Lattice & Boolean Algebra

Unit: 3

Discrete Mathematics

B Tech (DS) 2^{nd} Year

Ms. Kajol Kathuria
Assistant Professor
CSE Dept.

Content

- Course Objective
- Course Outcome
- CO-PO Mapping
- Syllabus
- Ordered Set
- Poset
- Hasse Diagram of partially ordered set
- Lattices-Introduction
- Isomorphic ordered set
- Well ordered set
- Properties of lattice

Content

- Bounded and complemented lattice
- Distributive Lattice
- Introduction of Boolean algebra
- Axioms and Theorems of Boolean algebra
- Algebraic manipulation of Boolean expressions
- Truth table & Logic gates
- Principle of duality
- Digital circuits and Boolean algebra
- Canonical form
- Simplification of Boolean Functions
- Karnaugh maps

Content

- Video links
- Daily Quiz
- Weekly Assignment
- MCQ
- Old Question papers
- Expected Question for University Exam
- Summary
- References.

Course Objective

- The subject enhances one’s ability to develop logical thinking and ability to problem solving.
- Demonstrate the ability to write and evaluate a proof or outline the basic structure of and give examples of each proof technique described.
- Apply logical reasoning to solve a variety of problems.
- Use Mathematically correct terminology and notation.

Course Outcome

Course Outcome (CO)	At the end of course , the student will be able to	Bloom's Knowledge Level (KL)
CO1	Apply the basic principles of sets, relations & functions and mathematical induction in computer science & engineering related problems.	K3
CO2	Understand the algebraic structures and its properties to solve complex problems	K2
CO3	Describe lattices and its types and apply Boolean algebra to simplify digital circuit.	K2, K3
CO4	Infer the validity of statements and construct proofs using predicate logic formulas.	K3,K5
CO5	Design and use the non-linear data structure like tree and graphs to solve real world problems..	K3,K6

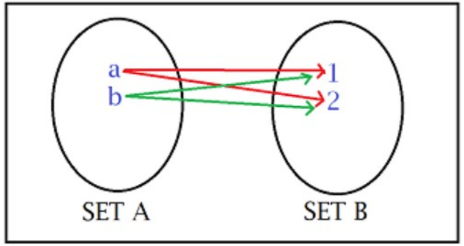
Ordered Sets & Posets

- A **Set** is a collection of elements. The elements that make up a set can be any kind of mathematical objects: numbers, symbols, points in space, lines, other geometrical shapes, variables, or even other sets.
- Example: A collection of natural numbers, as in, N = {1,2,3,4,5...}.
- A relation can also be used to define an order on a set.
- An **Ordered set** is a relational structure (S, ≤) such that the realtion ≤ is an ordering.
- An **ordered pair** is a pair of numbers (x,y) written in a particular order

Terminologies

- **Cartesian Product:** the product of two sets A and B such that every element of set A relates to every other element of set B to form ordered pairs.

  A*B = {(a,1),(a,2),(b,1),(b,2)}



- **Reflexive Relation:**

  A relation R, over a set A, is **reflexive** if every element of the set is 'related' to itself.

  Let's consider set A as follows:

  A = {p,q,r}

  Therefore, A*A = {(p,p),(p,q),(p,r),(q,p),(q,q),(q,r),(r,p),(r,q),(r,r)}

  Then the reflexive pairs in A*A would be all the diagonal elements of the matrix

  i.e. {(p,p),(q,q),(r,r)} as every element relates to itself.

A × A

	p	q	r
p	pp	pq	pr
q	qp	qq	qr
r	rp	rq	rr

Terminologies

- **Symmetric**: A relation R on a set A is called symmetric if (b, a) $\in$ R whenever (a, b) $\in$ R, for all a, b $\in$ A.

- **Anti-symmetric Relation:** : A relation R on a set A such that for all a, b $\in$ A, if (a, b) $\in$ R and (b, a) $\in$ R, then a = b is called antisymmetric.

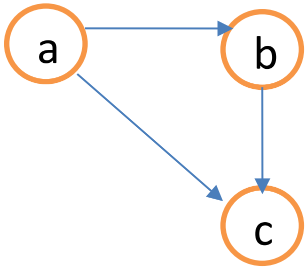
Note:

- A relation can be both symmetric and antisymmetric.
- A relation can be neither symmetric nor antisymmetric.

Terminologies

- **Transitive**: A relation R on a set A is called transitive if whenever (a, b) $\in$ R and (b, c) $\in$ R, then (a, c) $\in$ R, for all a, b, c $\in$ A.

Relation R={(1,2),(2,3),(1,3)} on set A={1,2,3} is transitive.



Asymmetric relation: Asymmetric relation is opposite of symmetric relation. A relation R on a set A is called asymmetric if no (b,a) € R when (a,b) € R.

POSET(Partially Ordered Set).

A relation R, over a set A, is a partial order relation if it is **reflexive**, **anti-symmetric** and **transitive**.

For this, we will

1.Check if it is reflexive

2.Check if it is anti-symmetric

3.Check if it is transitive

Step 1: The subset is **reflexive** as it contains the pairs, (p,p), (q,q) and (r,r).

Step 2: It is **anti-symmetric** as (p,r) and (q,r) do not have their symmetric pairs (r,p) or (r,q) in it. In addition, it also contains the pairs (p,p), (q,q), (r,r) in which the elements are equal to each other.

Step 3: The subset contains (p,p) and (p,r). Therefore according to the definition of a transitive relation, it must contain (p,r), which, you can see, is already present in it. Hence it is **transitive.**

Since all the three conditions are satisfied, we could now call the subset as a **partially ordered set**.

Hasse Diagram

A **POSET** can be represented in the form of a simple diagram called the Hasse diagram.

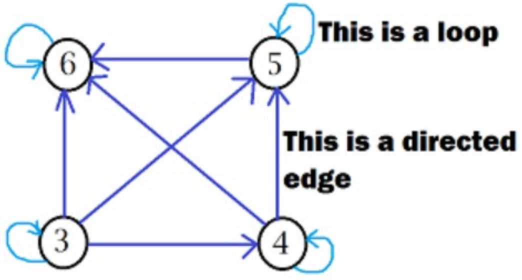
Let us consider a set B = {3,4,5,6}

B*B = {(3,3), (3,4), (3,5), (3,6), (4,3), (4,4), (4,5), (4,6), (5,3), (5,4), (5,5), (5,6), (6,3), (6,4), (6,5), (6,6)}

From this Cartesian product, let us create another set R having the following pairs of elements:

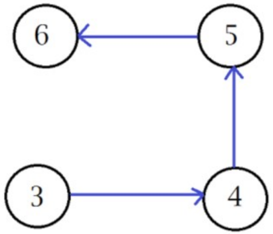
R = {(3,3), (3,4), (3,5), (3,6), (4,4)

Now represent each of the pair in the graph



To further simplify the graph omit the self-loops and transitive edges in order to avoid repetition.

