R.M.K GROUP OF ENGINEERING INSTITUTIONS

R.M.K
GROUP OF
INSTITUTIONS

R.M.K. ENGINEERING COLLEGE

22EC101 DIGITAL PRINCIPLES AND SYSTEM DESIGN

Department: **CSE,ECE, IT,AD, CD**

Batch/Year: **BATCH 2022-2026/I**

Created by:
Dr.S.Joshua Kumaresan, Professor/ ECE
Mr.T.Joel, ASP/ ECE
Mr.K.Vijayanand, ASP/ EIE
Dr.R.S.Ganesh , ASP/ ECE
Mr.J.Bharath Singh, AP / EIE
Ms.M.Vimala, AP / EEE
Mr.Alexander Jeevanandam, AP / EEE
Ms.Pavaiayarkarasi, AP / ECE
Mr.C.Karthikeyan , AP / ECE
Ms. Vishnu Priya, AP / ECE
Ms. S. Mahin Sultana, AP / ECE

Date: **1st November, 2022**

Table of Contents

R.M.K
GROUP OF
INSTITUTIONS

2.COURSE OBJECTIVES

- To acquire the knowledge in Digital fundamentals and its simplification methods.
- To familiarize the design of various combinational digital circuits using logic gates.
- To realize various sequential circuits using flip flops.
- To interpret various clocked sequential circuits.
- To elucidate various semiconductor memories and related technology.
- To build various logic functions using Programming Logic Devices.

3.PRE REQUISITES

NIL

4.Syllabus

22EC101 DIGITAL PRINCIPLES AND SYSTEM DESIGN (LAB INTEGRATED)

L	T	P	C
3	0	2	4

UNIT I BOOLEAN ALGEBRA AND LOGIC GATES 9

Review of number systems-representation-conversions, Review of Boolean algebra-theorems, sum of product and product of sum simplification, canonical forms, min term and max term, Simplification of Boolean expressions-Karnaugh map, Implementation of Boolean expressions using logic gates and universal gates.

UNIT II COMBINATIONAL LOGIC CIRCUITS 9

Design of combinational circuits - Half and Full Adders, Half and Full Subtractors, Binary Parallel Adder – Carry look ahead Adder, Magnitude Comparator, Decoder, Encoder, Priority Encoder, Mux/De-mux, Parity Generator/Checker

UNIT III SEQUENTIAL CIRCUITS 9

Flip flops – SR, JK, T, D, Master/Slave FF – operation and excitation tables, Asynchronous and Synchronous Counters Design - Shift registers, Universal Shift Register

UNIT IV SYNCHRONOUSSEQUENTIAL CIRCUITS DESIGN 9

Design of clocked sequential circuits - Moore/Mealy models, state minimization, state assignment, circuit implementation

UNIT V MEMORY AND PROGRAMMABLE LOGIC DEVICES 9

Basic memory structure ROM: PROM – EPROM – EEPROM –RAM – Static and dynamic RAM – Programmable Logic Devices: Programmable Logic Array (PLA) – Programmable Array Logic (PAL) – Implementation of combinational logic circuits using PLA, PAL.

TOTAL : 45PERIODS

LIST OF EXPERIMENTS : 30 PERIODS

- Implementation of Boolean expression using logic gates.
- Design of adders
- Design of subtractors.
- Design of binary adder using IC7483
- Design of Multiplexers & Demultiplexers.
- Design of Encoders and Decoders.
- Implementation of a boolean function using a multiplexer.
- Design and implementation of 3 bit ripple counters.
- Design and implementation of 3 bit synchronous counter
- Design and implementation of shift registers.

5. Course Outcomes

Course Outcomes	Description	Knowledge Level
CO1	Implement digital circuits using simplified Boolean functions.	K1
CO2	Realize Combinational circuits for a given function using logic gates.	K2
CO3	Demonstrate the operation of various counters and shift registers using Flip Flops.	K3
CO4	Analyze Synchronous Sequential circuits.	K4
CO5	Summarize the various types of memory devices.	K1
CO6	Design the Combinational circuits using Programmable Logic Devices.	K3

Knowledge Level	Description
K6	Create
K5	Evaluate
K4	Analyze
K3	Apply
K2	Understand
K1	Remember

6. CO – PO /PSO Mapping Matrix

CO	PO 1	PO 2	PO 3	PO 4	PO 5	PO 6	PO 7	PO 8	PO 9	PO 10	PO 11	PO 12	PS O1	PS O2	PS 03
1	3	2	1	1	-	-	-	-	-	-	-	-	3	-	-
2	3	3	2	2	-	-	-	-	-	-	-	-	3	-	-
3	3	3	2	2	-	-	-	-	-	-	-	-	3	-	-
4	3	3	2	2	-	-	-	-	-	-	-	-	3	-	-
5	3	3	2	2	-	-	-	-	-	-	-	-	3	-	-
6	3	2	1	1	-	-	-	-	-	-	-	-	3		

7. Lecture Plan – Unit 1 - BOOLEAN ALGEBRA AND LOGIC GATES

Sl. No.	Topic	Number of Periods	Proposed Date	Actual Lecture Date	CO	Taxonomy Level	Mode of Delivery
1	Review of number systems-representation-conversions	2	07.11.22 08.11.22		CO1	K1	Chalk & Board/ PPT
2	Review of Boolean algebra – theorems	1	09.11.22		CO1	K1	Chalk & Board/ PPT
3	sum of product and product of sum simplification	2	10.11.22 11.11.22		CO1	K1	Chalk & Board/ PPT
4	Canonical Forms	1	12.11.22		CO1	K1	Chalk & Board/ PPT
5	min term and max term	1	14.11.22		CO1	K1	Chalk & Board/ PPT
6	Simplification of Boolean expressions-Karnaugh map	2	15.11.22		CO1	K1	Chalk & Board/ PPT
7	Implementation of Boolean expressions using logic gates and universal gates	1	16.11.22		CO1	K1	Chalk & Board/ PPT

8.Activity Based Learning

1. Quiz

1. The largest two digit hexadecimal number is __________
a) $(FE)_{16}$
b) $(FD)_{16}$
c)$(FF)_{16}$
d)$(EF)_{16}$

2.Representation of hexadecimal number $(6DE)_{16}$ in decimal:
a) $6 \times 16^2 + 13 \times 16^1 + 14 \times 16^0$
b) $6 \times 16^2 + 12 \times 16^1 + 13 \times 16^0$
c) $6 \times 16^2 + 11 \times 16^1 + 14 \times 16^0$
d) $6 \times 16^2 + 14 \times 16^1 + 15 \times 16^0$

3.Octal to binary conversion: $(24)_8$ =?
a) $(111101)_2$
b) $(010100)_2$
c) $(111100)_2$
d) $(101010)_2$

4.1's complement of 1011101 is _______________
a) 0101110
b) 1001101
c) 0100010
d) 1100101

5.2's complement of 11001011 is _______________
a) 01010111
b) 11010100
c) 00110101
d) 11100010

6. 1's complement can be easily obtained by using ___________
a) Comparator
b) Inverter
c) Adder
d) Subtractor

7.The decimal number 10 is represented in its BCD form as ____________
a) 10100000
b) 01010111
c) 00010000
d) 00101011

8.A three digit decimal number requires___________for representation in the conventional BCD format.
a) 3 bits
b) 6 bits
c) 12 bits
d) 24 bits
9. The code where all successive numbers differ from their preceding number by
a) single bit is
b) Alphanumeric Code
c) BCD
d) Excess-3
e) Gray

10. According to Boolean law: A + 1 = ?
a) 1
b) A
c) 0
d) A′

11. The involution of A is equal to ____________
a) A
b) A′
c) 1
d) 0

12. DeMorgan's theorem states that ____________
a) (AB)′ = A′ + B′
b) (A + B)′ = A′ * B
c) A′ + B′ = A′B′
d) (AB)′ = A′ + B

13. Which gate is represented by the following truth table?

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

The gate is ___________
a) EX-OR
b) EX-NOR
c)AND
d)NOR

14.The output of an EX-OR gate is 1. Which input combination is correct? a) A = 1, B = 1
b) A = 0, B = 1
c) A = 0, B = 0

15. What is the minimum number of two input NAND gates used to perform the function of two input OR gates?
a) One
b) Two
c)Three
d)Four

16. How many AND gates are required to realize Y = CD + EF + G?
a) 4
b) 5
c) 3
d) 2

17. There are________cells in a 4-variable K-map.
a) 12
b) 16
c) 18
d) 8

18. Which of the following is not a positional number system?
a) Roman Number System
b) Octal Number System
c) Binary Number System
d) Hexadecimal Number System

Activity Based Learning

19. The weights used in Binary coded decimal code are:
a) 4,2,1
b) 8,4,2,1
c) 6,4,2,1
d) 2,1

20. The 1's complement of 1 in 4 bits is __________
a) 0001
b) 0
c) 1001
d) 1110

Activity Based Learning

1. Quiz - Answers

1. Answer: c
Explanation: $(FE)_{16}$ is 254 in decimal system, while $(FD)_{16}$ is 253. $(EF)_{16}$ is 239 in decimal system. And, $(FF)_{16}$ is 255. Thus, The largest two-digit hexadecimal number is $(FF)_{16}$.
2.Answer: a
Explanation: Hexadecimal to Decimal conversion is obtained by multiplying 16 to the power of base index along with the value at that index position.
In hexadecimal number D & E represents 13 & 14 respectively.
So, 6DE = $6 \times 16^2 + 13 \times 16^1 + 14 \times 16^0$.

3.Answer: b
Explanation: Each digit of the octal number is expressed in terms of group of 3 bits. Thus, the binary equivalent of the octal number is obtained.
$(24)_8 = (010100)_2$
4. Answer: c
Explanation: 1's complement of a binary number is obtained by reversing the binary bits. All the 1's to 0's and 0's to 1's.
Thus, 1's complement of 1011101 = 0100010.
5.Answer: c
Explanation: 2's complement of a binary number is obtained by finding the 1's complement of the number and then adding 1 to it.
2's complement of 11001011 = 00110100 + 1 = 00110101.

6. Answer: b
Explanation: With the help of inverter the 1's complement is easily obtained. Since, during the operation of 1's complement 1 is converted into 0 and vice- versa and this is well suited for the inverter.

7. Answer: c
Explanation: The decimal number 10 is represented in its BCD form as 0001 0000, in accordance to 8421 for each of the two digits.

8. Answer: c
Explanation: The number of bits needed to represent a given decimal number is always greater than the number of bits required for a straight binary encoding of the same. Hence, a three digit decimal number requires 12 bits for representation in BCD format.

Activity Based Learning

9. Answer: d
Explanation: The code where all successive numbers differ from their preceding number by single bit is gray code. It is an unweighted code. The most important characteristic of this code is that only a single bit change occurs when going from one code number to next. BCD Code is one in which decimal digits are represented by a group of 4-bits each, whereas, in Excess-3 Code, the decimal numbers are incremented by 3 and then written in their BCD format.

10. Answer: a
Explanation: $A + 1 = 1$, as per 1's Property.

11. Answer: a
Explanation: The involution of A means double inversion of A (i.e. A'') and is equal to A.
Proof: $((A)')' = A$

12. Answer: a
Explanation: The DeMorgan's law states that $(AB)' = A' + B'$ & $(A + B)' = A' * B'$, as per the Dual Property.

13. Answer: b
Explanation: The output of a logic gate is 1 when all inputs are at logic 0. The gate is NOR. The output of a logic gate is 1 when all inputs are at logic 0 or all inputs are at logic 1, then it is EX-NOR. (The truth tables for NOR and EX-NOR Gates are shown in above table).

14. Answer: b
Explanation: The output of EX-NOR gate is given by $AB + A'B'$. So, for $A = 0$ and $B = 0$ the output will be 1.
15. Answer: c
Explanation: $Y = A + B$. This is the equation of OR gate. We require 3 NAND gates to create OR gate. We can also write,1st, 2nd and 3rd NAND operations as: $Y = ((\text{NOT } A) \text{ AND } (\text{NOT } B))' = A'' + B'' = (A+B)$.

16. Answer: d
Explanation: To realize $Y = CD + EF + G$, two AND gates are required and two OR gates are required

Activity Based Learning

17. Answer: b
Explanation: There are 16 = (2^4) cells in a 4-variable K-map.
(2 Variables → 4 squares;
3 Variables → 8 squares;
4 Variables →16 squares;)

18. Answer: a
Explanation: The Roman number system isn't a positional number system since it uses symbols to represent numbers.
The octal number system uses digits from 0-7, the binary number system uses digits from 0-1 whereas, the hexadecimal number system uses digits from 0-15.

19. Answer: b
Explanation: BCD is a weighted code and it uses the weights 8,4,2,1 respectively. It is often called the 8421 code.
Since, it uses 4 bits for the representation therefore the weights are assigned as $2^3 = 8$, $2^2 = 4$, $2^1 = 2$, $2^0 = 1$.

20. Answer: d
Explanation: 1's complement is obtained by reversing the bits from 0 to 1 and vice-versa.
Binary of 1 is : 0001 and 1's complement is : 1110.

Activity Based Learning

2. Mind Map Activity

Create a Mind Map to describe the procedure of NAND/NOR implementation using the following tool:

https://www.apowersoft.us/mindmap?gclid=EAIaIQobChMIzLbL677i6gIVwQ0rCh2_SAeKEAAYAiAAEgIi5_D_BwE

3. Jigsaw Puzzle – Boolean Theorems

Learn boolean theorems through jigsaw activity

https://www.jigsawplanet.com/?rc=play&pid=25d6b6a021b3

4. Online Logic Circuit Simulator

Design and Simulate various logic gates, flip-flops, Integrated Circuits using the following online open source tool:

https://logic.ly/

9.Lecture Notes – Unit 1

UNIT I BOOLEAN ALGEBRA AND LOGIC GATES

Sl. No.	Contents
1	Digital Systems – Introduction; Advantages; Disadvantages; Applications
2	Number Systems
3	Number System Conversions: • To decimal (from binary; octal; hexadecimal; any base) • From decimal (to binary; octal; hexadecimal; any base) • To binary (from octal; hexadecimal) • From binary (to octal; hexadecimal) • Octal to hexa and vice versa • Other number system conversions – any base • Fractional conversions
4	Binary Arithmetic
5	Signed Binary Number Representation
6	Complements: 1’s; 2’s; 9’s & 10’s complement
7	Binary Arithmetic: Subtraction using 1’s; 2’s; 9’s & 10’s complement
8	Decimal Arithmetic using Complements
9	BCD Addition; BCD Subtraction (using 9’s & 10’s complement)
10	Binary Codes – Classification; examples
11	Boolean Algebra; Theorems and Properties of Boolean Algebra
12	Boolean functions; Simplification of Boolean functions using theorems
13	Canonical and Standard Forms
14	Converting Boolean functions to Standard SOP/POS form

UNIT I BOOLEAN ALGEBRA AND LOGIC GATES

1. Digital Systems

Introduction

Analog

• Information is represented by continuously varying values such as spatial position, voltage, etc. • Analog components have infinite number of possible values. • Discrete steps are not possible. Small change in one implies small change in another.	Analog Thermometer

Digital

• Digital describes electronic technology that generates, stores, and processes data in terms of two states: Positive & Non-positive. • Information is represented as numbers. • Digital components have discrete values.	Digital Thermometer

Digital Systems

- Digital System is a interconnection of digital modules.
- It is designed to manipulate logical information or physical quantities which are represented in digital form.
  - Ex: Digital computers, Digital cameras
- Signals are represented in binary form ('0'or '1')
- All communications within the system are carried out in digital manner.

1. Digital Systems

Advantages of digital systems

1. Easy to design: Since the circuits used in a digital system are switching circuits and there is no need of exact values of voltage and current.
2. Storage: The storage of digital information is easy.
3. Precision: Can increase the number of digits simply by adding switching circuits.
4. Noise: Less affected by noise, since can easily distinguish the binary ‘0’ and binary ‘1’ signal.
5. In digital IC’s, a large number of digital circuits can be incorporated compared to analog ICs.
6. Simple, inexpensive and safe implementation of complex data processing algorithms.
7. Simple design and implementation procedures.
8. High immunity to noise. Each digital component behaves as a filter eliminating the noise on its inputs.

Disadvantages of digital systems

1. All physical quantities are analog in nature.
2. Large number of transistors are required for performing some operations, that can be implemented using simpler analog hardware.
3. In some situations, the operation speed is lower than the speed offered by equivalent analog circuits.

Number System

System of naming or representing numbers. There are various types of number system.

1. Binary Number system – Base or radix 2
2. Decimal Number System – Base or radix 10
3. Octal Number System – Base or radix 8
4. Hexadecimal Number System – Base or radix 16.

1.1 Review of Number Systems – Representation

Number Systems

In digital electronics, the number systems are used for representing information. The number system has different bases and most common of them are **decimal, binary, octal** and **hexadecimal.**

A digital system can understand positional number system only where there are a few symbols called digits and these symbols represent different values depending on the position they occupy in the number.

A value of each digit in a number can be determined using

- The digit
- The position of the digit in the number
- The base of the number system (where base is defined as the total number of digits available in the number system)

Decimal Number system:

Decimal Number system has ***base 10 as it uses 10 digits from 0 to 9***.

- The number system that we use in our day-to-day life is the decimal number system.
- In decimal number system, the successive positions to the left of the decimal point represents units, tens, hundreds, thousands and so on. Each position represents a specific power of the base (10).
- For example, the decimal number 1234 consists of the digit 4 in the units position, 3 in the tens position, 2 in the hundreds position, and 1 in the thousands position, and its value can be written as,
- $(1\times1000) + (2\times100) + (3\times10) + (4\times l)$
- $(1\times10^3) + (2\times10^2) + (3\times10^1) + (4\times l0^0)$
- 1000 + 200 + 30 + 1
- 1234

1.1 Review of Number Systems – Representation

Binary Number system:

Base 2. Digits used in binary number system are 0 and 1

- Uses two digits, 0 and 1. Also called base 2 number system
- Each position in a binary number represents a 0 power of the base 2. Example: 2^0
- Last position in a binary number represents an x power of the base (2). Example: 2^x where x represents the last position – 1

Octal Number System:

Base 8. Digits used in octal number system are 0 to 7

- Uses eight digits, 0,1,2,3,4,5,6,7. Also called base 2 number system
- Each position in a Octal number represents a 0 power of the base 8. Example: 8^0
- Last position in a Octal number represents an x power of the base (8). Example: 8^x where x represents the last position – 1

Hexadecimal Number System:

Base 16. Digits used in hexadecimal number system are 0 to 9 and letters from A to F

- Uses 10 digits and 6 letters, 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
- Letters represents numbers starting from 10.
  A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.
- Also called base 16 number system.
- Each position in a hexadecimal number represents a 0 power of the base (16). Example 16^0
- Last position in a hexadecimal number represents an x power of the base (16). Example 16^x where x represents the last position - 1.

1.1 Review of Number Systems – Representation

Number System	Description
DECIMAL	Base 10. Digits used: **0 to 9**
BINARY	Base 2. Digits used: **0,1**
OCTAL	Base 8. Digits used: **0 to 7**
HEXA DECIMAL	Base 16. Digits used: **0 to 9**, Letters used: **A- F**

Number Systems – Representation

DECIMAL (BASE 10)	BINARY (BASE 2)	OCTAL (BASE 8)	HEXA DECIMAL (BASE 16)
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F
16	1 0000	20	10
17	1 0001	21	11
18	1 0010	22	12
19	1 0011	23	13
20	1 0100	24	14
21	1 0101	25	15
22	1 0110	26	16

Number Systems – Representation

DECIMAL (BASE 10)	BINARY (BASE 2)	OCTAL (BASE 8)	HEXA DECIMAL (BASE 16)
23	1 0111	27	17
24	1 1000	30	18
25	1 1001	31	19
26	1 1010	32	1A
27	1 1011	33	1B
28	1 1100	34	1C
29	1 1101	35	1D
30	1 1110	36	1E
31	1 1111	37	1F
32	10 0000	40	20
33	10 0001	41	21
34	10 0010	42	22
35	10 0011	43	23
36	10 0100	44	24
37	10 0101	45	25
38	10 0110	46	26
39	10 0111	47	27
40	10 1000	50	28
41	10 1001	51	29
42	10 1010	52	2A
43	10 1011	53	2B
44	10 1100	54	2C
45	10 1101	55	2D

R.M.K GROUP OF INSTITUTIONS

Conversion in Number Systems

DECIMAL TO OTHER BASE SYSTEM

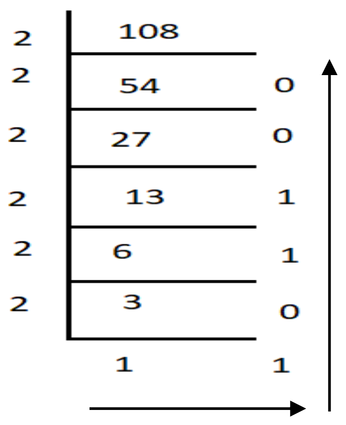
- ❖ **Step 1** – Divide the decimal number to be converted by the value of the new base.
- ❖ **Step 2** – Get the remainder from Step 1 as the rightmost digit (least significant digit) of new base number.
- ❖ **Step 3** – Divide the quotient of the previous divide by the new base.
- ❖ **Step 4** – Record the remainder from Step 3 as the next digit (to the left) of the new base number.
- ❖ Repeat Steps 3 and 4, getting remainders from right to left, until the quotient becomes zero in Step 3.
- ❖ The last remainder thus obtained will be the Most Significant Digit (MSD) of the new base number.

Conversion from Decimal To

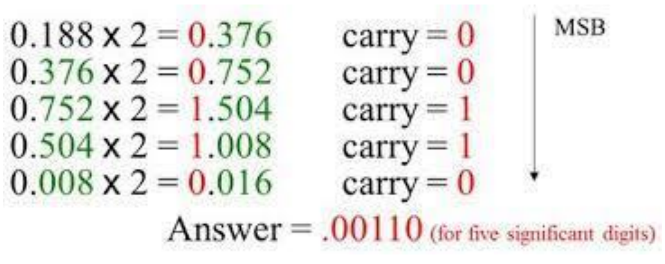
- ✓ **Binary**
- ✓ **Octal**
- ✓ **Hexadecimal**

✓ **1.Decimal to Binary Conversion**

Convert 108.188_{10} into binary number.



$108_{10} = 1101100_2$



$0.188 \times 2 = 0.376$ carry = 0
$0.376 \times 2 = 0.752$ carry = 0
$0.752 \times 2 = 1.504$ carry = 1
$0.504 \times 2 = 1.008$ carry = 1
$0.008 \times 2 = 0.016$ carry = 0

MSB

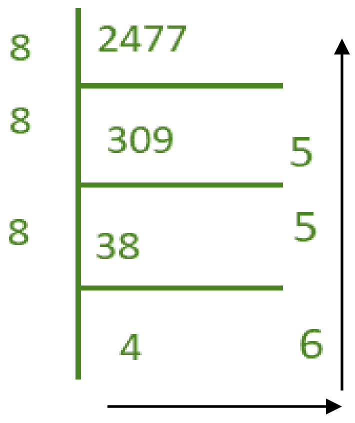
Answer = .00110 (for five significant digits)

$.188_{10} = .00110_2$

$$108.188_{10} = 1101100.00110_2$$

✓ 2.Decimal to Octal Conversion

Convert 2477.64_{10} into octal number.



$0.64 \times 8 = 5.12$

$0.12 \times 8 = 0.96$

$0.96 \times 8 = 7.68$

$0.68 \times 8 = 5.44$

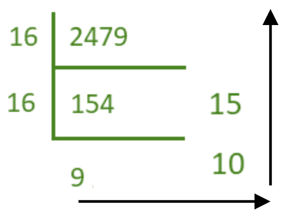
$2477_{10} = 4655_8$

$.64_{10} = .5075_8$ (approx.)

$$2477.64_{10} = 4655.5075_8 \text{ (approx.)}$$

✓ 3.Decimal to Hexadecimal Conversion

Convert 2479.565_{10} into Hexadecimal number.



$0.565 \times 16 = 9.04$	9
$0.04 \times 16 = 0.64$	0
$0.64 \times 16 = 10.24$	10 = A
$0.24 \times 16 = 3.84$	3
$0.84 \times 16 = 13.44$	13 = D
$0.44 \times 16 = 7.04$	7
$0.04 \times 16 = 0.64$	0

$(0.565)_{10} = (0.90A3D70...)_{16}$

$2479_{10} = 9AF_{16}$

$.565_{10} = .90A3D70_{16}$

$$2479.565_{10} = 9AF.90A3D70_{16}$$

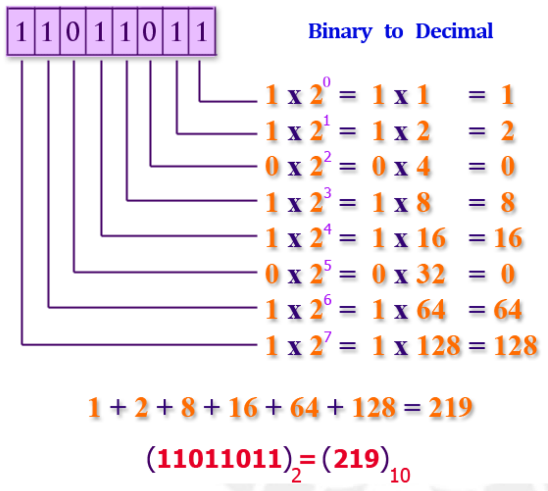
BINARY TO OTHER BASE SYSTEM

Conversion from Binary To

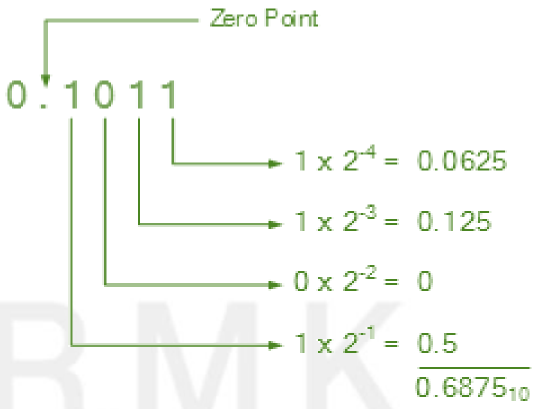
- ✓ **Decimal**
- ✓ **Octal**
- ✓ **Hexadecimal**

✓ 1.Binary to Decimal Conversion

Convert 11011011.1011_2 into decimal number.



1 1 0 1 1 0 1 1
Binary to Decimal
$1 \times 2^0 = 1 \times 1 = 1$
$1 \times 2^1 = 1 \times 2 = 2$
$0 \times 2^2 = 0 \times 4 = 0$
$1 \times 2^3 = 1 \times 8 = 8$
$1 \times 2^4 = 1 \times 16 = 16$
$0 \times 2^5 = 0 \times 32 = 0$
$1 \times 2^6 = 1 \times 64 = 64$
$1 \times 2^7 = 1 \times 128 = 128$
$1 + 2 + 8 + 16 + 64 + 128 = 219$
$(11011011)_2 = (219)_{10}$



$11011011_2 = 219_{10}$ $.1011_2 = .6875_{10}$

$$11011011.1011_2 = 219.6875_{10}$$

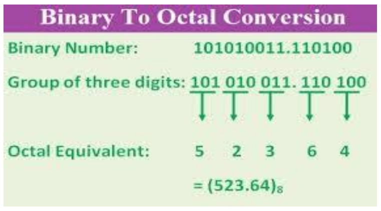
✓ 2.Binary to Octal Conversion

Convert 1010111100.1011_2 into Octal number.

001	010	111	100	. 101	100
1	**2**	**7**	**4**	**. 5**	**4**

$0110011_2 = 1274_8$ $.1011_2 = .54_8$

$$1010111100.1011_2 = 1274.54_8$$


Binary To Octal Conversion

Binary Number: 101010011.110100

Group of three digits: 101 010 011. 110 100

Octal Equivalent: 5 2 3 6 4

$= (523.64)_8$

✓ 3.Binary to Hexadecimal Conversion

Convert $1111101110111 0010.01110101010111 1_2$ into Hexadecimal number.

