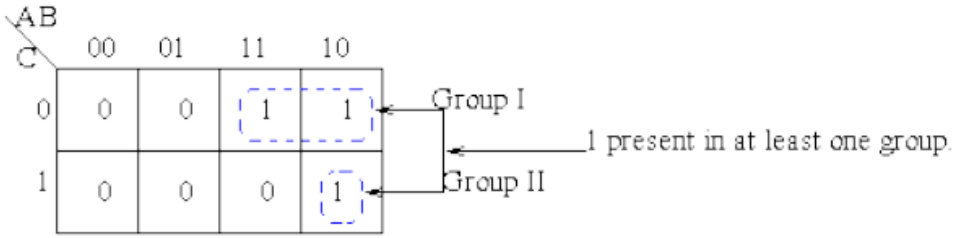
3. Groups must contain 1, 2, 4, 8, or in general 2^n cells. That is if n = 1, a group will contain two 1's since $2^1 = 2$. If n = 2, a group will contain four 1's since $2^2 = 4$.



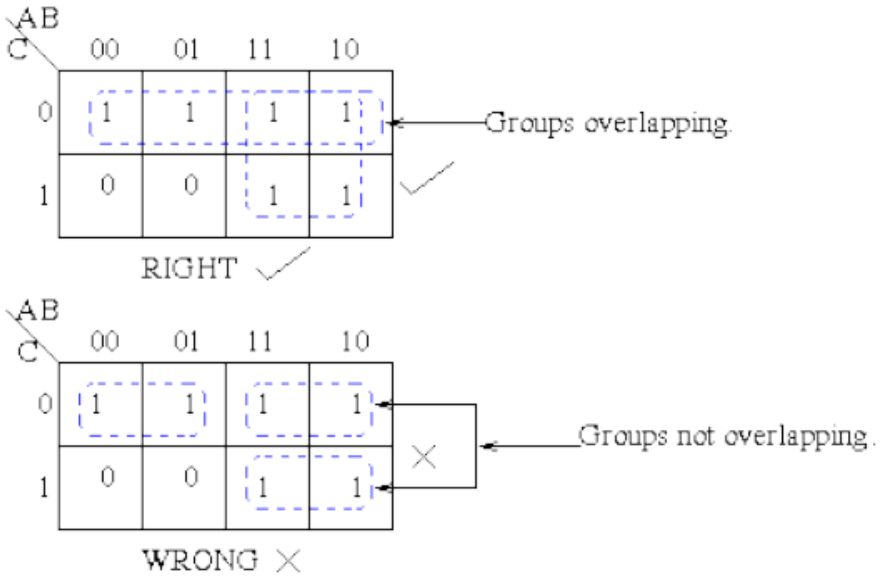
4. Each group should be as large as possible.



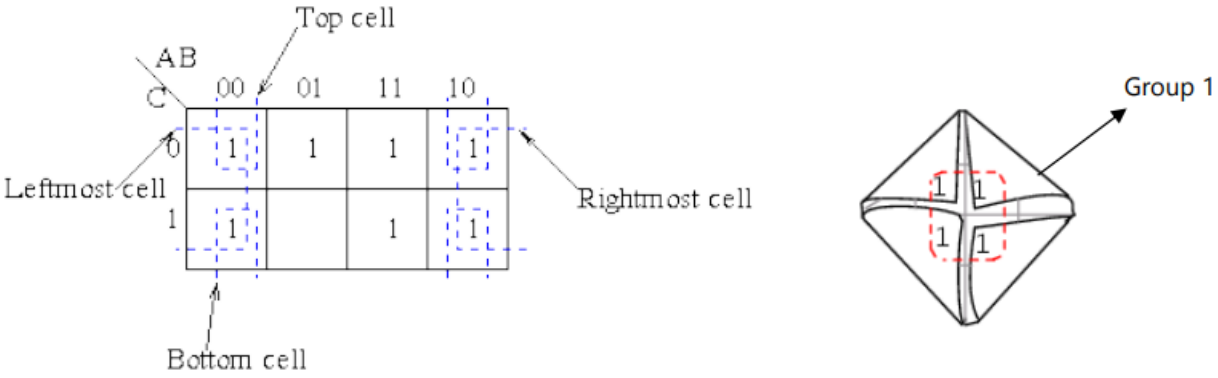
5. Each cell containing a one must be in at least one group.



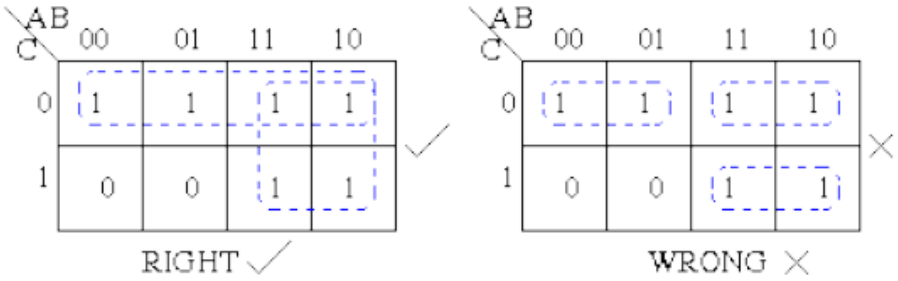
6. Groups may overlap.



