

# Digital Electronics ES-207A

**Rajeev Kumar**

**AP ECE Department (PIET)**

**Research Domain: Soft Computing, Li-Fi  
AI, ML & DL**

# Course Outcome

CO1	To introduce basic postulates of Boolean algebra and shows the correlation between Boolean expressions
CO2	To introduce the methods for simplifying Boolean expressions
CO3	To outline the formal procedures for the analysis and design of combinational circuits and sequential circuits
CO4	To introduce the concept of memories and programmable logic devices.

# Syllabus

## **UNIT 1 MINIMIZATION TECHNIQUES AND LOGIC GATES**

- Binary Digits, Logic Levels, and Digital Waveforms, Logic Systems-Positive and negative, Logic Operations, Logical Operators, Logic Gates-AND, OR, NOT, NAND, NOR, Exclusive-OR and Exclusive-NOR, Active high and Active low concepts, Universal Gates and realization of other gates using universal gates, Gate Performance Characteristics and Parameters. Boolean Algebra: Rules and laws of Boolean algebra, Demorgan's Theorems, Boolean Expressions and Truth Tables, Standard SOP and POS forms; Minterm and Maxterms, Canonical representation of Boolean expressions, Duality Theorem, Simplification of Boolean Expressions, Minimization Techniques for Boolean Expressions using Karnaugh Map and Quine McCluskey Tabular method. Introduction of TTL and CMOS Logic and their characteristics, Tristate gates.

## **UNIT 2 COMBINATIONAL CIRCUITS**

- Introduction to combinational Circuits, Adders-Half-Adder and Full-Adder, Subtractors- Half and Full Subtractor; Parallel adder and Subtractor; Look-Ahead Carry Adders. BCD adder, BCD subtractor, Parity Checker/Generator, Multiplexer, Demultiplexer, Encoder, Priority Encoder; Decoder, BCD to Seven segment Display Decoder/Driver, LCD Display, and Comparators.

# Cont...

## UNIT 3 SEQUENTIAL CIRCUITS

- **Introduction to Sequential Circuits**, Flip-Flops: Types of Flip Flops -RS, T, D, JK; Edge triggering, Level Triggering; Flip Flop conversions; Master-Slave JK.
- Introduction to shift registers, Basic Shift Register Operations, types of shift registers, Bidirectional Shift Registers, Shift Register Counters. Introduction to counters, Types of Counters-Asynchronous and synchronous counters, Up/Down Synchronous Counters, Modulo-n Counter, State table, excitation table concepts, Design of asynchronous and synchronous counters, Ring Counter, Applications of counters.

## UNIT 4 CONVERTER and MEMORY DEVICES

- **Digital to Analog Converter**, Weighted Register: R-2R Ladder Network: **Analog to Digital Conversion**, Successive Approximation Type, Dual Slope Type.
- **Classification of memories** - ROM: ROM organization, PROM, EPROM, EEPROM, EAPROM, RAM: - RAM organization - Write operation, Read operation, Memory cycle, Timing wave forms, memory expansion, Static RAM Cell, MOSFET RAM cell structure, Dynamic RAM cell structure, Programmable Logic Devices - Programmable Logic Array (PLA), Programmable Array Logic (PAL), Implementation of PLA, PAL using ROM.
- .

# Books

## Suggested Books

- R. P. Jain , “Modem Digital Electronics (Edition III)” ; TMH
- Anand Kumar , “Fundamentals of digital circuits” ; PHI
- Malvino & Leach, “Digital Principles and Applications”, McGraw Hill.
- Thomas L. Floyd, “Digital Fundamentals”, Pearson Education Inc,
- 
- **Note: The Examiner will be given the question paper template and will have to set the question paper according to the template provided along with the syllabus.**

# DE Lab

**ES- 209AL**

## **LIST OF EXPERIMENTS**

- Familiarization with Digital Trainer Kit and associated equipment.
- Study of TTL gates AND, OR, NOT, NAND, NOR, EX-OR, EX-NOR.
- Design and realize a given function using K-Maps and verify its performance.
- To verify the operation of Multiplexer and De-multiplexer.
- To verify the operation of Comparator.
- To verify the truth table of S-R, J-K, T, D Flip-flops.
- To verify the operation of Bi-directional shift register.
- To design and verify the operation of 3-bit asynchronous counter.
- To design and verify the operation of asynchronous Up/down counter using J-K FFs.
- To design and verify the operation of asynchronous Decade counter.
- Study of TTL logic family characteristics.
- Study of Encoder and Decoder.
- Study of BCD to 7 segment Decoder.

# Basic Definitions

## Binary Digit

- A binary digit, or bit, is **the smallest unit of information in a computer**. It is used for storing information and has a value of true/false, or on/off. An individual bit has a value of either 0 or 1, which is generally used to store data and implement instructions in groups of bytes.

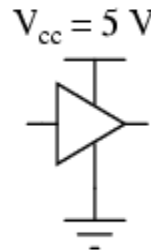
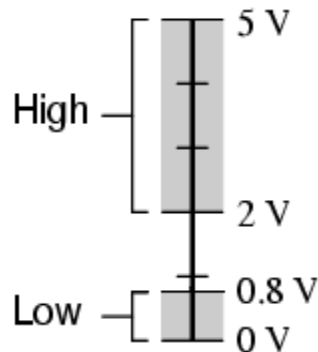
## Logic Level

- A logic level is one of several states that a digital signal can possess, expressed as a DC (direct-current) voltage with respect to electrical ground. Usually, the term refers to binary logic in which two levels, or states, can exist: logic 1 (also called the high state) and logic 0 (also called the low state).

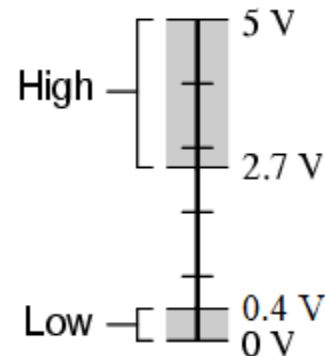
# Logic Level

- In most circuits, logic 1 is represented by approximately +5 V (positive 5 volts) relative to ground, while logic 0 is represented by approximately the same voltage as ground (0 V). This system is called positive or active-high logic.
- In some circuits the two voltage levels are reversed, so that the higher voltage represents logic 0 and the lower voltage represents logic 1. This system is known as negative or active-low logic.

*Acceptable TTL gate  
input signal levels*

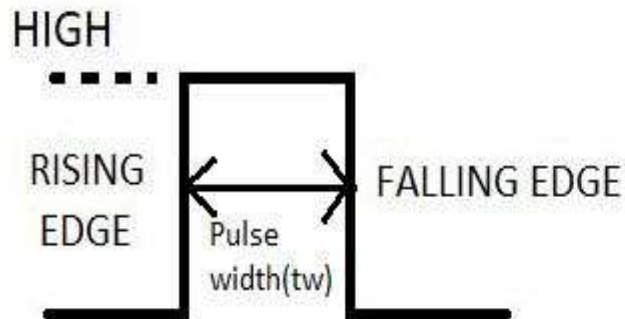


*Acceptable TTL gate  
output signal levels*

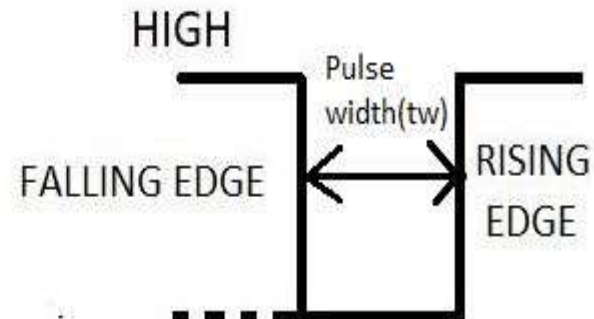


# Digital Waveform

- Digital waveforms are made of voltage levels that are HIGH AND LOW and also they contain a series of pulses.



(a) Positive Pulse



(b) Negative Pulse

# CHARACTERISTICS OF DIGITAL WAVEFORM

## Frequency

- As we represent digital working in form of waves it is going to have a frequency, which is the rate with which the pulse repeats itself

$$f=1/t$$

Where t is the period and f is the frequency.

## Duty cycle

- It is the ratio of the pulse width to the period and is expressed in percentage

$$\text{Duty cycle} = tw/t * 100\%$$

Where tw is the pulse width and t is the period.

## Amplitude

- The maximum displacement of a pulse.

## Rising and falling time

- The time at which a pulse rises and falls respectively.

# Logic Systems-Positive and negative

- There are two types of representations used in digital systems, the positive logic and the negative logic representations.
- In positive logic representation Bit 1 represents Logic high and Bit 0 represent a Logic low as shown in fig 2 a and b.
- High is represented by +5 Volts and low is represented by -5 Volts or 0 Volts.