The simplified graph would then look like this:



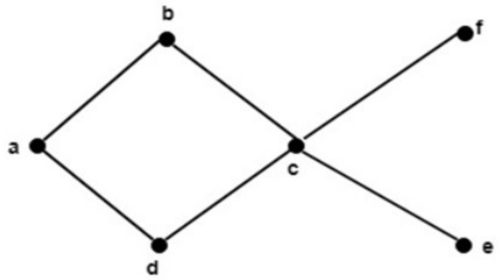
POSET(Partially Ordered Set).

Elements of POSET:

1.Maximal Element: An element a ∈ A is called a maximal element of A if there is no element in c in A such that a ≤ c.

2.Minimal Element: An element b ∈ A is called a minimal element of A if there is no element in c in A such that c ≤ b.

Example: Determine all the maximal and minimal elements of the poset whose Hasse diagram is shown in fig:



The maximal elements are b and f.

The minimal elements are d and e.

POSET(Partially Ordered Set).

Comparable Elements: Consider an ordered set A. Two elements a and b of set A are called comparable if $a \leq b$ or $b \leq a$

Non-Comparable Elements:

Consider an ordered set A. Two elements a and b of set A are called non-comparable if neither $a \leq b$ nor $b \leq a$.

Example: Consider A = {1, 2, 3, 5, 6, 10, 15, 30} is ordered by divisibility. Determine all the comparable and non-comparable pairs of elements of A.

Solution: The comparable pairs of elements of A are:

{1, 2}, {1, 3}, {1, 5}, {1, 6}, {1, 10}, {1, 15}, {1, 30}

{2, 6}, {2, 10}, {2, 30}

{3, 6}, {3, 15}, {3, 30}

{5, 10}, {5, 15}, {5, 30}

{6, 30}, {10, 30}, {15, 30}

The non-comparable pair of elements of A are:

{2, 3}, {2, 5}, {2, 15}

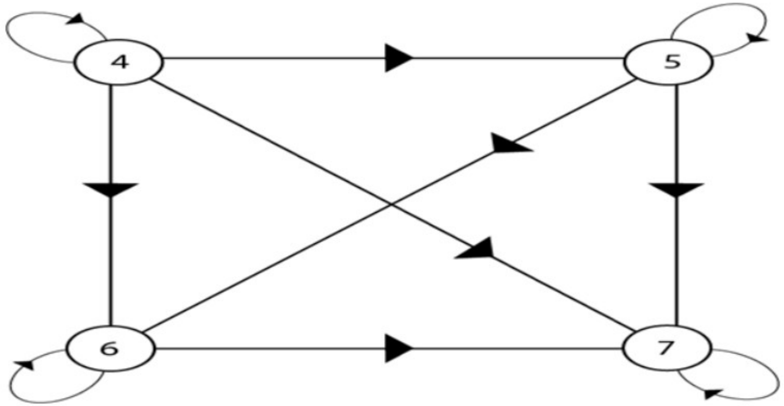
{3, 5}, {3, 10}, {5, 6}, {6, 10}, {6, 15}, {10, 15}

HASSE Diagram

Example: Consider the set A = {4, 5, 6, 7}. Let R be the relation ≤ on A. Draw the directed graph and the Hasse diagram of R.

Solution: The relation ≤ on the set A is given by

R = {{4, 5}, {4, 6}, {4, 7}, {5, 6}, {5, 7}, {6, 7}, {4, 4}, {5, 5}, {6, 6}, {7, 7}}



The directed graph of the relation R

HASSE Diagram

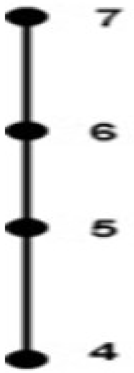
To draw the Hasse diagram of partial order, apply the following points:

1.Delete all edges implied by reflexive property i.e.
(4, 4), (5, 5), (6, 6), (7, 7)

2.Delete all edges implied by transitive property i.e.
(4, 7), (5, 7), (4, 6)

3.Replace the circles representing the vertices by dots.

4.Omit the arrows.



The Hasse diagram

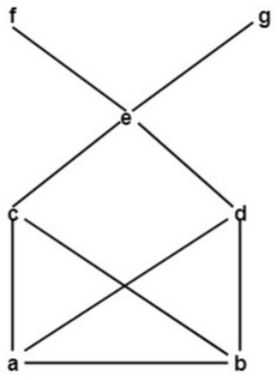
•**Upper Bound:** Consider B be a subset of a partially ordered set A. An element x $\in$ A is called an upper bound of B if $y \leq x$ for every y $\in$ B.

•**Lower Bound:** Consider B be a subset of a partially ordered set A. An element z $\in$ A is called a lower bound of B if $z \leq x$ for every x $\in$ B.

HASSE Diagram

- **Upper Bound:** Consider B be a subset of a partially ordered set A. An element x ∈ A is called an upper bound of B if y ≤ x for every y ∈ B.
- **Lower Bound:** Consider B be a subset of a partially ordered set A. An element z ∈ A is called a lower bound of B if z ≤ x for every x ∈ B.

Example: Consider the poset A = {a, b, c, d, e, f, g} be ordered shown in fig. Also let B = {c, d, e}. Determine the upper and lower bound of B.

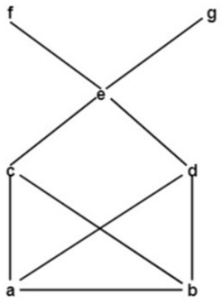


Solution:
The upper bound of B is e, f, and g because every element of B is '≤' e, f, and g.
The lower bounds of B are a and b because a and b are '≤' every elements of B.