7. Groups may wrap around the table. The leftmost cell in a row may be grouped with the rightmost cell and the top cell in a column may be grouped with the bottom cell.



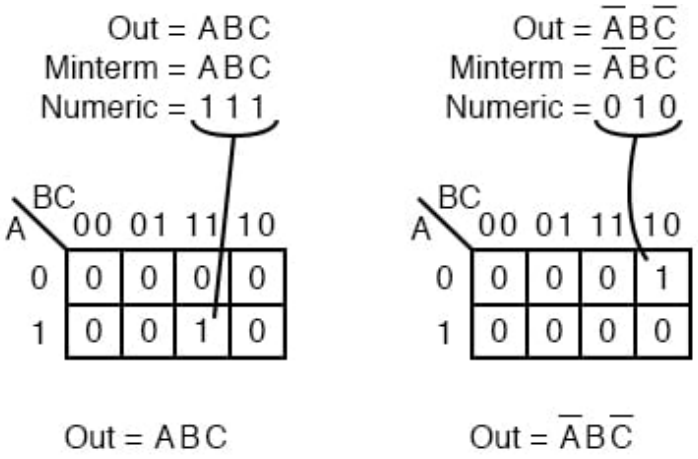
8. There should be as few groups as possible, as long as this does not contradict any of the previous rules.



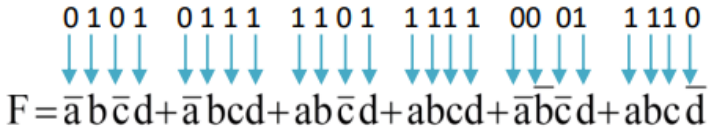
Simplifying Boolean expressions using K-maps.

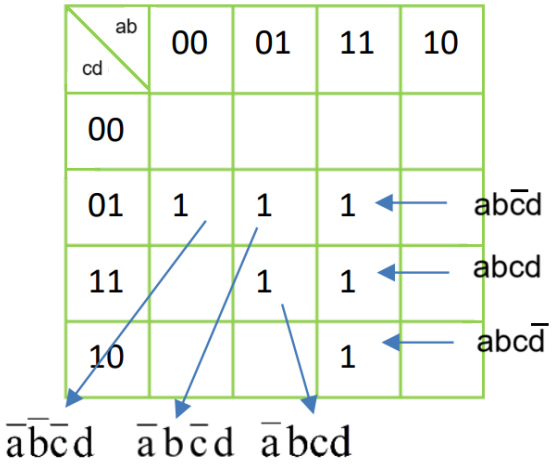
SOP format- In a Boolean expression, any variable with a bar, is considered as a 0 and others are considered as 1 (e.g. - A is 1 and $\bar{A}$ *is* 0)

- 3 input variables



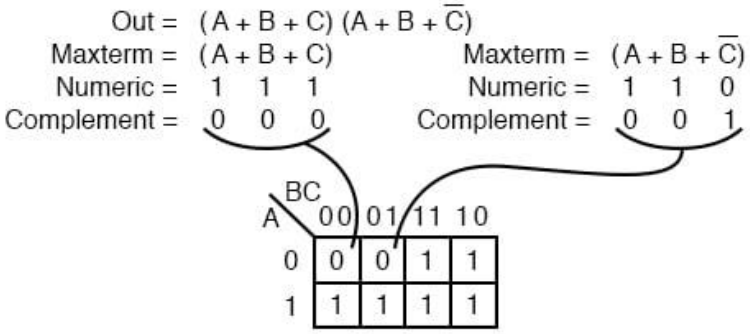
- 4 input variables



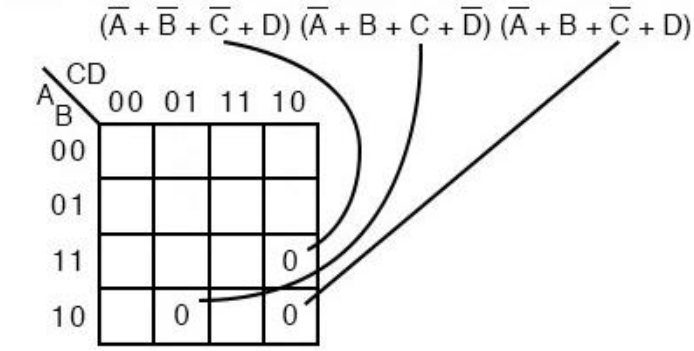


POS format- In a Boolean expression, any variable with a bar, is considered as a 1 and others are considered as 0 (e.g. - A is 0 and $\overline{A}$ *is* 1)