# Positive Logic

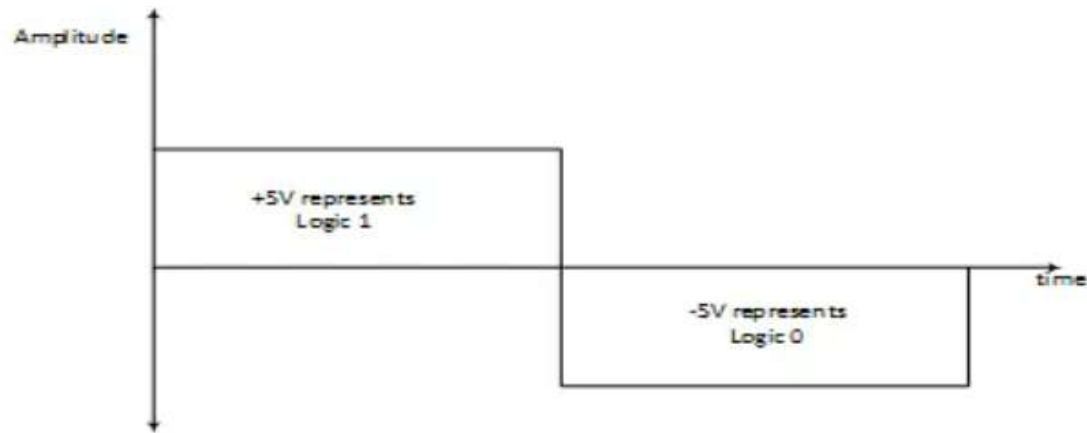


Fig. 2 a. Positive logic representation

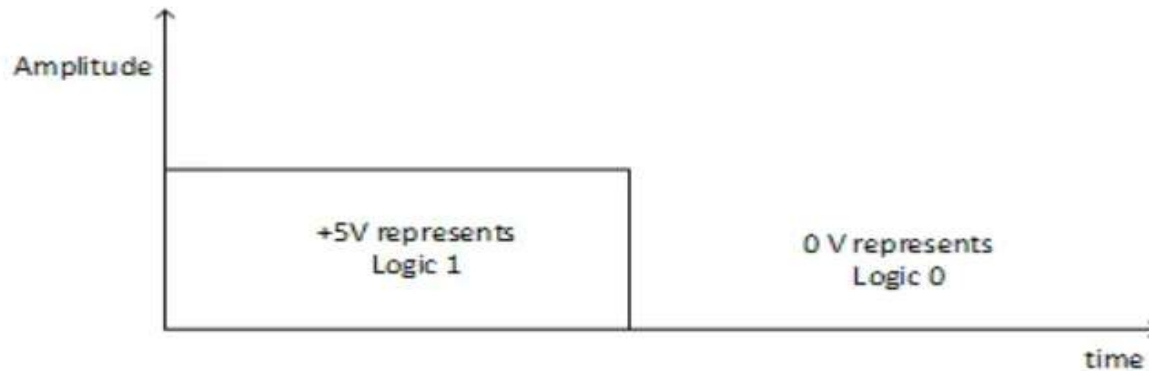


Fig. 2 b. Positive logic representation

# Negative Logic

- In Negative logic representation Bit 1 represents logic low and Bit 0 represents logic high as shown in Fig 3 a and b.
- In terms of voltage level, bit 1 can be represented as +5V and bit 0 can be represented as 0 V or -5 Volts

# Negative Logic

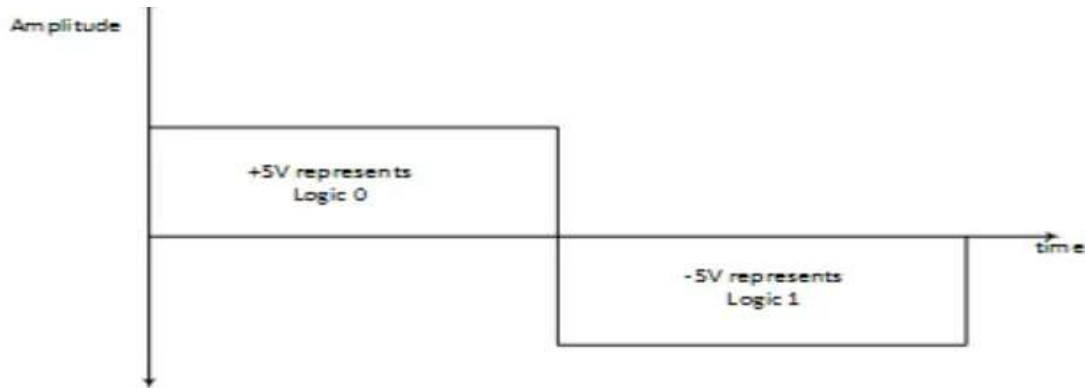


Fig. 3 a. Negative logic representation



Fig. 3 b. Negative logic representation

# Positive and Negative Logic

## Positive & Negative Logic

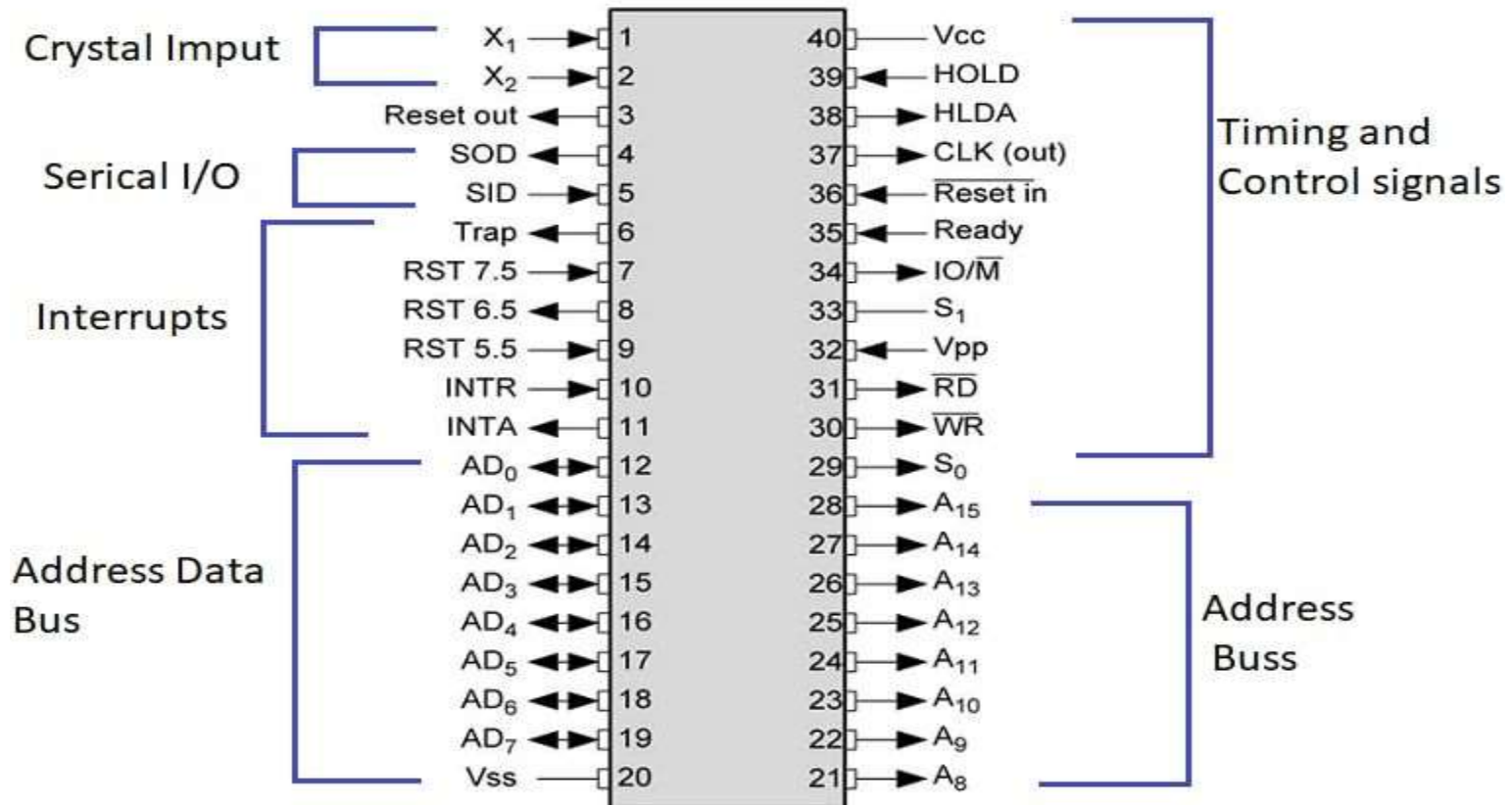
- In logic gates, usually:
  - ❖ H (high voltage, 5V) = 1
  - ❖ L (low voltage, 0V) = 0
- This convention – **positive logic**.
- However, the reverse convention, **negative logic** possible:
  - ❖ H (high voltage) = 0
  - ❖ L (low voltage) = 1
- Depending on convention, same gate may denote different Boolean function.

# Active high and Active low concepts

- When working with ICs and microcontrollers, you'll likely encounter pins that are active-low and pins that are active-high. Simply put, this just describes how the pin is activated. If it's an active-low pin, you must "pull" that pin LOW by connecting it to ground. For an active high pin, you connect it to your HIGH voltage (usually 3.3V/5V).
- For example, let's say you have a [shift register](#) that has a chip enable pin, CE. If you see the CE pin anywhere in the datasheet with a line over it like this, CE, then that pin is active-low. The CE pin would need to be pulled to GND in order for the chip to become enabled. If, however, the CE pin doesn't have a line over it, then it is active high, and it needs to be pulled HIGH in order to enable the pin.

# Active high and Active low concepts

## 8085 Pin Diagram



# Logic Operator

- Digital logic has three basic operators, **the AND, the OR and the NOT**.
- These three operators form the basis for everything in digital logic. In fact, almost everything your computer does can be described in terms of these three operations.
- One of the things that makes binary so useful in electronics is that it's very efficient at handling special operations called logical operations.
- Logical operations compare two binary bits and render a third binary bit as a result.