HASSE Diagram

- **Least Upper Bound**(SUPREMUM) **:** Let A be a subset of a partially ordered set S. An element M in S is called an upper bound of A if M succeeds every element of A, i.e. if, for every x in A, we have x <=M
  If an upper bound of A precedes every other upper bound of A, then it is called the supremum of A and is denoted by Sup (A)
- **Greatest Lower Bound** (INFIMUM) **:** An element m in a poset S is called a lower bound of a subset A of S if m precedes every element of A, i.e. if, for every y in A, we have m <=y
  If a lower bound of A succeeds every other lower bound of A, then it is called the infimum of A and is denoted by Inf (A)

Example: Determine the least upper bound and greatest lower bound of B = {a, b, c} if they exist, of the poset whose Hasse diagram is



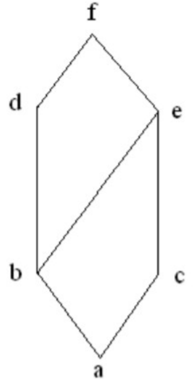
Solution: The least upper bound is c.
The greatest lower bound is k.

Lattice

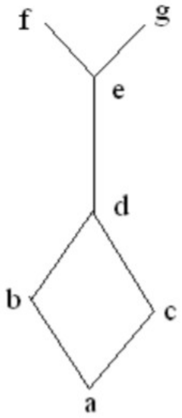
- A lattice is a poset in (L,≤) in which every subset {a,b} consisiting of two elements has a least upper bound and a greatest lower bound.

  LUB({a,b}) is denotead by a v b and is called the join of a and b.

  GLB({a,b}) is denoted by a Λ b and is called the meet of a and b.



(a)



a) Is a lattice

b) Is not a lattice because
f V g does not exist

Topic Objective: Boolean Algebra (CO1)

After learning Boolean algebra students will be able to

- Perform the three basic logic operations.
- Describe the operation of and construct the truth tables for the AND, NAND, OR, and NOR gates, and the NOT (INVERTER) circuit.
- Implement logic circuits using basic AND, OR, expression for the logic gates and combinations of logic gates. and NOT gates.
- Appreciate the potential of Boolean algebra to simplify complex logic expressions.
- Use De-Morgan’s theorems to simplify logic expressions.

Boolean Algebra(CO1)

- An algebra which is use for simplifying the presentation & manipulation of propositional logic.
- It is also known as Switching Algebra.
- Invented by George Boole in 1854.
- Boolean algebra is mainly use for simplify and manipulate electronic logic circuits in computers.

Basic concepts of Boolean algebra (CO1)

- It is denoted by (B, +, ., ‘, 0, 1), where
  - B = non-empty set
  - + is the binary operation denoting logical OR
  - . is the binary operation denoting logical AND
  - ‘ is the unary operation denoting logical NOT
  - 0 and 1 are distinct elements known as identity elements of + and . operation
- It deals with the binary number system {0,1}.
- Truth Table -It is a tabular representation of all the combination of values for input and their corresponding outputs.

AND Operation(CO1)

- AND operation is used for **logical multiplication**.
- The **dot** symbol (".")used for representing AND operation.
- AND operator has an output 1 if all the inputs are 1 otherwise the output is 0.

Input	Input	Output
A	B	C =A.B
0	0	0
0	1	0
1	0	0
1	1	1

OR Operation(CO1)

- OR operation is use for **logical addition**.
- The symbol ("+") used for represent OR operation.
- OR operator has an output 0 if all of the inputs are 0 otherwise the output is 1.

Input	Input	Output
A	B	C =A+B
0	0	0
0	1	1
1	0	1
1	1	1

NOT operation (CO1)

- NOT gate performs the logical complementation operation.
- It is an electronic circuit that generates the reverse of the input signal as output signal.

Input	Output
A	Y =A’
0	1
1	0

Postulates/Axioms of Boolean Algebra (CO1)

If a, b, c $\in$ B, then

1. Closure Law
   i. $a + b \in B$ ii. $a.b \in B$
2. Commutative Law
   i. $a + b = b + a$ ii. $a.b = b.a$
3. Distributive Law
   i. $a.(b + c) = a.b + a.c$ ii. $a + (b.c) = (a + b).(a + c)$
4. Identity Law
   i. $a + 0 = a$ ii. $a.1 = a$
5. Complement Law
   i. $a + a' = 1$ ii. $a.a' = 0$

Postulates/Axioms of Boolean Algebra (CO1)

6. Idempotent Law
   i. $a + a = a$ ii. $a.a = a$
7. Boundness Law
   i. $a + 1 = 1$ ii. $a.0 = 0$
8. Absorption Law
   i. $a + a.b = a$ ii. $a.(a + b) = a$
9. Associative Law
   i. $a + (b + c) = (a + b) + c$ ii. $a.(b.c) = (a.b).c$

Minimization of Boolean Algebra using laws (CO1)

1. $(A + B)(A + C) = A + BC$

This rule can be proved as follows:

$(A + B)(A + C) = AA + AC + AB + BC$ (Distributive law)

$= A + AC + AB + BC \quad (AA = A)$

$= A(1 + C) + AB + BC \quad (1 + C = 1)$

$= A.\ 1 + AB + BC$

$= A(1 + B) + BC \quad (1 + B = 1)$

$= A.\ 1 + BC \quad (A\ .\ 1 = A)$

$= A + BC$

Minimization of Boolean Algebra using laws (CO1)

2. ▪ $AB + \overline{A}C + BC = AB + \overline{A}C$ (Consensus Theorem)

Proof Steps	**Justification**
$AB + \overline{A}C + BC$	
$= AB + \overline{A}C + 1 \cdot BC$	Identity element
$= AB + \overline{A}C + (A + \overline{A}) \cdot BC$	Complement
$= AB + \overline{A}C + ABC + \overline{A}BC$	Distributive
$= AB + ABC + \overline{A}C + \overline{A}CB$	Commutative
$= AB \cdot 1 + ABC + \overline{A}C \cdot 1 + \overline{A}CB$	Identity element
$= AB\,(1+C) + \overline{A}C\,(1 + B)$	Distributive
$= AB \,.\, 1 + \overline{A}C \,.\, 1$	$1+X = 1$
$= AB + \overline{A}C$	Identity element

Minimization of Boolean Algebra using laws (CO1)

1. $\bar{A}\bar{B} + AB + \bar{A}B$

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

$\bar{A}\bar{B} + B\cdot 1$ (Complement law)

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

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

$(B + \bar{A})(B + \bar{B})$ (Distributive law)

$(B + \bar{A})\cdot 1$ (Complement law)

$B + \bar{A}$

$\bar{A} + B$

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

$A + B\bar{B}$ (Complement law)

$A + 0$ (Identity Law)

A

3. $\bar{A}B + A\bar{B} + AB + \bar{A}\bar{B}$

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

$B\cdot 1 + \bar{B}\cdot 1$ (Identity law)

$B + \bar{B}$ (Complement law)

1

Other laws of Boolean Algebra (CO1)