0001	1111	0111	0111	0010		0111	0101	0101	111**0**
1	F	7	7	2	.	6	5	5	E

$$11111011101110010.011101010101111_2 = 1F772.655E_H$$

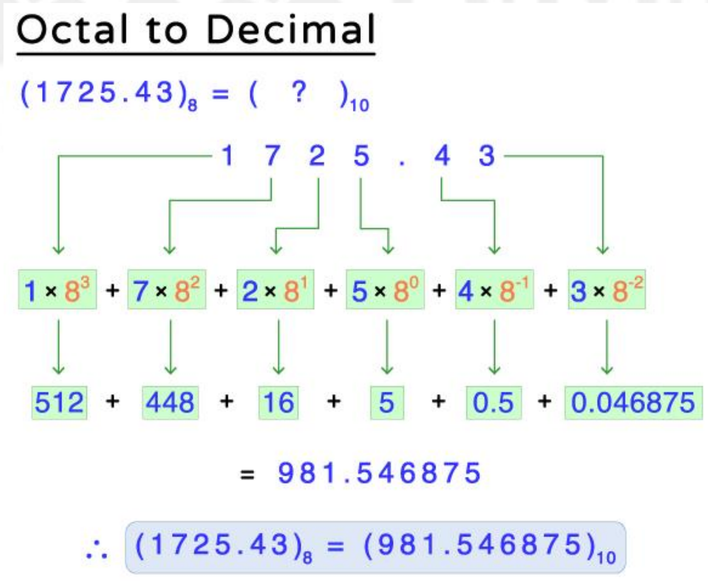
OCTAL TO OTHER BASE SYSTEM

Conversion from Octal To

- ✓ **Decimal**
- ✓ **Binary**
- ✓ **Hexadecimal**

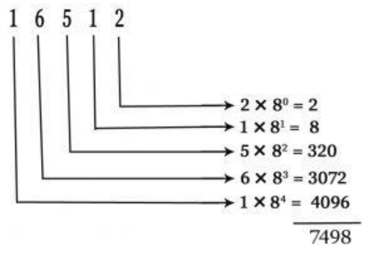
1.Octal to Decimal Conversion

Convert 1725.43_8 into decimal number.



Octal to Decimal
$(1725.43)_8 = (\ ?\ )_{10}$
1 7 2 5 . 4 3
$1\times8^3 + 7\times8^2 + 2\times8^1 + 5\times8^0 + 4\times8^{-1} + 3\times8^{-2}$
512 + 448 + 16 + 5 + 0.5 + 0.046875
= 981.546875
$\therefore (1725.43)_8 = (981.546875)_{10}$

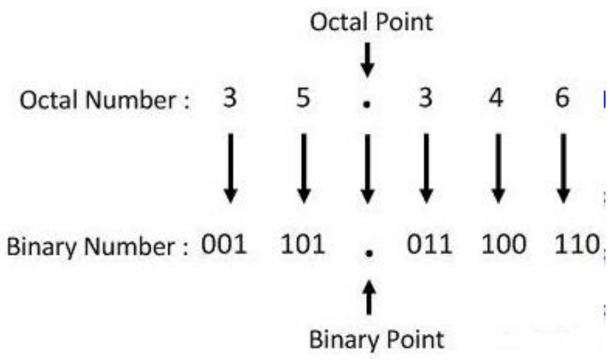
$$1725.43_8 = 981.546875_{10}$$



$$16512_8 = 7498_{10}$$

✓ 2.Octal to Binary Conversion

Convert 35.346_8 into Binary number.



Octal Point
Octal Number : 3 5 . 3 4 6
Binary Number : 001 101 . 011 100 110
Binary Point

$$35.346_8 = \mathbf{00}1101.01110011\mathbf{0}_2$$

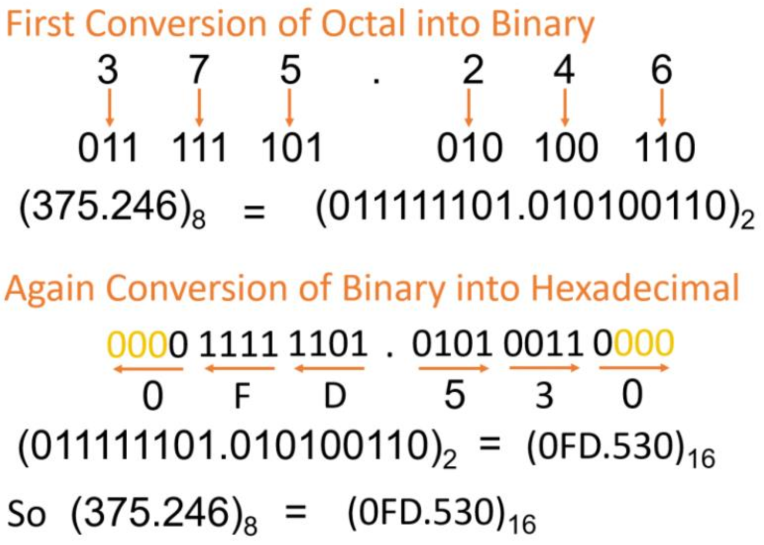
✓ 3.Octal to Hexadecimal Conversion

The following two ways to convert octal to hexadecimal number-

One: First octal to decimal then decimal to hexadecimal

Two: First octal to binary then binary to hexadecimal **(Easy method)**

Convert $(375.246)_8$ into Hexadecimal number.



First Conversion of Octal into Binary
3 7 5 . 2 4 6
011 111 101 010 100 110
(375.246)8 = (011111101.010100110)2
Again Conversion of Binary into Hexadecimal
0000 1111 1101 . 0101 0011 0000
0 F D 5 3 0
(011111101.010100110)2 = (0FD.530)16
So (375.246)8 = (0FD.530)16

HEXADECIMAL TO OTHER BASE SYSTEM

Conversion from Hexadecimal To

- ✓ **Decimal**
- ✓ **Binary**
- ✓ **Hexadecimal**

✓ 1.Hexadecimal to Decimal Conversion

Convert $54.D2_{16}$ into decimal number.

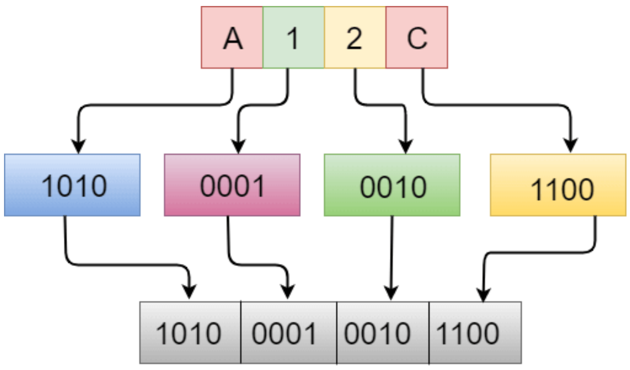
Digit	5	4	D	2
Place value	16^1	16^0	16^{-1}	16^{-2}

$$
\begin{aligned}
&54.D2_{16}\\
&= 5 \cdot 16^1 + 4 \cdot 16^0 + D \cdot 16^{-1} + 2 \cdot 16^{-2}\\
&= 5 \cdot 16^1 + 4 \cdot 16^0 + 13 \cdot 16^{-1} + 2 \cdot 16^{-2}\\
&= 80 + 4 + 0.8125 + 0.0078125\\
&= 84.8203125
\end{aligned}
$$

$$54.D2_{16} = 84.8203125_{10}$$

✓ 2. Hexadecimal to Binary Conversion

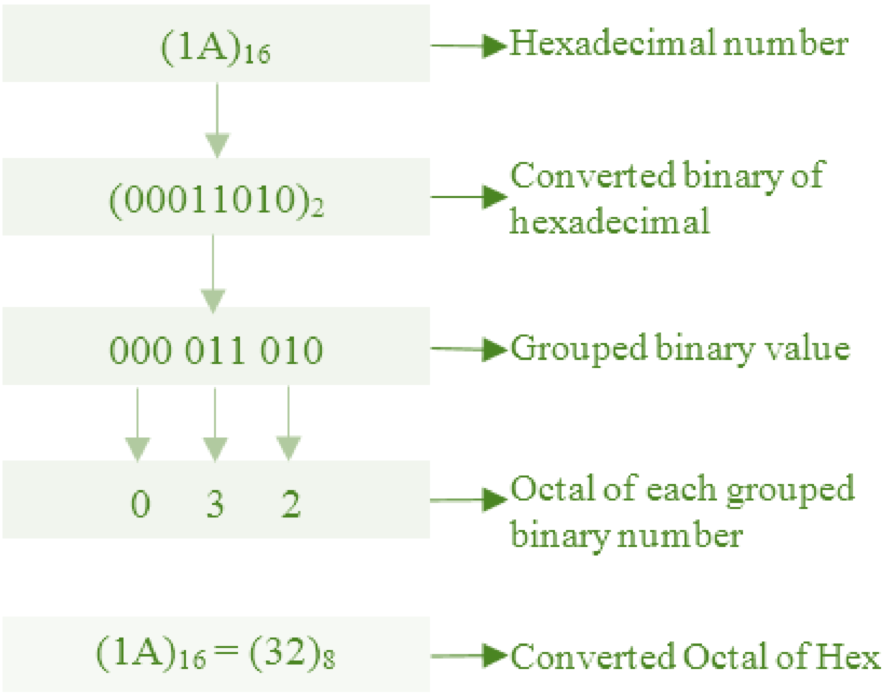
Convert $A12C_{16}$ into binary number.



$$A12C_{16} = 1010000100101100_2$$

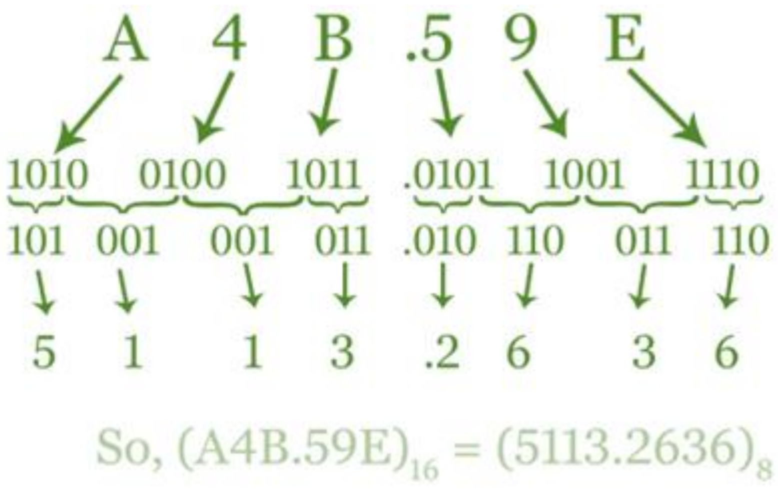
✓ 3. Hexadecimal to Octal Conversion

Convert $1A_{16}$ into Octal number.



$1A_{16} = 32_8$

Convert $A4B.59E_{16}$ into Octal number



$A4B.59E_{16} = 5113.2636_8$

Binary Arithmetic

In computers, numbers are represented in binary form. The arithmetic operations performed by a computer therefore involves binary numbers. Binary arithmetic is essential part of all the digital computers and many other digital system. In binary number system there are only 2 digits 0 and 1, and any number can be represented by these two digits.

Example: Electricity is Either On or Off.

1=On

0=Off

Types of Binary Arithmetic Operations

Arithmetic in binary is much like arithmetic in other numeral systems.

- Addition
- Subtraction
- Multiplication
- Division

Binary arithmetic operation starts from the least significant bit i.e. from the right most side.

Binary Addition

It is a key for binary subtraction, multiplication, division. The four rules of binary addition are as follows.

- 0 + 0 = 0
- 0 + 1 = 1
- 1 + 0 = 1
- 1 + 1 = 0 (carry 1 to the next significant bit)

Example Perform Addition

```
            1   1   1                        1   1    1   1

i) 13=     1  1  0  1₂             ii)     1   1   1   1   0₂
    5=     0  1  0  1₂                +         1   1   0   0₂
          ------------------------------        -------------------------
          1   0  0  1  0₂                       1  0   1   0   1   0₂
```

$i)\ 13 = 1101_2,\quad 5 = 0101_2,\quad 1101_2 + 0101_2 = 10010_2$

$ii)\ 11110_2 + 1100_2 = 101010_2$

Binary Arithmetic

Binary Subtraction

Normally the borrow method is used. The four rules of binary subtraction are as follows

- 0 – 0 = 0
- 0 – 1 = 1, borrow 1 from the next more significant bit
- 1 – 0 = 1
- 1 – 1 = 0

Example: if a = 1101_2 , b = 101_2 , find a - b

$$
\begin{array}{rcccc}
13= & 1 & 1 & 0 & 1_2 \\
-\ 5= & 0 & 1 & 0 & 1_2 \\
\hline
 & 1 & 0 & 0 & 0_2
\end{array}
$$

Example: if a = 10110_2 , b = 1001_2 , find a - b

$$
\begin{array}{rccccc}
 & 0 & 10 & & 0 & 10 \\
 & \cancel{1} & 0 & 1 & \cancel{1} & 0_2 \\
- & & 1 & 0 & 0 & 1_2 \\
\hline
 & 0 & 0 & 1 & 0 & 1_2
\end{array}
$$

Example: if a = 1100110_2 , b = 111101_2 , find a - b

$$
\begin{array}{rccccccc}
 & & 10 & & & & & \\
 & 0 & \cancel{0} & 1 & 10 & & 0 & 10 \\
 & & & \cancel{10} & & & & \\
 & \cancel{1} & \cancel{1} & \cancel{0} & \cancel{0} & 1 & \cancel{1} & \cancel{0}_2 \\
- & & 1 & 1 & 1 & 1 & 0 & 1_2 \\
\hline
 & & 1 & 0 & 1 & 0 & 0 & 1_2
\end{array}
$$

Binary Arithmetic

Binary Multiplication

Binary multiplication may sound like it would be more difficult than binary addition or subtraction – but is actually a simple process. The rules are as follows.

- 0×0=0
- 1×0=0
- 0×1=0
- 1×1=1 (there is no carry or borrow for this)

When performing binary multiplication, remember the following rules.

1.Copy the multiplicand when the multiplier digit is 1. Otherwise, write a row of zeros.

2.Shift the results one column to the left for a new multiplier digit.

3.Add the results using binary addition to find the product.

Example: if a = 100001_2 , b = 101_2 , find a x b

```
         1 0 0 0 0 1
               1 0 1
      ---------------
         1 0 0 0 0 1
       0 0 0 0 0 0 +
     1 0 0 0 0 1 + +
    -----------------
     1 0 1 0 0 1 0 1_2
     -----------------
```

Example: if a = 26_{10} , b = 3_{10} , find a x b

```
           1 1 0 1 0
                 1 1
      ---------------

         1
           1 1 0 1 0
         1 1 0 1 0 +
   -----------------
       1 0 0 1 1 1 0_2
    ---------------------
```

-

Binary Arithmetic

Binary Division

Division of binary numbers uses the same technique as division in the decimal system.

- 0/0=0
- 1/0=(indefinte)
- 0/1=0
- 1/1=1 (there is no carry or borrow for this)

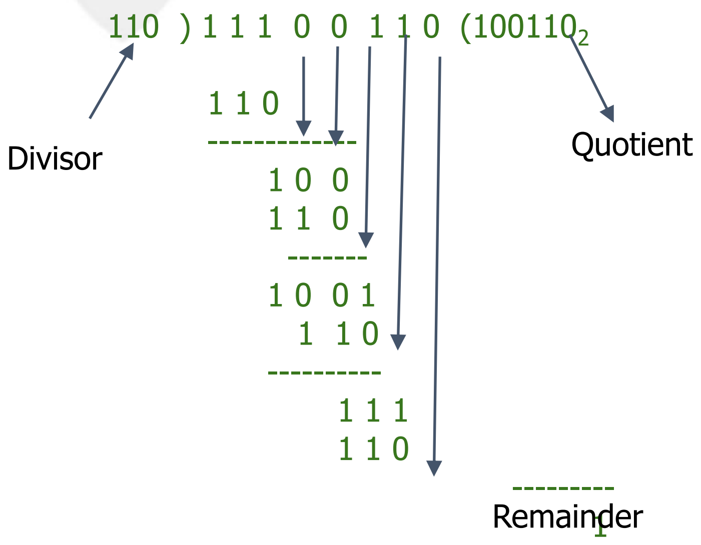
When doing binary division, some important rules need to be remembered.

(a) When the remainder is greater than or equal to the divisor, write a 1 in the quotient and subtract.

(b) When the remainder is less than the divisor, write a 0 in the quotient and add another digit from the dividend.

(c) If all the digits of the dividend have been considered and there is still a remainder, mark a radix point in the dividend and append a zero. Remember that some fractions do not have an exact representation in binary, so not all division problems will terminate.

Example: if a = 101101_2 , b = 101_2 , find a / b



Note:Perform subtraction using complements

Exercise Problems

1. Add and multiply the following numbers without converting to decimal.[CO1,K2]

   (a) $(367)_8$ and $(715)_8$.

   (b) $(15F)_{16}$ and $(A7)_{16}$

   (c) (110110), and $(\text{IlOIOI})_2$

2.

```
  10010          1011101          10011
+1100          + 1000000        + 1111101
------         ---------        ---------


  10011001       11000011         1001100
+   100111     +   101111       + 1100101
---------      ---------        ---------
```

3. If the numbers sixteen and nine are added in binary form, will the answer be any different than if the same quantities are added in decimal form? Explain.

Ans:

Unsigned and Signed Binary Numbers

Variables such as integers can be represent in two ways

- Signed
- Unsigned.

Signed numbers use sign flag or can be distinguish between negative values and positive values.

unsigned numbers stores only positive numbers but not negative numbers.

To represent negative integers, it requires a notation for negative values.

In ordinary arithmetic,

- Negative number is indicated by a (-) sign
- Positive number by a (+) sign.

With hardware limitations, computers represents everything with binary digits, commonly referred to as bits. The convention is to make the sign bit

- 0 for positive
- 1 for negative.

For example,

the string of bits 01001 represents

- 9 (unsigned binary) or
- +9 (signed binary) ; leftmost bit is 0.

*the string of bits **1**1001 represent the*

- 25 (unsigned binary) or
- - 9 (signed binary) ; leftmost bit is 1.

In each case: left-most bit indicates sign: positive (0) or negative (1).

Can't include the *sign bit* in 'Addition'

Sign	Magnitude	Decimal Equivalent
0	0001110	14_{10}
1	0001110	-14_{10}

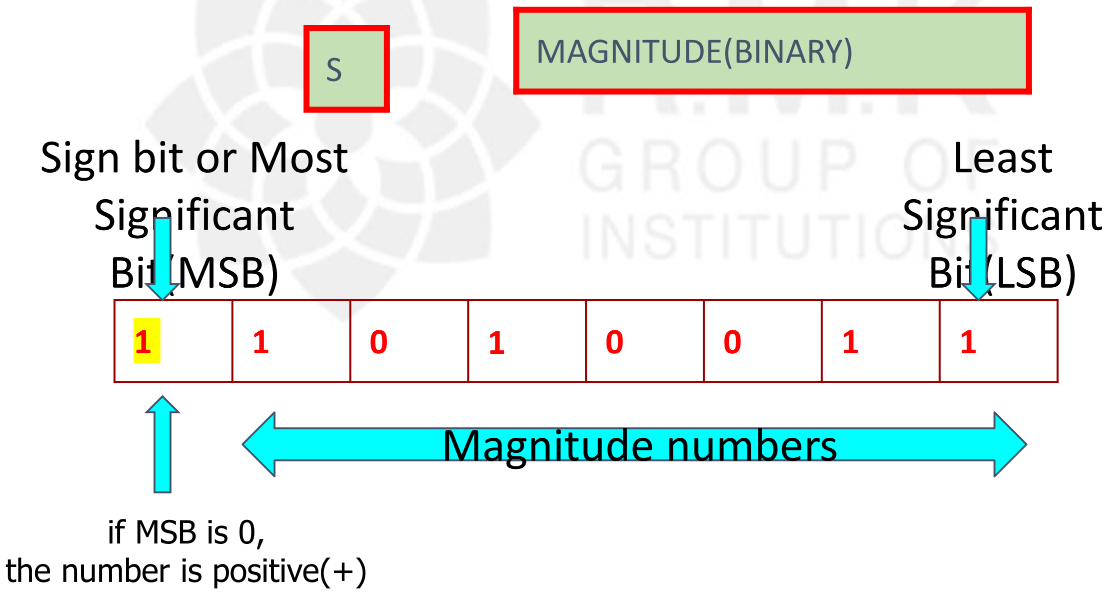
Representation of Signed Binary Numbers:

There are three types of representations for signed binary numbers. Because of extra signed bit, binary number zero has two representation, either positive (0) or negative (1)

- Sign-Magnitude form
- 1's complement form
- 2's complement form

Sign-Magnitude form

For n bit binary number, 1 bit is reserved for sign symbol. If the value of sign bit is 0, then the given number will be positive, else if the value of sign bit is 1, then the given number will be negative.


S
MAGNITUDE(BINARY)
Sign bit or Most Significant Bit (MSB)
Least Significant Bit (LSB)
1 1 0 1 0 0 1 1
Magnitude numbers
if MSB is 0,
the number is positive(+)

Examples

+0	0	00000
-0	1	00000
+31	0	11111
-31	1	11111

Complement Representation

Complements are used in digital computers to simplify the subtraction operation and for logical manipulation. There are Two types of complements for each base-r system:

- the diminished radix(r -1)'s complement
- the radix complement (r)'s complement

Binary Complement Representation

Complement for binary numbers are represented by two forms

- the diminished radix(2 -1)'s complement form
- the radix complement (2)'s complement form

1's complement form

The **1's complement** of a binary number is the number that results when we change all 1's to zeros and the zeros to ones. Complement Ranges from n-Bit Representation $-2^{n-1}+1 \leq \text{Number} \leq +2^{n-1} - 1$.where n is number of bits

Example Find 1's complement of binary number 10101110.

Simply invert each bit of given binary number, so 1's complement of given number will be 01010001.

1's Complementation in Signed Binary number Representation:

If the number is negative then it is represented using 1's complement. First represent the number with positive sign and then take 1's complement of that number.

Example Find 1's complement of -5_{10}

+5 is represented as it is represented in sign magnitude method. -5 is represented using the following steps:

(i) +5 = 0 0101

(ii) Take 1's complement of 0 0101 and that is ***1 1010.*** MSB is 1 which indicates that number is negative.

(iii) -5_{10} = ***1 1010*** ;; MSB is always 1 in case of negative numbers.

Complement Representation

Decimal	1's Complement
+ 7	0 1 1 1
+ 6	0 1 1 0
+ 5	0 1 0 1
+ 4	0 1 0 0
+ 3	0 0 1 1
+ 2	0 0 1 0
+ 1	0 0 0 1
+ 0	0 0 0 0
− 0	1 1 1 1
− 1	1 1 1 0
− 2	1 1 0 1
− 3	1 1 0 0
− 4	1 0 1 1
− 5	1 0 1 0
− 6	1 0 0 1
− 7	1 0 0 0

Advantages of 1's Complement:

•Still relatively simple to represent the numbers,

•Simpler Add/Subtract circuit design (subtracting a number from another involves complementing the subtracted and then adding it to the other number).

Disadvantages:

•Has the problem of double representing the 0 (-0 and +0),

•It may require two addition operations as if there is a carry out it has to be added to get the correct result

Overflow: occurs when the result is out of range

Complement Representation

2’s complement:

The **2’s complement** is the binary number that results when we add 1 to the 1’s complement. Complement Ranges from n-Bit Representation **$-2^{n-1} \leq$ Number $\leq +2^{n-1} - 1$.** **where** n is number of bits It is used to represent negative numbers.

4-Bit Representation

$2^4 = 16$ Combinations

$-8 \leq$ Number $\leq +7$

$-2^3 \leq$ Number $\leq +2^3 - 1$

2’s Complement=1’s Complement+1

Example Find 1’s complement of binary number 10101110.

Simply invert each bit of given binary number, so 1’s complement of given number will be ***01010010.***

2’s Complementation in Signed Binary number Representation:

If the number is negative then it is represented using 2’s complement. First represent the number with positive sign and then take 2’s complement of that number.

Example Find 2’s complement of -5_{10}

+5 is represented as it is represented in sign magnitude method. -5 is represented using the following steps:

(i) +5 = 0 0101

(ii) Take 2’s complement of 0 0101 and that is ***1 1011.*** MSB is 1 which indicates that number is negative.

(iii) -5_{10} = ***1 1011*** ;; MSB is always 1 in case of negative numbers.

Complement Representation

2’s complement:

Decimal	2’s Complement
+ 7	0 1 1 1
+ 6	0 1 1 0
+ 5	0 1 0 1
+ 4	0 1 0 0
+ 3	0 0 1 1
+ 2	0 0 1 0
+ 1	0 0 0 1
+ 0	0 0 0 0
− 1	1 1 1 1
− 2	1 1 1 0
− 3	1 1 0 1
− 4	1 1 0 0
− 5	1 0 1 1
− 6	1 0 1 0
− 7	1 0 0 1

Disadvantages of 2’s Complement:

•It is slightly more complex to obtain the 2's complement (it involves complementing and adding 1).

Overflow: occurs if the result is out of range

•Adding two positive numbers and getting a negative number

•Adding two negative numbers and getting a positive number

•If the last two carries are not equal

1’s and 2’s complements

1’s complements:

The one’s complement of a binary number is defined as the value obtained by inverting all the bits in the binary representation of the number. It behaves like a negative of the original number.

2’s complements:

2‘s complements of a binary number is 1 added to the 1’s complement of the binary number.

Problems:

1.Find 1’s complement for the given binary number 011001

Take complement of the given number

```
                011001
     Ans:       100110
```

2.Find 2’s complement for the given binary number 011001

Take 1’s complement and add 1 to the right most bit.

```
     100110
         +1
     100111
```

Ans: 100111

1’s complement subtraction:

1.Perform 1’s complement subtraction $10_{10} - 14_{10}$

Binary equivalent of 10 = 1010

Binary equivalent of 14 = 1110

```
Complement of 14 = 0001
Add 10                  = 1010

                          1011
```

Since there is no carry, take 1’s complement and prefix negative sign.
Hence the answer is   -0100

2.Perform 1's complement subtraction $14_{10} - 10_{10}$

Binary equivalent of 10 = 1010

Binary equivalent of 14 = 1110

```
      Complement of 10 = 0101
 Add 14                = 1110
                     1   0011
```

Since a carry is generated, add carry and write the answer.

```
                         0011
                            1
                  Answer 0100
```

2's complement subtraction

1.Perform 2's complement subtraction $10_{10} - 14_{10}$

Binary equivalent of 10 = 1010

Binary equivalent of 14 = 1110

```
  2's  Complement of 14 = 0001
                             1
   2's  Complement of 14= 0010
  Add 10                =  1010
                           1100
```

Since there is no carry, take 2's complement and prefix negative sign.

Hence the answer is -0100

2.Perform 2's complement subtraction $14_{10} - 10_{10}$

Binary equivalent of 10 = 1010

Binary equivalent of 14 = 1110

```
   2's  Complement of 10 = 0101
                              1
 2's Complement of 10     =  0110
    Add 14               =   1110
                       1     0100
```

Discard the carry and write the Answer

Hence, the answer is 0100.

- Perform 2’s complement subtraction $42_{10} - 68_{10}$

		64	32	16	8	4	2	1
Binary equation of 68	=	1	0	0	0	1	0	0
Binary equation of 42	=	0	1	0	1	0	1	0
2's complement of 68	=	0	1	1	1	0	1	1
Add 1								1
		0	1	1	1	1	0	0
Add 42		0	1	0	1	0	1	0
		1	1	0	0	1	1	0

No Carry.
Hence take 2' complement
and write the Answer.

	0	0	1	1	0	0	1	
							1	
−	0	0	1	1	0	1	0	Answer

Complement Representation

Decimal Complement Representation

Complement for binary numbers are represented by two forms

- **the diminished radix(10 -1)'s complement form**
- **the radix complement (10)'s complement form**

9 ’s complement form

The **9’s complement** of a decimal number is the subtraction of it's each digits from **9.**

Example Find 9’s complement of decimal number 456.

```
=     9 9 9
    - 4 5 6
   -------------
      5  4 3₁₀
   -------------
```

10 ’s complement form

The **10’s complement** of a decimal number is the adding 1 to the result of 9’s complement .

