Noida Institute of Engineering and Technology, Greater Noida

DIGITAL LOGIC & CIRCUIT DESIGN

Unit: 1

DIGITAL LOGIC & CIRCUIT DESIGN

SUBJECT CODE: ACSE0304

B Tech: 3rd Sem.

Ms. Nidhi Sharma
Associate Professor

Department of Electronics and Communication Engineering

Brief Introduction

Ms. Nidhi Sharma

- PhDAmity University, Pursuing
- M.Tech AKTU, Lucknow, India
- B.Tech. Kurukshetra University, Kkr. , Haryana

Research Interests

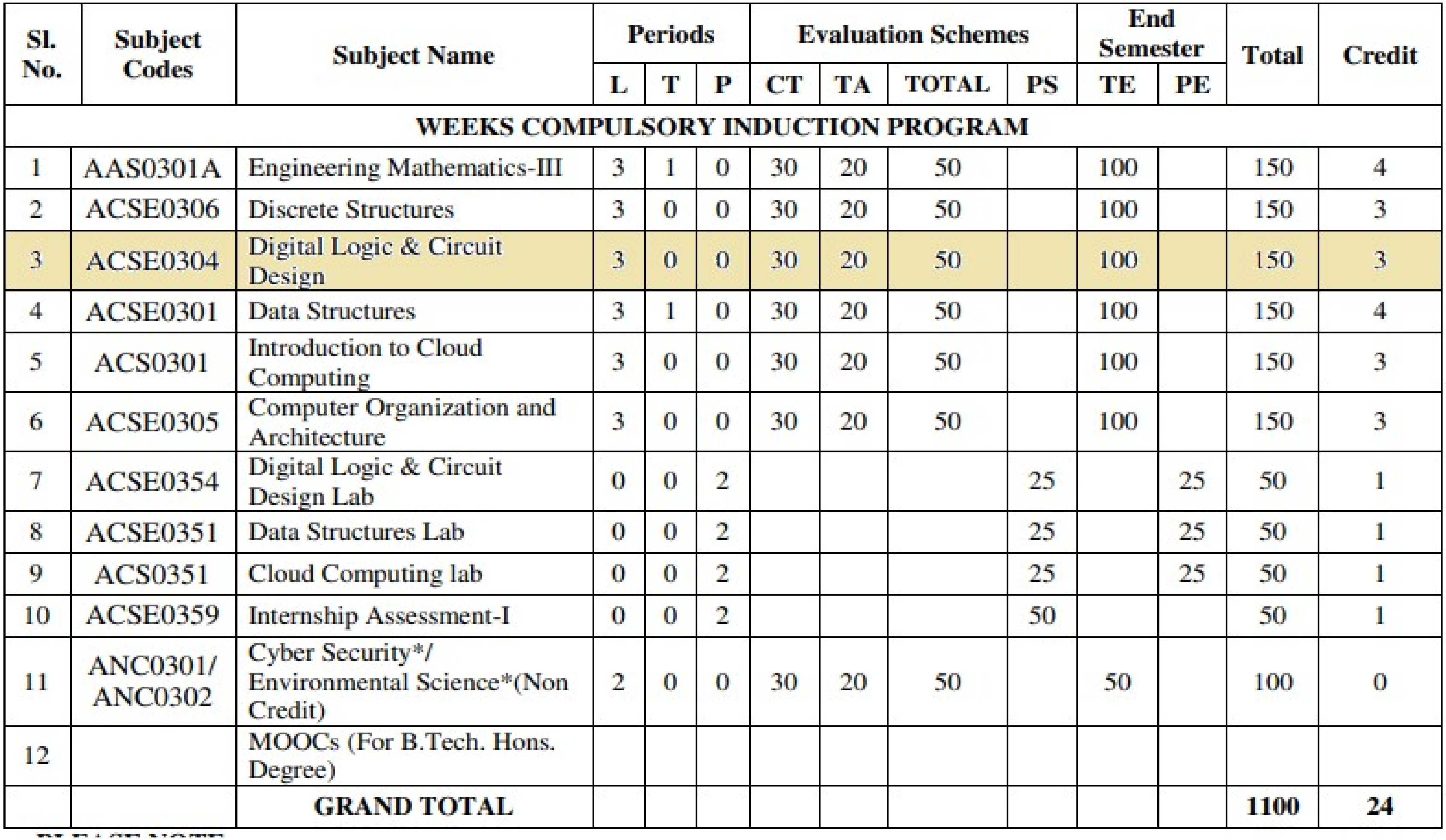
VLSI Design, Communication System, Internet of Things.

Industrial Experiance: (2.5 Years)

Academic Experience: (17+ years)

Publication : 08 (Peer Reviewed journals), 05(International Conference), 01 National conference.