# Logic Operation

- **An operation on logical values, producing a Boolean result** (see also Boolean algebra). The operations may be monadic or dyadic, and are denoted by symbols known as operators. In general there are 16 logic operations over one or two operands; they include AND, OR, NOT, NAND, NOR, exclusive-OR, and equivalence.

## Basic Logic Operations

**Table 1.8**  
*Truth Tables of Logical Operations*

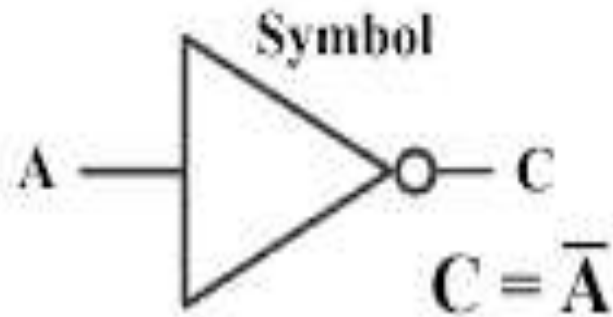
AND			OR			NOT	
$x$	$y$	$x \cdot y$	$x$	$y$	$x + y$	$x$	$x'$
0	0	0	0	0	0	0	1
0	1	0	0	1	1	1	0
1	0	0	1	0	1		
1	1	1	1	1	1		

# Basic Logic Gates & Operations

- **Fundamental Gates**
- AND
- OR
- NOT (Complement)
- **Universal Gates**
- NAND
- NOR
- **Arithmetic Gates**
- Ex-OR
- Ex-NOR

# NOT Gate

## NOT Gate

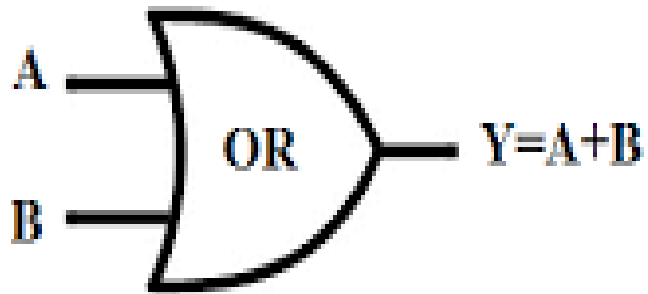


## Truth Table

INPUT	OUTPUT
A	NOT A
0	1
1	0

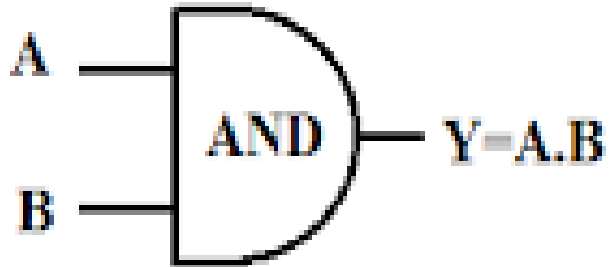
Projecthut123.com

# OR Gate



Inputs		Output
A	B	$Y=A+B$
0	0	0
0	1	1
1	0	1
1	1	1

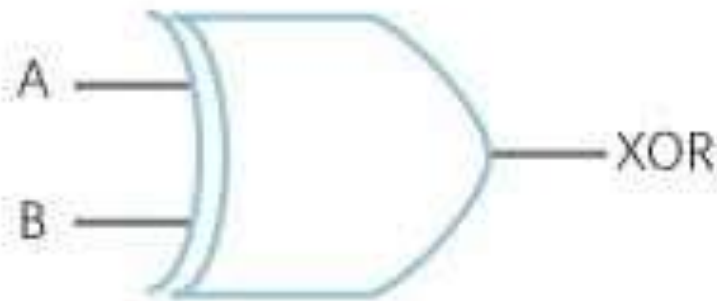
# AND Gate



Inputs		Output
A	B	$Y = A.B$
0	0	0
0	1	0
1	0	0
1	1	1

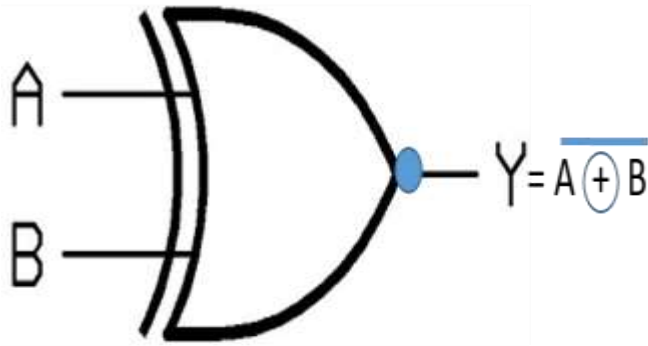
# Ex-OR Gate

$$X = A \oplus B$$



A	B	XOR
0	0	0
0	1	1
1	0	1
1	1	0

# Ex-NOR Gate



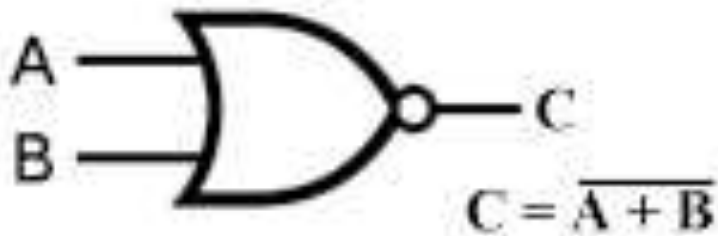
EX-NOR gate symbol

A	B	$Y = A \oplus B$
0	0	1
0	1	0
1	0	0
1	1	1

EX-NOR gate truth table

# NOR Gate

## NOR GATE



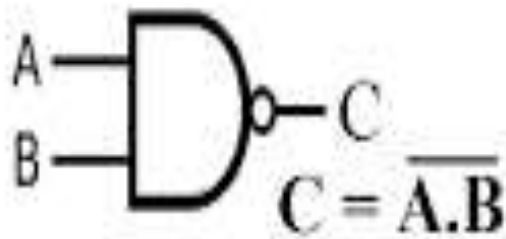
TRUTH TABLE

INPUT		OUTPUT
A	B	A NOR B
0	0	1
0	1	0
1	0	0
1	1	0

ProjectToT123.com

# NAND Gate

## NAND GATE



Truth Table

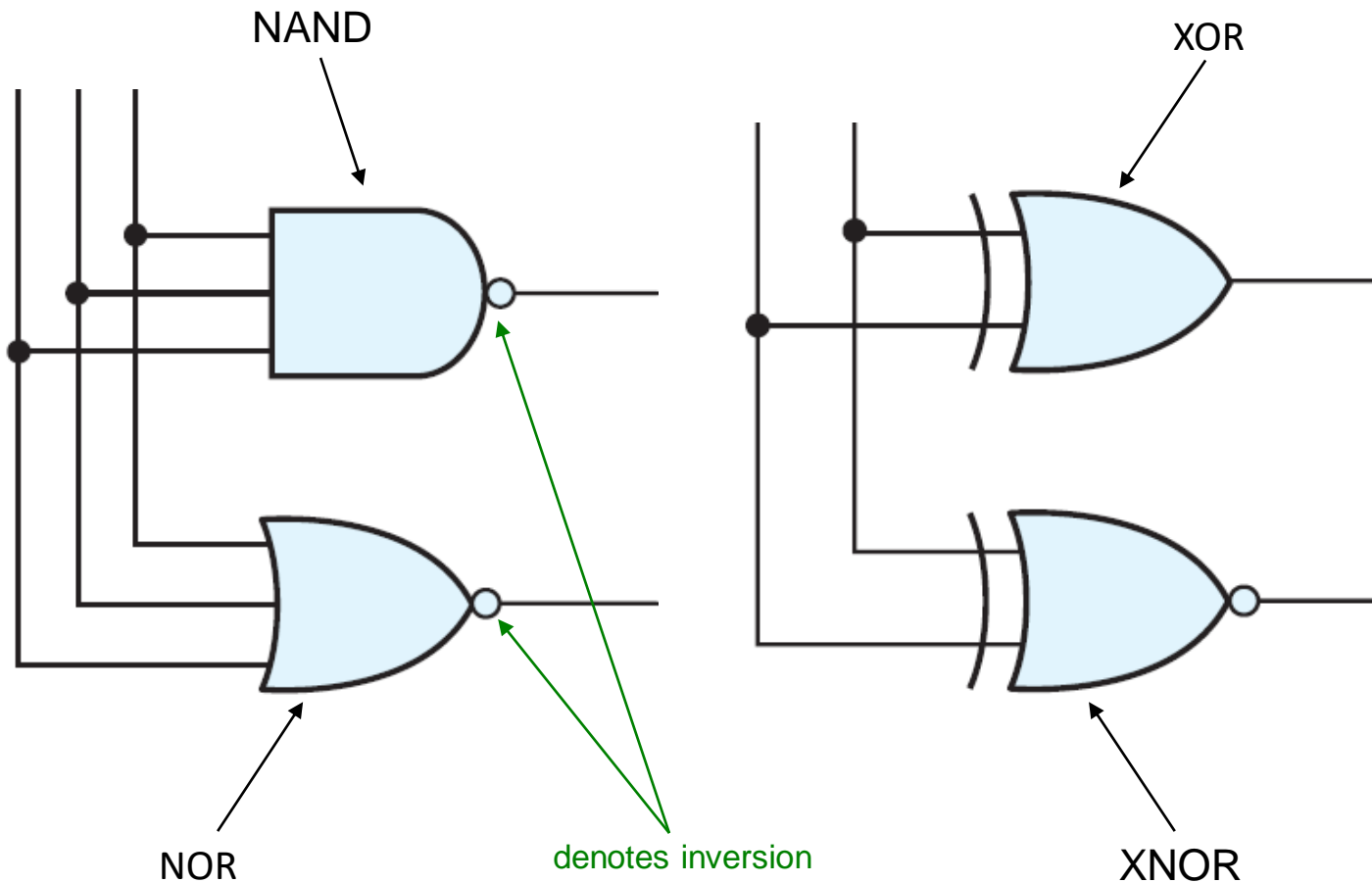
INPUT		OUTPUT
A	B	A NAND B
0	0	1
0	1	1
1	0	1
1	1	0