- 3 input variables

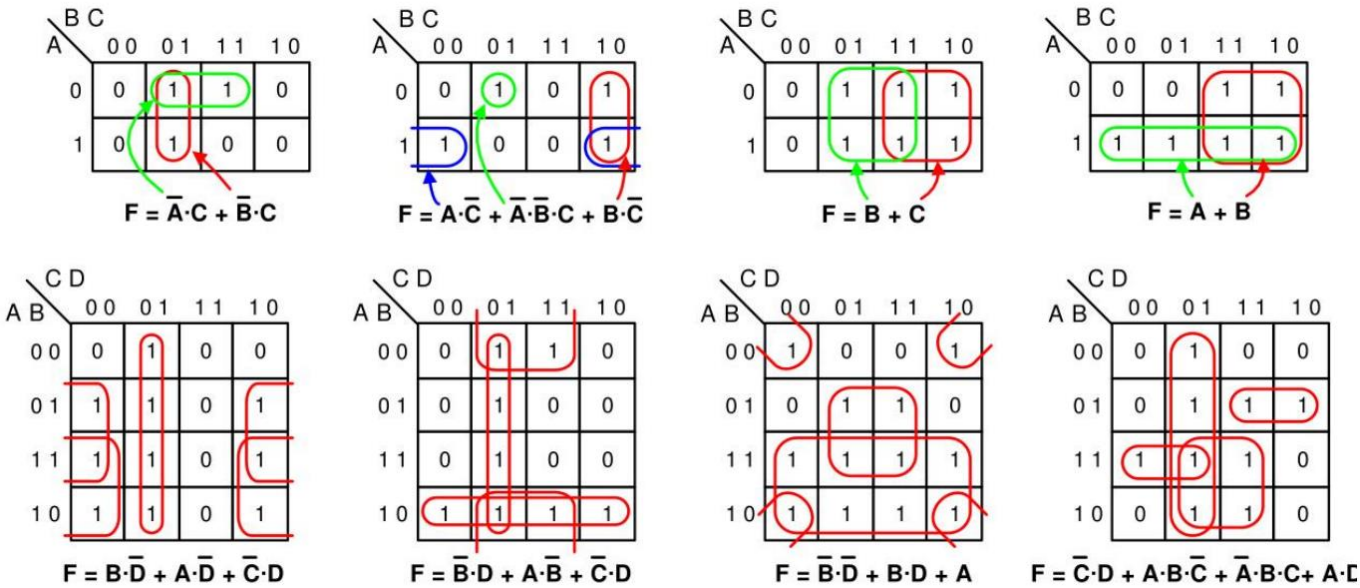


- 4 input variables

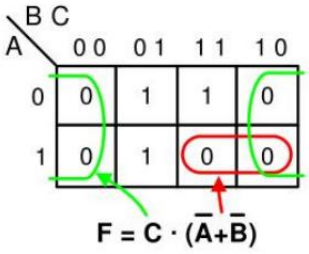


Examples:

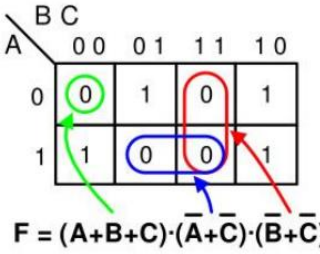
SOP loops



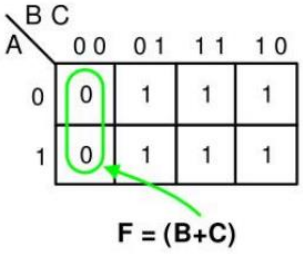
POS loops



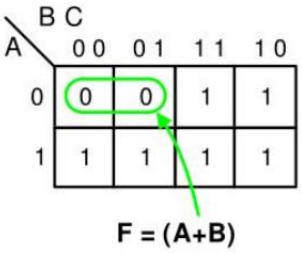
$F = C \cdot (\bar{A}+\bar{B})$



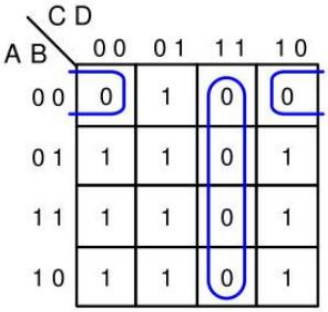
$F = (A+B+C)\cdot(\bar{A}+\bar{C})\cdot(\bar{B}+\bar{C})$



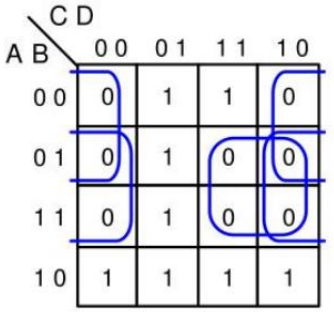
$F = (B+C)$



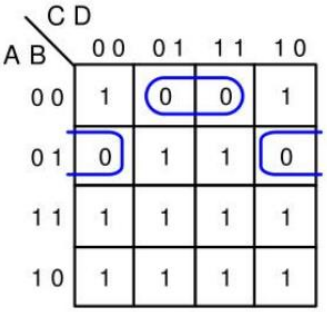
$F = (A+B)$



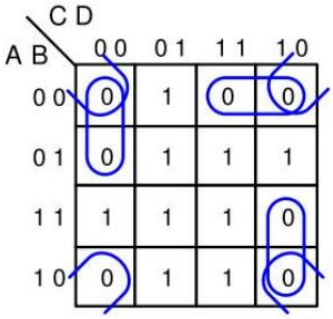
$F = (A+B+D)\cdot(\bar{C}+\bar{D})$



$F = (A+D)\cdot(\bar{B}+D)\cdot(\bar{B}+\bar{C})$



$F = (A+B+\bar{D})\cdot(A+\bar{B}+D)$



$F = (A+C+D)\cdot(A+B+\bar{C})\cdot(\bar{A}+\bar{C}+D)\cdot(B+D)$

Exercise 05

a) Find the simplified SOP of the k-maps given.