If a, b, c $\in$ B, then

1. Uniqueness of complement

   If a + x = 1 and a.x = 0, then x = a’

2. Involution Law

   (a’)’ = a

3. De-Morgan’s Law

   i. (a + b)’ = a’.b’ ii.(a.b) = a’ + b’

Simplification: Example

$$
\begin{aligned}
\overline{(A\bar{B} + \bar{A}B)}(A + B) &= \overline{A\bar{B}}\,\overline{\bar{A}B}(A+B) \\
&= (\bar{A}+B)(A+\bar{B})(A+B) \\
&= (\bar{A}+B)(AA+AB + \bar{B}A + \bar{B}B) \\
&= (\bar{A}+B)(A + AB + A\bar{B} + \bar{B}B) \\
&= (\bar{A}+B)(A(1 + B + \bar{B}) + \bar{B}B) \\
&= (\bar{A}+B)(A(1) + \bar{B}B) \\
&= (\bar{A}+B)A \\
&= A\bar{A} + AB \\
&= AB
\end{aligned}
$$

Example of De-Morgan Law(CO1)

- Example:

  Apply DeMorgan's theorem to the given expressions:

$$F_2 = \overline{(\overline{X} + Z)(\overline{XY})}$$

$$F_2 = \overline{(\overline{X} + Z)} + (\overline{\overline{XY}})$$

$$F_2 = \overline{(\overline{X} + Z)} + (XY)$$

$$F_2 = (\overline{\overline{X}}\,\overline{Z}) + (XY)$$

$$F_2 = (X\,\overline{Z}) + (XY)$$

$$F_2 = X\,\overline{Z} + X\,Y$$

Boolean Function(CO1)

- Boolean function is an expression formed with binary variable.
- It is the combination of Boolean operators such as AND, OR & NOT operators.

$$X.(X'+Y)=X.Y$$

Input		Output			
X	Y	$\overline{X}$	$\overline{X}+Y$	$X.(\overline{X}+Y)$	$=X.Y$
0	0	1	1	0	0
0	1	1	1	0	0
1	0	0	0	0	0
1	1	0	1	1	1

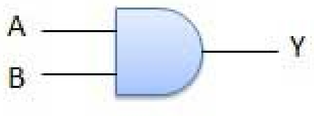
Truth table for Boolean function.

Logic Gate (CO1)

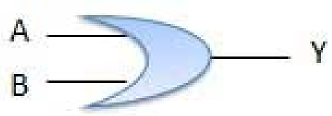
- Computer carries out all operations by the combination of signals that pass through standard blocks of built-in circuits.
- Logic gate is an elementary building block of digital electronic circuits that operates one or more input signals to produce standard output.
- The common use of logic gate elements is to act as switch.
- In computer logic gates are use to implement Boolean functions.

Logic Gates Symbol (CO1)

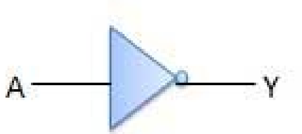
- AND Gate



- OR Gate



- NOT Gate



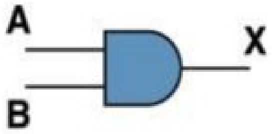
AND Gate (CO1)

- AND gate is use for logical multiplication input signals.
- If the input values for an AND gate are 1 then the output is 1 otherwise the output is 0.

Boolean Expression

$$X = A \cdot B$$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

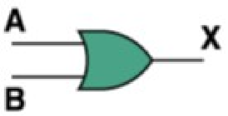
OR Gate (CO1)

- OR gate performs the logical addition operation.
- It is an electronic circuit that generates an output signal 1 if any of the input signal is 1.

Boolean Expression

$X = A + B$

Logic Diagram Symbol



Truth Table

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

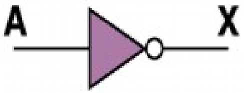
NOT Gate (CO1)

- NOT gate performs the logical complementation operation.
- It is an electronic circuit that generates the reverse of the input signal as output signal.

Boolean Expression

$X = A'$

Logic Diagram Symbol

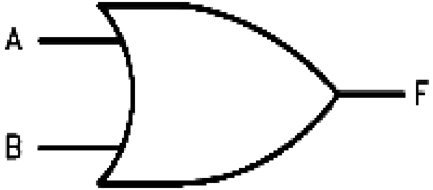


Truth Table

A	X
0	1
1	0

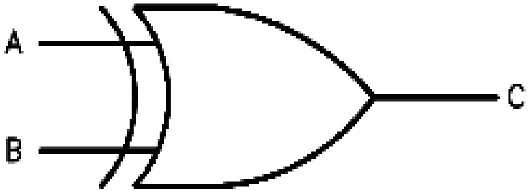
Exclusive OR gate(CO1)

OR Gate



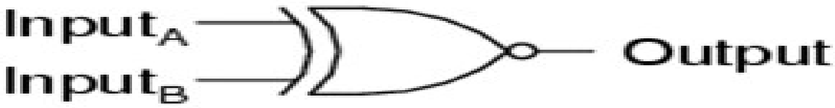
INPUT		OUTPUT
A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

Exclusive OR Gate



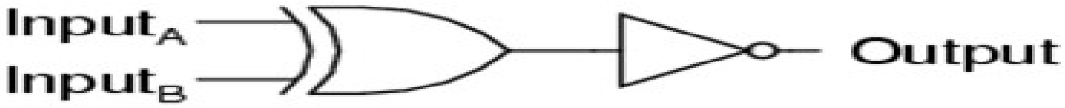
INPUT		OUTPUT
A	B	C
0	0	0
0	1	1
1	0	1
1	1	0

Exclusive-NOR gate



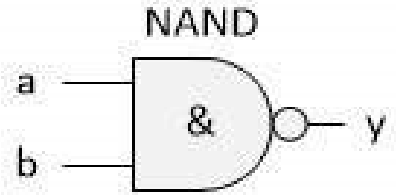
A	B	Output
0	0	1
0	1	0
1	0	0
1	1	1

Equivalent gate circuit

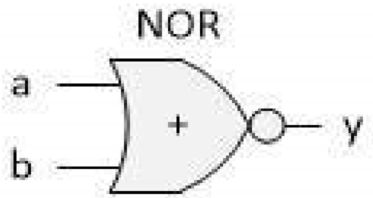


Universal Gates(CO1)