Project071231mm

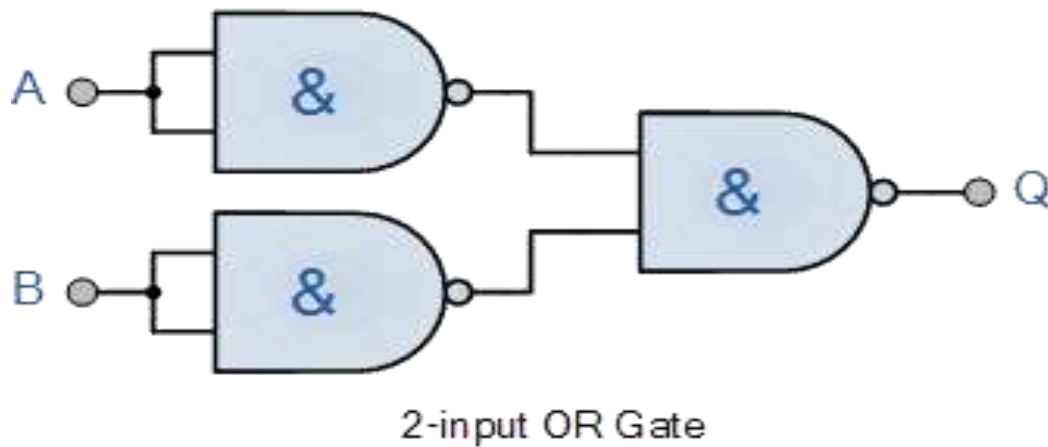
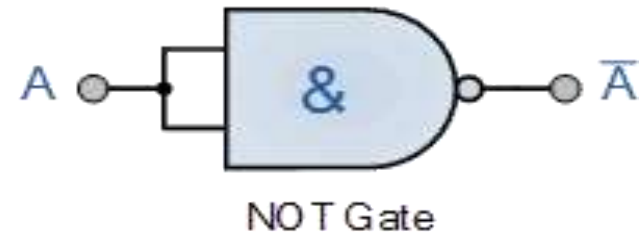
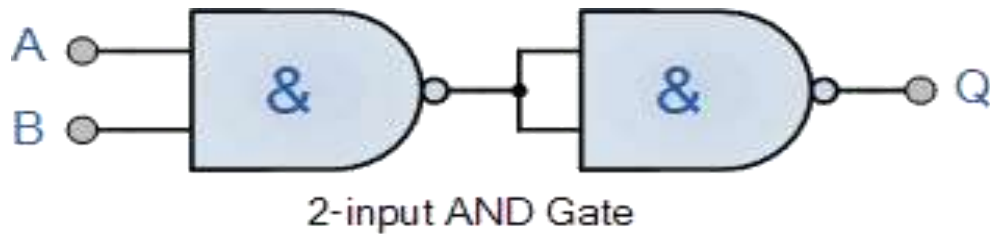
# Additional Logic Operations

- NAND
  - $F = (A \cdot B)'$
- NOR
  - $F = (A + B)'$
- XOR
  - Output is 1 iff either input is 1, but not both.
- XNOR (aka. Equivalence)
  - Output is 1 iff both inputs are 1 or both inputs are 0.

# Additional Logic Operations

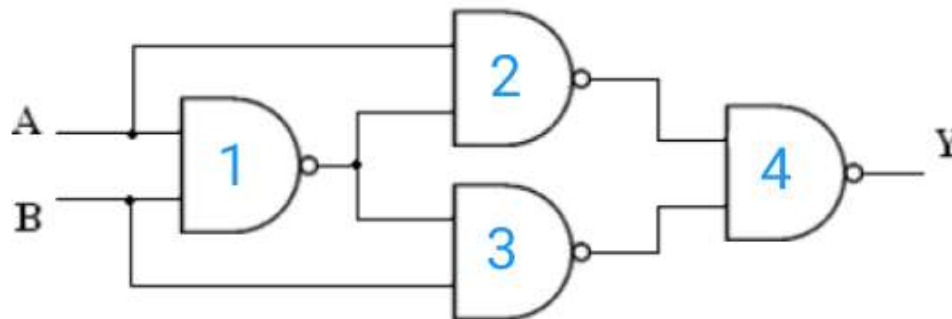


# Universal Gate-NAND Gate



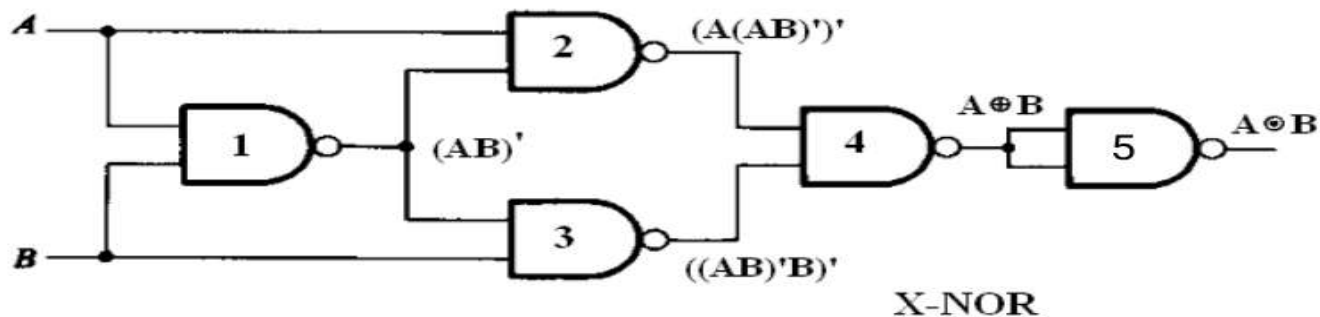
# NAND Gate

(e) Ex-OR gate:  $Y = A \oplus B$



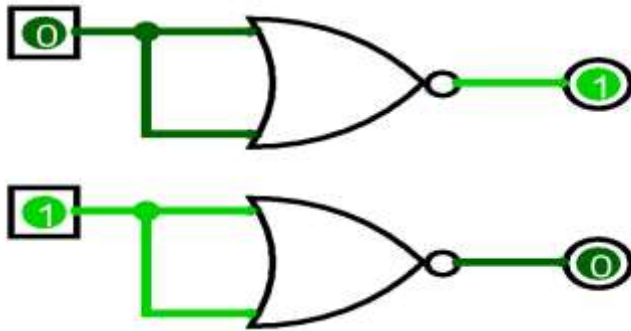
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