Z \ XY	00	01	11	10
0	1	1	1	1
1	1	0	1	1

C \ AB	00	01	11	10
0	1	0	1	0
1	0	1	0	1

CD \ AB	00	01	11	10
00	1	0	0	1
01	0	0	1	0
11	0	1	0	0
00	1	1	1	1

YZ \ WX	00	01	11	10
00	1	0	1	1
01	0	1	1	0
11	1	0	0	0
10	1	0	0	1

b) Find the simplified POS of the k-maps given

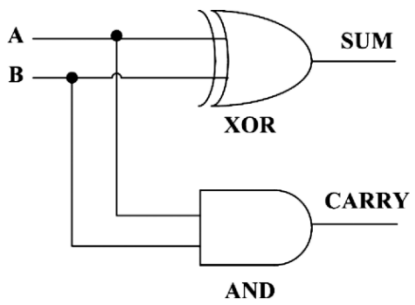
C \ AB	00	01	11	10
0	0	1	0	1
1	0	0	0	1

Z \ XY	00	01	11	10
0	0	1	0	1
1	0	0	1	0

Half adder

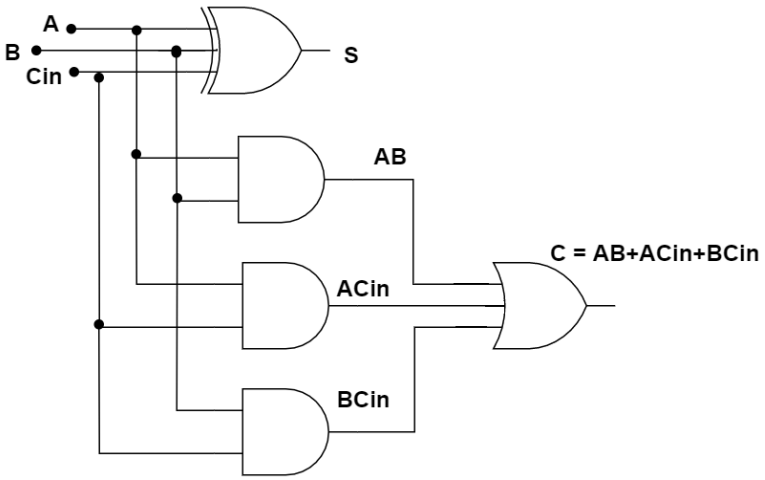
A **half adder** is a basic digital circuit used to add two single-bit binary numbers, producing two outputs: **Sum (S)** and **Carry (C)**. The sum represents the result of the addition, while the carry indicates any overflow to the next bit. The sum is calculated using an XOR gate (S = A ⊕ B), and the carry is determined using an AND gate (C = A . B). Unlike a full adder, a half adder does not account for any carry input from previous additions, making it suitable for simple, single-bit operations.

INPUTS		OUTPUTS	
A	**B**	**SUM**	**CARRY**
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1



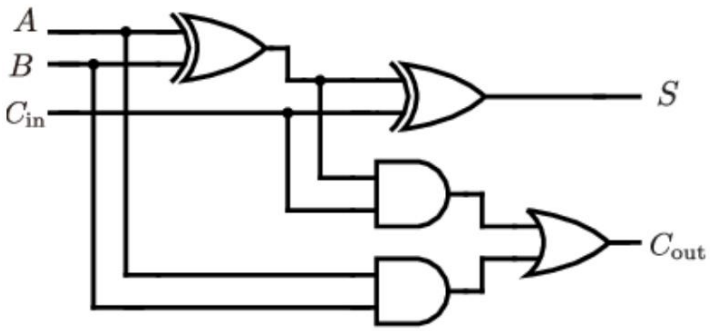
Full Adder

The main difference between a half-adder and a full-adder is that the **full-adder has three inputs** and two outputs. The first two inputs are A and B and the third input is an input carry designated as Carry In(C_{in}). The outputs, carry is designated as Carry Out (C_{out} = A . B . C) and the sum is designated as Sum (S = A⊕ B ⊕ C).