NAND & NOR gates

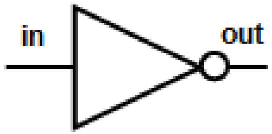


a	b	y
0	0	1
0	1	1
1	0	1
1	1	0



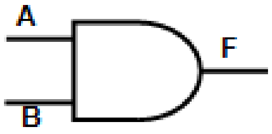
a	b	y
0	0	1
0	1	0
1	0	0
1	1	0

Summary of Logic Gates(CO1)



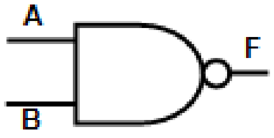
NOT

Input	Output
I	F
0	1
1	0



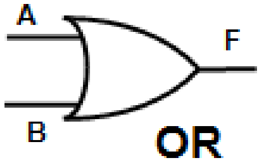
AND

Inputs		Output
A	B	F
0	0	0
1	0	0
0	1	0
1	1	1



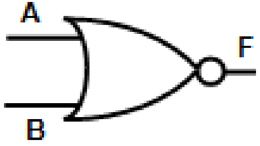
NAND

Inputs		Output
A	B	F
0	0	1
1	0	1
0	1	1
1	1	0



OR

Inputs		Output
A	B	F
0	0	0
1	0	1
0	1	1
1	1	1



NOR

Inputs		Output
A	B	F
0	0	1
1	0	0
0	1	0
1	1	0

Principle of Duality(CO1)

This principle states that any algebraic equality derived from these axioms will still be valid whenever the OR and AND operators, and identity elements 0 and 1, have been interchanged. i.e. changing every OR into AND and vice versa, and every 0 into 1 and vice versa.

Example:

1. $X + 0 = X$ 2. $X \bullet 1 = X$ (dual of 1)

3. $X + 1 = 1$ 4. $X \bullet 0 = 0$ (dual of 3)

5. $X + X = X$ 6. $X \bullet X = X$ (dual of 5)

7. $X + X' = 1$ 8. $X \bullet X' = 0$ (dual of 8)

Canonical/Standard forms(CO1)

We need to consider formal techniques for the simplification of Boolean functions.

- Identical functions will have exactly the same canonical form.
- Minterms and Maxterms.
- Sum-of-Minterms and Product-of- Maxterms.
- Product and Sum terms.
- Sum-of-Products (SOP) and Product-of-Sums (POS)

Terms of Canonical/standard forms(CO1)

- Literal: A variable or its complement.
- Product term: literals connected by •
- Sum term: literals connected by +
- Minterm: a product term in which all the variables appear exactly once, either complemented or un-complemented.
- Maxterm: a sum term in which all the variables appear exactly once, either complemented or un-complemented.

Minterms(CO1)

- Represents exactly one combination in the truth table.
- Denoted by mj, where j is the decimal equivalent of the minterm ’s corresponding binary combination (bj).
- A variable in mj is complemented if its value in bj is 0, otherwise is u-ncomplemented.
- Example: Assume 3 variables (A,B,C), and j=3. Then, bj = 011 and its corresponding minterm is denoted by mj = A’BC

Maxterms(CO1)

- Represents exactly one combination in the truth table.
- Denoted by Mj, where j is the decimal equivalent of the maxterm's corresponding binary combination (bj).
- A variable in Mj is complemented if its value in bj is 1, otherwise is uncomplemented.
- Example: Assume 3 variables (A,B,C), and j=3. Then, bj = 011 and its corresponding maxterm is denoted by Mj = A+B'+C'

Simplification of Boolean Functions (CO1)

- An implementation of a Boolean Function requires the use of logic gates.
- A smaller number of gates, with each gate (other then Inverter) having less number of inputs, may reduce the cost of the implementation.
- There are 2 methods for simplification of Boolean functions: algebraic method and graphical method (K-maps).

Topic objective: K-maps(CO1)

- Simplification of Boolean functions and expressions using K-map.
- K-map implementation for circuit designing.

Prerequisite and recap of Topic: K-maps(CO1)

Prerequisite

- Boolean functions.
- Binary logic.
- SOP form
- POS form

Recap

- K-map is a graphical representation to solve the Boolean functions to reduce redundancy. It is easy to find the minimized expression of lengthy Boolean functions using K-maps.

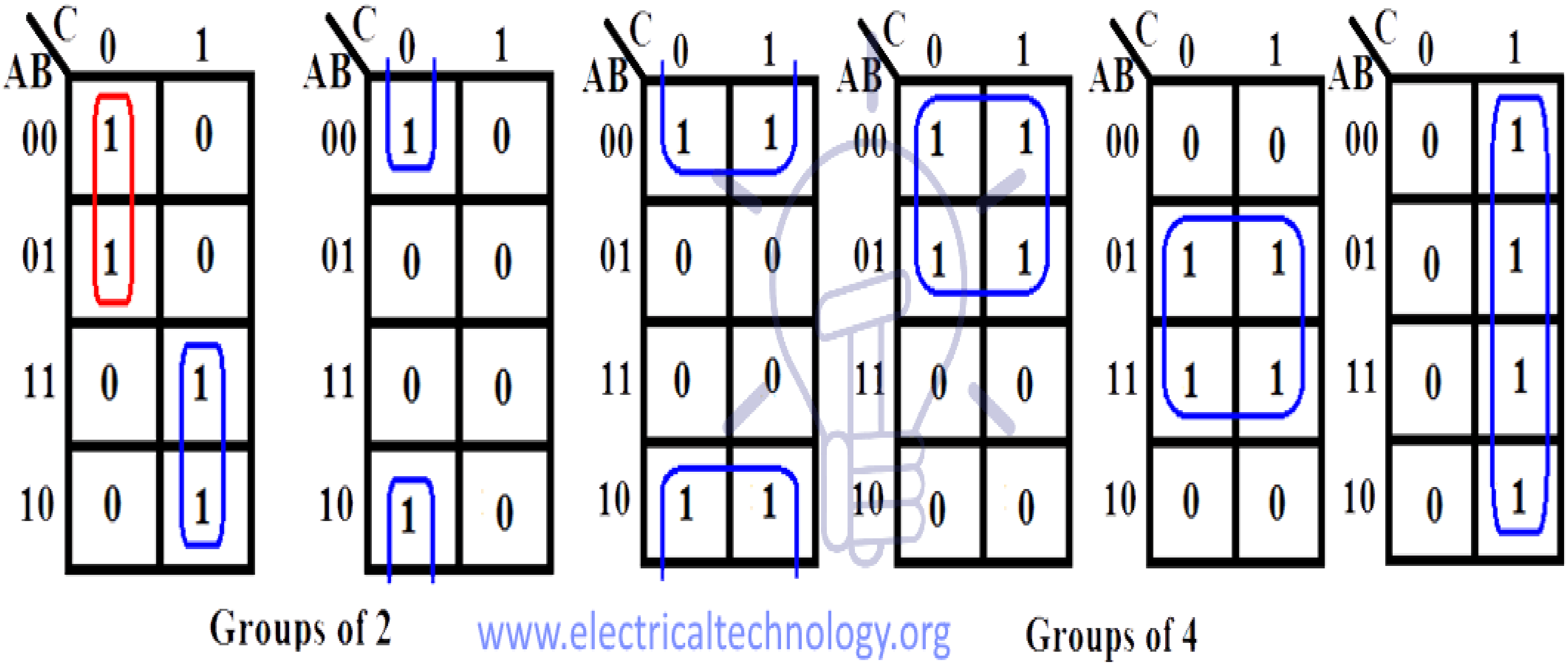
K-Map (CO1)