10’s complement=9’s Complement+1

Example Find 10’s complement of decimal number 456.

```
=     9 9 9
    - 4 5 6
    ----------
      5 4 3
    +       1
    ----------
      5 4 4₁₀
    ----------
```

Binary Arithmetic Using Complement

Subtraction using complement allow us to perform subtraction using addition.Thus only adder component is required and no need of separate subtractor component.

Subtraction using 1's complement

a) Subtraction of smaller number from a larger number:-

1. Calculate the 1's complement of a smaller number(subtrahend).

2. add the 1's complement to the larger number(minuend).

3. If carry comes in the MSB , **remove the carry and add it to the result**.

b) Subtraction of larger number from a smaller number:-

1.Calculate 1's complement of a larger number(subtrahend).

2. Add 1's complement in smaller number (minuend)

3. If no carry then the result will be in 1's complement form. Calculate 1's complement of final value and assign **–ve sign** to the result.

Advantages of using 1's complement subtraction

1. This can be easily obtained by simply inverting each bit in the number

2. This subtraction can be done with an binary adder. Thus ,it is useful in arithmetic logic circuits.

Binary Arithmetic Using Complement

Subtraction Using 1's Complement

Solve:10101_2 - 1001_2 = ?

Objective :

10101_2 - 1001_2 = ?

Input Data :

Binary Input 1 = 10101

Binary Input 2 = 1001

Work with Steps **:**

10101_2 + (- 01001_2)

i)Calculate the 1's complement of a subtrahend.

(- 1001_2)=>1's(**01001_2**)

= 10110

ii)add the 1's complement to the larger number(minuend).

```
   10101
 + 10110
 1 01011
```

$$\boxed{1\ \ 01011}$$

iii) If carry comes in the MSB , remove the carry and add it to the result.

```
   01011
 +     1
   01100
```

$$\boxed{01100}$$

Binary Arithmetic Using Complement

Subtraction Using 1's Complement

Solve:1001_2 - 10101_2 = ?

Objective :

1001_2 - 10101_2 = ?

Input Data :

Binary Input 1 = 1001

Binary Input 2 = 10101

Work with Steps **:**

01001_2 + (- 10101_2)

i)Calculate the 1's complement of a subtrahend.

(- 10101_2)=>1's(10101_2)

= 01010_2

ii)add the 1's complement to the larger number(minuend).

```
  01001
+ 01010
```

$\boxed{0\ \ 10011}$

iii) If no carry then the result will be in 1's complement form. Calculate 1's complement of final value and assign –ve sign to the result.

1's(10011)

$\boxed{-\ 01100}$

Binary Arithmetic Using Complement

Subtraction Using 2’s Complement:

Steps for Subtraction using 2’s complement

a) Subtraction of smaller number from a larger number:-

1. Calculate the 2’s complement of a smaller number.

2. add the 2’s complement to the larger number.

3. If carry comes in the MSB , discard the carry .

b) Subtraction of larger number from a smaller number:-

1.Calculate 2’s complement of a larger number.

2. Add 2’s complement in smaller number

3. The result will be in 2’s complement form. Calculate 2’s complement of final value and assign –ve sign to the result.

Advantages of 2’s Complement

It is optimally simple to implement adders that can do signed arithmetic when 2’s complement notation is used.

two's complement only has one value for zero

2's complement doesn't require carry values.

Binary Arithmetic Using Complement

Subtraction Using 2's Complement:

Solve: 10101_2 - 1001_2 = ?

Objective :

10101_2 - 1001_2 = ?

Input Data :

Binary Input 1 = 10101

Binary Input 2 = 1001

Work with Steps **:**

10101_2 + (- 01001_2)

i)Calculate the 2's complement of a subtrahend.

(- 1001_2)=>1's(**01001_2**)

= 10111

ii)add the 1's complement to the larger number(minuend).

```
   10101
+  10111
 1 01100
```

iii) If carry comes in the MSB , discard the carry.

$\boxed{01100}$

Binary Arithmetic Using Complement

Subtraction Using 2's Complement:

Solve: 1001_2 - 10101_2 = ?

Objective :

1001_2 - 10101_2 = ?

Input Data :

Binary Input 1 = 1001

Binary Input 2 = 10101

Work with Steps **:**

01001_2 + (- 10101_2)

i) Calculate the 2's complement of a subtrahend.

2's(- 10101_2)=>1's(10101_2)+1

= 01011_2

ii) add the 1's complement to the larger number(minuend).

```
  01001
+ 01011
0 10100
```

iii) If no carry then the result will be in 2's complement form. Calculate 2's complement of final value and assign –ve sign to the result.

2's(10100)

$\boxed{- 01100_2}$

Decimal Arithmetic Using Complement

Steps Subtraction using 9's complement

a) Subtraction of smaller number from a larger number:-

1. Calculate the 9's complement of a smaller number(subtrahend).
2. add the 9's complement to the larger number(minuend).
3. If carry comes in the MSB , remove the carry and add it to the result.

Example

b) Subtraction of larger number from a smaller number:-

1.Calculate 9's complement of a larger number(subtrahend).

2. Add 9's complement in smaller number (minuend)
3. If no carry then the result will be in 9's complement form. Calculate 9's complement of final value and assign –ve sign to the result.

Subtraction using 10's complement

a) Subtraction of smaller number from a larger number:-

1. Calculate the 10's complement of a smaller number.
2. add the 110's complement to the larger number.
3. If carry comes in the MSB , discard the carry .

b) Subtraction of larger number from a smaller number:-

1.Calculate 10's complement of a larger number.

2. Add 10's complement in smaller number
3. The result will be in 10's complement form. Calculate 10's complement of final value and assign –ve sign to the result.

Decimal Arithmetic Using Complement

Subtraction Using 9's Complement:

Solve:10101_2 - 1000_2 = ?

Objective :

$10101_2 - 1000_2 = ?= 21_{10} - 18_{10}$

Input Data :

Binary Input 1 = 21_{10}

Binary Input 2 = 18_{10}

Work with Steps **:**

$21_{10} + (- 18_{10})$

i)Calculate the 9's complement of a subtrahend.

$(- 18_{10}) \Rightarrow 9's(-18_{10})$

$= 9\ 9 - 1\ 8$

$= 8\ 1$

ii)add the 9's complement to the larger number(minuend).

$$\begin{array}{r} 2\ 1 \\ +\ \ 8\ 1 \\ \hline \boxed{1\ \ 0\ 2} \end{array}$$

iii) If carry comes in the MSB , Add carry to the result .

$$\begin{array}{r} 0\ \ 2 \\ +\ \ \ \ 1 \\ \hline \boxed{0\ \ 3_{10}} \end{array}$$

Decimal Arithmetic Using Complement

Subtraction Using 9's Complement:

Solve : 18_{10} - 21_{10}= ?

Objective :

18_{10} - 21_{10}= ?

Input Data :

Binary Input 1 = 18_{10}

Binary Input 2 = 21_{10}

Work with Steps **:**

18_{10} + (- 21_{10})

i)Calculate the 9's complement of a subtrahend.

(- 21_{10})=>9's(**-21_{10}**)

= 9 9 - 2 1

= 7 8

ii)add the 9's complement to the larger number(minuend).

$$\begin{array}{r} 1\ 8 \\ +\ \ 7\ 8 \\ \hline \boxed{0\quad 9\ 6} \end{array}$$

iii) If no carry then the result will be in 9's complement form. Calculate 9's complement of final value and assign –ve sign to the result.

$$\begin{array}{r} 9\ \ 9 \\ -\ \ 9\ \ 6 \\ \hline \boxed{-\ 0\ \ 3_{10}} \end{array}$$

Decimal Arithmetic Using Complement

Subtraction Using 10's Complement:

Solve:10101_2 - 1000_2 = ?

Objective :

$10101_2 - 1000_2$ = ?= 21_{10}- 18_{10}

Input Data :

Binary Input 1 = 21_{10}

Binary Input 2 = 18_{10}

Work with Steps **:**

21_{10} + (- 18_{10})

i)Calculate the 10's complement of a subtrahend.

(- 18_{10})=>10's(**-18_{10}**)

= 9 9 -1 8 + 1= 8 2

ii)add the 10's complement to the larger number(minuend).

$$\begin{array}{r} 2\ 1 \\ +\ \ 8\ 2 \\ \hline \boxed{1\ \ 0\ 3} \end{array}$$

iii) If carry comes in the MSB , discard the carry.

$$\boxed{0\ 3_{10}}$$

Decimal Arithmetic Using Complement

Subtraction Using 10's Complement:

Solve:1000_2 - 10101_2= ?

Objective :

$1000_2 - 10101_2 = ? = 18_{10} - 21_{10}$

Input Data :

Binary Input 1 = 18_{10}

Binary Input 2 = 21_{10}

Work with Steps **:**

$18_{10} + (-21_{10})$

i)Calculate the 10's complement of a subtrahend.

(-21_{10})=>10's(-21_{10})

= 9 9 -2 1 + 1= 7 9

ii)add the 10's complement to the larger number(minuend).

```
    1 8
+   7 9
0   9 7
```

iii) The result will be in 10's complement form. Calculate 10's complement of final value and assign –ve sign to the result.

10's(9 7) =9 9 - 9 7 + 1

$\boxed{= 0\ 3_{10}}$

Exercise Problems

1. Find the 9’s and 10's complement of the following 6-digit decimal numbers:

    123900;

    090657;

    100000;

    and 000000.

2. Find the I's and 2's complements of the following 8-digit binary numbers:

    10101110;

    10000001;

    10000000;

    00000001; and 00000000.

3. Perform subtraction with the following unsigned decimal numbers by taking the 9’s and 10's complement of the subtrahend.

    (a) 5250 - 1321

    (b) 1753 - 8640

4.Perform the arithmetic operations (+42) + (-13) and (-42) - (-13) in binary using the signed-2's-complement representation for negative numbers

The (r - 1)'s complement of base-6 numbers is called the 5's complement.

(a) Determine a procedure for obtaining the 5's complement of base-6 numbers.

(b) **Obtain the 5's complement of $(543210)_6$.**

(c) Design a 3-bit code to represent each of the six digits of the base-6 number system.

Make the binary code self-complementing so that the 5's complement is obtained by changing 1's to O's and O's to I's in all the bits of the coded number.

Codes

Binary Codes:

The digital data is represented, stored and transmitted as group of bits. This group of bits is also called as binary code.

Binary codes can be classified into two types

- Weighted codes
- Unweighted codes

Weighted codes

- In a weighted code each bit position is assigned by a weighting factor.
- Example: 8421, 5421, 2421, 84-2-1
- 8421 code is also known as Binary Coded decimal(BCD code)

Unweighted codes

- Example: Gray code, Excess 3-code

Gray code

Gray code system is a binary number system in which every successive pair of numbers differ in only one bit.

Binary to Gray Code Conversion

- Example: Convert 1011 binary to gray code

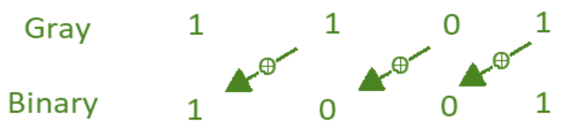
Binary	=	1	⊕	0	⊕	0	⊕	1
Gray	=	1		1		0		1

- Example: Convert 0101 binary to gray code.

Binary	=	0	⊕	1	⊕	0	⊕	1
Gray	=	0		1		1		1

Gray to Binary Conversion

- Example: Convert gray code 1101 to binary.



Excess-3 Code

- This code doesn’t have any weights. So, it is an un-weighted code.
- Excess-3 code of a decimal number is obtained by adding three 0011 to the binary equivalent of that decimal number.
- Excess-3 Code is also known as Self-complementing code.

Value	BCD Code	Excess-3 Code
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Alphanumeric codes

- Alphanumeric codes, also called character codes used to represent alphanumeric data.
- The alphanumeric data includes letters of the alphabet, numbers, mathematical symbols and punctuation marks

Binary Coded Decimal - Arithmetic

Binary-Coded Decimal (BCD) is a class of binary encodings of decimal numbers where each digit is represented by a fixed number of bits, usually four or eight. Sometimes, special bit patterns are used for a sign or other indications (e.g. error or overflow)

In this code each decimal digit is represented by a 4-bit binary number. BCD is a way to express each of the decimal digits with a binary code. In the BCD, with four bits we can represent sixteen numbers (0000 to 1111). But in BCD code only first ten of these are used (0000 to 1001). The remaining six code combinations i.e. 1010 to 1111 are invalid in BCD.

Decimal	Binary Coded Decimal
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Decimal	Binary Coded Decimal
10	1 0000
11	1 0001
12	1 0010
13	1 0011
…	…
…..	….
25	0010 0101
:	:
149	0001 0100 1001
1024	0001 0000 0010 0100

Binary Coded Decimal - Arithmetic

BCD is a binary code of the ten decimal digits. It is not a binary equivalent.

Steps to perform BCD addition

Add the BCD digits as regular binary numbers.

If the sum is 9 or less and no carry was generated, it is a valid BCD digit.

If the sum produces a carry, the sum is invalid and the number 6 (0110) must be added to the digit.

If the sum is greater than nine, the sum is invalid and the number 6 (0110) must be added to the digit.

Repeat for each of the BCD digits.

Example For 599 and 015, Find BCD Addition method

Solution: Add 599 and 015 using BCD addition

BCD code for 599 : 0101 1001 1001

BCD code for 015 : 0000 0001 0101

Addition : 0101 1010 1110

If Invalid BCD then add 6 : 0110 0110

Addition : 0101 10000 10100

Remaining bits except carry : 0101 0000 0100

Carry : 1 1

Addition : 0110 0001 0100_{BCD}

BCD value : 6 1 4

So final answer of BCD addition is 614_{10}

Binary Coded Decimal - Arithmetic

Example Add 7123_{10} and 3395_{10} using BCD addition

Solution:

For A+B

1. Add each digit of A and B using binary addition
2. If sum of two digits is more than 9 then result is Invalid BCD and add 6 to the result, Otherwise result is valid BCD.
3. If carry then add it to the next bits

Add 7123 and 3395 using BCD addition

BCD code for 7123 :		0111	0001	0010	0011
BCD code for 3395 :		0011	0011	1001	0101
Addition :		1010	0100	1011	1000
If Invalid BCD then add 6 :		0110		0110	
Addition		10000	0100	10001	1000
Remaining bits except carry :		10000	0100	0001	1000
Carry :			1		
Addition :	1	0000	0101	0001	1000_{BCD}
BCD value :	1	0	5	1	8

So final answer of BCD addition is 10518_{10}

Binary Coded Decimal - Arithmetic

BCD Subtraction using 9's complement

Steps for BCD subtraction using 9’s complement

For A-B

1. Take 9’s complement for B
2. Add it to A using BCD addition
3. If addition is invalid BCD then add 6
4. If carry then add it to the next bits
5. In final result, if carry is occured then add it the remaining result and if there is no any carry over, then take 9’s complement of the result and it is negative.

BCD Subtraction using 9's complement

Steps for BCD subtraction using 9’s complement

For A-B

1. Take 10’s complement for B
2. Add it to A using BCD addition
3. If addition is invalid BCD then add 6
4. If carry then add it to the next bits
5. In final result, if carry is occured then it is ignored and if

there is no any carry over, then take 10’s complement of the result and it is negative.

Binary Coded Decimal - Arithmetic

BCD Subtraction using 9's Complement

Find BCD Subtraction for 599 and 015, using 9's complement method

1. Take 9’s complement for 015

Note : 9's complement of a number is obtained by subtracting all bits from 999.

9's complement of 015 is

```
      9   9   9
 -    0   1   5
 ---------------
      9   8   4
```

2. Add 599 and 984 using BCD addition

BCD code for 599 :		0101	1001	1001
BCD code for 984 :		1001	1000	0100
Addition		1110	10001	1101
If Invalid BCD then add 6 :		0110	0110	0110
Addition :		10100	10111	10011
Remaining bits except carry :		10100	0111	0011
Carry :		1	1	
Addition :	1	0101	1000	0011_{BCD}
BCD value :	1	5	8	3

The left most bit of the result is 1, called carry and This will be added to 583

583+1=584

So final answer of BCD Subtraction is 584_{10}

Binary Coded Decimal - Arithmetic

BCD Subtraction Using 10's Complement

Find BCD Subtraction for 015 and 599 , using 10's complement method

1. Take 10’s complement for 599

Note : 10's complement of a number is obtained by subtracting all bits from 999 and then add 1 or directly subracting from 1000.

10's complement of 599 is

```
      9   9   9
 -    5   9   9
----------------
      4   0   0
```

Now add 1 : 400 + 1 = 401

2. Add 015 and 401 using BCD addition

```
BCD code for 015 :            0000          0001          0101
BCD code for 401 :            0100          0000          0001
  -----------------------------------------------------------------
Addition                      0100          0001          0110
If Invalid BCD then add 6 :   0000          0000          0000
  -----------------------------------------------------------------
Addition :                    0100          0001          0110
  -----------------------------------------------------------------
```

There is no carry over, So 10’s complement of 416 is the final result and it is negative.

Note : 10's complement of a number is 1 added to it's 9's complement number.

9's complement of 416 is

```
      9   9   9
 -    4   1   6
----------------
      5   8   3
```

Now add 1 : 583 + 1 = 584_{10} is the final result.

Exercise Problems

1. Perform BCD addition with the following unsigned decimal numbers by taking the 9’s and 10's complement of the subtrahend.[CO1,K3]

   (a) 2850 - 1490

   (b) 7553 - 8646

2. Perform BCD subtraction with the following unsigned decimal numbers by taking the 9’s and 10's complement of the subtrahend.[CO1,K3]

   (a) 1350 - 1591

   (b) 7753 - 8640

3. Perform BCD subtraction with the following unsigned decimal numbers by taking the 1’s and 2's complement of the subtrahend.[CO1,K3]

   (a) 131250 - 15191

   (b) 97853 - 28780

Exercise Problems

1. Perform BCD addition with the following unsigned decimal numbers by taking the 9’s and 10's complement of the subtrahend.[CO1,K3]

   (a) 2850 - 1490

   (b) 7553 - 8646

2. Perform BCD subtraction with the following unsigned decimal numbers by taking the 9’s and 10's complement of the subtrahend.[CO1,K3]

   (a) 1350 - 1591

   (b) 7753 - 8640

3. Perform BCD subtraction with the following unsigned decimal numbers by taking the 1’s and 2's complement of the subtrahend.[CO1,K3]

   (a) 131250 - 15191

   (b) 97853 - 28780

BOOLEAN ALGEBRA

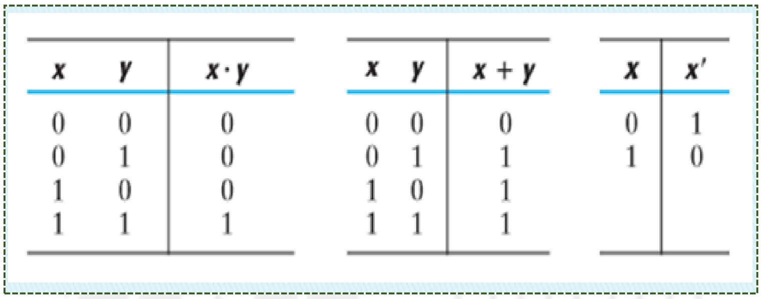
Definition of Boolean Algebra

- Boolean algebra is a mathematical system that is defined with
  - A set of elements
  - Operators
  - A number of postulates or axioms

Boolean Operations using binary operators

- The rules for the binary operators '**+**' and '**.**' on a set of two elements S={0,1} are shown in the following operator tables.

Table 1 : Boolean Operations



x	y	$x \cdot y$
0	0	0
0	1	0
1	0	0
1	1	1

x	y	$x + y$
0	0	0
0	1	1
1	0	1
1	1	1

x	x'
0	1
1	0

1. POSTULATES OF BOOLEAN ALGEBRA

- The postulates are the basic assumptions from which it is possible to deduce the rules and theorems.
- The most common postulates used to formulate various algebraic structures are as follows:

1. Closure Property
2. Commutative Property
3. Distributive Property
4. Identity Property
5. Inverse Property

(Refer Table.2)

BOOLEAN ALGEBRA

2. BASIC THEOREMS AND PROPERTIES OF BOOLEAN ALGEBRA

1. Duality Principle
2. Consensus Theorem
3. Associative Theorem
4. Absorption Theorem
5. DeMorgan Theorem
6. Idempotent Theorem
7. Involution Theorem

DUALITY PRINCIPLE

- The important property of Boolean algebra is the duality principle. It states that every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged.
- Duality principle states that a Boolean expression can be obtained from other expression by,
    - Converting AND operation to OR operation
    - Converting OR operation to AND operation
    - Complementing/negating the 0's or 1's appearing in the expression.

CONSENSUS THEOREM

i) $AB+A'C+BC = AB+A'C$

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

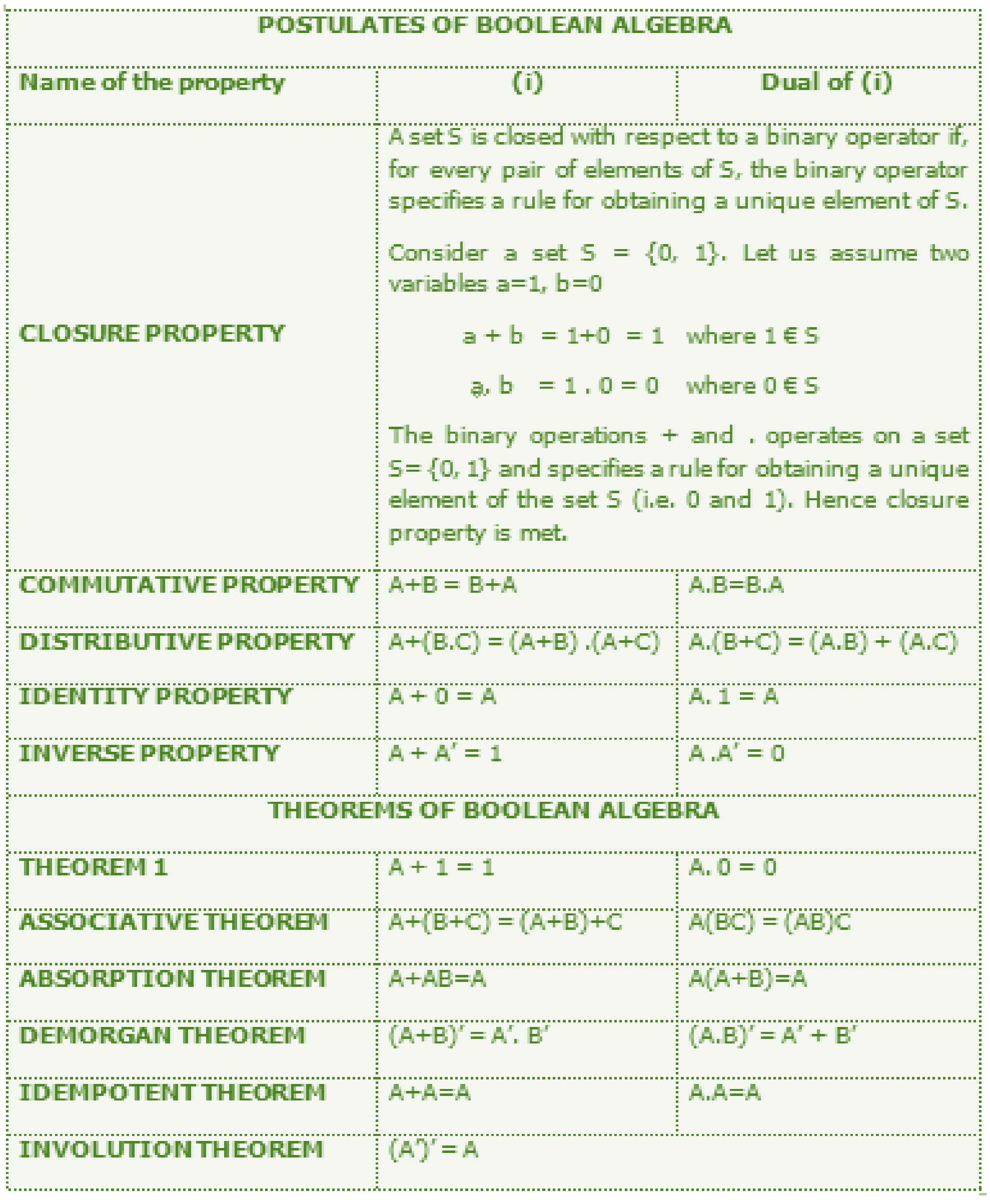
- A consensus term/redundant term is the conjunction of unique literals/variables of the terms, excluding the literal that appears negated in one term and unnegated in the other. The redundant term is eliminated.

BOOLEAN ALGEBRA

- Conditions to apply Consensus Theorem

1. Three literals must be present in the expression.
2. Each literal is repeated twice.
3. One literal must be present in both negated and unnegated form.

Table .2 Boolean Algebra Postulates and Theorems



POSTULATES OF BOOLEAN ALGEBRA		
Name of the property	**(i)**	**Dual of (i)**
CLOSURE PROPERTY	A set S is closed with respect to a binary operator if, for every pair of elements of S, the binary operator specifies a rule for obtaining a unique element of S. Consider a set S = {0, 1}. Let us assume two variables a=1, b=0 a + b = 1+0 = 1 where 1 € S a. b = 1 . 0 = 0 where 0 € S The binary operations + and . operates on a set S= {0, 1} and specifies a rule for obtaining a unique element of the set S (i.e. 0 and 1). Hence closure property is met.	
COMMUTATIVE PROPERTY	A+B = B+A	A.B=B.A
DISTRIBUTIVE PROPERTY	A+(B.C) = (A+B) .(A+C)	A.(B+C) = (A.B) + (A.C)
IDENTITY PROPERTY	A + 0 = A	A. 1 = A
INVERSE PROPERTY	A + A′ = 1	A .A′ = 0
THEOREMS OF BOOLEAN ALGEBRA		
THEOREM 1	A + 1 = 1	A. 0 = 0
ASSOCIATIVE THEOREM	A+(B+C) = (A+B)+C	A(BC) = (AB)C
ABSORPTION THEOREM	A+AB=A	A(A+B)=A
DEMORGAN THEOREM	(A+B)′ = A′. B′	(A.B)′ = A′ + B′
IDEMPOTENT THEOREM	A+A=A	A.A=A
INVOLUTION THEOREM	(A′)′ = A	

BOOLEAN ALGEBRA

3. PROOF OF THEOREMS

Operator Precedence

- The operator precedence for evaluating Boolean expressions is (1) parentheses, (2) NOT, (3) AND, and (4) OR.
- The Boolean theorems can be **proved in two ways**:

  (i) Using the Boolean algebra Postulates (ii) Using the truth table.

Examples:

Prove A+A =A (Prove the Idempotent theorem)

L.H.S	= A+A	
	= (A+A) . 1	(Identity property)
	= (A+A) . (A+A’)	(Inverse property)
	= A+AA’	(Distributive property)
	= A + 0	(Inverse property)
	= A	(Identity property)
	= R.H.S	

Hence proved.

Prove A.A=A (Prove the Idempotent theorem)

L.H.S	= A.A	
	= (A.A) +0	(Identity property)
	= (A.A)+(A.A’)	(Inverse property)
	= A.(A+A’)	(Distributive property)
	= A.1	(Inverse property)
	= A	(Identity property)
	= R.H.S	

Hence proved.

Prove A+1 = 1 (Theorem 1)

L.H.S	= A + 1	
	= (A+1) . 1	(Identity property)
	= (A+1) . (A + A')	(Inverse property)
	= A + 1.A’	(Distributive property)
	= A+A’	(Identity property)
	= 1	(Inverse property)
	= R.H.S	

Hence Proved

BOOLEAN ALGEBRA

Prove A+AB = A (Prove the Absorption theorem)

L.H.S = A+AB
= A.1 + AB (Identity property)
= A(1+B) (Distributive property)
= A.1 (Theorem 1)
= A (Identity property)
= R.H.S

Hence proved.