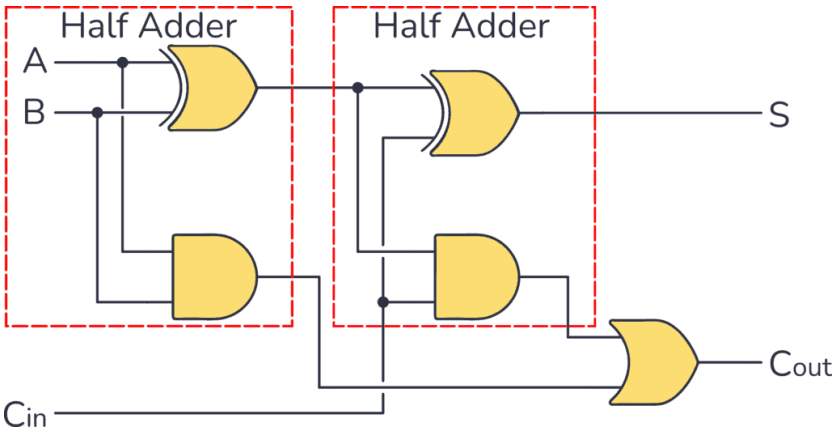


Inputs			Outputs	
A	B	C_{in}	S	C_{out}
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Method 2:



- **Full adder using two half adders**



Combinational circuits Vs Sequential circuits

Feature	Combinational Circuits	Sequential Circuits
Dependence	Only on current inputs	On current inputs and previous states
Memory Elements	None	Present
Applications	Arithmetic operations, Data routing	Memory devices, Digital clocks, State machines
Examples	Adders, Multiplexers, Encoders, Decoders	Flip-flops, Counters, Registers

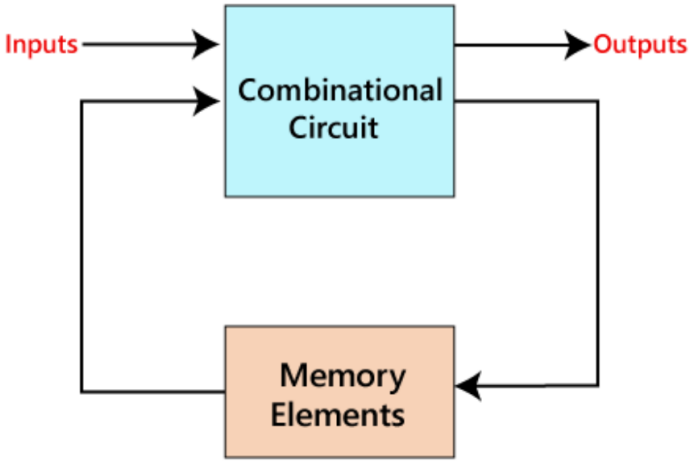


Figure – structure of a sequential circuit

Flip Flops

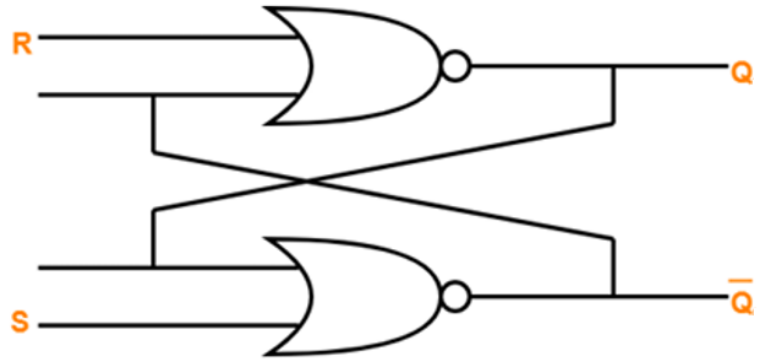
Flip flops is an application of logic gates. With the help of Boolean logic, you can create memory with them. Flip flops can also be considered as the most basic idea of a Random-Access Memory. When a certain input value is given to them, they will be remembered.

- **Latch Flip-flop**

R-S (Reset Set) flip flop is the simplest flip flop of all and easiest to understand. It is basically a device which has two outputs one output being the inverse or complement of the other, and two inputs. A pulse on one of the inputs to take on a particular logical state. The outputs will then remain in this state until a similar pulse is applied to the other input. The two inputs are called the Set and Reset input (sometimes called the preset and clear inputs).

Such flip flop can be made simply by cross coupling two inverting gates, either NAND or NOR gate.