- The **Karnaugh map** (**KM** or **K-map**) is a method of simplifying Boolean algebra expressions. Maurice Karnaugh introduced it in 1953.
- The Karnaugh map reduces the need for extensive calculations by taking advantage of humans' pattern-recognition capability. It also permits the rapid identification and elimination of potential race conditions.
- The required Boolean results are transferred from a truth table onto a two-dimensional grid where, in Karnaugh maps, the cells are ordered in Gray code, and each cell position represents one combination of input conditions.
- Cells are also known as minterms, while each cell value represents the corresponding output value of the boolean function.
- Optimal groups of 1s or 0s are identified, which represent the terms of a canonical form of the logic in the original truth table. These terms can be used to write a minimal Boolean expression representing the required logic.

Rules for K-Map Simplification(CO1)

- Groups may not contain zero.
- We can group 1,2,4,8 ,…cells.
- Each group should be a large as possible.
- Groups may overlap.
- Opposite grouping and corner grouping are allowed.
- Grouping can not possible in diagonally manner.
- Cells contains 1 must be grouped.

Grouping :(CO1)



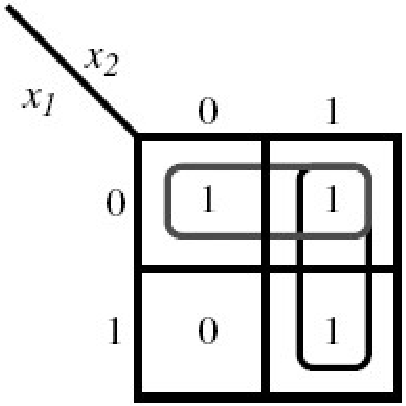
K-Maps Numbering (CO1)

- Cell numbers are written in the cells.
- 2-variable K-map

A \ B	0	1
0	0	1
1	2	3

Examples of 2-variable K-Maps (CO1)

x_1	x_2	f
0	0	1
0	1	1
1	0	0
1	1	1



$$f = \overline{x}_1 + x_2$$

2-Variable K-Map SOP & POS:(CO1)

A. SOP: -

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

B. POS: -

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

3-Variable K-Map Numbering:(CO1)

A \ BC	00	01	11	10
0	0	1	3	2
1	4	5	7	6

3-Variable K-Map Examples:(CO1)

$f = \sum(0,4) = \overline{B}\,\overline{C}$

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

$f = \sum(4,5) = A\,\overline{B}$

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

$f = \sum(0,1,4,5) = \overline{B}$

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

$f = \sum(0,1,2,3) = \overline{A}$

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

$f = \sum(0,4) = \overline{A}\,C$

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

$f = \sum(4,6) = A\,\overline{C}$

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

$f = \sum(0,2) = \overline{A}\,\overline{C}$

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

$f = \sum(0,2,4,6) = \overline{C}$

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

4-variable K-map Numbering (CO1)

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7	6
11	12	13	15	14
10	8	9	11	10

4-variable K-map Examples (CO1)

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

$f = \sum(0,8) = \overline{B} \bullet \overline{C} \bullet \overline{D}$

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

$f = \sum(5,13) = B \bullet \overline{C} \bullet D$

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

$f = \sum(13,15) = A \bullet B \bullet D$

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

$f = \sum(4,6) = \overline{A} \bullet B \bullet \overline{D}$

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

$f = \sum(2,3,6,7) = \overline{A} \bullet C$

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

$f = \sum(4,6,12,14) = B \bullet \overline{D}$

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

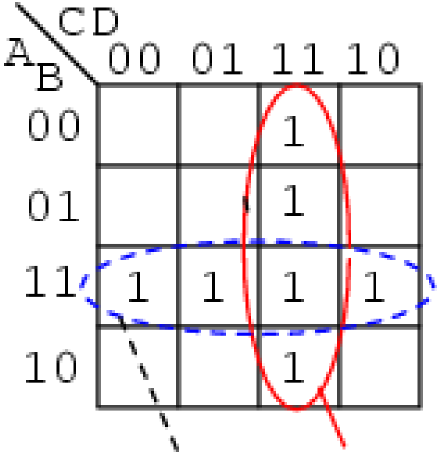
$f = \sum(2,3,10,11) = \overline{B} \bullet C$

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

$f = \sum(0,2,8,10) = \overline{B} \bullet \overline{D}$

4-variable K-map simplification(CO1)

$$Out = \bar{A}\bar{B}CD + \bar{A}BCD + ABCD + A\bar{B}CD + AB\bar{C}\bar{D} + AB\bar{C}D + ABC\bar{D}$$

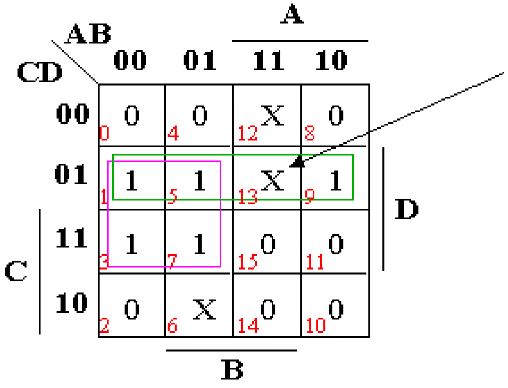


$$Out = AB + CD$$

Don’t cares in K map(CO1)

A **don't-care** term (abbreviated DC, historically also known as redundancies, irrelevancies, optional entries, invalid combinations, vacuous combinations, forbidden combinations, or unused states) for a function is an input-sequence (a series of bits) for which the function output does not matter.