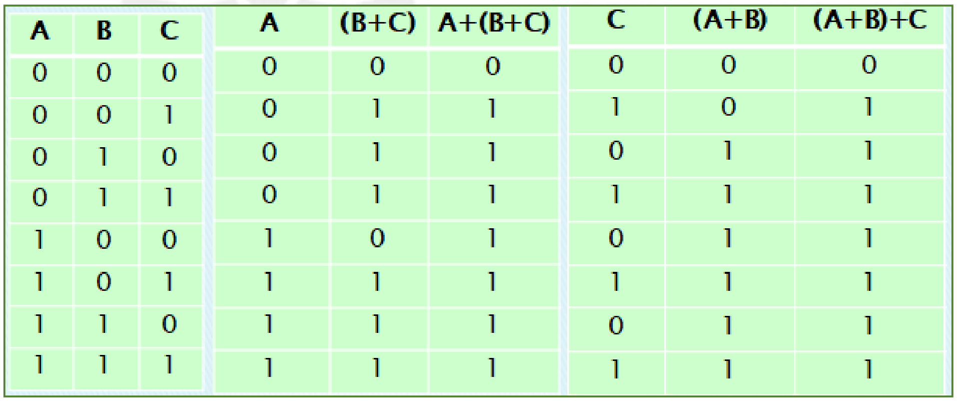
Prove A.(A+B) =A (Prove the Absorption theorem)

L.H.S = A(A+B)
= A.A+A.B (Distributive property)
= A+ AB (Idempotent property)
= A.1 + AB (Identity property)
= A(1 + B) (Distributive property)
= A. 1 (Theorem 1)
= A (Identity property)
= R.H.S

Hence proved.

Prove A +(B+C) = (A+B)+C (Prove the Associative theorem)

Since the algebraic proof of Associative theorem is long, it can be proved using the truth table



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

$\uparrow$ L.H.S

$\uparrow$ R.H.S

L.H.S = R.H.S.

Hence proved.

BOOLEAN ALGEBRA

Prove A .(B.C) = (A.B).C (Prove the Associative theorem)

Since the algebraic proof of Associative theorem is long, it can be proved using the truth table.

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

$\uparrow$ L.H.S $\uparrow$ R.H.S

L.H.S = R.H.S

Hence proved.

Prove (A+ B)' = A' . B' (Prove the DeMorgan theorem)

Since the algebraic proof of DeMorgan theorem is long, it can be proved using the truth table.

A	B	A+B	(A+B)'	A'	B'	A' . 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

$\uparrow$ L.H.S $\uparrow$ R.H.S

L.H.S = R.H.S

Hence proved.

BOOLEAN ALGEBRA

Prove (A . B)′ = A′ + B′ (Prove the DeMorgan theorem)

Since the algebraic proof of DeMorgan theorem is long, it can be proved using the truth table.

A	B	A.B	(A.B)'	A'	B'	A'+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

$\uparrow$ L.H.S $\uparrow$ R.H.S

L.H.S = R.H.S

Hence proved.

Prove that AB + BC + B′C = AB + C

L.H.S = AB+BC+B′C

= AB+C(B+B′)

= AB+ C.1 (Inverse property)

= AB+ C (Identity property)

= R.H.S

Hence proved.

Prove that x + xyz + yzx′ + wx + w′x + x′y = x + y

L.H.S = x + xyz + yzx′ + wx + w′x + x′y

= x(1+yz+w+w′) + x′(y+yz)

= x(1) + x′ (y) (Theorem 1 and Absorption theorem)

=x + x′y (Identity property)

=(x+x′) .(x+y) (Distributive property)

= 1. (x+y) (Inverse property)

= x+y (Identity property)

= R.H.S

Hence proved.

BOOLEAN FUNCTIONS

- A Boolean function described by an algebraic expression consists of:
  - binary variables,
  - the constants 0's and 1's and
  - the logic operation symbols.
- For a given value of the binary variables, the function can be equal to 0 or 1.
- A Boolean function can be represented in a truth table and it can be transformed into a circuit diagram composed of logic gates.

SIMPLIFICATION OF BOOLEAN FUNCTIONS USING ALGEBRAIC METHOD

- The original Boolean expression can be converted to an equivalent simpler form of Boolean expression by using the Boolean theorems and postulates.

Advantages of Simplification/Minimization

1. Circuit gets simplified.
2. Number of gates required is reduced.
3. Cost of the circuit is reduced.

Simplify the Boolean function F= x(x'+y) to minimum number of literals

F = x(x'+y)

= xx'+xy (Distributive property)

= 0 + xy (Inverse property)

= xy (Identity property)

The simplified Boolean function is F = xy

Simplify the Boolean function F= x+x'y to minimum number of literals

F = x + x'y

= (x + x') (x + y) (Distributive property)

= 1 . (x+y) (Inverse property)

= x+y (Identity property)

The simplified Boolean function is F= x+y

BOOLEAN FUNCTIONS

Simplify the Boolean function F= (x + y)(x + y′) using algebraic method

F= (x + y)(x + y′)	
= x + yy′	(Distributive property)
= x + 0	(Inverse property)
= x	(Identity property)

The simplified Boolean function is F=x

Reduce the given function F= xy + x′z + yz to minimum number of literals

F = xy + x′z + yz	
= xy + x′z + yz(x+x′)	(Identity property)
= xy + x′z + xyz + x′yz	(Distributive property)
= xy(1+z) +x′z(1+y)	
= xy +x′z	(Theorem 1)

By applying consensus theorem, yz is identified as a consensus term/ redundant term and it is eliminated from the expression.

Hence, the reduced Boolean function is F= xy + x′z

Minimize the Boolean function F = (x + y)(x′ + z)(y + z)

By applying the consensus theorem (y + z) is identified as a consensus/redundant term and it is eliminated from the expression.

Hence, the minimized Boolean function is F= (x + y)(x′ + z)

BOOLEAN FUNCTIONS

2. COMPLEMENT OF A FUNCTION

The complement of a function F is represented as F’. It can be identified in **two** ways:

Method 1: Algebraic method (using DeMorgan theorem)

Method 2: Finding Dual of F and Complementing each literal

Example: Find the complement of the Boolean function F= (x’yz’ + x’y’z)

Method 1: Algebraic method (using DeMorgan theorem)

Given F= (x’yz’ + x’y’z)

F ’= (x’yz’ + x’y’z) ’

=(x’yz’)’ . (x’y’z)’ (DeMorgan theorem)

= ((x’)’+y’+(z’)’) . ((x’)’ + (y’)’+z’) (DeMorgan theorem)

= (x+y’+z) . (x+y+z’) (Involution theorem)

The complement of the given Boolean function is F’ = (x+y’+z) . (x+y+z’)

Method 2: Finding Dual of F and Complementing each literal

Given F= (x’yz’ + x’y’z)

i. Dual of F = (x’+y+z’). (x’+y’+z)

ii. Complementing each literal = (x+y’+z)(x+y+z’)

The complement of the given Boolean function is F’ = (x+y’+z) . (x+y+z’)

Note: The resulting complement of the function can further be reduced by applying theorems and postulates if required.

CANONICAL AND STANDARD FORMS

A Boolean function is expressed in two form.

1. Sum of Minterms
2. Product of Maxterms

- Minterms are called products because they are the logical AND of a set of variables, and maxterms are called sums because they are the logical OR of a set of variables.

Minterms and Maxterms:

A binary variable may appear either in its normal form (x) or in its complement form (x′). Now either two binary variables x and y combined with an AND operation. Since each variable may appear in either form, there are four possible combinations:

x′y′, x′y, xy′ and xy Each of these four AND terms is called a '***minterm***'.

In a similar fashion, when two binary variables x and y combined with an OR operation, there are four possible combinations:

x′+ y′, x′+ y, x+ y′ and x+ y Each of these four OR terms is called a '***Maxterm***'.

The minterms and Maxterms of a 3- variable function can be represented as in table below.

A	B	C	Minterms	Maxterms
0	0	0	A′B′C′	A + B + C
0	0	1	A′B′C	A + B + C′
0	1	0	A′BC′	A + B′ + C
0	1	1	A′BC	A + B′ + C′
1	0	0	AB′C′	A′ + B + C
1	0	1	AB′C	A′ + B + C′
1	1	0	ABC′	A′ + B′ + C
1	1	1	ABC	A′ + B′ + C′

CANONICAL AND STANDARD FORMS

Canonical Products of Sum

If each term in POS form contains all literals then the POS is known as standard (or) Canonical POS form. Each individual term in standard POS form is called Maxterm canonical form.

F (A, B, C) = (A+ B+ C). (A+ B′+ C). (A+ B+ C′)

F (x, y, z) = (x+ y′+ z′). (x′+ y+ z). (x+ y+ z)

Steps to convert general POS to standard POS form:

If each term in SOP form contains all the literals then the SOP is known as standard (or) canonical SOP form. Each individual term in standard SOP form is called Find the missing literals in each product term if any.

1. AND each product term having missing literals by ORi g the literal and its complement.

3. Expand the term by applying distributive law and reorder the literals in the product term.

4. Reduce the expression by omitting repeated product terms if any.

Obtain the canonical SOP form of the function:

1. Y(A, B) = A+ B

= A. (B+ B′)+ B (A+ A′) = AB+ AB′+ AB+ A′B = AB+ AB′+ A′B.

2. Y (A, B, C) = A+ ABC

= A. (B+ B′). (C+ C′)+ ABC = (AB+ AB′). (C+ C′)+ ABC = ABC+ ABC′+ AB′C+ AB′C′+ ABC = ABC+ ABC′+ AB′C+ AB′C′ = m7+ m6+ m5+ m4 = ∑m (4, 5, 6, 7). 3. Y (A, B, C) = A+ BC

= A. (B+ B′). (C+ C′)+(A+ A′). BC = (AB+ AB′). (C+ C′)+ ABC+ A′BC = ABC+ ABC′+ AB′C+ AB′C′+ ABC+ A′BC = ABC+ ABC′+ AB′C+ AB′C′+ A′BC = m7+ m6+ m5+ m4+ m3 = ∑m (3, 4, 5, 6, 7).

SUM OF PRODUCT AND PRODUCT OF SUM SIMPLIFICATION

1. Representation of Boolean expression can be primarily done in two ways. They are as follows:
   a. Sum of Products (SOP) form
   b. Product of Sums (POS) form
2. The terms "product" and "sum" have been borrowed from mathematics to describe AND and OR logic operations.
3. Note: If the number of input variables are n, then the total number of combinations in Boolean algebra is 2^n.

1.1 Sum of Products (SOP) Form :

1. It is one of the ways of writing a boolean expression, formed by adding (OR operation) the product terms.
2. Product Term: Is defined as either a literal or a product of literals (also called as at are ANDed together.
   Eg.: $F(A,B,C) = \bar{A}\,B\,\bar{C} + A\,\bar{B}\,C + \bar{A}\,B\,C + A\,B\,\bar{C} + A\,B\,C$
   Eg: ABC , XYZ
3. SOP: Is defined as group of product terms that are ORed together.
   Eg: F(A,B,C) = AB + BC + AC
4. Other Names: Disjunctive Normal Form, Disjunctive Normal Formula.
5. Standard SOP / Canonical SOP: SOP is said to be a Standard SOP / Canonical SOP, if each product term consists of all literals in either complemented form or uncomplemented form.
   Eg.: F(A,B,C) = A B C + A B C + A B C + A B C+A B C
6. Minterms (∑m): Is defined as each individual product term in Standard SOP / Canonical SOP.

0	**Complement of a Variable**
1	**Variable**

❁ 1.2 Product of Sums (POS) Form :

1. It is one of the ways of writing a boolean expression, formed by multiplying (AND operation) the sum terms.
2. Sum Term: Is defined as either a literal or a sum of literals (also called as disjunction). It is also defined as a group of literals that are ORed together.

   Eg: A+B+C , X+Y+Z
3. POS: Is defined as group of sum terms that are ANDed together.

   Eg: F(A,B,C) = (A+B) (B+C) (A+C)
4. Other Names: Conjunctive Normal Form, Conjunctive Normal Formula.
5. Standard POS / Canonical POS: POS is said to be a Standard POS / Canonical POS, if each sum term consists of all literals in either complemented form or uncomplemented form.

   Eg.: $F(A,B,C) = (\bar{A}+B+\bar{C})\ (\bar{A}+B+C)\ (A+\bar{B}+C)\ (A+B+\bar{C})\ (A+B+C)$
6. Maxterms (ΠM): Is defined as each individual sum term in Standard POS / Canonical POS.

0	**Variable**
1	**Complement of a Variable**

7. POS Form is obtained by writing one sum term for each input combination that produces an output 0.

14. Converting Boolean functions to Standard SOP/POS form

Canonical Sum of Products

If each term in SOP form contains all the literals then the SOP is known as standard (or) canonical SOP form. Each individual term in standard SOP form is called minterm canonical form.

F (A, B, C) = AB′C+ ABC+ ABC′

Steps to convert general SOP to standard SOP form:

1. Find the missing literals in each product term if any.
2. AND each product term having missing literals by ORi g the literal and its complement.
3. Expand the term by applying distributive law and reorder the literals in the product term.
4. Reduce the expression by omitting repeated product terms if any.

Obtain the canonical SOP form of the function:

1. Y(A, B) = A+ B

= A. (B+ B′)+ B (A+ A′) = AB+ AB′+ AB+ A′B = AB+ AB′+ A′B.

2. Y (A, B, C) = A+ ABC

= A. (B+ B′). (C+ C′)+ ABC = (AB+ AB′). (C+ C′)+ ABC = ABC+ ABC′+ AB′C+ AB′C′+ ABC = ABC+ ABC′+ AB′C+ AB′C′

= m7+ m6+ m5+ m4 = ∑m (4, 5, 6, 7).

3. Y (A, B, C) = A+ BC

= A. (B+ B′). (C+ C′)+(A+ A′). BC = (AB+ AB′). (C+ C′)+ ABC+ A′BC = ABC+ ABC′+ AB′C+ AB′C′+ ABC+ A′BC = ABC+ ABC′+ AB′C+ AB′C′+ A′BC = m7+ m6+ m5+ m4+ m3 = ∑m (3, 4, 5, 6, 7).

4. Y (A, B, C) = AC+ AB+ BC

= AC (B+ B′)+ AB (C+ C′)+ BC (A+ A′) = ABC+ AB′C+ ABC+ ABC′+ ABC+ A′BC = ABC+ AB′C+ ABC′+ A′BC = ∑m (3, 5, 6, 7).

14. Converting Boolean functions to Standard SOP/POS form

Canonical Product of Sum

If each term in POS form contains all literals then the POS is known as standard (or) Canonical POS form. Each individual term in standard POS form is called Maxterm canonical form.

F (A, B, C) = (A+ B+ C). (A+ B′+ C). (A+ B+ C′)

F (x, y, z) = (x+ y′+ z′). (x′+ y+ z). (x+ y+ z)

Steps to convert general POS to standard POS form:

1. Find the missing literals in each sum term if any
2. OR each sum term having missing literals by ANDing the literal and its complement.
3. Expand the term by applying distributive law and reorder the literals in the sum term.
4. Reduce the expression by omitting repeated sum terms if any.

Obtain the canonical POS form of the function:

1. Y= A+ B′C

= (A+ B′) (A+ C) [A+ BC = (A+B) (A+C)] = (A+ B′+ C.C′) (A+ C+ B.B′)

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

= (A+ B′+C). (A+ B′+C′). (A+ B+ C)

= M2. M3. M0 = ∏M (0, 2, 3)

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

= (A+B+ C.C′) (B+ C+ A.A′) (A+C+B.B′)

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

= (A+B+C) (A+B+C′) (A′+B+C) (A+B′+C)

= M0. M1. M4. M2 = ∏M (0, 1, 2,

14. Converting Boolean functions to Standard SOP/POS form

3. Y= A. (B+ C+A)

= (A+ B.B′+ C.C′). (A+ B+ C)

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

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

= M0. M1. M2. M3 = ∏M (0, 1, 2, 3)

4. Y= (A+B′) (B+C) (A+C′)

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

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

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

= M2. M3. M0. M4. M1 = ∏M (0, 1, 2, 3, 4)

5. Y= xy+ x′z

= (xy+ x′) (xy+ z) Using distributive law, convert the function into OR terms.

= (x+x′) (y+x′) (x+z) (y+z) [x+ x′=1]

= (x′+y) (x+z) (y+z) = (x′+y+ z.z′) (x+z+y.y′) (y+z+ x.x′)

= (x′+ y+ z) (x′+ y+ z′) (x+ y+ z) (x+ y′+ z) (x+ y+ z) (x′+ y+ z)

= (x′+ y+ z) (x′+ y+ z′) (x+ y+ z) (x+ y′+ z)

= M4. M5. M0. M2 = ∏M (0, 2, 4, 5).

15. Introduction to Karnaugh Map

- The simplification of the functions using Boolean laws and theorems becomes complex with the increase in the number of variables and terms.
- The map method, first proposed by Veitch and slightly improvised by Maurice Karnaugh, a telecommunications engineer, at Bell Labs in 1953 while designing digital logic based telephone switching circuits.
- This map method is called **Veitch diagram or Karnaugh map.** Karnaugh Map usually abbreviated as **K-map** which is a systematic approach used for simplifying Boolean expressions or logic functions.
- It is majorly used method for **minimizing the Boolean expressions.** K-maps reduce logic functions more quickly and easily compared to Boolean algebra.
- A K-map provides a pictorial method of grouping together expressions with common factors and therefore eliminating unwanted variables.
- A K-map is a diagram made up of squares, with each square representing one minterm of the function that is to be minimized.
- For n variables on a K-map there are **2^n numbers of squares.** It can be drawn directly from either minterm (sum-of-products) or maxterm (product-of-sums) Boolean expressions.
- K-maps can be used for expressions with two, three, four and five variables.
- **Number of squares = number of combinations**
- 2 Variables → 4 squares; 3 Variables → 8 squares; 4 Variables →16 squares;
- The K-map can also be described as a **special arrangement of a truth table**; instead of being organized (i/p and o/p) into columns and rows, the K-map is an array of squares in which each square represents a binary value of the input variables.

15. Introduction to Karnaugh Map

Types of K-Map

- 2-variable K-Map
- 3-variable K-Map
- 4-variable K-Map
- 5-variable K-Map

2-variable K-map

- A two-variable map is shown in the following diagram. There are **four minterms** for two variables; hence the map consists of four squares, one for each minterm.

m_0	m_1
m_2	m_3

x \ y	0	1
0	m_0	m_1
1	m_2	m_3

Decimal Value	**Input**		**Minterm**	
	X	Y		
0	0	0	m_0	x′y′
1	0	1	m_1	x′y
2	1	0	m_2	xy′
3	1	1	m_3	xy

- The 0's and 1's marked for each row and each column designate the values of variables x and y, respectively.
- Notice that x appears primed in row 0 and unprimed in row 1. Similarly, y appears primed in column 0 and unprimed in column 1.

15. Introduction to Karnaugh Map

3-variable K-map

- The structure of three variable K-map is shown in the following diagram. There are **eight minterms for three binary variables.** Therefore, the map consists of eight squares.

x \ yz	00	01	11	10
0	x′y′z′	x′y′z	x′yz	x′yz′
1	xy′z′	xy′z	xyz	xyz′

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6

- The minterms are not arranged in binary sequence. The characteristic of this sequence is that only one bit changes from 1 to 0 or from 0 to 1 in the listing sequence.

4-variable K-map

- The structure of three variable K-map is shown in the following diagram. There are **eight minterms for three binary variables.**
- Therefore, the map consists of eight squares.

m_0	m_1	m_3	m_2
m_4	m_5	m_7	m_6
m_{12}	m_{13}	m_{15}	m_{14}
m_8	m_9	m_{11}	m_{10}

wx \ yz	00	01	11	10
00	w′x′y′z′	w′x′y′z	w′x′yz	w′x′yz′
01	w′xy′z′	w′xy′z	w′xyz	w′xyz′
11	wxy′z′	wxy′z	wxyz	wxyz′
10	wx′y′z′	wx′y′z	wx′yz	wx′yz′

15. Introduction to Karnaugh Map

Don’t Care Condition

- Unused combinations of the input variables are called as don’t care inputs or condition.
- As a result, we don’t care what the function output is to be for those input combinations of the variables because they are guaranteed never to occur.
- To distinguish the don’t care conditions from 1’s and 0’s an ‘X’ mark will be used.
- When choosing adjacent squares to simplify the function in the map, the X’s may be assumed to be either 0 or1, whichever gives the simplest expression.
- In addition, an X need not be used at all if it does not contribute to covering a large area.

Number locations in a K-Map

- In a K-Map, the numbers are located as shown below:

x \ y	0	1
0	0	1
1	2	3

4 cells

x \ yz	00	01	11	10
0	0	1	3	2
1	4	5	7	6

8 cells

yz \ x	0	1
00	0	1
01	2	3
11	6	7
10	4	5

8 cells

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

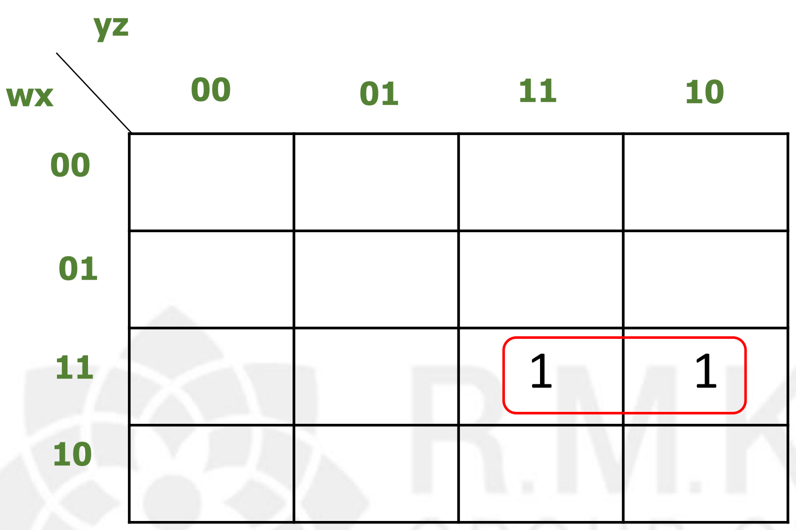
16 cells

15. Introduction to Karnaugh Map

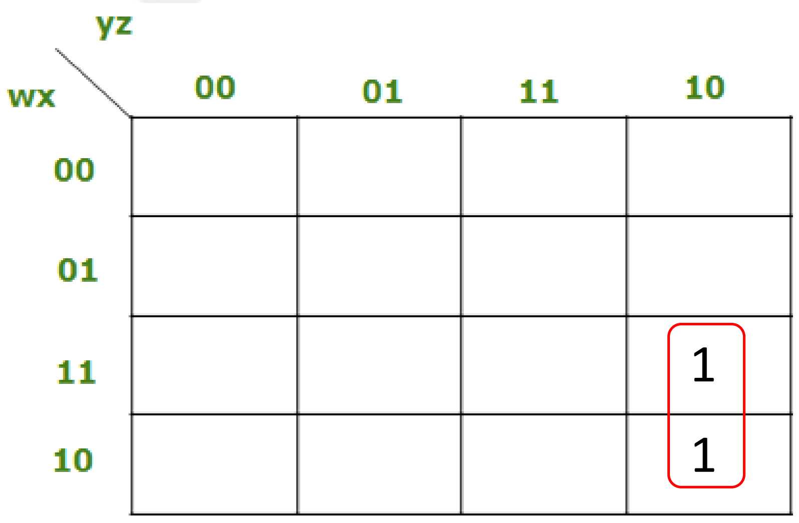
Grouping of K-map

- Pairs
- Quads and
- Octets

Pair Grouping of K-map



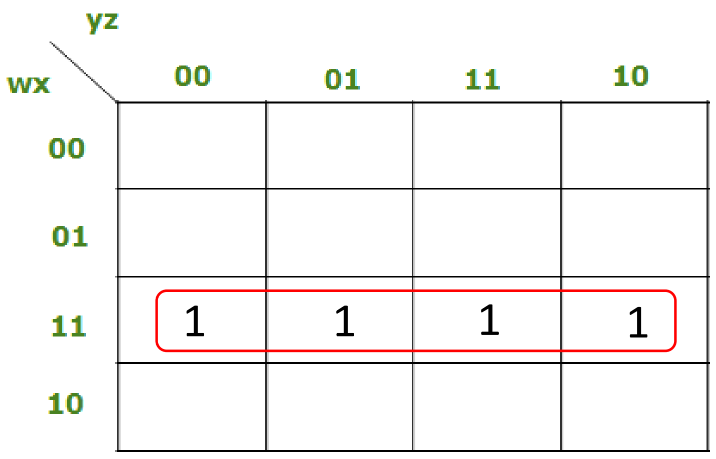
F = W X Y Z + W X Y Z′
= W X Y (Z+Z′) = W X Y



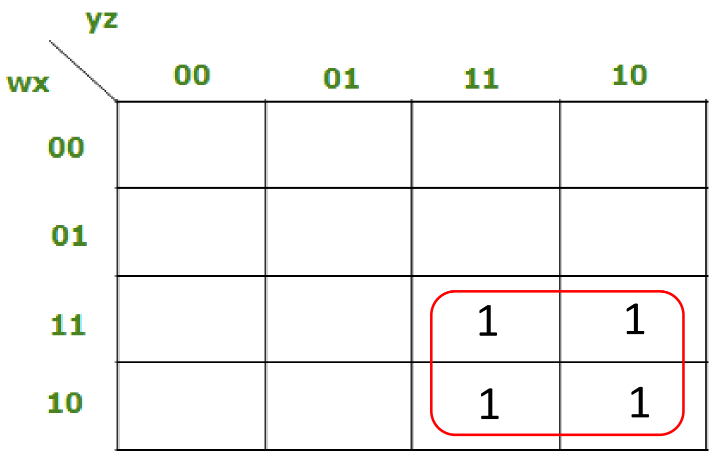
F = W X Y Z′ + W X′ Y Z′
= W Y Z′ (X+X′) = W Y Z′

15. Introduction to Karnaugh Map

Quad Grouping of K-map



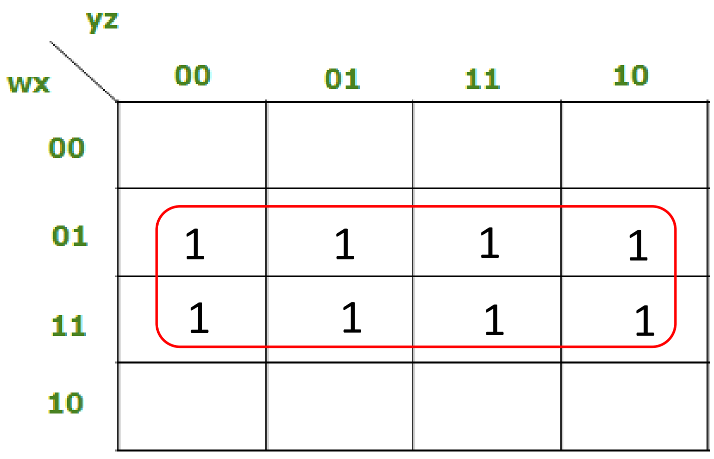
$$F = W X Y' Z' + W X Y' Z + W X Y Z + W X Y Z'$$
$$= W X Y' (Z'+Z) + W X Y (Z+Z')$$
$$= W X Y' + W X Y$$
$$= W X (Y'+Y)$$
$$= W X$$



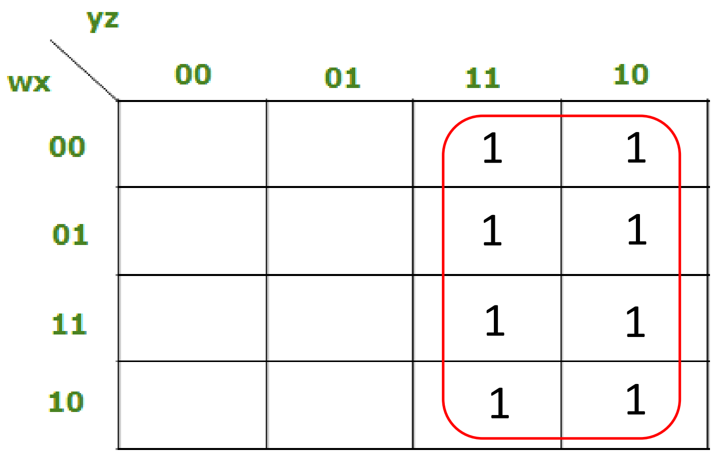
$$F = W X Y Z + W X Y Z' + W X' Y Z + W X' Y Z'$$
$$= W X Y (Z+Z') + W X' Y (Z+Z')$$
$$= W X Y + W X' Y$$
$$= W Y (X+X')$$
$$= W Y$$