- Latch constructed using **NOR** gates

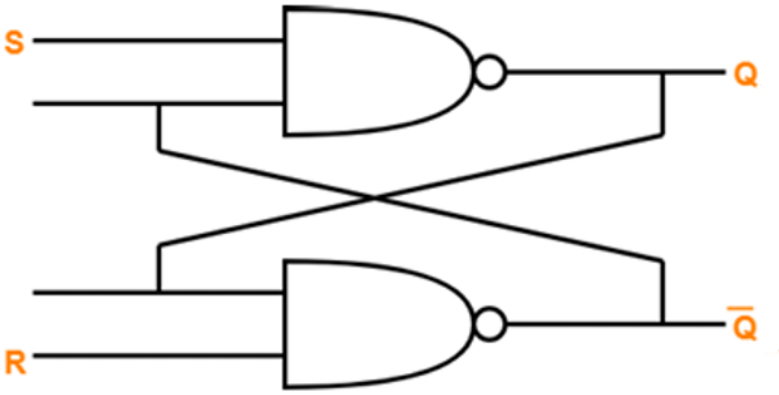


Truth Table

Set	Reset	Output
0	0	No change*
1	0	Q=1
0	1	Q=0
1	1	Invalid state

*can be used for data storage

- Latch constricted using **NAND** gates



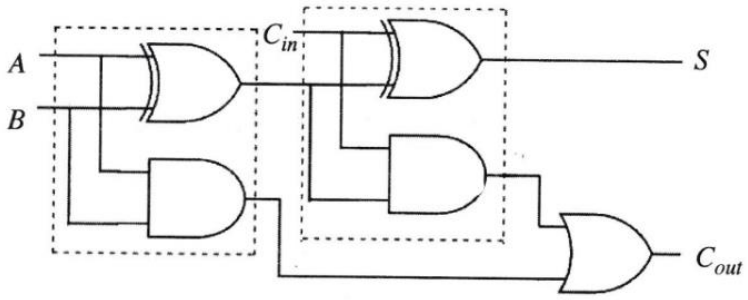
Truth Table

Set	Reset	Output
1	1	No change*
0	1	Q=1
1	0	Q=0
0	0	Invalid state

*can be used for data storage

AL Questions
2019 AL

- Consider the following logic circuit diagram to answer the questions 37 and 38:



37. Which of the following statements is/are correct about the above circuit?

I - It implements a full adder.
II - The logic function of S can be stated as $S = A \oplus B \oplus C_{in}$.
III - The logic function of C_{out} can be stated as $C_{out} = AB + BC_{in} + AC_{in}$.

(1) I only (2) II only (3) I and II only (4) I and III only (5) All I, II and III

38. Which of the following statements is/are correct about the part of the circuit within the area surrounded by the dotted line?

I - It implements a half adder.
II - It can be implemented using only AND and OR gates.
III - It can be implemented using only NAND gates.

(1) I only (2) II only (3) III only (4) I and III only (5) All I, II and III

39. Consider the Karnaugh map shown below:

C \ AB	00	01	11	10
0	0	0	0	1
1	1	1	0	1

Which of the following is the correct logic expression that corresponds to the two marked segments on the Karnaugh map?

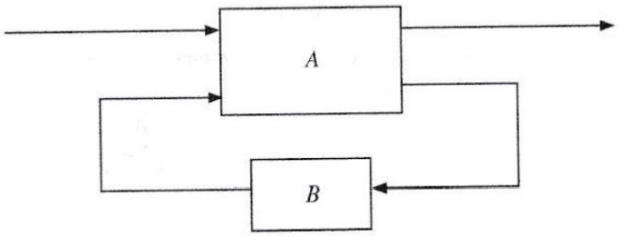
(1) $A\bar{B} + B\bar{C}$ (2) $\bar{A}\bar{C} + AB$ (3) $(\bar{A} + \bar{C})(A+B)$ (4) $(A+C)(\bar{A} + \bar{B})$ (5) $AC + \bar{A}\bar{B}$

2020 AL

9. Which of the following is the most simplified expression equivalent to A$\bar{B}\bar{C}$ + B$\bar{C}$ + $\bar{A}\bar{B}$C + BC?

(1) A$\bar{B}\bar{C}$ + $\bar{A}\bar{B}$C+ B
(2) $\bar{B}$ (A$\bar{C}$ +$\bar{A}$C)+ B
(3) $\bar{C}$ (A$\bar{B}$+B)+ C($\bar{A}\bar{B}$+B)
(4) A$\bar{C}$ + $\bar{A}$C + B
(5) $\bar{A}\bar{C}$ +B

10. A block diagram of a sequential logic circuit is shown below, with one block labelled as "A" and the other labelled as "B".

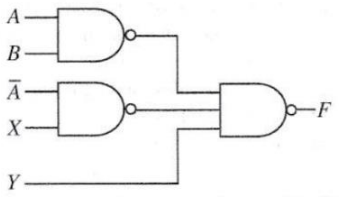


Which of the following statements about the above block diagram are correct?

I - The block A is a combinational logic circuit.
II - The block B is a memory element.
III - Only the block A can be implemented using logic gates.

(1) Only I
(2) Only II
(3) Only I and II
(4) Only I and III
(5) All I, II and III

11. Consider the following logic circuit consisting of NAND gates:



If the required output of the above circuit is AB+AB+C, then what should the inputs X and Y be?

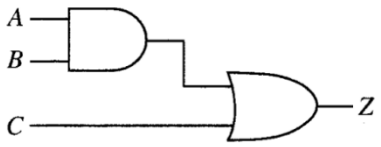
(1) X = B and Y = C
(2) X = B and Y = $\bar{C}$
(3) X = B and Y = C
(4) X = $\bar{B}$ and Y = $\bar{C}$
(5) X = $\bar{C}$ and Y = B

2021 AL

9. Consider the following Karnaugh map and the logic circuit implemented based on it where A, B and C are the inputs and Z is the output:

C \ AB	00	01	11	10
0	0	e	f	0
1	1	g	h	1

(*a*) Karnaugh map



(*b*) Logic circuit based on Karnaugh map

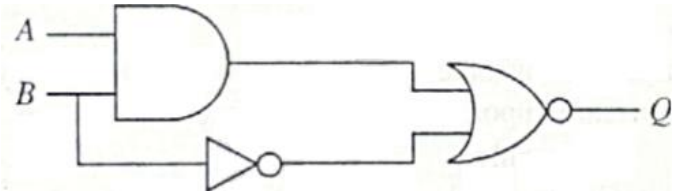
For the logic circuit to correctly implement the logic function represented in the Karnaugh map, what should be the values of **e, f, g, h**?