- Don't cares can be treated as 0s or 1s
  - Example: minimize $F(A,B,C,D) = \Sigma m(1,3,5,7,9) + d(6,12,13)$
    - With and without using don't cares



Use X == "1"
⇒ allows a 2-cube rather than a 1-cube

F = A'D + B'C'D   *without using* don't cares
F = A'D + C'D   *using* don't cares

Don’t cares in K map(CO1)

Example: Simplify the Boolean function

$$F(w,x,y,z)=\Sigma(1,3,7,11,15),\quad dc(w,x,y,z)=\Sigma(0,2,5)$$

wx \ yz	00	01	11	10
00	X	1	1	X
01	0	X	1	0
11	0	0	1	0
10	0	0	1	0

(a) $F = yz + w'x'$

wx \ yz	00	01	11	10
00	X	1	1	X
01	0	X	1	0
11	0	0	1	0
10	0	0	1	0

(a) $F = yz + w'z$

Practice Problems on K-map (CO1)

Simplify using SOP

1) $Z= (A,B,C)=\sum(1,3,6,7)$ Ans: $(A'C+AB)$
2) $F(P,Q,R,S)=\sum(0,2,5,7,8,10,13,15)$ Ans: $(QS+Q'S')$

Simplify using POS

1) $F(A,B,C)=\pi(0,3,6,7)$

Ans: $(A' + B' + C) (B' + C') (A + B + C)$

2) $F(A,B,C,D)=\pi(3,5,7,8,10,11,12,13)$

Ans: $(C+D'+B').(C'+D'+A).(A'+C+D).(A'+B+C')$

Faculty Video Links, Youtube & NPTEL Video Links and Online Courses Details (CO1)

Video Links

- https://www.youtube.com/watch?v=C-C2mtqMVfo
- https://www.youtube.com/watch?v=35YXM8PcloY
- https://www.youtube.com/watch?v=bXqqfizJIyA
- https://www.youtube.com/watch?v=OpmVzizWCqI

NEPTEL Video Link:

- https://nptel.ac.in/courses/106/106/106106183/#

Daily Quiz (CO1)

1. Algebra of logic is termed as _______________
   a) Numerical logic
   b) Boolean algebra
   c) Arithmetic logic
   d) Boolean number

2. Boolean algebra can be used _____________
   a) For designing of the digital computers
   b) In building logic symbols
   c) Circuit theory
   d) Building algebraic functions

3. What is the definition of Boolean functions?
   a) An arithmetic function with k degrees such that $f:Y \to Y^k$
   b) A special mathematical function with n degrees such that $f:Y^n \to Y$
   c) An algebraic function with n degrees such that $f:X^n \to X$
   d) A polynomial function with k degrees such that $f:X^2 \to X^n$

Daily Quiz (CO1)

4. F(X,Y,Z,M) = X\`Y\`Z\`M\`. The degree of the function is ________
a) 2
b) 5
c) 4
d) 1

5. K-map is used for _______
a) logic minimization
b) expression maximization
c) summing of parity bits
d) logic gate creation

6. A Poset in which every pair of elements has both a least upper bound and a greatest lower bound is termed as _______
a) sublattice
b) **lattice**
c) trail
d) walk

Daily Quiz (CO1)

7. Which of the following is a Simplification law?

a) M.(~M+N) = M.N

b) M+(N.O) = (M+N)(M+O)

c) ~(M+N) = ~M.~N

d) M.(N.O) = (M.N).O

8. What are the canonical forms of Boolean Expressions?

a) OR and XOR

b) NOR and XNOR

c) MAX and MIN

d) SOM and POM

9. Which of the following is/are the universal logic gates?

a) OR and NOR

b) AND

c) **NAND and NOR**

d) NOT

Daily Quiz (CO1)

11. The ___________ of all the variables in direct or complemented from is a maxterm.
a) addition
b) product
c) moduler
d) subtraction

12. _________ is used to implement the Boolean functions.
a) Logical notations
b) Arithmetic logics
c) Logic gates
d) Expressions

13. Inversion of single bit input to a single bit output using _________
a) NOT gate
b) NOR gate
c) AND gate
d) NAND gate

Daily Quiz (CO1)

14. There are __________ numbers of Boolean functions of degree n.
a) n
b) $2^{(2*n)}$
c) n^3
d) $n^{(n*2)}$

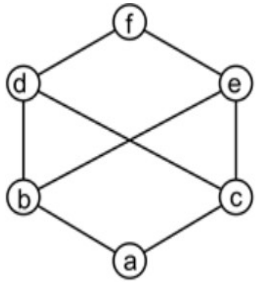
15. Minimization of function F(A,B,C) = A*B*(B+C) is __________
a) AC
b) B+C
c) B`
d) AB

16. The set for which the Boolean function is functionally complete is ___________
a) {*, %, /}
b) {., +, -}
c) {^, +, -}
d) {%, +, *}

Daily Quiz (CO1)

17. ______ and _______ are the two binary operations defined for lattices.
a) **Join, meet**
b) Addition, subtraction
c) Union, intersection
d) Multiplication, modulo division

18. The graph given below is an example of
a) **non-lattice poset**
b) semilattice
c) partial lattice
d) bounded lattice



19. In the poset (Z^+, |) (where Z^+ is the set of all positive integers and | is the divides relation) are the integers 9 and 351 comparable?
a) **comparable**
b) not comparable
c) comparable but not determined
d) determined but not comparable

Weekly Assignment (CO1)

1. Consider the Boolean function F(x1,x2,x3,x4)= x1+(x2.(x1’+x4)+x3.(x2’+x4’))
   i) Simplify F algebraically
   ii) Draw the logic circuit of F and reduction of F.
2. Draw the K-Map and simplify the following Boolean expressions:
   i) ABC+ A’BC’+ABC’+A’BC
   ii) A’B’C+A’B’C’+ABC’+ABC’+A’BC’
   iii) AB’C’+ABC’+ABC+AB’C+AB’C’
   iv) ABC’D’+AB’C’D’+AB’C’D+AB’C’D
3. Evaluate the expression: (X + Z)(X + XZ’) + XY + Y.
4. Simplify the expression: A’(A + BC) + (AC + B’C).
5. What is the simplification value of MN(M + N’) + M(N + N’)?
6. Simplify the expression XZ’ + (Y + Y’Z) + XY.
7. If an expression is given that x+x’y’z=x+y’z, find the minimal expression of the function F(x,y,z) = x+x’y’z+yz?