NAND Gates as EX-NOR Gate



# Universal Gate-NOR Gate

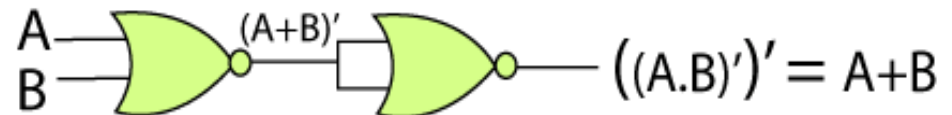
NOT USING NOR GATE



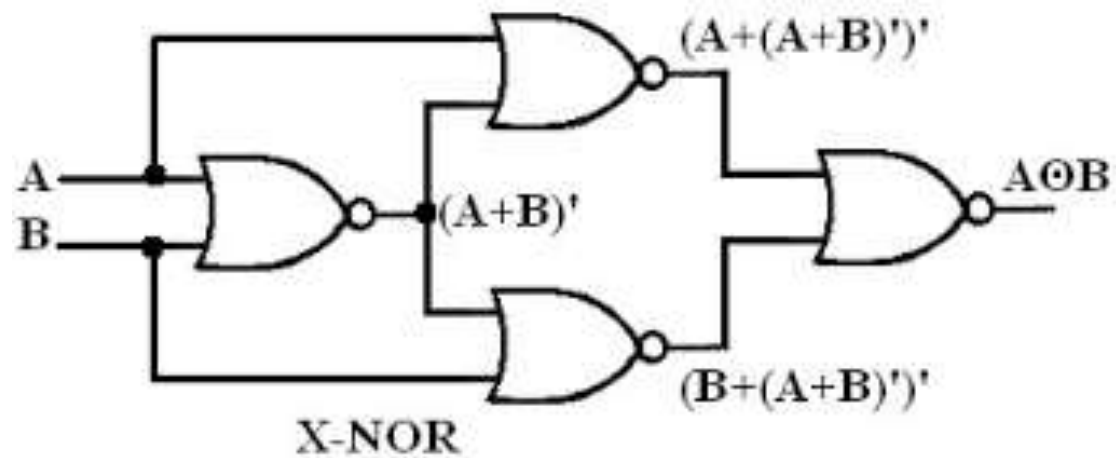
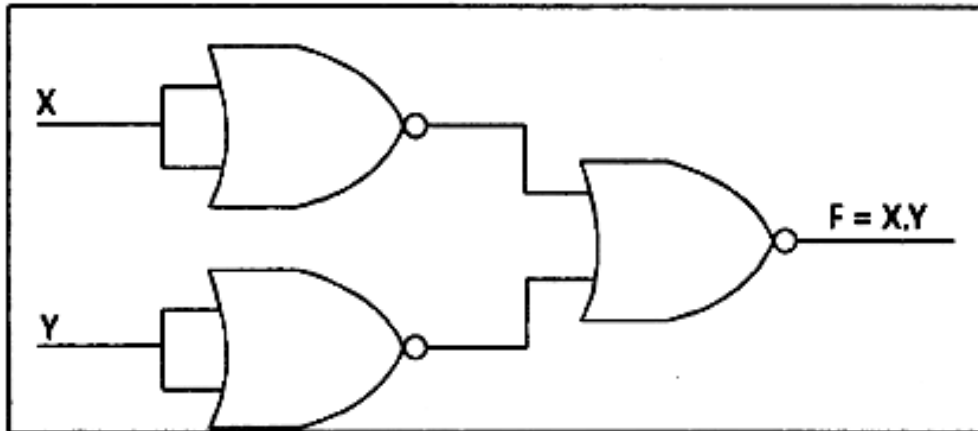
OR Gate using NOR Gate



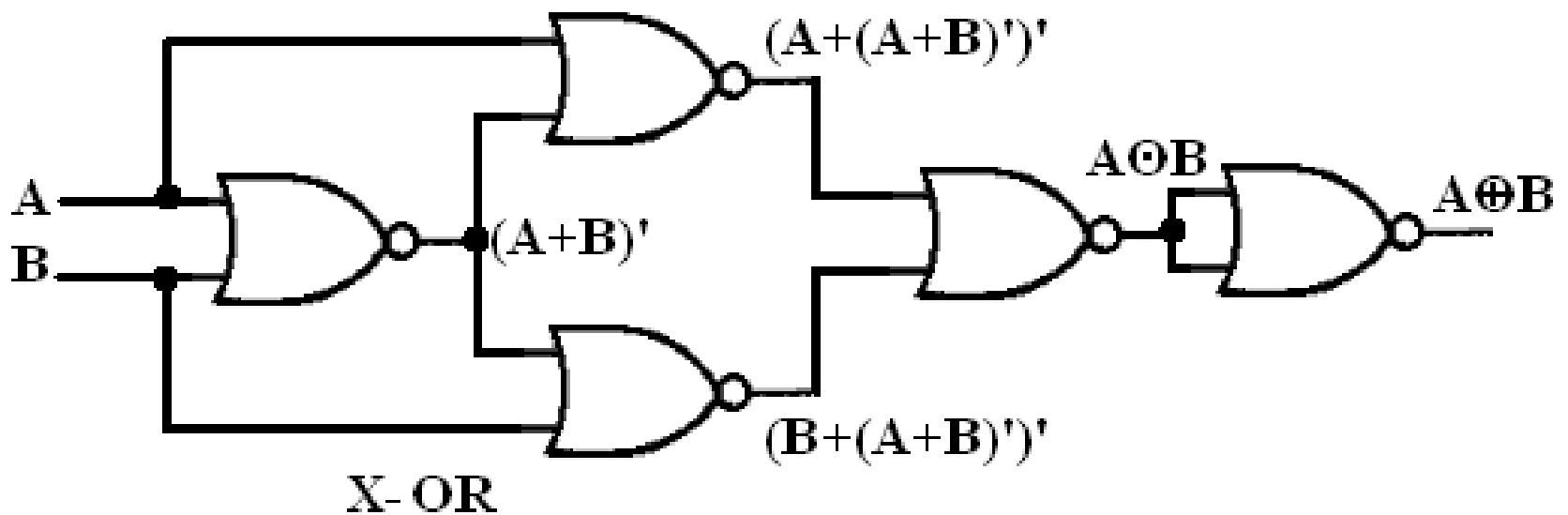
$$A+B = ((A.B)')' \quad \text{Involution law}$$



# NOR Gate



# NOR Gate



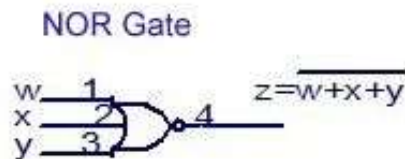
# Truth Tables

- Used to describe the functional behavior of a Boolean expression and/or Logic circuit.
- Each row in the truth table represents a unique combination of the input variables.
  - For  $n$  input variables, there are  $2^n$  rows.
- The output of the logic function is defined for each row.
- Each row is assigned a numerical value, with the rows listed in ascending order.
- The order of the input variables defined in the logic function is important.

# 3-input Truth Table

$F(A,B,C)$  = Boolean expression

## 3 Input NOR Gate

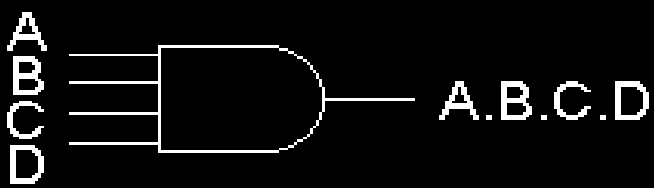


TRUTH TABLE

INPUTS			OUTPUT
W	X	Y	Z
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

# 4-input Truth Table

$F(A,B,C,D)$  = Boolean expression

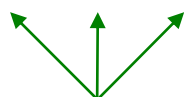
				
4 Input AND gate				
A	B	C	D	A.B.C.D
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	1

# Boolean Expressions

- Boolean expressions are composed of
  - Literals – variables and their complements
  - Logical operations

- Examples

- $F = A.B'.C + A'.B.C' + A.B.C + A'.B'.C'$



literals



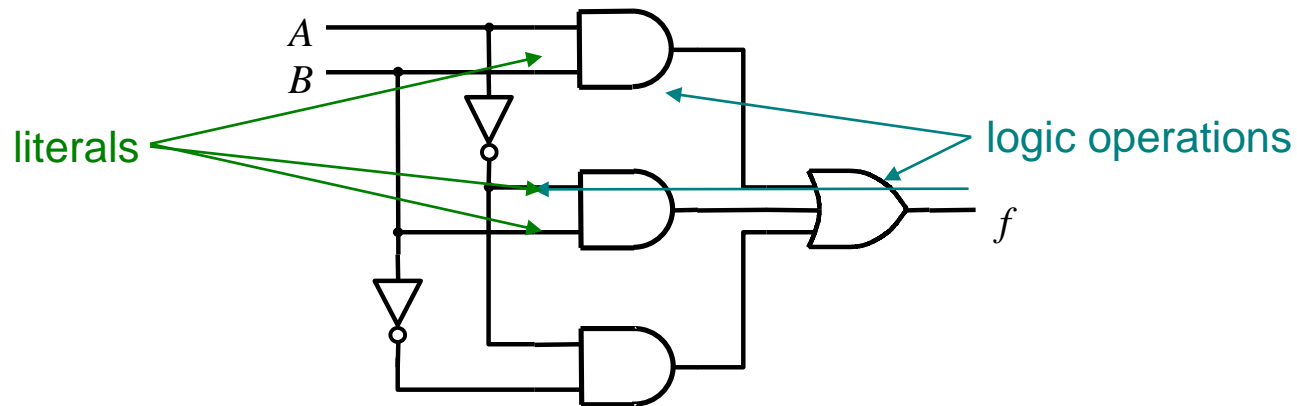
logic operations

- $F = (A+B+C').(A'+B'+C).(A+B+C)$

- $F = A.B'.C' + A.(B.C' + B'.C)$

# Boolean Expressions

- Boolean expressions are realized using a network (or combination) of logic gates.
  - Each logic gate implements one of the logic operations in the Boolean expression
  - Each input to a logic gate represents one of the literals in the Boolean expression



# Boolean Algebra

- **George Boole** developed an algebraic description for processes involving logical thought and reasoning.
  - Became known as Boolean Algebra
- **Claude Shannon** later demonstrated that Boolean Algebra could be used to describe switching circuits.
  - Switching circuits are circuits built from devices that switch between two states (e.g. 0 and 1).
  - Switching Algebra is a special case of Boolean Algebra in which all variables take on just two distinct values
- Boolean Algebra is a powerful tool for analyzing and designing logic circuits.