15. Introduction to Karnaugh Map

Octet Grouping of K-map



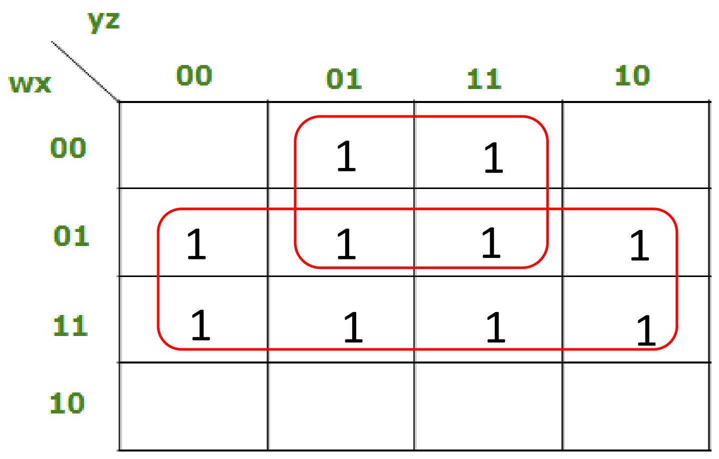
F = X



F = Y

15. Introduction to Karnaugh Map

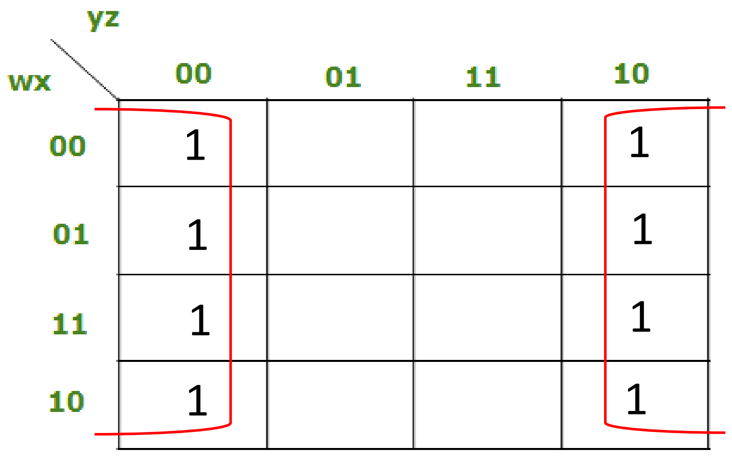
Overlapping of K-map



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

F = X + W Z

Rolling of K-map



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

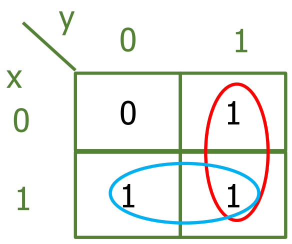
F = Z’

15. Simplification of Boolean Function Using Two Variable Karnaugh Map (SOP Form)

Example: 1

Simplify the following Boolean function in SOP $F(x,y) = \Sigma(1,2,3)$

Solution:



m_0	m_1
m_2	m_3

- Since, the simplification of given Boolean function is to be minimized in SOP form, we need to assign '1' in those respective minterm squares of the K-map. (The subscript decimal values in the right hand side diagram is used for easily identifying the respective minterm square easily).
- After assigning the 1s in the respective squares, the next step is to combine or group adjacent 1's using the following rules:
  - Adjacent 1's should be grouped in powers of 2 (i.e. $2^1=2$, Two horizontal or vertical squares, $2^2=4$, Four adjacent squares, $2^3=8$, eight adjacent squares and so on).
  - While grouping, search for maximum number of 1's (in powers of 2) present in adjacent squares.
  - To understand the usefulness of the map for simplifying Boolean functions, we must recognize the basic property possessed by adjacent squares.
  - Any two adjacent square in the map differ by only one variable which is primed in one square and unprimed in the other.
  - From the postulates of Boolean algebra, it follows that the sum of two minterms in adjacent squares can be simplified to a single AND term consisting of only two literals.

15. Simplification of Boolean Function Using Two Variable Karnaugh Map (SOP Form)

- The next step is to write the simplified Boolean expression from the grouped minterms. Identify, which rows and which columns of the map are combined and then write the expression in SOP form.

- The Boolean expression for the Example:1 problem is as follows:

  - ❖ By identifying the map, we can identify that two groupings are done by combining the minterms m_1,m_3 and m_2,m_3.

  - ❖ The minterms (m_1,m_3) belongs to Row 1, Row 2 and Column 2, which corresponds to $(x'+x).(y)=(1).y=y$.

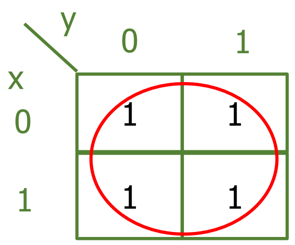
  - ❖ Similarly, the minterms (m_2,m_3) belongs to Row 2, Column 1 and Column 2, which corresponds to $x.(y'+y)=x.(1)=x$.

  - ❖ Finally, combining the two results, we get **F = y+x** as the simplified Boolean expression.

Example: 2

Simplify the following Boolean function in SOP F(x,y) = Σ(0,1,2,3)

Solution:



❖ In this example, we can combine all four adjacent 1's. Following the method for identifying the expression we get,

$F(x,y) = (x'+x)+(y'+y)=1+1=1$

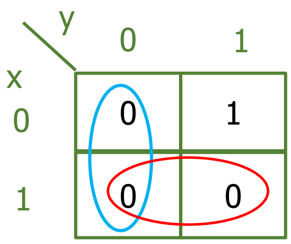
❖ So, **F(x,y)=1**

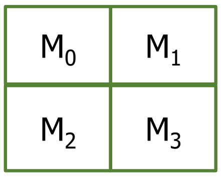
15. Simplification of Boolean Function Using Two Variable Karnaugh Map (POS Form)

Example: 3

Simplify the following Boolean function in SOP F(x,y) = Π(0,2,3)

Solution:





- Since, the simplification of given Boolean function is to be minimized in POS form, we need to assign '0' in those respective minterm squares of the K-map. (The subscript decimal values in the right hand side diagram is used for easily identifying the respective minterm square easily).
- After assigning the 0s in the respective squares, the next step is to combine or group adjacent 0's using the following rules:
  - Adjacent 0's should be grouped in powers of 2 (i.e. 2^1=2, Two horizontal or vertical squares, 2^2=4, Four adjacent squares, 2^3=8, eight adjacent squares and so on).
  - While grouping, search for maximum number of 0's (in powers of 2) present in adjacent squares.
  - To understand the usefulness of the map for simplifying Boolean functions, we must recognize the basic property possessed by adjacent squares.
  - Any two adjacent square in the map differ by only one variable which is primed in one square and unprimed in the other.
  - From the postulates of Boolean algebra, it follows that the sum of two maxterms in adjacent squares can be simplified to a single OR term consisting of only two literals.

15. Simplification of Boolean Function Using Two Variable Karnaugh Map (POS Form)

- The next step is to write the simplified Boolean expression from the grouped maxterms. Identify, which rows and which columns of the map are combined and then write the expression in POS form.

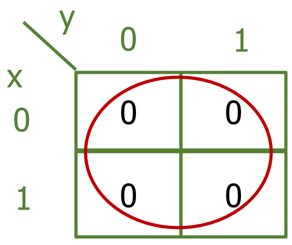
- The Boolean expression for the Example:3 problem is as follows:

  - ❖ By identifying the map, we can identify that two groupings are done by combining the maxterms M_0,M_2 and M_2,M_3.
  - ❖ The maxterms (M_0,M_2) belongs to Row 1, Row 2 and Column 1, which corresponds to (x′.x)+(y′)=(0)+y′=y′.
  - ❖ Similarly, the maxterms (m_2,m_3) belongs to Row 2, Column 1 and Column 2, which corresponds to x+(y′.y)=x+(0)=x.
  - ❖ Finally, combining the two results, we get **F = y′x** as the simplified Boolean expression.

Example: 4

Simplify the following Boolean function in POS F(x,y) = Σ(0,1,2,3)

Solution:



❖ In this example, we can combine all four adjacent 0′s. Following the method for identifying the expression we get,

F(x,y) = (x′.x).(y′.y)=0.0=0

❖ So, **F(x,y)=0**

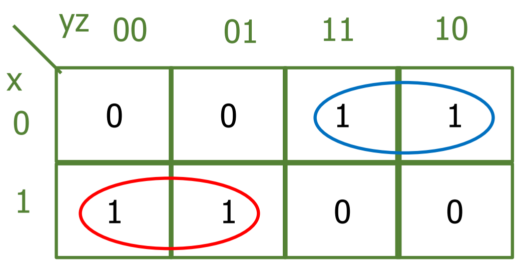
15. Simplification of Boolean Function Using Three Variable Karnaugh Map (SOP Form)

Example: 5

Simplify the Boolean function F = x′yz + x′yz′ + xy′z′ + xy′z

Solution:

❖ The given function is in canonical SOP form.



❖ First step, assign '1' in the respective squares of the given minterms, and '0' in the remaining squares.

❖ The next step is to combine the adjacent 1's forming the minterms m_2,m_3 and m_4,m_5 as two pairs.

❖ The next step is to write the simplified Boolean expression from the grouped minterms.

❖ Identify, which rows and which columns of the map are combined and then write the expression in SOP form as follows,

➢ For m_2,m_3, we get, $x'(y+y)(z+z') = x'y$

➢ For m_4,m_5, we get, $x(y'+y')(z'+z) = xy'$

❖ Finally, write the expression as,

$$F = x'y + xy'$$

15. Simplification of Boolean Function Using Three Variable Karnaugh Map (SOP Form)

Example: 6

Simplify the given function using K-map, F = A′C + A′B + AB′C + BC

Solution:

❖ The given expression has three variables A,B and C. Hence, we need to use three variable K-map.

❖ But, the given expression is not in canonical SOP form.

❖ As the first step, we need to convert the given expression into the canonical SOP form.

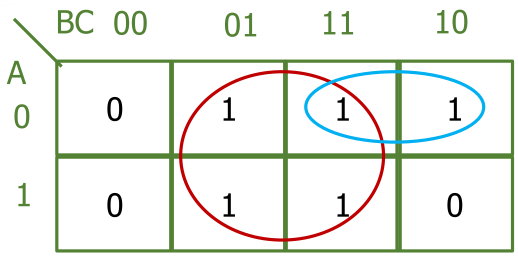
F = A′C[B+B′] + A′B[C+C′] + AB′C + BC[A+A′]

F = A′BC + A′B′C + A′BC + A′BC′ + AB′C + ABC + A′BC

F = A′BC + A′B′C + A′BC′ + AB′C + ABC

❖ Rearranging the expression we get, F = A′B′C + A′BC′ + A′BC + AB′C + ABC

❖ Hence, **F = Σ(1,2,3,5,7)**



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

❖ Expression for m_2,m_3, we get, A′(B+B)(C+C′) = A′B

❖ Expression for m_1,m_3,m_5,m_7, we get, (A′+A)(B′+B)(C+C) = C

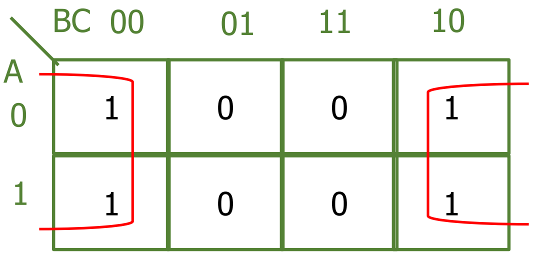
F = A′B + C

15. Simplification of Boolean Function Using Three Variable Karnaugh Map (POS Form)

Example: 7

Simplify the following Boolean function F = $\Sigma(0,2,4,6)$

Solution:

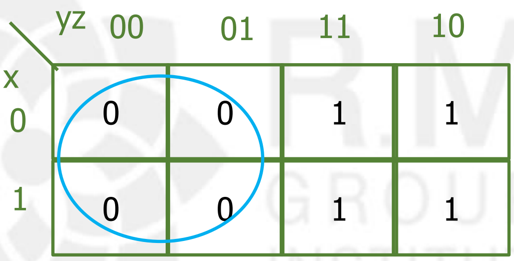


❖ Combining m_1,m_3,m_5,m_7, as four adjacent squares (Visualize that we are folding the map) we get, (A′+A)(B′+B)(C′+C′) = C′; **F = C′.**

Example: 8

Obtain the simplified expression in product of sums F(x,y,z) = $\Pi(0,1,4,5)$

Solution:

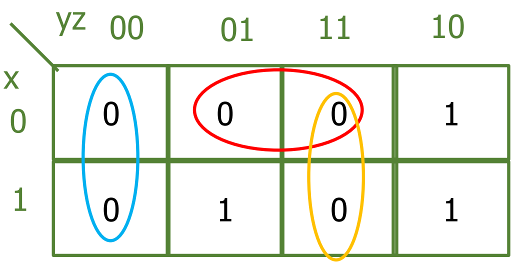


❖ Combining maxterms M_0,M_1,M_4,M_5, we get, F = (x.x′) + (y.y) + (z.z′) = y; **F = y.**

Example: 9

Obtain the simplified expression in product of sums F(x,y,z) = $\Pi(0,1,3,4,7)$

Solution:



❖ Combining maxterms M_0,M_1, we get, (x.x′) + (y) + (z) = (y+z)

❖ Combining maxterms M_1,M_3, we get, (x) + (y.y′) + (z′.z′) = (x+z′)

❖ Combining maxterms M_3,M_7, we get, (x.x′) + (y′) + (z′) = (y′+z′)

❖ So, **F = (y+z) (x+z′) (y′+z′)**

15. Simplification of Boolean Function Using Four Variable Karnaugh Map (SOP Form)

Example: 10

Simplify the Boolean function F = W′X′Y′ + X′YZ′ + W′XYZ′ + WX′Y′

Solution:

❖ The given function is not in canonical SOP form.

❖ We need to convert the given function into canonical SOP form.

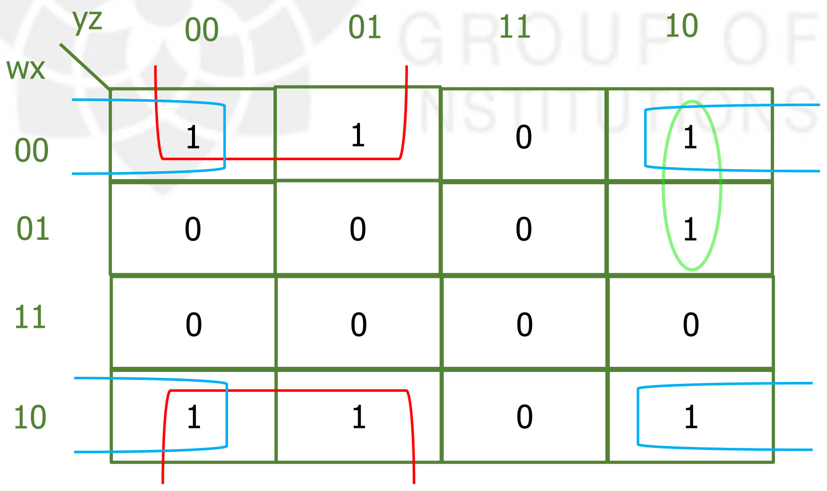
F = W′X′Y′[Z+Z′] + X′YZ′[W+W′] + W′XYZ′ + WX′Y′[Z+Z′]

F = W′X′Y′Z + W′X′Y′Z′ + WX′YZ′ + W′X′YZ′ + W′XYZ′ + WX′Y′Z + WX′Y′Z′

❖ Rearranging the equation, we get,

F = W′X′Y′Z′ + W′X′Y′Z + W′X′YZ′ + W′XYZ′ + WX′Y′Z′ + WX′Y′Z + WX′YZ′

F = Σ(0,1,2,6,8,9,10)



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

❖ Combining minterms m_0,m_2,m_8,m_{10}, we get (W′+W)(X′+X′)(Y′+Y)(Z′+Z′) = X′Z′

❖ Combining minterms m_0,m_1,m_8,m_9, we get (W′+W)(X′+X′)(Y′+Y′)(Z′+Z) = X′Y′

❖ Combining minterms m_2,m_6, we get (W′+W′)(X′+X)(Y)(Z′) = W′YZ′

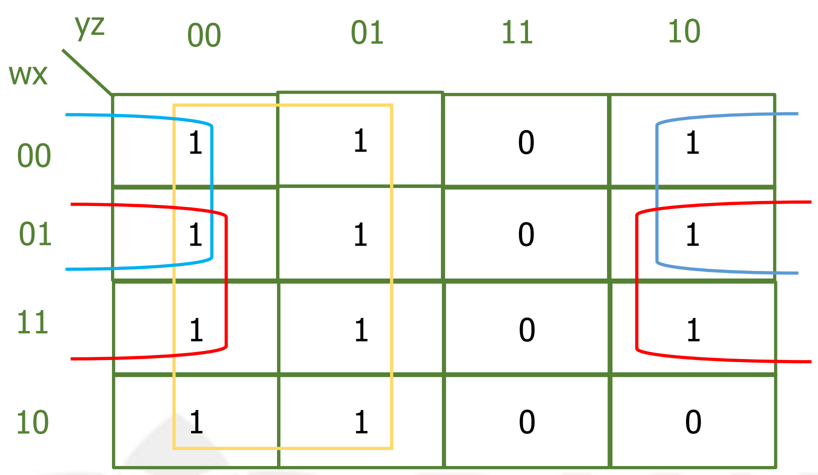
F = X′Z′ + X′Y′ + W′YZ′

15. Simplification of Boolean Function Using Four Variable Karnaugh Map (SOP Form)

Example: 11

Simplify the Boolean function F(w,x,y,z) = Σ(0,1,2,4,5,6,8,9,12,13,14)

Solution:



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

❖ Combining minterms $m_0,m_1,m_4,m_5,m_8,m_9,m_{12},m_{13}$, we get

$(w'+w'+w+w)(x'+x+x'+x)(y'+y')(z'+z) = y'$

❖Combining minterms m_0,m_2,m_4,m_6, we get $(w'+w')(x'+x)(y'+y)(z'+z') = w'z'$

❖Combining minterms m_4,m_6,m_{12},m_{14}, we get $(w'+w)(x+x)(y'+y)(z'+z') = xz'$

$$F = y' + w'z' + xz'$$

Rule of Thumb in four variable K-map

➢ One square represents one minterm, giving a term of four literals

➢ Two adjacent squares represent a term of three literals

➢ Four adjacent squares represent a term of two literals

➢ Eight adjacent squares represent a term of one literal

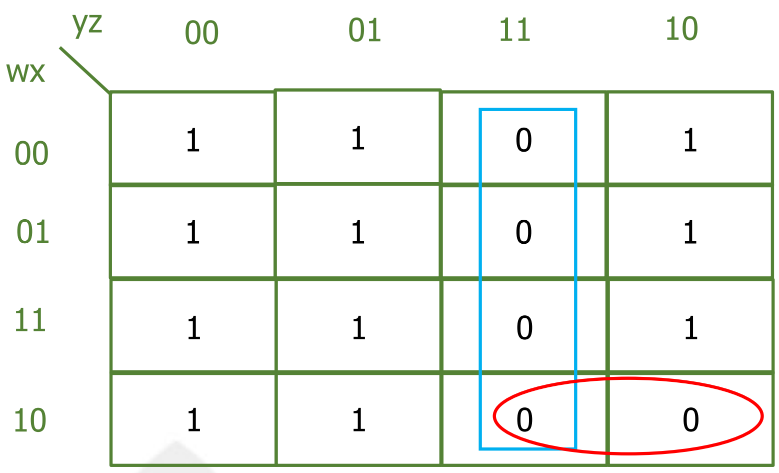
➢ Sixteen adjacent squares represent the function equal to 1

15. Simplification of Boolean Function Using Four Variable Karnaugh Map (POS Form)

Example: 12

Simplify the Boolean function F(w,x,y,z) = Π(3,7,10,11,15)

Solution:



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

❖ Combining maxterms M_3, M_7, M_{11}, M_{15}, we get (w.w.w′.w′) + (x.x′.x′.x) + (y′) + (z′) = (y′+z′)

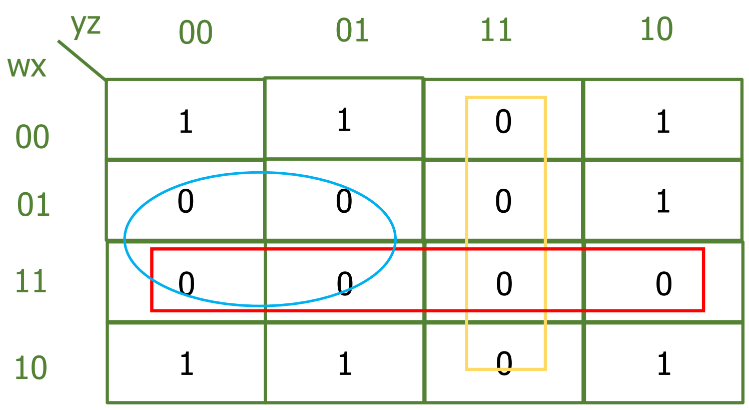
❖ Combining maxterms M_{10}, M_{11}, we get (w′) + (x) + (y′.y′) + (z′.z) = (w′+x+y′)

F = (y′+z′) (w′+x+y′)

Example: 13

Simplify the Boolean function F(w,x,y,z) = Π(3,7,10,11,15)

Solution:



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

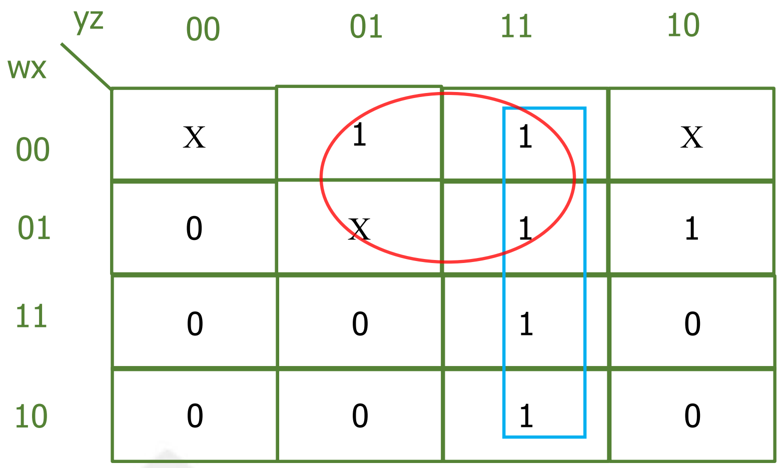
F = (y′+z′) (w′+x′) (x′+y)

15. Simplification of Boolean Function Using Don’t Care Conditions

Example: 14

Simplify the Boolean function F(w,x,y,z) = Σ(1,3,7,11,15) + d(w,x,y,z) = Σ(0,2,5)

Solution:



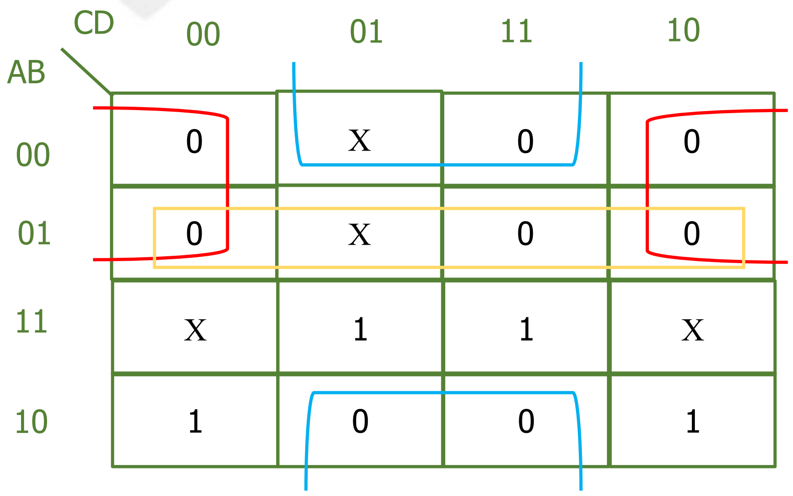
wx \ yz	00	01	11	10
00	X	1	1	X
01	0	X	1	1
11	0	0	1	0
10	0	0	1	0

F = w’z + yz

Example: 15

Simplify the Boolean function F in product of sums using the don’t care conditions d: F = Π(0,2,3,4,7,9,11) + d= Π(1,5,6,12,14)

Solution:



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

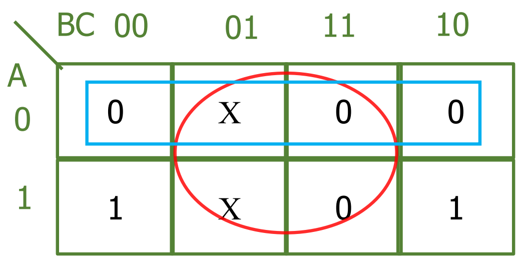
F = (A+D) (A+B’) (B+D’)

15. Simplification of Boolean Function Using Don’t Care Conditions

Example: 16

Simplify the Boolean function F = Π(0,2,3,7) + d= Π(1,5)

Solution:



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

F = A + C′

Example: 17

Simplify the Boolean function F in sum of products using the don’t care conditions d: F = B′C′D′ + BCD′ + ABCD′ ; d= B′CD′ + A′BC′D

Solution:

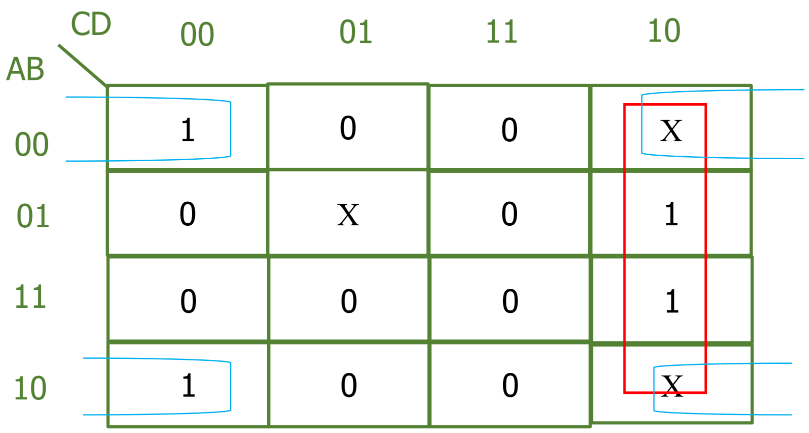
Convert F and d into canonical SOP form

F = B′C′D′[A+A′] + BCD′[A+A′] + ABCD′

F= A′B′C′D′ + AB′C′D′ + ABCD′ + A′BCD′ + ABCD′

F = A′B′C′D′ + A′BCD′ + AB′C′D′ + ABCD′ = Σ(0,6,8,14)

d = B′CD′[A+A′] + A′BC′D; d = AB′CD′ + A′B′CD′ + A′BC′D = Σ(2,5,10)



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

F = B′D′ + CD′

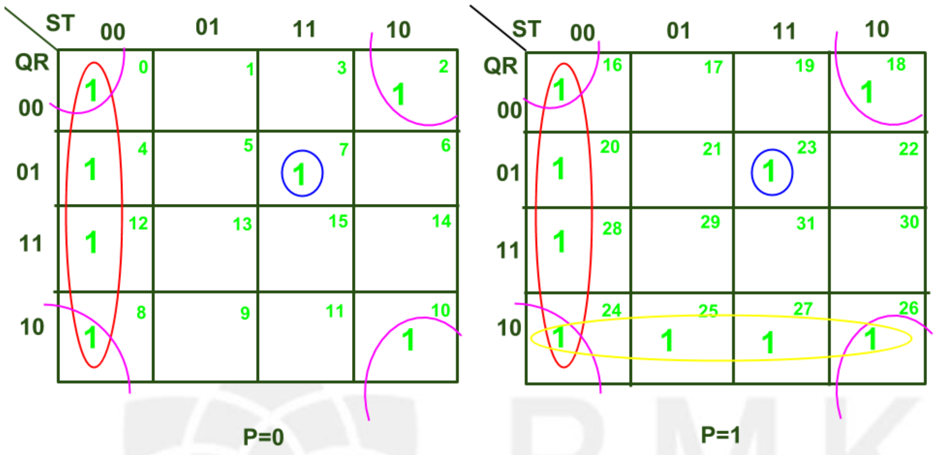
15. Simplification of Boolean Function Using Five Variable Karnaugh Map