MCQs (CO1)

1. Algebra of logic is termed as _______________
   a) Numerical logic
   b) Boolean algebra
   c) Arithmetic logic
   d) Boolean number

2. Boolean algebra can be used _____________
   a) For designing of the digital computers
   b) In building logic symbols
   c) Circuit theory
   d) Building algebraic functions

3. What is the definition of Boolean functions?
   a) An arithmetic function with k degrees such that $f:Y \rightarrow Y^k$
   b) A special mathematical function with n degrees such that $f:Y^n \rightarrow Y$
   c) An algebraic function with n degrees such that $f:X^n \rightarrow X$
   d) A polynomial function with k degrees such that $f:X^2 \rightarrow X^n$

MCQs (CO1)

4. F(X,Y,Z,M) = X\`Y\`Z\`M\`. The degree of the function is ________

   a) 2 b) 5

   c) 4 d) 1

5. K-map is used for _______

   a) logic minimization b) expression maximization

   c) summing of parity bits d) logic gate creation

6. Simplify the expression: XY’ + X’ + Y’X’.

   a) X’ + Y

   b) XY’

   c) (XY)’

   d) Y’ + X

7. Find the simplified term Y’ (X’ + Y’) (X + X’Y)?

   a) XY’

   b) X’Y

   c) X + Y

   d) X’Y’

MCQs (CO1)

8. To display time in railway stations which digital circuit is used?
    a) seven segment decoder
    b) eight segment encoder
    c) 8:3 multiplexer
    d) 9 bit segment driver

9. When designing a circuit to emulate a truth table, both Product-of-Sums (POS) expressions and Sum-of-Products (SOP) expressions can be derived from?
    a) k-map
    b) NAND gate
    c) NOR gate
    d) X-NOR gate

10. Addition of two or more bits produces how many bits to construct a logic gate?
    a) 108
    b) 2
    c) 32
    d) 64

MCQs (CO1)

11. Who has invented K-map?

    a) Maurice Karnaugh
    b) Edward Veitch
    c) George Boole
    d) Adam Smith

12. In Gray coding, the adjacent code values differ by ________

    a) single bit
    b) 3 bits
    c) 10 bits
    d) 0 bit

13. Simplify the expression using K-maps: $F(A,B,C) = \pi(0,2,4,5,7)$.

    a) $(x+y)(y+z)(x+z)(x'+z')$
    b) $(x+z')(y+z)(x+y)$
    c) $(x+y'+z)(x+z')$
    d) $(y'+z')(x'+y)(z+y')$

Old Question Papers (CO1)

1. Draw Karnaugh map and simplify the Boolean expression

   $A'B'C'D' + A'B C' D + A' B' C D + A'B'C D' + A'B C D$

2. Simplify the following Boolean function using K-map:

   $F(x,y,z)=\Sigma(0,2,3,7)$

3. Find the values of the Boolean function represented by $F (x, y, z) = xy + z'$.
4. Simplify the given 2-variable Boolean equation by using K-map $F = X Y' + X' Y + X'Y'$.
5. Simplify the given 4-variable Boolean equation by using k-map. $F (W, X, Y, Z) = (1, 5, 12, 13)$.
6. What is De-Morgan's Law. Explain with example.
7. What are Boolean axioms?
8. Minimize the Boolean function without using K map: $A+AB$.
9. How many NAND gates are required for XOR gate?
10. Describe Boolean duality principle.

Old Question Papers (CO1)

11. Design logic circuit using AND, OR, Not gate to solve following problem:
    (i) input two bits x,y and output two bits representing x-y(1-1=00,1-0=01,0-0=00,0-1=11)
    (ii) input three bits x,y,z and output one bit which is majority of three input bits.

For more Previous year Question papers:

https://drive.google.com/drive/folders/1xmt08wjuxu71WAmO9Gxj2iDQ0lQf-so1

Expected Questions for University Exam (CO1)

1. Draw Karnaugh map and simplify the Boolean expression: - A'B'C'D'+ A'B C' D + A' B' C D + A'B'C D' + A'B C D
2. Simplify the following Boolean function using K-map:
   $F(x,y,z)=\Sigma(0,2,3,7)$
3. Simplify the given 4-variable Boolean equation by using k-map. F (W, X, Y, Z) = (1, 5, 12, 13).
4. How many NAND gates are required for XOR gate?
5. Describe Boolean duality principle.
6. Define SOP and POS.
7. Design logic circuit using AND, OR, Not gate to solve following problem:
   (i) input two bits x,y and output two bits representing x-y(1-1=00,1-0=01,0-0=00,0-1=11)
   (ii) input three bits x,y,z and output one bit which is majority of three input bits.
8. How many NAND gates are required for XOR gate?
9. What is the definition of Boolean functions?
10. Simplify the expression: A'(A + BC) + (AC + B'C).

Text Books & Reference Books

Text books:

1. Koshy, Discrete Structures, Elsevier Pub. 2008 Kenneth H. Rosen, Discrete Mathematics and Its Applications, 6/e, McGraw-Hill, 2006.
2. B. Kolman, R.C. Busby, and S.C. Ross, Discrete Mathematical Structures, 5/e, Prentice Hall, 2004.
3. E.R. Scheinerman, Mathematics: A Discrete Introduction, Brooks/Cole, 2000.
4. R.P. Grimaldi, Discrete and Combinatorial Mathematics, 5/e, Addison Wesley, 2004

Reference Books:

1. Liptschutz, Seymour, " Discrete Mathematics", McGraw Hill.
2. Trembley, J.P & R. Manohar, "Discrete Mathematical Structure with Application to Computer Science", McGraw Hill.
3. Narsingh Deo, "Graph Theory With application to Engineering and Computer Science.", PHI.
4. Krishnamurthy, V., "Combinatorics Theory & Application", East-West Press Pvt. Ltd., New Delhi

Summary (CO1)

- Boolean algebra simplifies logic circuits to increase work efficiency of digital device.

- Logic circuits can be built for any binary electric or electronic devices including switches, relays, electron tubes and transistors.

- The subject enhances one's ability to develop logical thinking and ability to problem solving.

References

- B. Kolman, R.C. Busby, and S.C. Ross, Discrete Mathematical Structures, 5/e, Prentice Hall, 2004.
- Liptschutz, Seymour, “ Discrete Mathematics”, McGraw Hill.
- Trembley, J.P & R. Manohar, “Discrete Mathematical Structure with Application to Computer Science”, McGraw Hill
- Koshy, Discrete Structures, Elsevier Pub. 2008 Kenneth H. Rosen, Discrete Mathematics and Its Applications, 6/e, McGraw-Hill, 2006.

Thank You