# Basic Laws and Theorems

Commutative Law	$A + B = B + A$	$A.B = B.A$
Associative Law	$A + (B + C) = (A + B) + C$	$A . (B . C) = (A . B) . C$
Distributive Law	$A.(B + C) = AB + AC$	$A + (B . C) = (A + B) . (A + C)$
Null Elements	$A + 1 = 1$	$A . 0 = 0$
Identity	$A + 0 = A$	$A . 1 = A$
Idempotence	$A + A = A$	$A . A = A$
Complement	$A + A' = 1$	$A . A' = 0$
Involution	$A'' = A$	
Absorption (Covering)	$A + AB = A$	$A . (A + B) = A$
Simplification	$A + A'B = A + B$	$A . (A' + B) = A . B$
DeMorgan's Rule	$(A + B)' = A'.B'$	$(A . B)' = A' + B'$
Logic Adjacency (Combining)	$AB + AB' = A$	$(A + B) . (A + B') = A$
Consensus	$AB + BC + A'C = AB + A'C$	$(A + B) . (B + C) . (A' + C) = (A + B) . (A' + C)$

# Idempotence

$$A + A = A$$

$$F = ABC + ABC' + ABC$$

$$F = ABC + ABC'$$

*Note: terms can also be added using this theorem*

$$A . A = A$$

$$G = (A' + B + C').(A + B' + C).(A + B' + C)$$

$$G = (A' + B + C') + (A + B' + C)$$

*Note: terms can also be added using this theorem*

# Complement

$$A + A' = 1$$

$$F = ABC'D + ABCD$$

$$F = ABD.(C' + C)$$

$$F = ABD$$

$$A . A' = 0$$

$$G = (A + B + C + D).(A + B' + C + D)$$

$$G = (A + C + D) + (B . B')$$

$$G = A + C + D$$

# Distributive Law

$$A.(B + C) = AB + AC$$

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

$$F = WX.(Y + Z)$$

$$F = WX + (Y.Z)$$

$$F = WXY + WXZ$$

$$F = (WX + Y).(WX + Z)$$

$$G = B'.(AC + AD)$$

$$G = B' + (A.C.D)$$

$$G = AB'C + AB'D$$

$$G = (B' + A).(B' + C).(B' + D)$$

$$H = A.(W'X + WX' + YZ)$$

$$H = A + ( (W'X).(WX') )$$

$$H = AW'X + AWX' + AYZ$$

$$H = (A + W'X).(A + WX')$$

# Absorption (Covering)

$$A + AB = A$$

$$A.(A + B) = A$$

$$F = A'BC + A'$$
$$F = A'$$

$$F = A'.(A' + BC)$$
$$F = A'$$

$$G = XYZ + XY'Z + X'Y'Z' + XZ$$
$$G = XYZ + XZ + X'Y'Z'$$
$$G = XZ + X'Y'Z'$$

$$G = XZ.(XZ + Y + Y')$$
$$G = XZ.(XZ + Y)$$
$$G = XZ$$

$$H = D + DE + DEF$$
$$H = D$$

$$H = D.(D + E + EF)$$
$$H = D$$

# Simplification

$$A + A'B = A + B$$

$$F = (XY + Z).(Y'W + Z'V') + (XY + Z)'$$

$$F = Y'W + Z'V' + (XY + Z)'$$

$$A.(A' + B) = A . B$$

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

$$G = (X + Y) . WZ$$

# Logic Adjacency (Combining)

$$A.B + A.B' = A$$

$$F = (X + Y).(W'X'Z) + (X + Y).(W'X'Z)'$$
$$F = (X + Y)$$

$$(A + B).(A + B') = A$$

$$G = (XY + X'Z).(XY + (X'Z)')$$
$$G = XY$$

# Boolean Algebra: Example

**Using Boolean Algebra, simplify the following Boolean expression.**

$$F(A,B,C) = A'.B.C + A.B'.C + A.B.C$$

$$F(A,B,C) = (A'+B'+C').(A'+B+C').(A+B'+C')$$

# Importance of Boolean Algebra

- Boolean Algebra is used to simplify Boolean expressions.
  - Through application of the Laws and Theorems discussed
- **Simpler expressions lead to simpler circuit realization, which, generally, reduces cost, area requirements, and power consumption.**
- The objective of the digital circuit designer is to design and realize **optimal digital circuits**.

# Algebraic Simplification

- **Boolean (or Switching) expressions can be simplified using the following methods:**
  1. Multiplying out the expression
  2. Factoring the expression
  3. Combining terms of the expression
  4. Eliminating terms in the expression
  5. Eliminating literals in the expression
  6. Adding redundant terms to the expression

As we shall see, there are other tools that can be used to simplify Boolean Expressions. Namely, **Karnaugh Maps**.

# Boolean Expression Reduction

**EXAMPLE 5.5** Reduce the expression

$$A + B[AC + (B + \bar{C})D]$$

*Solution*

The given expression is

$$A + B[AC + (B + \bar{C})D]$$

Expand  $(B + \bar{C})D$

$$= A + B(AC + BD + \bar{C}D)$$

Expand  $B(AC + BD + \bar{C}D)$

$$= A + BAC + BBD + B\bar{C}D$$

Write in order

$$= A + ABC + BD + BD\bar{C}$$

Factor

$$= A(1 + BC) + BD(1 + \bar{C})$$

Reduce

$$= A \cdot 1 + BD \cdot 1$$

Simplify

$$= A + BD$$

# Boolean Expression Reduction

**EXAMPLE 5.6** Reduce the expression

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

*Solution*

The given expression is

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

Demorganize  $\overline{(A + \overline{BC})}$

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

Simplify

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

Multiply

$$= \overline{A} BC A\overline{B} + \overline{A} BC ABC$$

Rearrange

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

$$= 0 + 0 = 0$$

# Boolean Expression Reduction

**EXAMPLE 5.7** Reduce the expression

$$(B + BC) (B + \overline{B}C) (B + D)$$

*Solution*

The given expression is

Multiply the first two terms

Reduce

$$(B + BC) (B + \overline{B}C) (B + D)$$

$$= (BB + BCB + B\overline{B}C + BC\overline{B}C) (B + D)$$

$$= (B + BC + 0 + 0) (B + D)$$

Factor

Reduce

Expand

Simplify

$$= B(1 + C) (B + D)$$

$$= B(B + D)$$

$$= BB + BD$$

$$= B(1 + D) = B$$

# Boolean Expression Reduction

**EXAMPLE 5.8** Show that

$$AB + A\bar{B}C + B\bar{C} = AC + B\bar{C}$$

**Solution**

$$\begin{aligned} AB + A\bar{B}C + B\bar{C} &= A(B + \bar{B}C) + B\bar{C} \\ &= A(B + \bar{B})(B + C) + B\bar{C} \\ &= AB + AC + B\bar{C} \\ &= AB(C + \bar{C}) + AC + B\bar{C} \\ &= ABC + AB\bar{C} + AC + B\bar{C} \\ &= AC(1 + B) + B\bar{C}(1 + A) \\ &= AC + B\bar{C} \end{aligned}$$

# Boolean Expression Reduction

**EXAMPLE 5.9** Show that  $A\bar{B}C + B + B\bar{D} + AB\bar{D} + \bar{A}C = B + C$

*Solution*

$$\begin{aligned} A\bar{B}C + B + B\bar{D} + AB\bar{D} + \bar{A}C &= A\bar{B}C + \bar{A}C + B(1 + \bar{D} + A\bar{D}) \\ &= C(\bar{A} + A\bar{B}) + B \\ &= C(\bar{A} + A)(\bar{A} + \bar{B}) + B \\ &= C\bar{A} + C\bar{B} + B \\ &= (B + C)(B + \bar{B}) + C\bar{A} \\ &= B + C + C\bar{A} \\ &= B + C(1 + \bar{A}) \\ &= B + C \end{aligned}$$

# Boolean Function & Their Representation

- A function of 'n' variable denoted by  $f(x_1, x_2, x_3, \dots, x_n)$  is another variable of algebra and takes one of the two possible values 0, and 1.
- The various ways of representing a given function are given as:

# SOP & POS

- This form is called Disjunctive Normal Form (DNF)
- For example:

$$\text{For example, } f(A, B, C) = \bar{A}B + \bar{B}C.$$

- This form is called Conjunctive Normal Form (CNF)
- For example:

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

# Truth Table form

- In this form the function is specified by listing all possible combination of values assumed by the variables and the corresponding values of the function

**Table 5.1** Truth table for  $f(A, B, C) = \bar{A}B + \bar{B}C$

Decimal code	A	B	C	f(A, B, C)
0	0	0	0	0
1	0	0	1	1
2	0	1	0	1
3	0	1	1	1
4	1	0	0	0
5	1	0	1	1
6	1	1	0	0
7	1	1	1	0

# Standard SOP

- This form is called Disjunctive Canonical Form (DCF), also called expanded SOP form, Canonical SOP form.
- In this form function is the sum of a number of product term where each product term contains all the variables of the function either in complemented or uncomplemented form.
- This can be derived from the truth table by finding the sum of all the terms that corresponds to those combinations (rows) for which 'f' assumes the value '1'. **It can also be obtained from SOP form algebraically.**

# Standard SOP

- A product term which contain all the variable in term of complemented or uncomplemented form is **called minterm**.
- A minterm assumes the value 1 only for one combination of the variables. An n variables function can have in all  $2^n$  minterms.
- The sum of minterms whose value is equal to 1 is the **standard SOP** form of the function.
- The minterms are often denoted as  $m_0, m_1, m_2, \dots$ . Where suffix are the decimal codes of the combinations.

Thus

$$f(A, B, C) = m_1 + m_2 + m_3 + m_5$$

Yet another way of representing the function in DCF is by listing the decimal codes of the minterms for which  $f = 1$ .

Thus

$$f(A, B, C) = \Sigma m(1, 2, 3, 5)$$

where  $\Sigma m$  represents the sum of all the minterms whose decimal codes are given in the parenthesis.