Example: 18

Simplify the Boolean function F in sum of products

F(P,Q,R,S,T) = Σm(0,2,,4,7,8,10,12,16,18,20,23,24,25,26,27,28)

Solution:



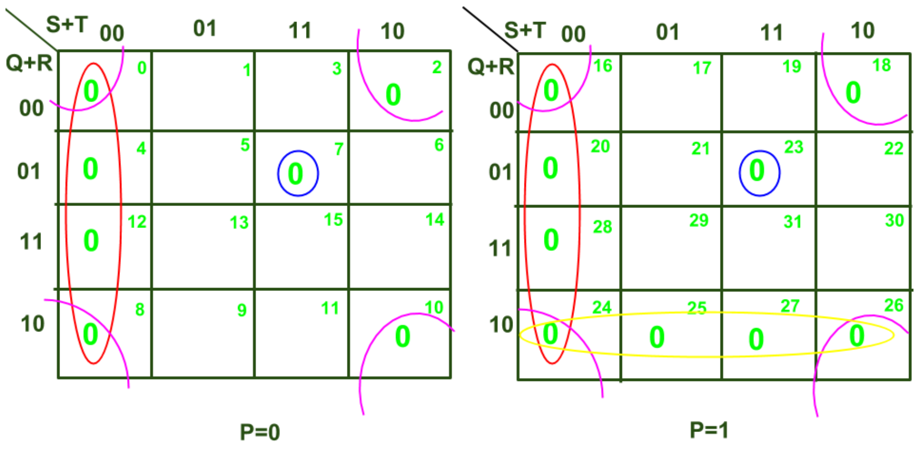
F = S’T’ + Q’RST + R’T’ + PQR’

Example: 19

Simplify the Boolean function F in product of sums

F(P,Q,R,S,T) = Πm(0,2,,4,7,8,10,12,16,18,20,23,24,25,26,27,28)

Solution:



F = (S+T) (Q+R’+S’+T’) (R+T) (P’+Q’+R)

15. Simplification using K-Maps

Gate Level Implementation :

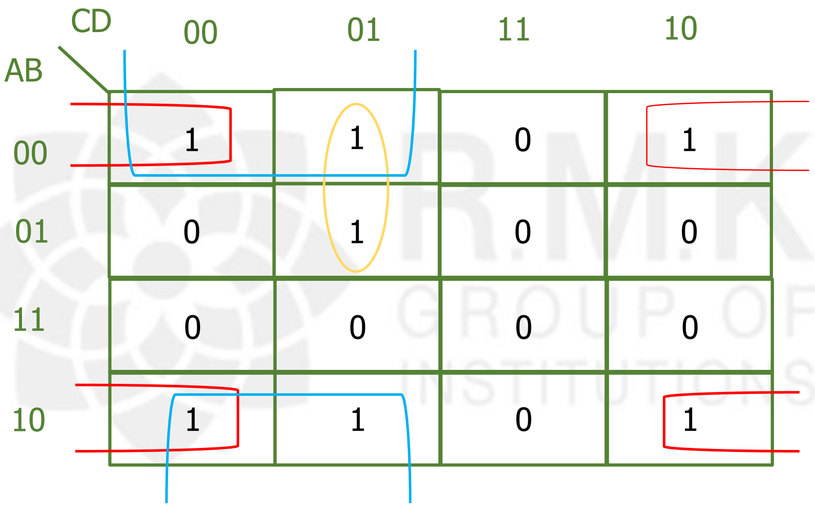
Example: 21

Implement the following Boolean function F, after simplifying in a) sum of products and b) product of sums

F(A,B,C,D) = Σm(0,1,2,5,8,9,10)

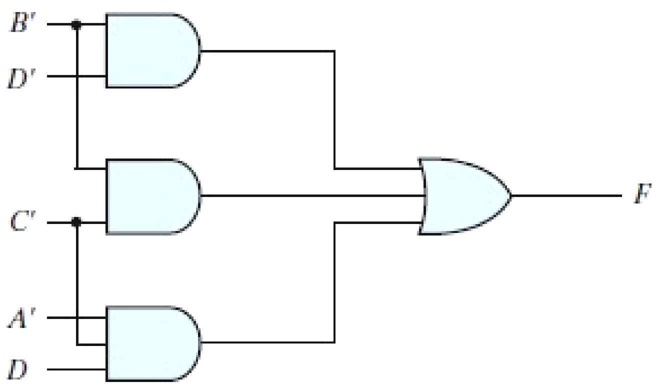
Solution:

a)



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

F = B′D′ + B′C′ + A′C′D



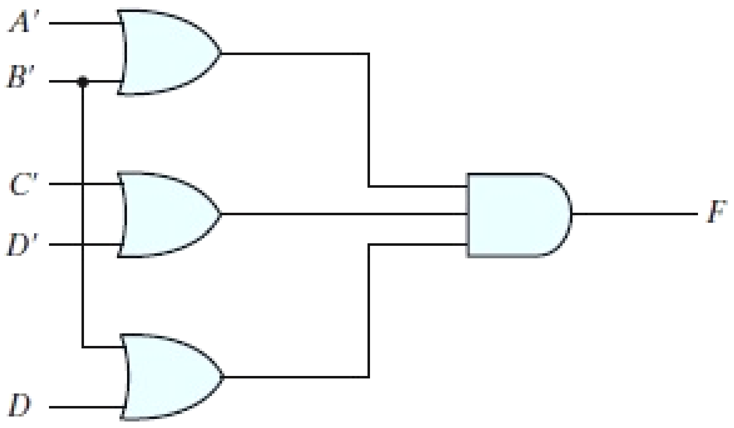
(a) $F = B'D' + B'C' + A'C'D$

15. Simplification using KMpas

b)

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

F = (B′+D)(A′+B′)(C′+D′)



(b) $F = (A' + B')(C' + D')(B' + D)$

16. LOGIC GATES

- Logic gates are electronic circuits that can be used to implement the Boolean Functions.
- Logic gates are also called switches.
- Logic gates are the basic building blocks of any digital system having one or more than one input and only one output. The relationship between the input and the output is based on a certain logic. Based on this, logic gates are named as AND gate, OR gate, NOT gate etc.

1-bit logic AND resembles binary multiplication:

$0 \bullet 0 = 0, \quad 0 \bullet 1 = 0,$

$1 \bullet 0 = 0, \quad 1 \bullet 1 = 1$

1-bit logic OR resembles binary addition, except for one operation:

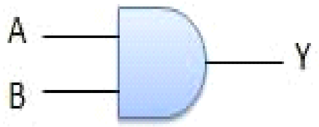
$0 + 0 = 0, \quad 0 + 1 = 1,$

$1 + 0 = 1, \quad 1 + 1 = 1 \ (\neq 10_2)$

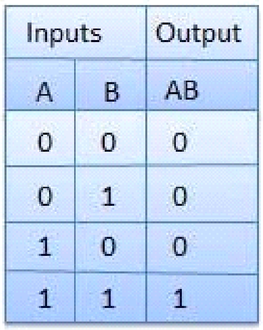
AND Gate

The AND gate performs logical multiplication, commonly known as AND function. The AND gate has two or more inputs and single output. The output of AND gate is HIGH only when all its inputs are HIGH (i.e. even if one input is LOW, Output will be LOW).

Logic diagram



Truth Table



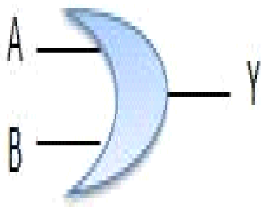
Inputs		Output
A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

16. LOGIC GATES

OR Gate

The OR gate performs logical addition, commonly known as OR function. The OR gate has two or more inputs and single output. The output of OR gate is HIGH only when any one of its inputs are HIGH (i.e. even if one input is HIGH, Output will be HIGH).

Logic diagram



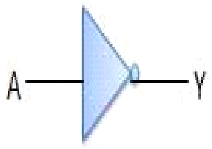
Truth Table

Inputs		Output
A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

NOT Gate

The NOT gate performs the basic logical function called inversion or complementation. NOT gate is also called inverter. The purpose of this gate is to convert one logic level into the opposite logic level. It has one input and one output. When a HIGH level is applied to an inverter, a LOW level appears on its output and vice versa

Logic diagram



Truth Table

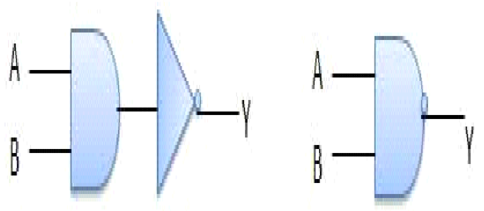
Inputs	Output
A	B
0	1
1	0

16. LOGIC GATES

NAND Gate

NAND gate is a cascade of AND gate and NOT gate, as shown in the figure below. It has two or more inputs and only one output. The output of NAND gate is HIGH when any one of its input is LOW (i.e. even if one input is LOW, Output will be HIGH).

Logic diagram



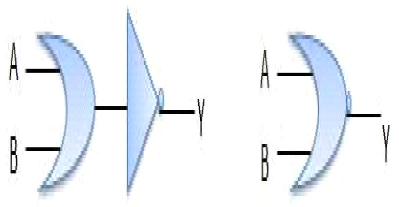
Truth Table

Inputs		Output
A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

NOR Gate

NOR gate is a cascade of OR gate and NOT gate, as shown in the figure below. It has two or more inputs and only one output. The output of NOR gate is HIGH when any all its inputs are LOW (i.e. even if one input is HIGH, output will be LOW).

Logic diagram



Truth Table

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

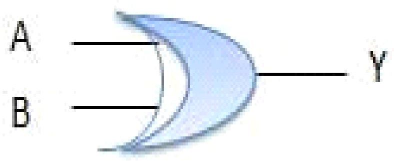
16. LOGIC GATES

XOR Gate

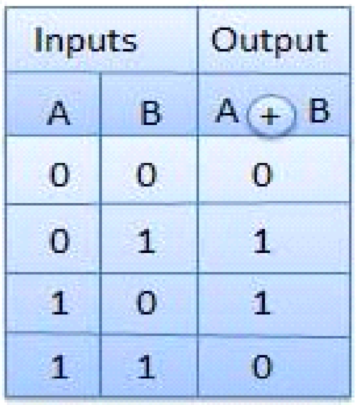
An Exclusive-OR (XOR) gate is gate with two or three or more inputs and one output. The output of a two-input XOR gate assumes a HIGH state if one and only one input assumes a HIGH state. This is equivalent to saying that the output is HIGH if either input X or input Y is HIGH exclusively, and LOW when both are 1 or 0 simultaneously.

It can be used in the half adder, full adder and subtractor. The exclusive-OR gate is abbreviated as EX-OR gate or sometime as X-OR gate. It has n input (n >= 2) and one output.

Logic diagram



Truth Table



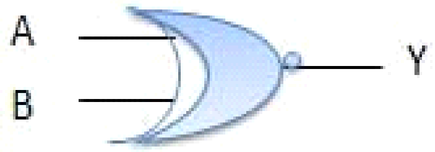
Inputs		Output
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

XNOR Gate

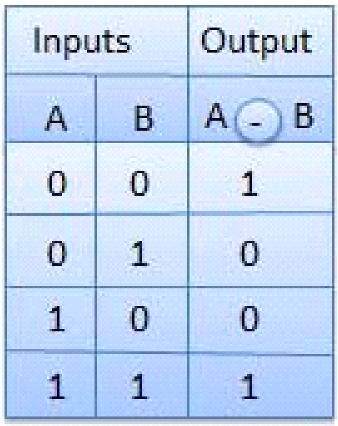
An Exclusive-NOR (XNOR) gate is gate with two or three or more inputs and one output. The output of a two-input XNOR gate assumes a HIGH state if all the inputs assumes same state. This is equivalent to saying that the output is HIGH if both input X and input Y is HIGH exclusively or same as input X and input Y is LOW exclusively, and LOW when both are not same. If X and Y are two inputs, then output F can be represented mathematically as F = X Y, Here denotes the XNOR operation. X Y and is equivalent to X.Y + X'.Y'

16. LOGIC GATES

Logic Diagram



Truth Table



Inputs		Output
A	B	$A \odot B$
0	0	1
0	1	0
1	0	0
1	1	1

17. UNIVERSAL GATES

- In addition to AND, OR, and NOT gates, other logic gates like NAND and NOR are also used in the design of digital circuits.
- A **universal gate** is a **gate** which can implement any Boolean function without need to use any other **gate**.
- The NAND and NOR gates are universal gates since NAND and NOR gates are economical and easier to fabricate all the basic gates like AND, OR and NOT that are used in all IC digital logic families.
- Any logic function can be implemented using NAND and NOR gates

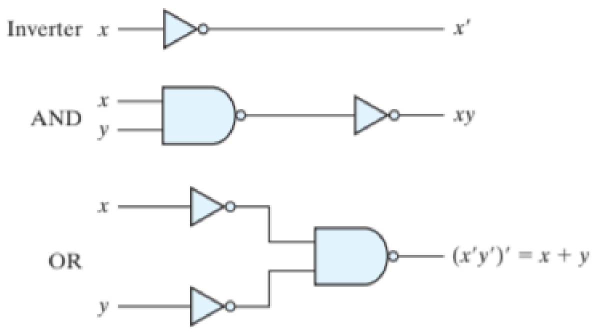
To prove that any Boolean function can be implemented using only NAND gates, we will show that the AND, OR, and NOT operations can be performed using only these gates.

NAND IMPLEMENTATION

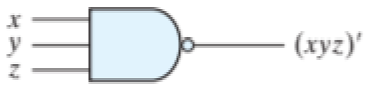
To prove that any Boolean function can be implemented using only NAND gates, we will show that all the basic logic operations can be performed using only these gates.

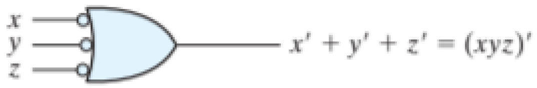
18. BASIC GATES IMPLEMENTATION USING NAND

- The NAND gate is called as a Universal Gate because any logic circuit or Boolean function can be implemented with NAND gate.
- The AND, OR implementations using NAND gate needs additional inverters in order to take the complement.
- The logic operations with NAND gates are shown below:



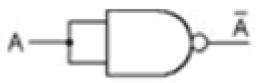
- The OR operation is achieved through a NAND gate with additional inverters in each input.
- The AND operation requires one NAND gate and an invertor.
- Two equivalent graphic symbols for the NAND gate are : AND-Invert and Invert-OR gates





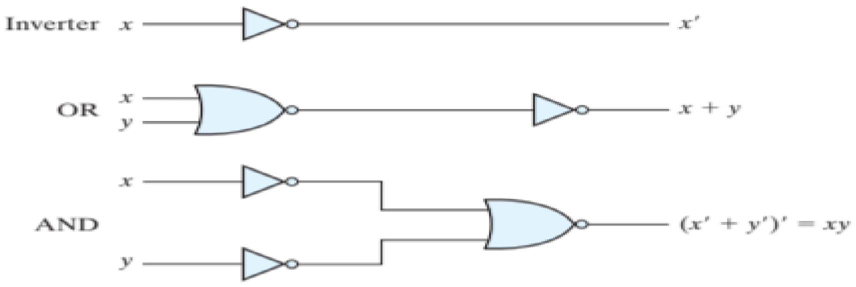
a) AND-Invert **b) Invert-OR**

- The AND-invert symbol consists of an AND graphic symbol followed by a small circle negation indicator referred to as a bubble.
- Invert-OR graphic symbol is preceded by a bubble in each input.
- The complement operation can also be obtained from a one-input NAND gate that behaves exactly like an inverter. The symbol is shown below.

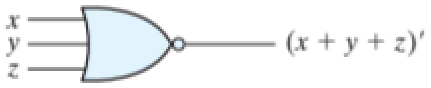


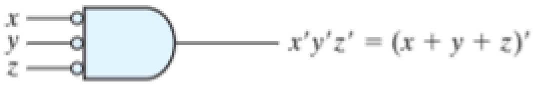
18. BASIC GATES IMPLEMENTATION USING NOR

- The NOR gate is another Universal Gate because any logic circuit or Boolean function can be implemented with NOR gate.
- The OR, AND implementations using NOR gate needs additional inverters in order to perform complement.
- NOR operation is the dual of the NAND operation.
- The logic operations with NOR gates are shown below:



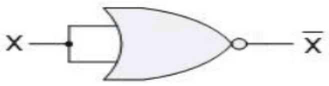
- The OR operation is achieved through a NOR gate followed with an additional inverter.
- AND operation is obtained with a NOR gate that has inverters in each input.
- Two equivalent graphic symbols for the NAND gate are : OR-Invert and Invert-AND





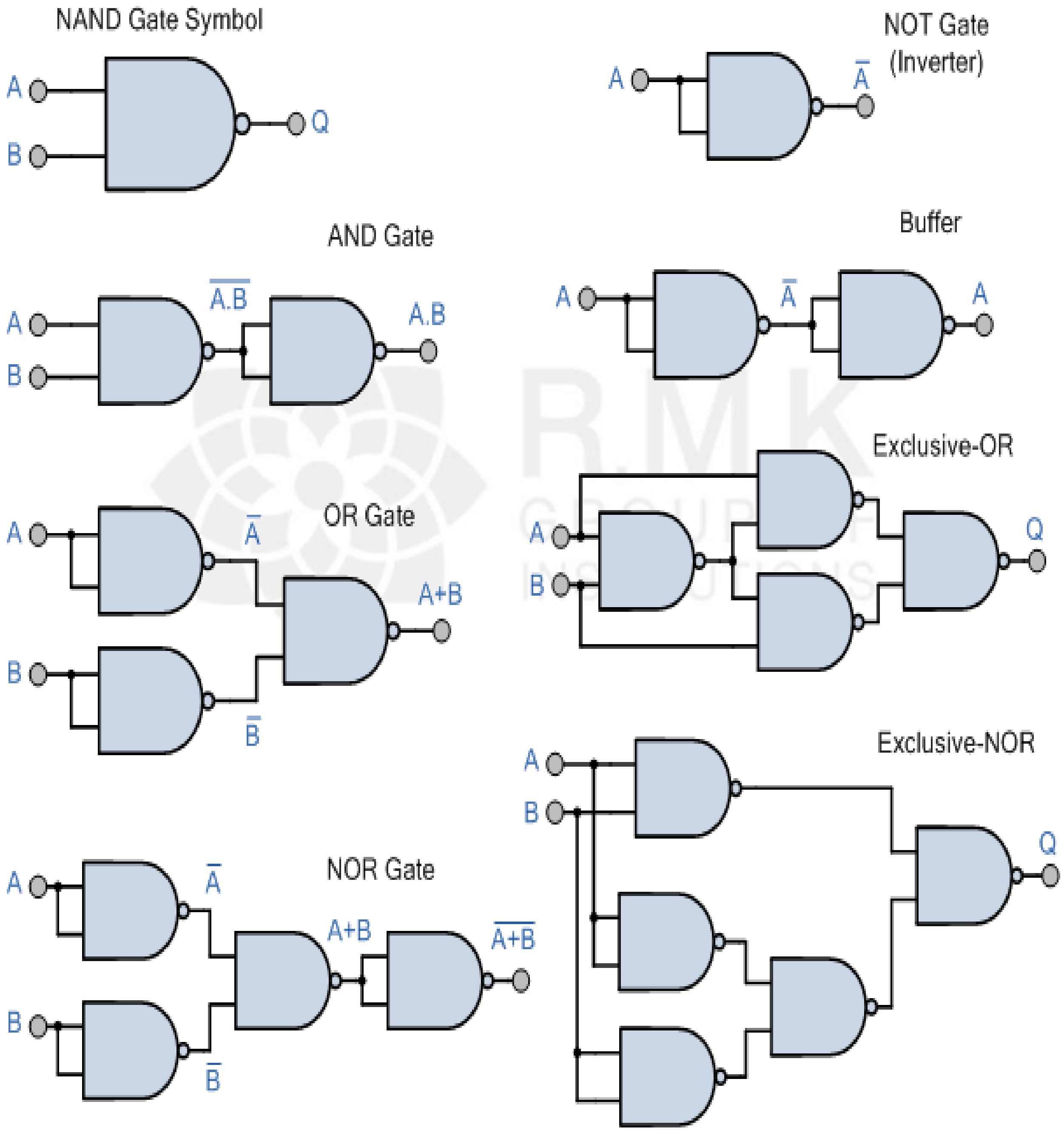
a) OR-Invert **b) Invert-AND**

- The OR-invert symbol defines the NOR operation as an OR followed by a complement.
- The invert-AND symbol complements each input and then performs an AND operation.
- The complement operation is obtained from a one input NOR gate that behaves exactly like an inverter and is shown below:



18. BASIC GATES USING UNIVERSAL GATE

NAND GATE AS UNIVERSAL GATE



18. BASIC GATES USING UNIVERSAL GATE

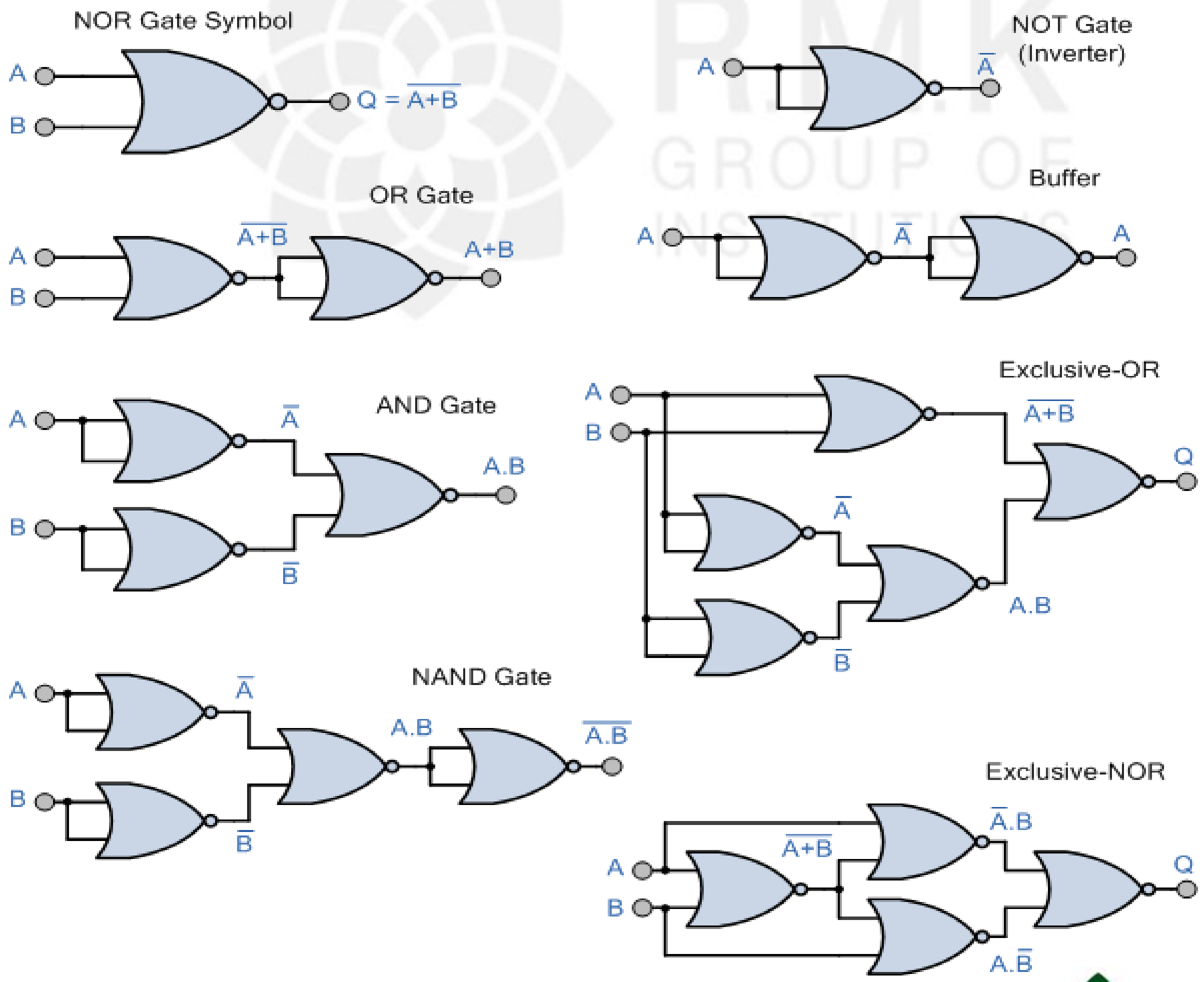
NORGATE AS UNIVERSAL GATE

The NAND and NOR gates are the complements of the previous AND and OR functions respectively and are individually a complete set of logic as they can be used to implement any other Boolean function or gate. But as we can construct other logic switching functions using just these gates on their own, they are both called a minimal set of gates. Thus the NAND and the NOR gates are commonly referred to as Universal Logic Gates.

NOR IMPLEMENTATION

The NOR gate represents the complement of the OR operation. Its name is an abbreviation of NOT OR. The graphic symbol for the NOR gate consists of an OR symbol with a bubble on the output, denoting that a complement operation is performed on the output of the OR gate

NOR GATE AS UNIVERSAL GATE



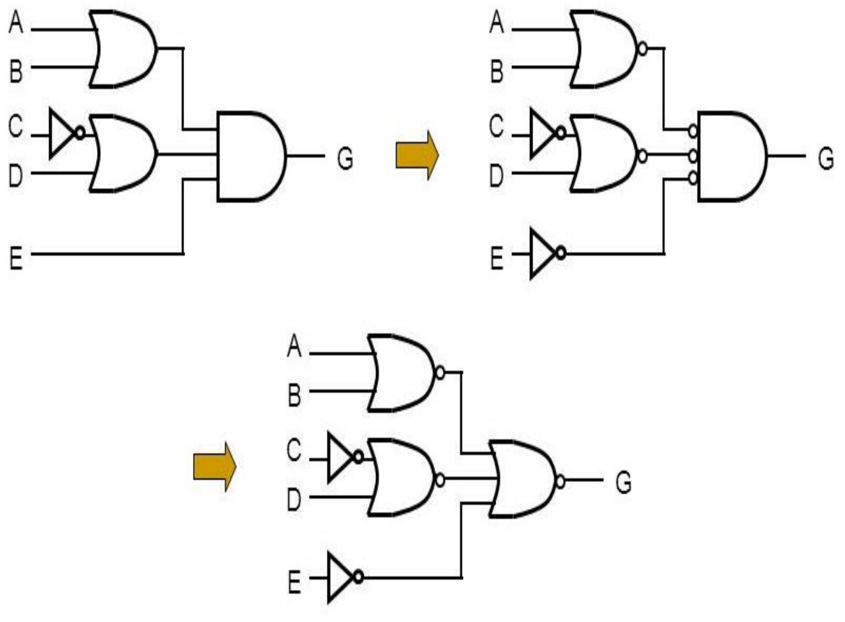
19. IMPLEMENT BOOLEAN FUNCTIONS USING NOR GATES

EXAMPLE 1:

IMPLEMENT THE FUNCTION G= (A+B)(C′+D)E
USING NOR GATE

Solution:

•Draw the logic diagram
•Add bubble to the output of OR gate
•Add bubble to the input of AND gate
•Do bubble compensation
Represent all gates by standard NOR representation



19. NAND IMPLEMENTATION

Procedure for NAND Implementation

Step 1. Draw the AND-OR-NOT logic diagram.