(1) e=0, f=0, g=1, h=1 (2) e=0, f=1,g=1, h=1 (3) e=1, f=0, g=1, h=1

(4) e=1,f=1, g=0, h=0 (5) e=1,f=1, g=0, h=1

2022 AL

9. Consider the following logic circuit:



When B=1, what would definitely be the output at Q?

(1) A (2) $\bar{A}$ (3) B (4) $\bar{B}$ (5) 0

10. Simplified Boolean expressions help to obtain simpler circuits.

Which of the following is a simplified form of X + $\bar{X}$Y ?

(1) X (2) Y (3) XY (4) $\bar{X}$Y (5) X+ Y

11. Consider the following truth table:

A	B	C	Z
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

What is the correct Karnaugh map for the above truth table?

(1)

A \ BC	00	01	10	11
0	0	0	1	1
1	1	1	0	0

(2)

A \ BC	00	01	11	10
0	0	0	1	1
1	1	1	0	0

(3)

A \ BC	00	10	01	11
0	0	1	0	1
1	1	0	1	0

(4)

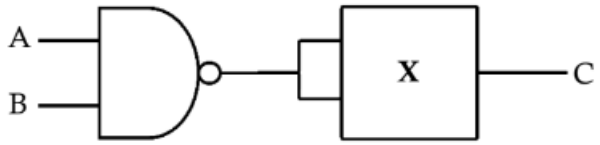
A \ BC	00	10	11	01
0	0	1	1	0
1	1	0	0	1

(5)

A \ BC	00	11	10	01
0	0	1	1	0
1	1	0	0	1

2023 AL

12. Consider the following logic circuit in which X indicates a two-input logic gate.



Which of the following should X be so that when A = 0 and B = 1, the output C would be 0?

I – a NAND gate
II – a NOR gate
III – an XOR gate

(1) I only (2) I and II only (3) I and III only (4) II and III only (5) All I, II and III

13. Which of the following is the simplified form of the Boolean expression $X(\overline{X} + Y)$?

(1) X (2) Y (3) XY (4) $\overline{XY}$ (5) X + Y

2019 AL (Part II) (Part B)

1. The Boolean function known as the majority function takes n binary inputs and outputs 1 if a majority (at least half) of the inputs are 1, otherwise it outputs 0. Let us consider the case when n=3, which is the 3-input majority function, whose inputs are A, B and C and the output is Z.

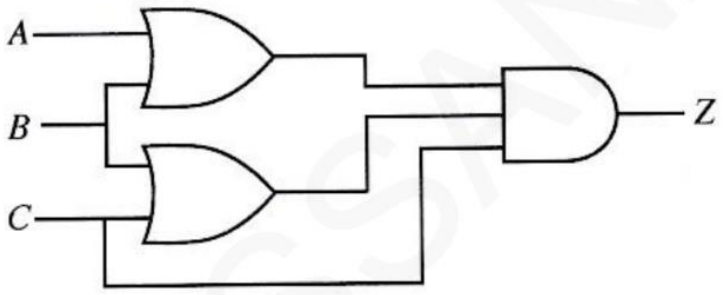
(a) Give the truth table for the 3-input majority function.

(b) Using Karnaugh maps, derive a simplified Boolean expression for the output Z in the 3-input majority function.

(c) Construct a logic circuit for the 3-input majority function using NAND gates only.

2020 AL (Part II) (Part B)

5. Consider the logic circuit shown in the figure, in which A, B and C are the inputs and Z is the output.



(a) Give the complete truth table for the given circuit.

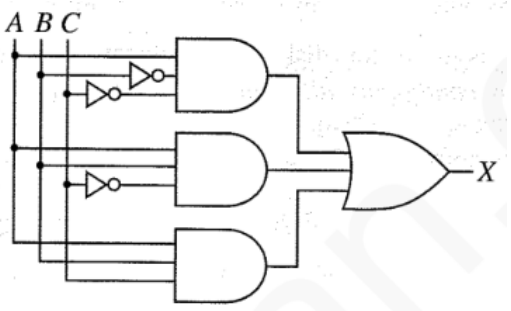
(b) Using a Karnaugh map, derive a simplified sum-of-products (SOP) expression for the output Z.

(c) Using a Karnaugh map, derive a simplified product-of-sums (POS) expression for the output Z.

(d) Of the two expressions (SOP and POS) you obtained in (b) and (c) above, which one is better to implement a more simplified logic circuit than the given logic circuit above? Explain your answer.

2021 AL (Part II) (Part B)

5. Consider the logic circuit shown in the figure, in which A, B and C are the inputs and X is the output.



(a) Show the complete truth table for the given circuit.

(b) Complete the Karnaugh map, according to the following format.