# Standard POS

- This form is called Conjunctive Canonical Form (CCF), also called expanded POS form, Canonical POS form.
- This is derived by considering the combinations for which  $f=0$ . Each term is sum of all the variables.
- A variable appears in uncomplemented form if it has a value of 0 in the combination and appears in complemented form if it has a value of 1 in the combination.

Thus, the function  $f(A, B, C) = (\bar{A} + \bar{B})(A + B)$  is given

by the product of sums

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

# Standard POS

- A sum term which contain each of the n variable in either complimented or uncomplimented form is **called maxterm**
- A maxterm assumes the value 0 only for one combination of the variables. An n variables function can have in all  $2^n$  maxterms.
- The product of maxterms whose value is equal to 0 is the **standard POS** form of the function.
- The **maxterms** are often denoted as  $M_0, M_1, M_2, \dots$ . Where suffix are the decimal codes of the combinations.

$$f(A, B, C) = M_0 \cdot M_4 \cdot M_6 \cdot M_7 \quad \text{or simply}$$

$$f(A, B, C) = \Pi M(0, 4, 6, 7)$$

# Standard SOP & POS

- Sum of Product
- Minterm
- The sum of minterms whose value is equal to 1 is the **standard SOP** form of the function.
- The minterms are often denoted as  $m_0, m_1, m_2, \dots$ . Where suffix are the decimal codes of the combinations.
- Product of Sum
- Maxterm
- The product of maxterms whose value is equal to 0 is the **standard POS** form of the function.
- The **maxterms** are often denoted as  $M_0, M_1, M_2, \dots$ . Where suffix are the decimal codes of the combinations.

$$f(A, B, C) = m_1 + m_2 + m_3 + m_5$$

$$f(A, B, C) = \Sigma m(1, 2, 3, 5)$$

$$f(A, B, C) = M_0 \cdot M_4 \cdot M_6 \cdot M_7 \quad \text{or simply}$$

$$f(A, B, C) = \Pi M(0, 4, 6, 7)$$

# Expansion of a Boolean Expression to SOP Form

**EXAMPLE 6.1** Expand  $\overline{A} + \overline{B}$  to minterms and maxterms.

*Solution*

The given expression is a two-variable function. In the first term  $\overline{A}$ , the variable  $B$  is missing; so, multiply it by  $(B + \overline{B})$ . In the second term  $\overline{B}$ , the variable  $A$  is missing; so, multiply it by  $(A + \overline{A})$ . Therefore,

$$\begin{aligned}\overline{A} + \overline{B} &= \overline{A}(B + \overline{B}) + \overline{B}(A + \overline{A}) \\ &= \overline{A}B + \overline{A}\overline{B} + \overline{B}A + \overline{B}\overline{A} \\ &= \overline{A}B + \overline{A}\overline{B} + A\overline{B} + \overline{A}\overline{B} \\ &= \overline{A}B + \overline{A}\overline{B} + A\overline{B} \\ &= 01 + 00 + 10 \\ &= m_1 + m_0 + m_2 \\ &= \Sigma m(0, 1, 2)\end{aligned}$$

**EXAMPLE 6.2** Expand  $A + B\bar{C} + AB\bar{D} + ABCD$  to minterms and maxterms.

**Solution**

The given expression is a four-variable function. In the first term  $A$ , the variables  $B$ ,  $C$ , and  $D$  are missing. So, multiply it by  $(B + \bar{B})(C + \bar{C})(D + \bar{D})$ . In the second term  $B\bar{C}$ , the variables  $A$  and  $D$  are missing. So, multiply it by  $(A + \bar{A})(D + \bar{D})$ . In the third term,  $AB\bar{D}$ , the variable  $C$  is missing. So, multiply it by  $(C + \bar{C})$ . In the fourth term  $ABCD$ , all the variables are present. So, leave it as it is. Therefore,

$$\begin{aligned} A &= A(B + \bar{B})(C + \bar{C})(D + \bar{D}) \\ &= ABCD + ABC\bar{D} + AB\bar{C}D + AB\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}C\bar{D} + A\bar{B}\bar{C}D + A\bar{B}\bar{C}\bar{D} \\ B\bar{C} &= B\bar{C}(A + \bar{A})(D + \bar{D}) = AB\bar{C}D + AB\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D} \\ AB\bar{D} &= AB\bar{D}(C + \bar{C}) = ABC\bar{D} + AB\bar{C}\bar{D} \end{aligned}$$

or

$$\begin{aligned} A + B\bar{C} + AB\bar{D} + ABCD &= ABCD + ABC\bar{D} + AB\bar{C}D + AB\bar{C}\bar{D} + A\bar{B}CD + A\bar{B}C\bar{D} \\ &\quad + A\bar{B}\bar{C}D + A\bar{B}\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}B\bar{C}\bar{D} \\ &= m_{15} + m_{14} + m_{13} + m_{12} + m_{11} + m_{10} + m_9 + m_8 + m_5 + m_4 \\ &= \Sigma m(4, 5, 8, 9, 10, 11, 12, 13, 14, 15) \end{aligned}$$

In the SOP form, the minterms 0, 1, 2, 3, 6, and 7 are missing. So in the POS form, the maxterms 0, 1, 2, 3, 6, and 7 will be present. Therefore, the POS form is

$$\Pi M(0, 1, 2, 3, 6, 7)$$

# Expansion of a Boolean Expression to POS Form

**EXAMPLE 6.3** Expand  $A(\bar{B} + A)B$  to maxterms and minterms.

*Solution*

The given expression is a two-variable function in the POS form. The variable B is missing in the first term A. So, add  $B\bar{B}$  to it. The second term contains all the variables. So, leave it as it is. The variable A is missing in the third term B. So, add  $A\bar{A}$  to it. Therefore,

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

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

or

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

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

$$= (00)(01)(10)$$

$$= M_0 \cdot M_1 \cdot M_2$$

$$= \prod M(0, 1, 2)$$

The maxterm  $M_3$  is missing in the POS form. So, the SOP form will contain only the minterm  $m_3$ .

**EXAMPLE 6.4** Expand  $A(\bar{A} + B) (\bar{A} + B + \bar{C})$  to maxterms and minterms.

**Solution**

The given expression is a three-variable function in the POS form. The variables B and C are missing in the first term A. So, add  $B\bar{B}$  and  $C\bar{C}$  to it. The variable C is missing in the second term  $(\bar{A} + B)$ . So, add  $C\bar{C}$  to it. The third term  $(\bar{A} + B + \bar{C})$  contains all the three variables. So, leave it as it is. Therefore,

$$\begin{aligned} A &= A + B\bar{B} + C\bar{C} = (A + B) (A + \bar{B}) + C\bar{C} = (A + B + C\bar{C}) (A + \bar{B} + C\bar{C}) \\ &= (A + B + C) (A + B + \bar{C}) (A + \bar{B} + C) (A + \bar{B} + \bar{C}) \\ \bar{A} + B &= \bar{A} + B + C\bar{C} = (\bar{A} + B + C) (\bar{A} + B + \bar{C}) \end{aligned}$$

Therefore,

$$\begin{aligned} A(\bar{A} + B) (\bar{A} + B + \bar{C}) &= (A + B + C) (A + B + \bar{C}) (A + \bar{B} + C) (A + \bar{B} + \bar{C}) \\ &\quad (\bar{A} + B + C) (\bar{A} + B + \bar{C}) \\ &= (000) (001) (010) (011) (100) (101) \\ &= M_0 \cdot M_1 \cdot M_2 \cdot M_3 \cdot M_4 \cdot M_5 \\ &= \prod M(0, 1, 2, 3, 4, 5) \end{aligned}$$

The maxterms  $M_6$  and  $M_7$  are missing in the POS form. So, the SOP form will contain the minterms 6 and 7. Therefore, the given expression in the SOP form is  $\Sigma m(6, 7)$ .

Also

# Two Variable K-Map

## Mapping of SOP Expressions

A two-variable K-map has  $2^2 = 4$  squares. These squares are called cells. Each square on the K-map represents a unique minterm. The minterm designations of the squares are shown in Fig. 6.1. A 1 placed in any square indicates that the corresponding minterm is included in the output expression, and a 0 or no entry in any square indicates that the corresponding minterm does not appear in the expression for output.

A \ B	0	1
0	$\bar{A}\bar{B}^0$	$\bar{A}B^1$
1	$A\bar{B}^2$	$AB^3$

Fig. 6.1 Two-variable K-map.

Consider the expression,  $\bar{A}\bar{B} + A\bar{B} + AB$ . It can be expressed using minterms as

$$m_0 + m_2 + m_3 = \Sigma m(0, 2, 3)$$

A \ B	0	1
0	1 <sup>0</sup>	0 <sup>1</sup>
1	1 <sup>2</sup>	1 <sup>3</sup>

6.2 K-map of  $\Sigma m(0, 2, 3)$ .

**EXAMPLE 6.8** Map the expression  $\bar{A}B + A\bar{B}$ .

*Solution*

The given expression in minterms is

$$m_1 + m_2 = \Sigma m(1, 2).$$

The K-map is shown below.