Step 2. Convert all AND gates to NAND gates with AND-invert graphic symbols and Convert all OR gates to NAND gates with invert-OR graphic symbols.

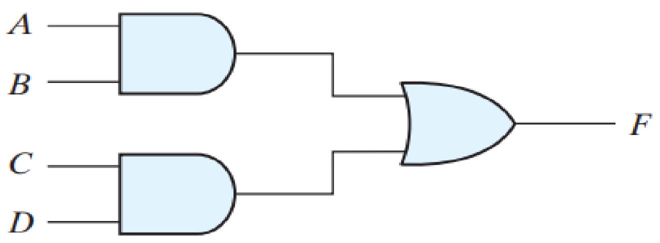
Step 3. Check all the bubbles in the diagram. For every bubble that is not compensated by another small circle along the same line, insert an inverter (a one-input NAND gate) or complement the input literal.

Step 4. Convert all Invert-OR gates to NAND gate.

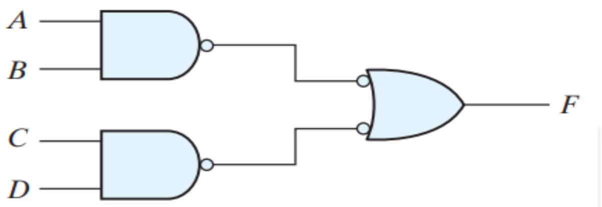
Converting AND-OR circuit to all NAND diagram

Example 1: Implement the Boolean Function F = AB + CD using NAND Gates. (Two-Level Circuit)

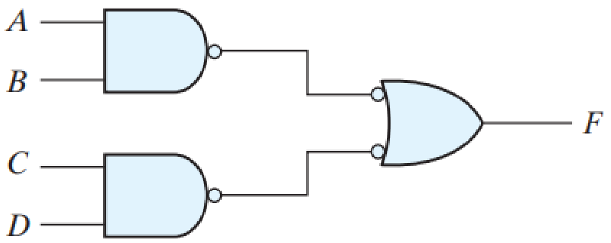
- Step 1 : Draw the AND-OR-NOT logic diagram



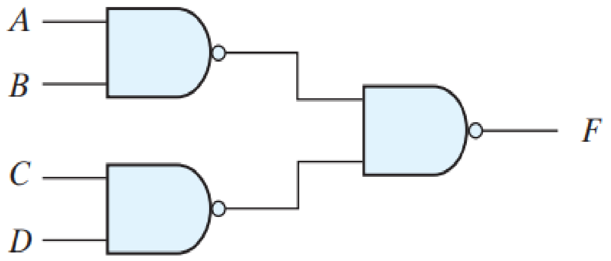
- Step 2: Convert all AND gates to NAND gates with AND-invert graphic symbols and Convert all OR gates to NAND gates with invert-OR graphic symbols.



- Step 3. Check all the bubbles in the diagram. For every bubble that is not compensated by another small circle along the same line, insert an inverter (a one-input NAND gate) or complement the input literal.



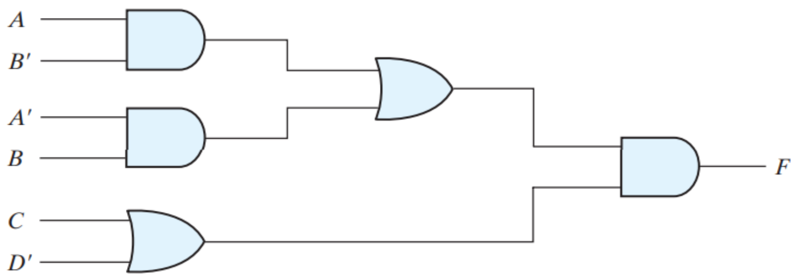
- Step 4. Convert all Invert-OR gates to NAND gate.



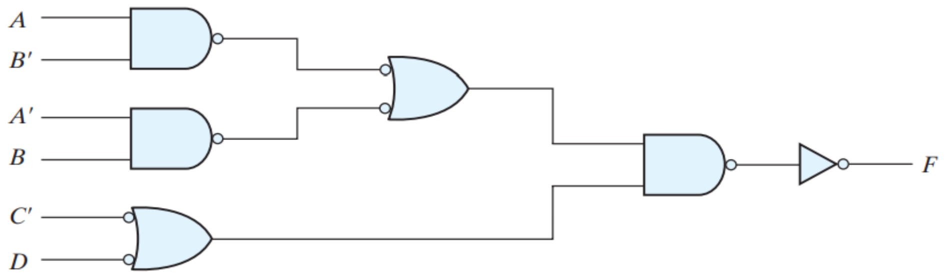
Converting AND-OR circuit to all NAND diagram

Example 2: Implement the Boolean Function F = (AB' + A'B)(C + D') using NAND Gates. (Multi-Level Circuit)

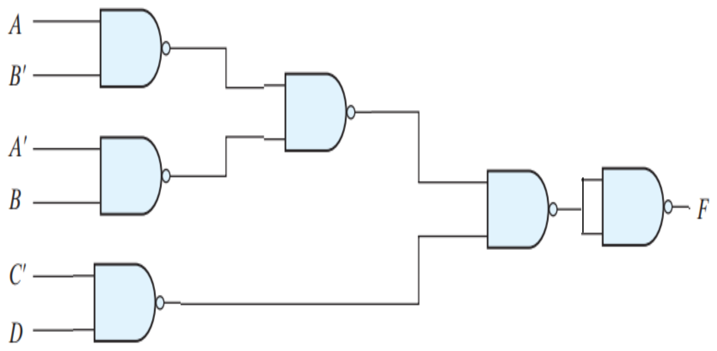
- Step 1 : Draw the AND-OR-NOT logic diagram



- Step 2: Convert all AND gates to NAND gates with AND-invert graphic symbols and Convert all OR gates to NAND gates with invert-OR graphic symbols. Step 3. Check all the bubbles in the diagram. For every bubble that is not compensated by another small circle along the same line, insert an inverter (a one-input NAND gate) or complement the input literal.



- Step 4. Convert all Invert-OR gates to NAND gate.



20. NOR IMPLEMENTATION

Procedure for NOR Implementation

Step 1. Draw the AND-OR-NOT logic diagram

Step 2. Convert all OR gates to NOR gates with OR-invert graphic symbols and Convert all AND gates to NOR gates with invert-AND graphic symbols.

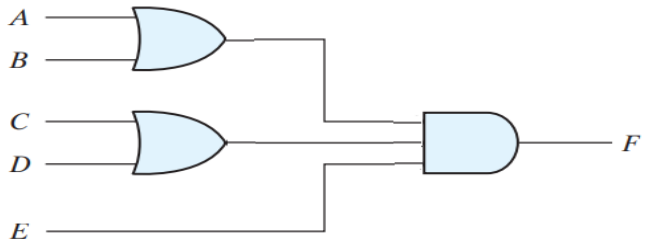
Step 3. Check all the bubbles in the diagram. For every bubble that is not compensated by another small circle along the same line, insert an inverter (a one-input NOR gate) or complement the input literal.

Step 4. Convert all Invert-AND gates to NOR gates

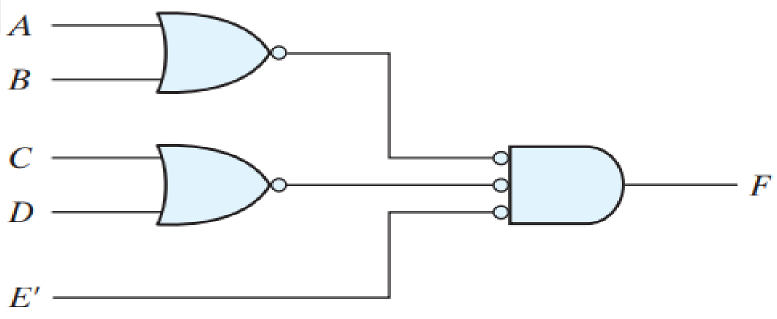
Converting OR-AND circuit to all NOR diagram

Example 1: Implement the Boolean Function F = (A + B)(C + D)E using NOR Gates. (Two-Level Circuit)

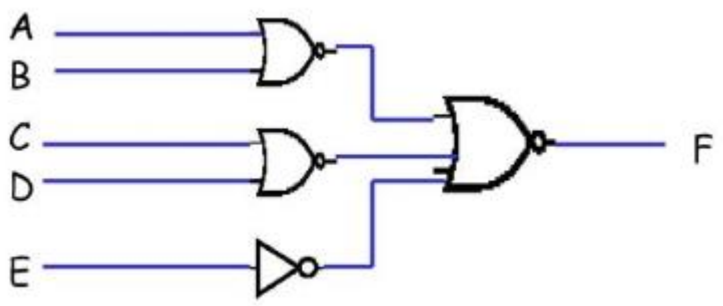
- Step 1 : Draw the AND-OR-NOT logic diagram



- Step 2: Convert all OR gates to NOR gates with OR-invert graphic symbols and Convert all AND gates to NOR gates with invert-AND graphic symbols. Step 3. Check all the bubbles in the diagram. For every bubble that is not compensated by another small circle along the same line, insert an inverter (a one-input NOR gate) or complement the input literal.



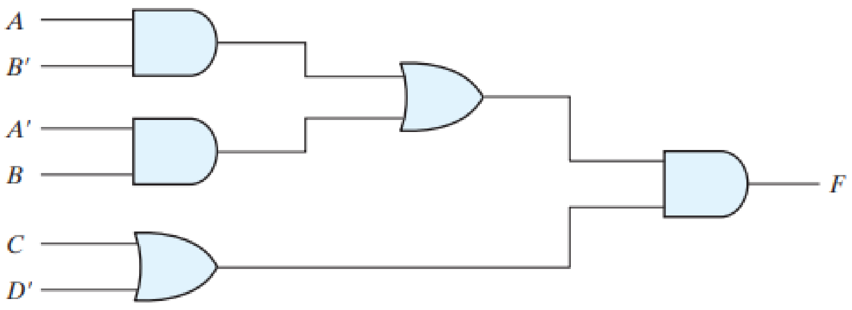
- Step 4. Convert all Invert-AND gates to NOR gates



Converting OR-AND circuit to all NOR diagram

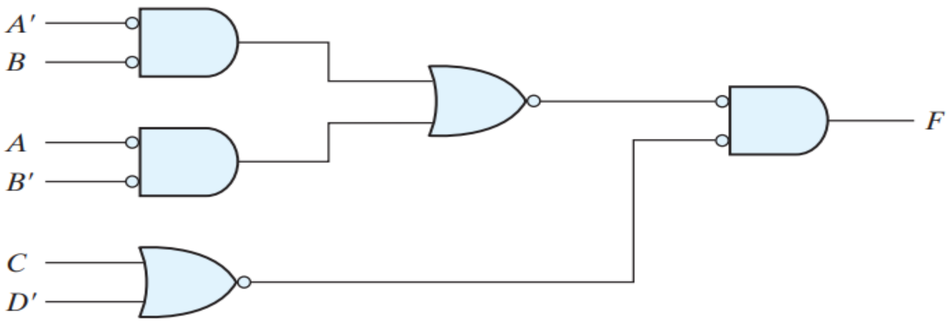
Example 2: Implement the Boolean Function F = (A' B + AB')(C + D') using NOR Gates. (Multi-Level Circuit)

- Step 1 : Draw the AND-OR-NOT logic diagram

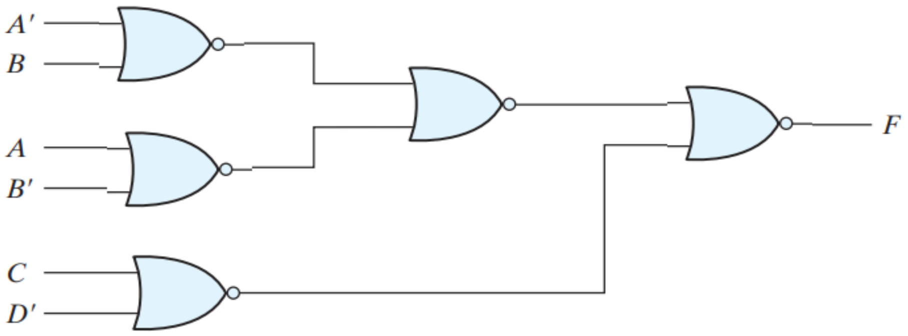


(a) AND–OR gates

- Step 2: Convert all OR gates to NOR gates with OR-invert graphic symbols and Convert all AND gates to NOR gates with invert-AND graphic symbols. Step 3. Check all the bubbles in the diagram. For every bubble that is not compensated by another small circle along the same line, insert an inverter (a one-input NOR gate) or complement the input literal.



- Step 4. Convert all Invert-AND gates to NOR gates



Video Links

Sl. No.	Topic	Video Link
1	Number Systems	https://www.youtube.com/watch?v=pLTDDsvMnFQ
2	Boolean Algebra	https://www.youtube.com/watch?v=czFn9Wg3k4o
3	BCD Addition	https://www.youtube.com/watch?v=a2gbCeIilBU
4	Universal gates	https://www.youtube.com/watch?v=wIeqYG4c6Ho
5	Logic Minimization using Karnaugh Maps	https://www.youtube.com/watch?v=ygm25sqqepg
6	Karnaugh Minimization using Maxterms	https://www.youtube.com/watch?v=i_HYxdri69Y

Assignment Questions

1. Express the following numbers in decimal: (**CO1, K1)**

   (a) $(10110.010)_2$ (b) $(16.5)_{16}$ (c) $(26.24)_8$ (d) $(FAFA)_{16}$

   (e) $(1010.1010)_2$ (f) $(DADA.B)_{16}$

2. Convert the following numbers with the indicated bases to decimal: (**CO1, K1)**

   (a) $(4310)_5$ (b) $(198)_{12}$ (c) $(735)_8$ d) $(525)_6$

3. Convert hexadecimal number 64CD to binary and then convert it from binary to octal number system. (**CO1, K1)**

4. Convert the decimal number 431 to binary in two ways. (**CO1, K1)**

   i) Convert directly to binary.

   ii) Convert first to hexadecimal and then from hexadecimal to binary.

5. Perform 10's complement subtraction for the following:

   (a) 6428 - 3409 (b) 1631 - 745

6. Perform 2's complement subtraction for the following:

   (a) 1001 - 101000 (b) 110000 - 10101

7. Formulate a weighted binary code for the decimal digits using weights

   (a) 6, 3, 1, 1 (b) 6, 4, 2, 1

8. Find the complement of the following expressions:

   (a) xy' + x'y (b) (A 'B + CD) E' + E

   (c) (x' + y + z')(x + y')(x + z)

9. Express the complement of the following functions in sum-of-minterms form:

   (a) F(A, B,C, D) = ∑ (3, 5,9, 11, 15) (b) F(x, y, z) = ∏ (2, 4, 5, 7)

10. Convert each of the following expressions into sum of products and product of sums:

    (a) (AB + C)(B + C′ D) (b) x' + x(x + y')(y + z')

11. Simplify the following Boolean functions, using three-variable maps:

    (a) F(x, y, z) = ∑ (0 , 1, 5, 7)

    (b) F(x, y, z) = ∑(1, 2, 3, 6, 7)

    (c) F(x, y, z) = ∑ (0. 1 , 6, 7)

    (d) F(x, y, z) = ∑ (0, 1, 3, 4, 5)

    (e) F(x, y, z) = ∑ (1,3, 5, 7)

    (f) F(x, y, z) = ∑ (1,4,5,6, 7)

Assignment Questions

12. Find all the prime implicants for the following Boolean functions and determine which are essential:
    - a) F (w, x, y, z) = ∑ (0, 2, 4, 5, 6, 7, 8, 10, 13, 15)
    - b) F (A, B, C, D) = ∑(0, 2, 3, 5, 7, 8, 10, 11 14,15)
    - c) F (A, B, C, D) = ∑ (1, 3, 4, 5, 10, 11, 12, 13, 14,15)
    - d) F (w, x, y, z) = ∑ (1, 3, 6, 7, 8, 9, 12, 13, 14, 15)
    - e) F(A, B,C,D) = ∑ (0, 2, 3, 5, 7, 8, 10, 11, 13, 15)
    - f) F (w, x, y, z) = ∑ (0, 2, 7, 8, 9, 10, 12, 13, 14, 15)

13. Simplify the following Boolean functions to product-of-sums form:

(a) F (w, x, y, z) = ∑ (0, 1, 2, 5, 8, 10, 13)

(b) F (A, B, C, D) = ∏ (1, 3, 5, 7, 13,15)

(c) F (A, B, C, D) = ∏ (1, 3, 6, 9, 11,12, 14)

14. Simplify the following expressions to (1) sum-of-products and (2) products-or-sums:
    - a) x'y' + y'z' + yz’ + xy
    - b) ACD' + C' D + AB' + ABCD
    - c) ((A + C' + D')(A ' + B' + D')(A ' + B + D')(A' + B + C')
    - d) ABC' + AB'D + BCD

15. Simplify the following Boolean function F, together with the don't-care conditions d, and then express the simplified function in sum-of-minterms form:

(a) F(x, y, z) = ∑ (2, 3, 4, 6, 7) + d (x, y, z) = ∑ (0, 1, 5)

(b) F(A, B, C, D) = ∑ (0, 6, 8, 13, 14) + d (A,B,C,D) = ∑ (2, 4, 10)

16. Simplify the following functions and implement them with two-level NAND gate circuits:

(a) F(A, B, C, D) = A'B'C + AC' + ACD + ACD' + A' B'D'

(b) F(A, B, C, D) = AB + A'BC + A'B’C’D

16. Simplify the following functions, and implement them with two-level NOR gate circuits:

(a) F = wx' + y' z' + w'yz'

(b) F (w, x, y, z) = ∑ (1, 2, 13, 14)

(c) F(x, y, z) = [(x + y) (x' + z)]'

17. Implement the following Boolean function F, using the two-level forms of logic (a) NAND-AND, (b) AND-NOR, (c) OR-NAND, and (d) NOR-OR

F(A, B, C, D) = ∑ (0, 4, 8, 9, 10,11,12, 14)

Part A Questions & Answers

Q. No.	Questions	K Level	CO Mapping
1.	What is the difference between analog and digital systems? In a digital system the physical quantities or signals can assume only discrete values, while in analog systems the physical quantities or signals vary continuously over a specified range.	K1	CO1
2	What is a binary number system ? The number system with base (or radix) two is known as the binary number system. Only two symbols are used to represent the numbers in the system and these are 0 and 1	K1	CO1
3	Define a Digital system Electronic systems are most reliable when designed for two state operation and this binary system is made use of in digital system.	K1	CO1
4	What is a decimal number system? A decimal number system has a radix (Base r=10) and uses symbols 0,1,2,3,4,5,6,7,8,9. Eg. $(237)_{10}$	K1	CO1
5	Convert $(196.062)_{10}$ to octal. $(196.062)_{10} = (304.03757)_8$	K1	CO1
6	Convert $(10.175)_8$ to its decimal equivalent. $(10.175)_8 = (8.244135)_{10}$	K1	CO1
7	Define bit, byte and nibble A binary digit is called bit . A group of 8 bits is called byte. A group of 4 bits is called Nibble.	K1	CO1
8	Convert $(1128)_{16}$ to decimal. $(1128)_{16} = (4392)_{10}$	K1	CO1
9	Convert $(0.6875)_{10}$ to binary $(0.6875)_{10} = (0.1011)_2$	K1	CO1
10.	Find the octal equivalent of hexadecimal numbers AB.CD $(AB.CD)_8 = (253.632)_8$	K1	CO1

R.M.K GROUP OF INSTITUTIONS

Part A Questions & Answers

Q. No.	Questions	K Level	CO Mapping
11.	**Convert $(126)_{10}$ to octal number and binary number** $126)_{10} = (176)_8$ $(126)_{10} = (111111)_2$	K1	CO1
12	**Subtract the following numbers:** **i) 101_2 from 1001_2** Solution: 101 from 1001 $\begin{array}{r} 0\ 10 \quad \text{Borrow} \\ \cancel{1}\ \cancel{0}\ 0\ 1 \\ \underline{1\ 0\ 1} \\ 1\ 0\ 0_2 \end{array}$ **ii) 111_2 from 1000_2** Solution: 111 from 1000 $\begin{array}{r} 0\ \ 1\ 1\ 10 \quad \text{borrow} \\ \cancel{1}\ \cancel{0}\ 0\ 0 \\ \underline{1\ 1\ 1} \\ 0\ 0\ 0\ 1 \end{array}$	K1	CO1
13.	**iii) 1010101.10_2 from 1111011.11_2** Solution: 1010101.10 from 1111011.11 $\begin{array}{r} 1 \quad \text{Borrow} \\ 1\ 1\ 1\ 1\ 0\ 1\ 1\ .\ 1\ 1 \\ \underline{1\ 0\ 1\ 0\ 1\ 0\ 1\ .\ 1\ 0} \\ 1\ 0\ 0\ 1\ 1\ 0\ .\ 0\ 1_2 \end{array}$ **iv) 11010.101_2 from 101100.011_2** Solution:11010.101 from 101100.011 $\begin{array}{r} 1 \\ 10 \quad 0\ \cancel{10}\ \ 10 \quad \text{Borrow} \\ \cancel{1}\ 0\ 1\ \cancel{1}\ \cancel{0}\ 0\ .\ 0\ 1\ 1 \\ \underline{1\ 1\ 0\ 1\ 0\ .\ 1\ 0\ 1} \\ 1\ 0\ 0\ 0\ 1\ .\ 1\ 1\ 0_2 \end{array}$	K1	CO1

GROUP OF INSTITUTIONS

Part A Questions & Answers

Q. No.	Questions	K Level	CO Mapping
14.	**list out the operation carried out in 1’s complement** The operation is carried out by means of the following steps: (i) At first, 1’s complement of the subtrahend is found. (ii) Then it is added to the minuend. (iii) If the final carry over of the sum is 1, it is dropped and the result is positive. (iv) If there is no carry over, the 1’s complement of the sum will be the result and it is negative.	K1	CO1
15.	**list out the operation carried out in 2’s complement** The operation is carried out by means of the following steps: (i) At first, 2’s complement of the subtrahend is found. (ii) Then it is added to the minuend. (iii) If the final carry over of the sum is 1, it is dropped and the result is positive. (iv) If there is no carry over, the two’s complement of the sum will be the result and it is negative.	K1	CO1
16.	**list out the operation carried out in 9’s complement.** the operation is carried out by means of the following steps: (i) At first, 9’s complement of the subtrahend is found. (ii) Then it is added to the minuend. (iii) If the final carry over of the sum is 1, it is dropped and the result is positive. (iv) If there is no carry over, the 9’s complement of the sum will be the result and it is negative.	K1	CO1
17.	**list out the operation carried out in 10’s complement.[CO1,K1]** The operation is carried out by means of the following steps: (i) At first, 10’s complement of the subtrahend is found. (ii) Then it is added to the minuend. (iii) If the final carry over of the sum is 1, it is dropped and the result is positive. (iv) If there is no carry over, the 10’s complement of the sum will be the result and it is negative.	K1	CO1

Part A Questions & Answers

Q. No.	Questions	K Level	CO Mapping
18.	**List the advantages and disadvantages of BCD code?** The advantages of BCD code are a. Any large decimal number can be easily converted into corresponding binary number b. A person needs to remember only the binary equivalents of decimal number from 0 to 9. c. Conversion from BCD into decimal is also very easy. The disadvantages of BCD code are a. The code is least efficient. It requires several symbols to represent even small numbers. b. Binary addition and subtraction can lead to wrong answer. c. Special codes are required for arithmetic operations. d. This is not a self-complementing code. e. Conversion into other coding schemes requires special methods.	K1	CO1
19.	**Represent (28)10 and (53)10 in 8421 BCD notation** Solution: (28)10 in BCD notation can be represented as (0010 1000)bCD. Similarly, (53)10 in BCD notation can be represented as (0101 0011)BCD.	K2	CO1
20.	**What are the needs for binary codes? (CO1, K1)** Code is used to represent letters, numbers and punctuation marks. Coding is required for maximum efficiency in single transmission. Binary codes are the major components in the synthesis (artificial generation) of speech and video signals. By using error detecting codes, errors generated in signal transmission can be detected. Codes are used for data compression by which large amounts of data are transmitted in very short duration of time.	K1	CO1
21.	**What is a weighted code? Give example. (CO1,K1)** Weighted binary code are those which obey the positional weighting principle. Example. 8421 BCD code	K1	CO1

Part A Questions & Answers

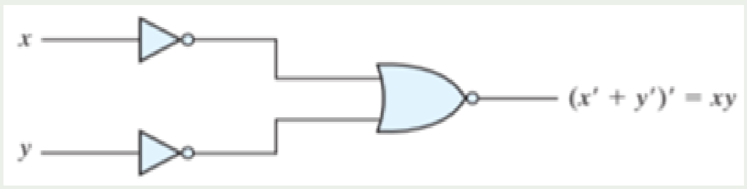
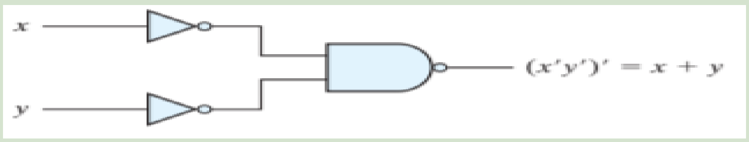
Q. No.	Questions	K Level	CO Mapping
21	**What is a non-weighted binary code ?Give example (CO1,K1)** Non-weighted codes are the codes that are not positional weighted. That is each position within the binary number is not assigned a fixed value. Example. Excess 3 code, Gray code.	K1	CO1
22.	**Define BCD code. (CO1,K1)** BCD is a commonly used Weighted Numeric code. Each digit of a decimal number (0-9) is represented by a 4-bit binary number (i.e. 0000 through 1001) Consider the decimal number 185. Its corresponding BCD and binary equivalent are given as follows. $(185)_{10} = (0001\ 1000\ 0101)_{BCD} = (10111001)_2$	K1	CO1
23.	**What is a gray code? (CO1,K1)** The gray code is a binary code in which each successive number differs in only one bit position. It is a non-weighted code and not an arithmetic code. Gray code is also called as unit distance code. The Gray code is used in applications in which the normal sequence of binary numbers generated by the hardware may produce an error or ambiguity during the transition from one number to the next.	K1	CO1
24.	**What is an Excess-3 code? (CO1,K1)** Excess-3 is an unweighted code in which each coded combination is obtained from the corresponding binary value plus 3. The excess-3 code is a self complementing code.	K1	CO1
25	**State the sequence of operator precedence in Boolean expression** Parenthesis NOT AND OR	K1	CO1
26	**State DeMorgan theorem** The complement of the sum is equal to the product of complements i) $(x + y)' = x' . y'$ The complement of the product is equal to sum of complements. ii) $(x . y)' = x' + y'$	K1	CO1

R.M.K GROUP OF INSTITUTIONS

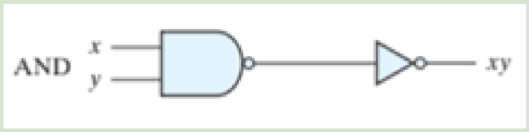
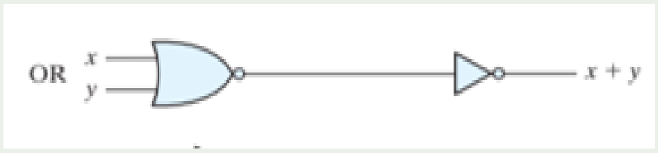
Part A Questions & Answers

Q. No.	Questions	K Level	CO Mapping
27	**State the Duality principle** The important property of Boolean algebra is the duality principle. It states that every algebraic expression deducible from the postulates of Boolean algebra remains valid if the operators and identity elements are interchanged. Duality principle states that a Boolean expression can be obtained from other expression by, Converting AND to OR Converting OR to AND Complementing/negating the 0's or 1's appearing in the expression	K1	CO1
28	**State the Consensus theorem** A consensus term/redundant term is the conjunction of unique literals of the terms, excluding the literal that appears negated in one term and unnegated in the other. The redundant term is eliminated. <u>Conditions to apply Consensus Theorem</u> Three variables/literals must be present in the expression. Each variable/literal is repeated twice. One variable/variable must be present in both negated and unnegated form. $AB+A'C+BC = AB+A'C$ $(A+B).(A'+C).(B+C) =(A+B).(A'+C)$	K1	CO1