C \ AB	00	01	11	10
0				
1				

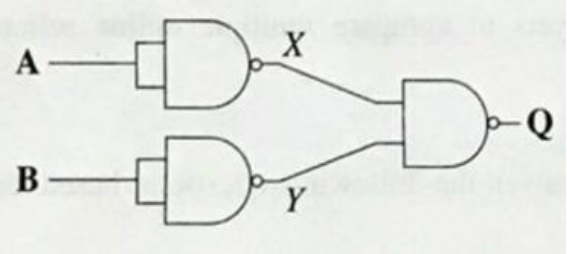
(c) Using the Karnaugh map, derive an optimal (most simplified) sum-of-products (SOP) expression for the output X. Show the loops clearly on the Karnaugh map.

(d) Using the Karnaugh map, derive an optimal (most simplified) product-of-sums (POS) expression for the output X. Show the loops clearly on the Karnaugh map.

(e) Of the optimal SOP and POS expressions you obtained in (c) and (d) above, which is better (or more suitable) to implement a simplified logic circuit? Explain your answer.

2022 AL (Part II)(Part A Q4)

(d) (i) Complete the truth table for the given logic circuit.



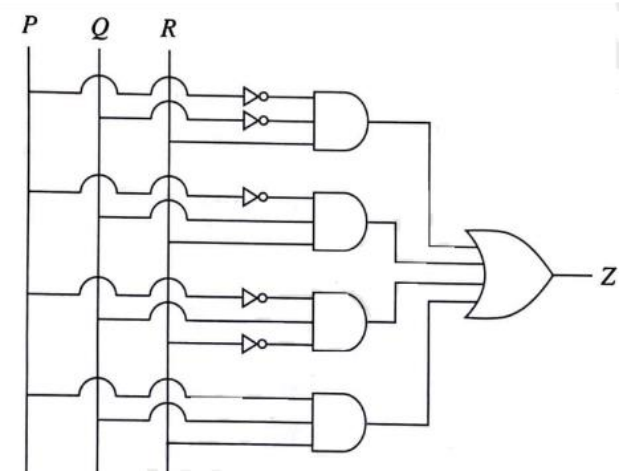
A	B	X	Y	Q
0	0			
0	1			
1	0			
1	1			

(ii) What is the basic logic gate that has the above truth table (with inputs A, B and the output Q)?

.....

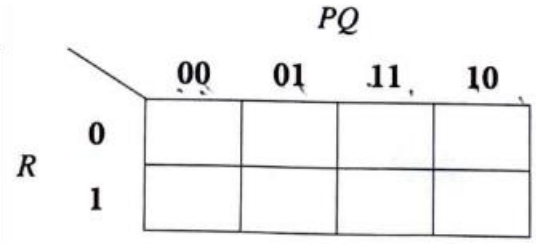
(Part B)

5. (a) Consider the logic circuit shown in the following figure in which P, Q and R are the input and Z is the output.



(i) Draw the complete truth table for the above circuit.

(ii) Complete the Karnaugh map relevant to the above circuit according to the following format.



(iii) Using the Karnaugh map, derive the most simplified sum-of-products expression for the output Z. Show the loops clearly on the Karnaugh map.

(b) (i) Using Boolean algebra, show that the Boolean expression $\bar{A}$BC + A$\bar{B}$C + AB$\bar{C}$ + ABC is equivalent to BC + AC + AB.

(ii) Draw a logic circuit for the above simplified expression in (b)(i) by only using OR and AND gates.
(iii) Draw a logic circuit for the above simplified expression in (b)(i) by only using NAND gates.

2023 AL (Part II)(Part B)

5. (a) A circuit with three inputs (A, B, C) and one output (Z) is to be designed. The output should be equal to 1 when the binary value combination of the three inputs is either 1, 3 or 6. The output should be 0 for other cases.

(i) Draw the complete truth table for the above circuit.

(ii) Complete the Karnaugh map relevant to the above circuit according to the following format:

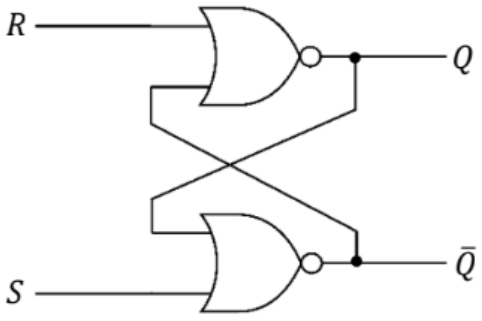
C \ AB	00	01	11	10
0				
1				

(iii) Using the Karnaugh map, derive the most simplified product-of-sums (POS) expression for the output Z. Show the loops clearly on the Karnaugh map.

(iv) Draw a logic circuit for the simplified expression derived in (iii) by only using NOR gates assuming that the complemented inputs $\bar{A}$, $\bar{B}$ and $\bar{C}$ are also available.

(b) Using Boolean Algebra show that$\bar{A}$C + $\bar{A}$B + A$\bar{B}$C + BC is equivalent to C + $\bar{A}$B.

(c) Consider the flip flop circuit shown on the right:



(i) Assume that the S input is 1 and the R input is 0. What will be the output at Q?

(ii) What will be the output at Q if the S input is now made 0?

(iii)What will be the output at Q when the R input is now made 1?