A \ B	0	1
0	0 <sup>0</sup>	1 <sup>1</sup>
1	1 <sup>2</sup>	0 <sup>3</sup>

# Three Variable K-Map

A function in three variables (A, B, C) expressed in the SOP form can have eight possible combinations:  $\overline{A}\overline{B}\overline{C}$ ,  $\overline{A}\overline{B}C$ ,  $\overline{A}B\overline{C}$ ,  $\overline{A}BC$ ,  $A\overline{B}\overline{C}$ ,  $A\overline{B}C$ ,  $AB\overline{C}$ , and  $ABC$ . Each one of these combinations designated by  $m_0$ ,  $m_1$ ,  $m_2$ ,  $m_3$ ,  $m_4$ ,  $m_5$ ,  $m_6$ , and  $m_7$ , respectively, is called a minterm. A is the MSB of the minterm designator and C is the LSB.

In the POS form, the eight possible combinations are:  $A + B + C$ ,  $A + B + \overline{C}$ ,  $A + \overline{B} + C$ ,  $A + \overline{B} + \overline{C}$ ,  $\overline{A} + B + C$ ,  $\overline{A} + B + \overline{C}$ ,  $\overline{A} + \overline{B} + C$ , and  $\overline{A} + \overline{B} + \overline{C}$ . Each one of these combinations designated by  $M_0$ ,  $M_1$ ,  $M_2$ ,  $M_3$ ,  $M_4$ ,  $M_5$ ,  $M_6$ , and  $M_7$ , respectively, is called a maxterm. A is the MSB of the maxterm designator and C is the LSB.

A three-variable K-map has, therefore,  $8(= 2^3)$  squares or cells, and each square on the map represents a minterm or maxterm as shown in Figs. 6.6a and b.

A \ BC				
	00	01	11	10
0	$\overline{A}\overline{B}\overline{C}^0$	$\overline{A}\overline{B}C^1$	$\overline{A}B\overline{C}^3$	$\overline{A}BC^2$
1	$A\overline{B}\overline{C}^4$	$A\overline{B}C^5$	$AB\overline{C}^7$	$ABC^6$

(a) Minterms

A \ BC				
	00	01	11	10
0	$A + B + C^0$	$A + B + \overline{C}^1$	$A + \overline{B} + \overline{C}^3$	$A + \overline{B} + C^2$
1	$\overline{A} + B + C^4$	$\overline{A} + B + \overline{C}^5$	$\overline{A} + \overline{B} + \overline{C}^7$	$\overline{A} + \overline{B} + C^6$

(b) Maxterms

Fig. 6.6 The minterms and maxterms of a three-variable K-map.

# Four Variable K-Map

A four-variable (A, B, C, D) expression can have  $2^4 = 16$  possible combinations of input variables such as  $\overline{A}\overline{B}\overline{C}\overline{D}$ ,  $\overline{A}\overline{B}\overline{C}D$ , . . . , ABCD with minterm designations  $m_0, m_1, \dots, m_{15}$ , respectively, in the SOP form and as  $A + B + C + D, \dots, \overline{A} + \overline{B} + \overline{C} + \overline{D}$  with maxterm designations  $M_0, M_1, \dots, M_{15}$ , respectively, in the POS form.

A four-variable K-map has  $2^4 = 16$  squares or cells and each square on the map represents either a minterm or a maxterm as shown in Fig. 6.9.

The binary number designations of the rows and columns are in Gray code. The binary numbers along the top of the map indicate the conditions of C and D along any column and binary numbers along the left side indicate the conditions of A and B along any row. The numbers in the top right corners of the squares indicate the minterm or maxterm designations as usual.

# Four Variable K-Map

AB \ CD		CD			
		00	01	11	10
AB	00	0 $\bar{A}\bar{B}\bar{C}\bar{D}$	1 $\bar{A}\bar{B}\bar{C}D$	3 $\bar{A}\bar{B}CD$	2 $\bar{A}\bar{B}C\bar{D}$
	01	4 $\bar{A}B\bar{C}\bar{D}$	5 $\bar{A}B\bar{C}D$	7 $\bar{A}BCD$	6 $\bar{A}BC\bar{D}$
	11	12 $AB\bar{C}\bar{D}$	13 $AB\bar{C}D$	15 $ABCD$	14 $ABC\bar{D}$
	10	8 $A\bar{B}\bar{C}\bar{D}$	9 $A\bar{B}\bar{C}D$	11 $A\bar{B}CD$	10 $A\bar{B}C\bar{D}$

SOP form

AB \ CD		CD			
		00	01	11	10
AB	00	0 $A+B+C+D$	1 $A+B+C+\bar{D}$	3 $A+B+\bar{C}+\bar{D}$	2 $A+B+\bar{C}+D$
	01	4 $A+\bar{B}+C+D$	5 $A+\bar{B}+C+\bar{D}$	7 $A+\bar{B}+\bar{C}+\bar{D}$	6 $A+\bar{B}+\bar{C}+D$
	11	12 $\bar{A}+\bar{B}+C+D$	13 $\bar{A}+\bar{B}+C+\bar{D}$	15 $\bar{A}+\bar{B}+\bar{C}+\bar{D}$	14 $\bar{A}+\bar{B}+\bar{C}+D$
	10	8 $\bar{A}+B+C+D$	9 $\bar{A}+B+C+\bar{D}$	11 $\bar{A}+B+\bar{C}+\bar{D}$	10 $\bar{A}+B+\bar{C}+D$

POS form

Fig. 6.9 The minterms and maxterms of a four-variable K-map.

# K-Map Numerical

- **Procedure for Minimization**
- Octet            3 Variable
- Quad            2 Variable
- Pair            1 Variable
- Single           0 Variable
- Remove Redundant Terms

# K-Map Numerical

- **Implicit:** Each minterm given in expression is known as implicit
- **Prime Implicit:** PI is product term which is obtained by combining maximum possible cell in K-Map
- **Essential Prime Implicit:** EPI is PI which is necessary or compulsory in minimize expression

# Two Variable K-Map

SOP  $f(A, B) = \bar{A}B + A\bar{B} + AB$

$\begin{array}{c} B \\ \swarrow \searrow \\ A \end{array}$		0	1
			$\bar{A}B$
0			
1		$A\bar{B}$	$AB$

$\rightarrow A + B$

✓

$\begin{array}{c} B \\ \swarrow \searrow \\ A \end{array}$		0	1
0		1	1
1		1	

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

# Three Variable K-Map

Three Variable

31

$$\Sigma f(A, B, C) = \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}BC + ABC$$

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

	BC			
	00	01	11	10
$\bar{A}$		1	1	1
A			1	

$$f(A, B, C) = \Sigma m(0, 1, 2, 4, 5)$$

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

	00	01	11	10
0	1	1		1
1	1	1		1

# Three Variable K-Map

$$f(A, B, C) = \sum m(1, 2, 3, 6)$$

Minimized  $\rightarrow B\bar{C} + C\bar{A}$

Hazard free  $\rightarrow \bar{A}C + B\bar{C} + \bar{A}B$

	00	01	11	10
0		1	1	1
1				1

Hazard free

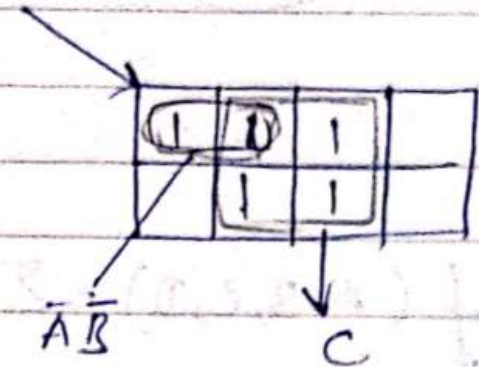
# Three Variable K-Map

✓  $f(A, B, C) = \sum m(0, 1, 3, 5, 7)$

I - 5

PI - 2

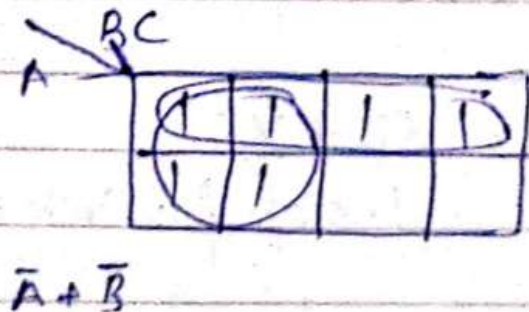
EPI - 2



$f(A, B, C) = \sum m(0, 1, 2, 3, 4, 5)$

PI - 2

EPI - 2



# Three Variable K-Map

Implicant - 4

PI - 3

EPI - 3

$$f(A, B, C) = \sum m(3, 5, 6, 7)$$

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

$\Rightarrow AC + AB + BC$  unique

$$f(A, B, C) = \sum m(1, 3, 6, 7)$$

	$\bar{B}\bar{C}$	$\bar{B}C$	$BC$	$B\bar{C}$
$\bar{A}$		1	1	
A			1	1

I - 4

PI -  $\bar{A}C, AB, BC$  3

EPI  $\rightarrow \bar{A}C + AB$  - 2

✓

$\bar{A}C + AB$

# Three Variable K-Map

$$f(A, B, C) = \sum m(2, 3, 4, 5, 7)$$

$$I = 5$$

$$PI = 4 - \bar{A}\bar{B}, AC, BC, \bar{A}B$$

$$EPI = 2, \bar{A}\bar{B}, \bar{A}B$$

	00	01	11	10
0		1	1	1
1	1	1	1	

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

$$f(A, B, C) = \sum m(0, 1, 2, 5, 6, 7)$$

$$I = 6$$

$$PI = 6$$

$$EPI = 0$$

	00	01	11	10
0	1	1		1
1		1	1	1

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

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