Part A Questions & Answers

Q. No.	Questions	K Level	CO Mapping
29	**State the Absorption theorem** i) A + AB = A ii) ii) A(A+B) = A	K1	CO1
30	What are the advantages of simplifying a Boolean expression? The circuit gets simplified The number of gates required is reduced Cost of the circuit is reduced	K1	CO1
31	**Define Universal Gates** A universal gate is a gate which can implement any Boolean function without need to use any other gate type. The NAND and NOR gates are universal gates. In practice, this is advantageous since NAND and NOR gates are economical and easier to fabricate and are the basic gates used in all IC digital logic families	K1	CO1
32	**Which gates are called as universal gates?** •NAND gate NOR gate	K1	CO1
33	**What are the advantages of universal gates?** Ans: 1. NAND and NOR are called as universal gates. 2. They are economical and easier to fabricate 3. Any logic circuit or Boolean function can be implemented with NAND/NOR gates	K1	CO1
34	**Implement AND operation using NOR gate.** The AND operation can be implemented using NOR as follows 	K1	CO1
35	**Implement OR operation using NAND gate.** The OR operation can be implemented using NAND as follows 	K1	CO1

Part A Questions & Answers

Q. No.	Questions	K Level	CO Mapping
36	**Implement AND operation using NAND gate . (CO1, K2)** The AND operation can be implemented using NAND gate as follows 	K1	CO1
37	**Implement OR operation using NOR gate** The OR operation can be implemented using NOR gate as follows 	K1	CO1
38	**List out the advantages and disadvantages of K-map method?** **The advantages of the K-map method are** •It is a fast method for simplifying expression up to four variables. •It gives a visual method of logic simplification. •Prime implicants and essential prime implicants are identified fast. •Suitable for both SOP and POS forms of reduction. •It is more suitable for class room teachings on logic simplification. •Kmap is a more orderly process with well defined steps as compared with the trial and error process sometimes used in algebraic simplification •Requires fewer steps especially for expressions containing many terms, and it always produces a minimum expression	K1	CO1
39	**The disadvantages of the K-map method are (CO1, K1)** i. It is not suitable for computer reduction. ii. K-maps are not suitable when the number of variables involved exceed four. iii. Care must be taken to fill in every cell with the relevant entry, such as a 0, 1 (or) don't care terms.	K1	CO1
40	**What are don't cares? (CO1, K1)** Don't care term is a minterm for which the combinational logic may output either 1 or 0.	K1	CO1

Part B Questions

Q. No.	Questions	K Level	CO Mapping
1	Explain the various binary codes used in digital systems	K2	CO1
2	Show that the Excess – 3 code is self – complementing	K4	CO1
3	Explain how you will construct an (n+1) bit Gray code from an n bit Gray code	K4	CO1
4	Prove that (x1+x2).(x1′. x3′+x3) (x2′ + x1.x3) =x1′x2	K4	CO1
5	State and Prove idempotent laws of Boolean algebra	K2	CO1
6	State and Prove Demorgan’s theorem	K2	CO1
7	State and Prove the postulates and laws of Boolean algebra	K2	CO1
8	Show that the NAND operation is not distributive over the AND operation	K4	CO1
9	Simplify the following Boolean expressions to a minimum number of literals: i. ABC + A′B + ABC′ ii. x′yz + xz iii. (x + y)′ (x′ + y′) iv. xy + x(wz + wz′) v. (BC′ + A′D) (AB′ + CD′)	K3	CO1
10	Simplify the following boolean expressions: i. AB′ + ABD + ABD′ + A′C′D′ + A′BC′ ii. BD + BCD′ + AB′C′D′ iii. x′z′ + y′z′ + yz′ + xy iv. AC′ + B′D + A′CD + ABCD	K3	CO1
11	Find the Minterm expansion of f(a,b,c,d) = a′(b′+d) + acd′	K3	CO1
12	Represent the Boolean function in standard / canonical SOP form: F(A,B,C) = A+B′C	K3	CO1
13	Represent the Boolean function in standard / canonical POS form: F(A,B,C) = (A+B)(B+C)(A+C)	K3	CO1

R.M.K GROUP OF INSTITUTIONS

Part B Questions

Q. No.	Questions	K Level	CO Mapping
14	Find the dual and complement of the following Boolean expression. F=xyz’ + x’yz + z(xy+w)	K3	CO1
15	Show that if all the gates in a two – level AND-OR gate networks are replaced by NAND gates the output function does not change.	K4	CO1
16	Why does a good logic designer minimize the use of NOT gates?	K4	CO1
17	Show that if all the gate in a two – level OR-AND gate network are replaced by NOR gate, the output function does not change.	K4	CO1
18	Implement Y = (A+C) (A+D’) (A+B+C’) using NOR gates only	K4	CO1
19	Check if NOR operator is associative.	K4	CO1
20	Implement Boolean expression for RXOR gate using NAND and NOR gates.	K4	CO1
21	Simplify the following functions, and implement them with two-level NAND gate circuits: (a) F(A, B, C, D) = AC'D' + A'C + ABC + AB'C + A'C'D' (b) F(A, B, C, D) = A'B'C'D + CD + AC'D (c) F(A, B, C) = (A' + C' + D’) (A' + C’) (C' + D’) (d) F(A, B, C, D) = A' + B + D' + B'C	K3	CO1

Part B Questions

Q. No.	Questions	K Level	CO Mapping
22	Draw a NAND logic diagram that implements the complement of the following function: $F(A, B, C, D) = \Sigma(0, 1, 2, 3, 6, 10, 11, 14)$	K3	CO1
23	Draw the multiple-level NOR circuit for the following expression: $CD(B + C)A + (BC' + DE')$	K3	CO1
24	Draw the multiple-level NAND circuit for the following expression: $w(x + y + z) + xyz$	K3	CO1
25	Simplify using K-map to obtain a minimum POS expression: $(A' + B'+C+D)\ (A+B'+C+D)\ (A+B+C+D')$ $(A+B+C'+D')\ (A'+B+C'+D')(A+B+C'+D)$	K3	CO1
26	Using a K-Map ,Find the MSP from of $F= \sum(0,4,8,12,3,7,11,15) +\sum d(5)$	K3	CO1
27	With the help of a suitable example ,explain the meaning of an redundant prime implicant.	K3	CO1
28	Using a K-Map, Find the MSP form of $F= \sum (0\text{-}3, 12\text{-}15) + \sum d\ (7, 11)$	K3	CO1
29	Determine the MSP and MPS focus of $F= \sum (0, 2, 6, 8, 10, 12, 14, 15)$	K3	CO1
30	Determine the MSP form of the Switching function $F = \sum (0,1,4,5,6,11,14,15,16,17,20\text{-}22,30,32,33,36,37,48,49,52,53,56,63)$	K3	CO1
31	Determine the MSP form of the Switching function $F(a,b,c,d) =\sum (0,2,4,6,8) + \sum d(10,11,12,13,14,15)$	K3	CO1
32	Find a Min SOP and Min POS for $f = b'c'd + bcd + acd' + a'b'c + a'bc'd$	K3	CO1

R.M.K GROUP OF INSTITUTIONS

Part B Questions

Q. No.	Questions	K Level	CO Mapping
33	Find the MSP representation for F(A,B,C,D,E) = ∑m(1,4,6,10,20,22,24,26) + ∑d (0,11,16,27) using K-Map method. Draw the circuit of the minimal expression using only NAND gates.	K3	CO1
34	Simplify the Boolean function F(A,B,C,D) = ∑ m (1,3,7,11,15) + ∑d (0,2,5). If don't care conditions are not taken care, What is the simplified Boolean function .What are your comments on it? Implement both circuits.	K3	CO1

35	F3 = f(a,b,c,d) = ∑ (2,4,5,6)
	F2 = f(a,b,c,d) = ∑ (2,3,,6,7)
	F1 = f(a,b,c,d) = ∑ (2,5,6,7) .
	Implement the above Boolean functions
	i. When each is treated separately and
	ii. When sharing common term
		K4	CO1

36	Find a network of AND and OR gate to realize f(a,b,c,d) = ∑m (1,5,6,10,13,14)	K3	CO1
37	Given the following Boolean function F= A''C + A'B + AB'C + BC Express it in sum of minterms & Find the minimal SOP expression.	K3	CO1
38	Simplify the given Boolean function in POS form using K-Map and draw the logic diagram using only NOR gates. F(A,B,C,D) = ∑ m (0,1,4,7, 8, 10, 12,15) + ∑d (2, 6, 11, 14).	K3	CO1

R.M.K GROUP OF INSTITUTIONS

Supportive online Certification courses (NPTEL, Swayam, Coursera, Udemy, etc.,)

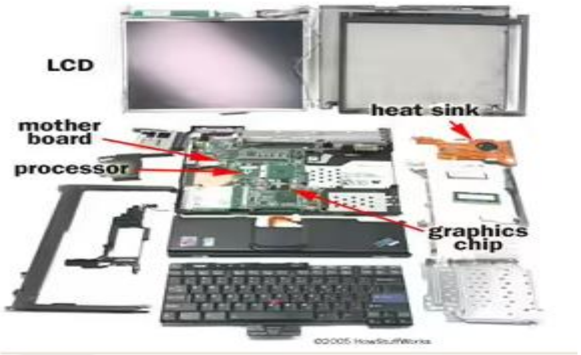
Supportive Online Certification Courses

- Swayam:
  - Digital Circuits By Prof. Santanu Chattopadhyay | IIT Kharagpur
  - https://swayam.gov.in/nd1_noc19_ee51/preview
- Coursera:
  - Digital Systems: From Logic Gates to Processors offered by Universitat Autònoma de Barcelona
  - https://www.coursera.org/learn/digital-systems
- Classcentral.com:
  - **Online Course - Digital Electronic Circuits by** Indian Institute of Technology, Kharagpur and NPTEL via Swayam
  - https://www.classcentral.com/course/swayam-digital-electronic-circuits-12953
- Udemy:
  - **Master The Digital Electronics- Minimisation And Basic Gates – [**Learn about the digital gates, boolean algebra, k-map| Update your digital from base to pro]
  - https://www.udemy.com/course/professional-digital-electronics/

14. Real time Applications in day to day life and to Industry

A few real time applications of digital systems is shown and explained.

Calculators, digital watches, mobile phones, laptops, and computers are some of the essential devices that we use in our daily lives. Apart from these devices, we use appliances like television, refrigerators, air conditioners, micro ovens, mixers, blenders, grinders, and many more.



- The microprocessor, or CPU, works with the operating system to control the computer. It essentially acts as the computer's brain. The CPU produces a lot of heat, so a desktop computer uses circulating air, a fan and a **heat sink** -- a system of plates, channels and radiator fins used to draw heat off of the processor -- to cool off. Since a laptop has far less room for each of these cooling methods, its CPU usually:
- **Runs at a lower voltage and clock speed** -- This reduces heat output and power consumption but slows the processor down. Most laptops also run at a higher voltage and clock speed when plugged in, and at lower settings when using the battery.
- **Mounts to the motherboard without using pins** -- Pins and sockets take up a lot of room in desktop PCs. Some motherboard processors mount directly to the motherboard without the use of a socket. Others use a **Micro-FCBGA** (Flip Chip Ball Grid Array), which uses balls instead of pins. These designs save space, but in some cases mean that the processor cannot be removed from the motherboard for replacement or upgrading.
- **Has a sleep or slow-down mode** -- The computer and the operating system work together to reduce the CPU speed when the computer is not in use or when the processor does not need to run as quickly. The Apple G4 processor also prioritizes data to minimize battery drain.

14. Real time Applications in day to day life and to Industry

Applications of Digital Circuits

- Digital electronics or digital circuits are an integral part of electronic devices, and here are the uses of digital circuits:
- The display of digital watches is designed based on digital circuits.
- Rocket science and quantum computing use digital electronics.
- The automatic doors work on the principle of digital electronics.
- Everyday encounters with traffic lights are based on digital circuits.
- **Quantum Computing in Practice**
- Many businesses are already using quantum computing. For example, IBM is working with Mercedes-Benz, ExxonMobil, CERN, and Mitsubishi Chemical to implement quantum computing into their products and services:
- Mercedes-Benz is exploring quantum computing to create better batteries for its electric cars. The company is hoping to shape the future of modernized electrically powered vehicles and make an impact on the environment by implementing quantum computing into its products in an effort to be carbon neutral by 2039. Simulating what happens inside batteries is extremely difficult, even with the most advanced computers today. However, using quantum computing technology, Mercedes-Benz can more accurately simulate the chemical reactions in car batteries.
- ExxonMobil is using quantum algorithms to more easily discover the most efficient routes to ship clean-burning fuel across the world. Without quantum computing, calculating all of the routing combinations and finding the most efficient one would be nearly impossible.
- The European Organization for Nuclear Research, known as CERN, is trying to discover the secrets of the universe. Using quantum computing, CERN can find algorithms that pinpoint the complex events of the universe in a more efficient way.

Content Beyond Syllabus

Quine-McCluskey Method

- A systematic simplification procedure to reduce a minterm expansion to a minimum sum of products.
- Use XY + XY′ = X to eliminate as many as literals as possible.
  - The resulting terms = prime implicants.
- Use a prime implicant chart to select a minimum set of prime implicants.
- **Determination of Prime Implicants:**
- Eliminate literals

  Two terms can be combined if they differ in exactly one variable.

  AB′CD′ + AB′CD = AB′C

  $\underset{X}{\underline{1\,0\,1}}\,\underset{Y}{0} + \underset{X}{\underline{1\,0\,1}}\,\underset{Y'}{1} = \underset{X}{\underline{1\,0\,1}}$

  A′BC′D + A′BCD′ (won′t combine)

  0 1 0 1 + 0 1 1 0 (check # of 1′s)

  We need to compare and combine whenever possible.

- **Sorting to Reduce Comparison:**

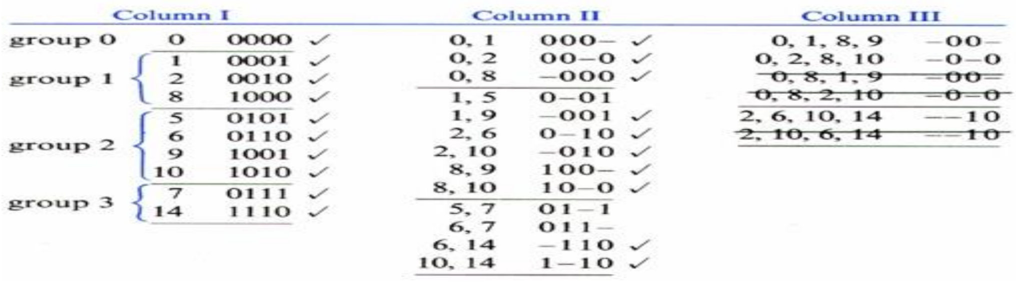
√ Sort into groups according to the number of 1′s.

Group	0	0	0000
Group	1	1	0001
		2	0010
		8	1000
Group	2	5	0101
		6	0110
		9	1001
		10	1010
Group	3	7	0111
		14	1110

F(a,b,c.d) = Σm(0,1,2,5,6,7,8,9,10,14)

- No need for comparisons
  - (1) Terms in nonadjacent group
  - (2) Terms in the same group

- **Comparison of Adjacent Groups:**
  - Use X + X = X repeatedly between adjacent groups
  - Those combined are checked off.
  - Combine terms that have the same dashes and differ one in the number of 1′s. (for column II and column III)



Column I				Column II			Column III	
group 0	0	0000	✓	0, 1	000–	✓	0, 1, 8, 9	–00–
group 1	1	0001	✓	0, 2	00–0	✓	0, 2, 8, 10	–0–0
	2	0010	✓	0, 8	–000	✓	~~0, 8, 1, 9~~	~~–00–~~
	8	1000	✓	1, 5	0–01		~~0, 8, 2, 10~~	~~–0–0~~
group 2	5	0101	✓	1, 9	–001	✓	2, 6, 10, 14	––10
	6	0110	✓	2, 6	0–10	✓	~~2, 10, 6, 14~~	~~––10~~
	9	1001	✓	2, 10	–010	✓		
	10	1010	✓	8, 9	100–	✓		
group 3	7	0111	✓	8, 10	10–0	✓		
	14	1110	✓	5, 7	01–1			
				6, 7	011–			
				6, 14	–110	✓		
				10, 14	1–10	✓		

$$f = \underset{(1,\,5)}{a'c'd} + \underset{(5,\,7)}{a'bd} + \underset{(6,\,7)}{a'bc} + \underset{(0,\,1,\,8,\,9)}{b'c'} + \underset{(0,\,2,\,8,\,10)}{b'd'} + \underset{(2,\,6,\,10,\,14)}{cd'}$$

$$f = a'bd + b'c' + cd'$$

Quine-McCluskey Method

- The terms that have not been checked off are called prime implicants.

  f = 0-01 + 01-1+011- + -00-

  + -0-0 + --10

  = $\underline{a'c'd}$ + a'bd + $\underline{a'bc}$ + b'c' +

  $\underline{b'd'}$ + cd'

- Each term has a minimum number of literals, but minimum SOP for f:

  f = a'bd + b'c' + cd'

  (a'bd, cd' => a'bc)

  (a'bd, b'c' => a'c'd)

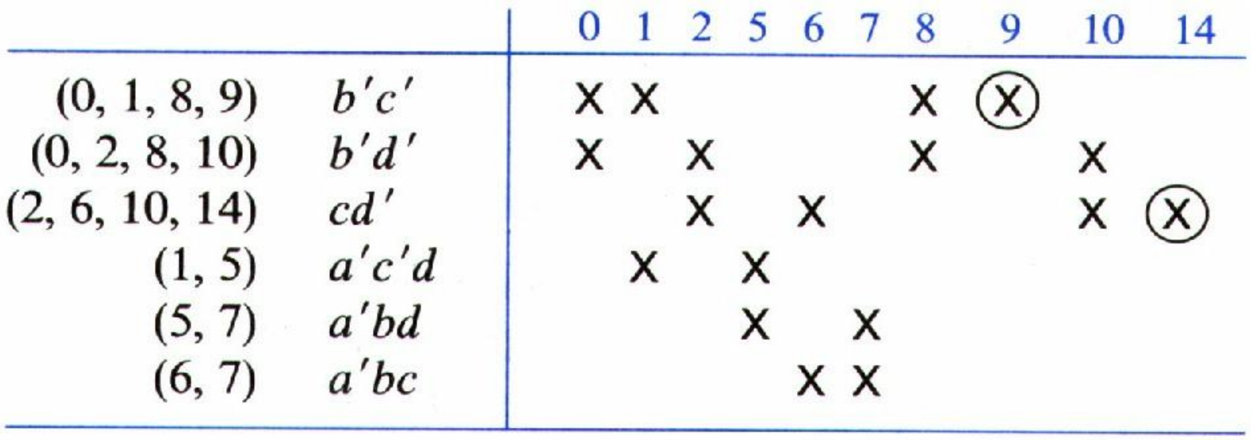
  (b'c', cd' => b'd')

Definition of Implicant:

- Definition
  - Given a function of F of n variables, a product term P is an implicant of F iff for every combination of values of the n variables for which P = 1, F is also equal to 1.
    - Every minterm of F is an implicant of F.
    - Any term formed by combining two or more minterms is an implicant.
    - If F is written in SOP form, every product term is an implicant.
- Example: f(a,b,c) = a'b'c' + ab'c' + ab'c + abc = b'c' + ac
    - If a'b'c' = 1, then F = 1, if ac = 1, then F = 1. a'b'c' and ac are implicants.
    - If bc = 1, (but a = 0), F = 0, so bc is not an implicant of F.
  - A prime implicant of a function F is a product term implicant which is no longer an implicant if any literal is deleted from it.
  - Example: f(a,b,c) = a'b'c' + ab'c' + ab'c + abc = b'c' + ac
    - Implicant a'b'c' is not a prime implicant. Why? If a' is deleted, b'c' is still an implicant of F.
    - b'c' and ac are prime implicants.
  - Each prime implicant of a function has a minimum number of literals that no more literals can be eliminated from it or by combining it with other terms.

Quine-McCluskey Method

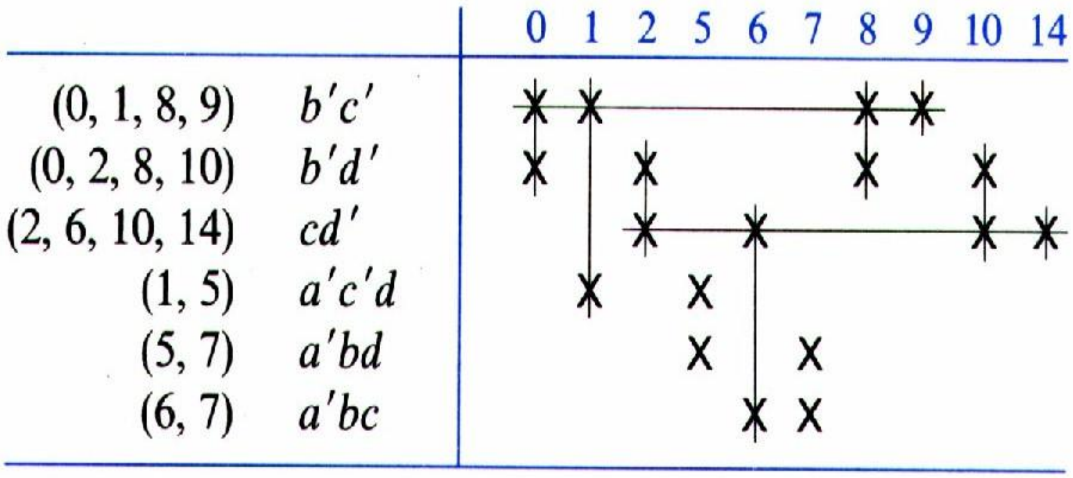
- QM procedure:
  - Find all product term implicants of a function
  - Combine non-prime implicants.
    - Remaining terms are prime implicants.
  - A minimum SOP expression consists of a sum of some (not necessarily all) of the prime implicants of that function.
    - We need to select a minimum set of prime implicants.
  - If an SOP expression contains a term which is not a prime implicant, the SOP cannot be minimum.
- Chart layout
  - Top row lists minterms of the function
  - All prime implicants are listed on the left side.
  - Place x into the chart according to the minterms that form the corresponding prime implicant.
- Essential prime implicant
  - If a minterm is covered only by one prime implicant, that prime implicant is called essential prime implicant. (9 & 14).
    - Essential prime implicant must be included in the minimum sum of the function.



		0	1	2	5	6	7	8	9	10	14
(0, 1, 8, 9)	$b'c'$	X	X					X	⊗		
(0, 2, 8, 10)	$b'd'$	X		X				X		X	
(2, 6, 10, 14)	cd'			X		X				X	⊗
(1, 5)	$a'c'd$		X		X						
(5, 7)	$a'bd$				X		X				
(6, 7)	$a'bc$					X	X				

Quine-McCluskey Method

- Selection of Prime Implicants:
  - √ Cross out the row of the selected essential prime implicants
  - √ The columns which correspond to the minterms covered by the selected prime implicants are also crossed out.
  - √ Select a prime implicant that covers the remaining columns. This prime implicant is not essential.



		0	1	2	5	6	7	8	9	10	14
(0, 1, 8, 9)	$b'c'$	X	X					X	X		
(0, 2, 8, 10)	$b'd'$	X		X				X		X	
(2, 6, 10, 14)	cd'			X		X				X	X
(1, 5)	$a'c'd$		X		X						
(5, 7)	$a'bd$				X		X				
(6, 7)	$a'bc$					X	X				

Assessment Schedule (Proposed Date & Actual Date)

Assessment	Proposed Date	Actual Date
First Internal Assessment	07/12/2022 to 14/12/2022	
Second Internal Assessment	23/01/2023 to 30/01/2023	
Model Examination	15/02/2023 to 24/02/2023	

Prescribed Text Books & Reference

TEXT BOOK:

1. M. Morris Mano and Michael D. Ciletti, Digital Design, With an Introduction to the Verilog HDL, VHDL, and System Verilog, 6th Edition, Pearson, 2018.
2. S.Salivahanan and S.Arivazhagan,Digital Circuits and Design, 5th Edition, Oxford University Press, 2018.

REFERENCES:

1. A.Anandkumar, Fundamental of digital circuits, 4th Edition, PHI Publication,2016.
2. William Kleitz, Digital Electronics-A Practical approach to VHDL, Prentice Hall International Inc, 2012.
3. Charles H.Roth, Jr. andLarry L. Kinney, Fundamentals of Logic Design, 7th Edition, Thomson Learning, 2014.

4.Thomas L. Floyd, Digital Fundamentals, 11th Edition, Pearson Education Inc, 2017.

5. John.M Yarbrough, Digital Logic: Applications and Design, 1st Edition, Cengage India, 2006.

NPTEL LINK: https://nptel.ac.in/courses/108/105/108105132/

Mini Project Suggestions

Mini Project Suggestions

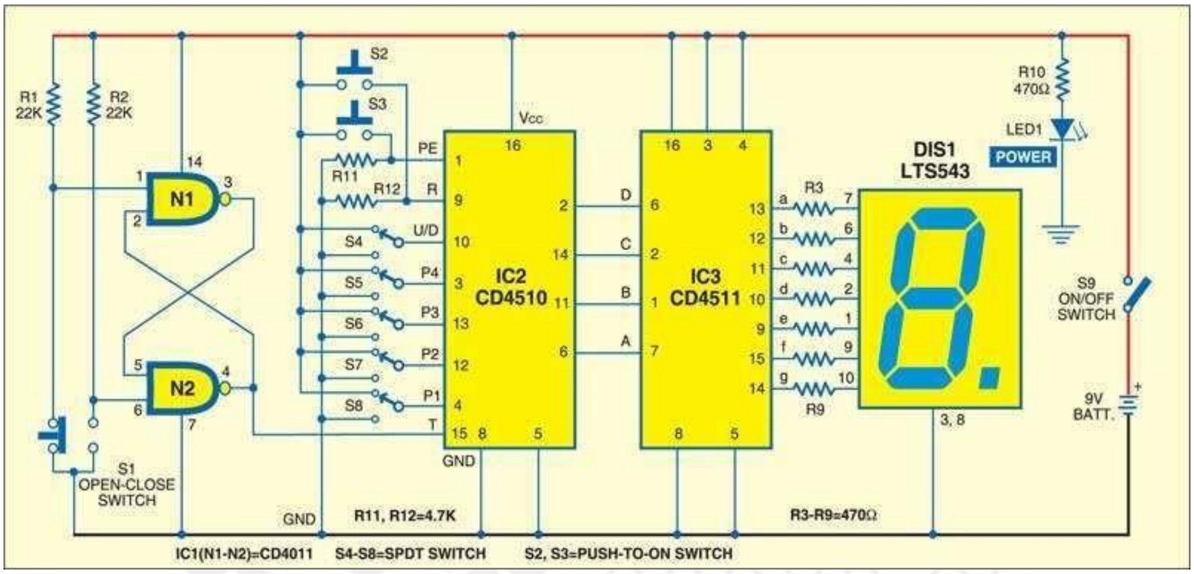
- 1. Smart Digital School Bell With Timetable Display
- 2. Digital Car Turning and Braking Indicator
- 3. Digitally Controlled Home Automation Project
- 4. Design push button lock using logic gates
- 5. Design burglar alarm using logic gates
- 6. Simple Water Level Indicator

Digital counter circuit

Circuit operation:

Momentarily press of micro-switch S1 once, sends one clock pulse.

The display shows '1.'



- Continuous pressing and releasing switch S1 increments the 7-segment display by one digit for each clock pulse.
- Continued pressing is done until the display shows '9.'
- The next press-and-release operation will change the display to '0,' indicating that the counter has been reset and that it has completed its one cycle.
- On changing the position of switch S4 to GND and giving a clock pulse from switch S1, the display will show '9'.
- This indicates that the counter has now started counting downwards. Reverse counting is justified as the down-count mode is selected.
- For parallel data loading, the parallel data by switches S5 through S8 (either GND or Vcc) is set and then switch S3 is pressed.
- The set parallel data appears on the 7-segment display.

https://www.electronicsforu.com/electronics-projects/digital-counter

Thank you