Evaluation Scheme



Sl. No.	Subject Codes	Subject Name	Periods			Evaluation Schemes				End Semester		Total	Credit
			L	T	P	CT	TA	TOTAL	PS	TE	PE		
		WEEKS COMPULSORY INDUCTION PROGRAM											
1	AAS0301A	Engineering Mathematics-III	3	1	0	30	20	50		100		150	4
2	ACSE0306	Discrete Structures	3	0	0	30	20	50		100		150	3
3	ACSE0304	Digital Logic & Circuit Design	3	0	0	30	20	50		100		150	3
4	ACSE0301	Data Structures	3	1	0	30	20	50		100		150	4
5	ACS0301	Introduction to Cloud Computing	3	0	0	30	20	50		100		150	3
6	ACSE0305	Computer Organization and Architecture	3	0	0	30	20	50		100		150	3
7	ACSE0354	Digital Logic & Circuit Design Lab	0	0	2				25		25	50	1
8	ACSE0351	Data Structures Lab	0	0	2				25		25	50	1
9	ACS0351	Cloud Computing lab	0	0	2				25		25	50	1
10	ACSE0359	Internship Assessment-I	0	0	2				50			50	1
11	ANC0301/ ANC0302	Cyber Security*/ Environmental Science*(Non Credit)	2	0	0	30	20	50		50		100	0
12		MOOCs (For B.Tech. Hons. Degree)											
		GRAND TOTAL										1100	24

PLEASE NOTE

Course Contents / Syllabus

UNIT-I: Digital System and Binary Numbers: Number System and its arithmetic, Signed binary numbers, Binary codes, Cyclic codes, Hamming Code, Simplification of Boolean Expression: K-map method up to five variable, SOP and POS Simplification Don't Care Conditions, NAND and NOR implementation, Quine Mc-Clusky Method (Tabular Method).

UNIT II : Combinational Logic: Combinational Circuits: Analysis Procedure, Design Procedure, Code Converter, Binary Adder-Subtractor, Decimal Adder, Binary Multiplier, Magnitude Comparator, Decoders, Encoders Multiplexers, Demultiplexers.

UNIT III: Sequential Logic and Its Applications: Storage elements: Latches & Flip Flops, Characteristic Equations of Flip Flops, Excitation Table of Flip Flops, Flip Flop Conversion, Registers, Shift Registers, Ripple Counters, Synchronous Counters, Other Counters: Johnson & Ring Counter.

UNIT IV: Synchronous & Asynchronous Sequential Circuits: Analysis of clocked Sequential Circuits with State Machine Designing, State Reduction and Assignments, Design Procedure.
Analysis procedure of Asynchronous Sequential Circuits, Circuit with Latches, Design Procedure, Reduction of State and flow Table, Race-free State Assignment, Hazards.

UNIT-V: Memory & Programmable Logic Devices: Basic concepts and hierarchy of Memory, Memory Decoding, RAM: SRAM, DRAM, ROM: PROM, EPROM, Auxiliary Memories, PLDs: PLA, PAL; Circuit Implementation using ROM, PLA and PAL; CPLD and FPGA..

CONTENT

- Course Objective
- Unit Objective
- Course Outcome
- Co and Po Mapping
- Topic Objective
- Prerequisite
- Introduction to Number Systems
- Code conversion
- Boolean algebra
- SOP & POS form
- K-Map
- QM Algorithm
- Daily quiz & MCQs
- Old Question Papers
- Recap
- Video Links
- Weekly Assignments
- References

COURSE OBJECTIVE

The intended objectives of this course are given as follows:

- To learn number representation and conversion between different representation.
- To analyze logic processes and implement logical operations using combinational logic circuits.
- To learn concepts of sequential circuits and analyze sequential circuits in terms of state machine.
- To learn to design synchronus and asynchronus circuits.
- To learn concept of memories and PLDs.

UNIT OBJECTIVE

- To learn number system & its conversions
- To learn basic rules of Boolean algebra.
- To learn the concept of logic gates and SOP& POS.
- To minimize function using K-Map & QM method.

COURSE OUTCOME

At the end of this course students will able to:

Course outcomes	DIGITAL LOGIC AND CIRCUIT DESIGN**(ACSE0304**)
ACSE0304.1	Apply concepts of Digital Binary System and implementation of Gates.
ACSE0304.2	Analyze and design of Combinational logic circuits.
ACSE0304.3	Analyze and design of Sequential logic circuits with their applications.
ACSE0304.4	Implement the Design procedure of Synchronous & Asynchronous Sequential Circuits.
ACSE0304.5	Apply the concept of Programmable Logic devices with circuit implementation.

CO-PO and PSO Mapping

COs	PO1	PO2	PO3	PO4	PO5	PO6	PO7	PO8	PO9	PO10	PO11	PO12
ACSE0304.1	3	2	-	-	-	-	-	-	-	-	-	2
ACSE0304.2	3	3	2	-	-	-	-	-	-	-	-	2
ACSE0304.3	2	3	2	2	-	-	-	-	-	-	-	2
ACSE0304.4	3	3	3	2	-	-	-	-	-	-	-	2
ACSE0304.5	3	2	1	-	-	-	-	-	-	-	-	2
AVERAGE	2.8	2.6	2	2	-	-	-	-	-	-	-	2

Course Outcome	PSO1	PSO2	PSO3
ACSE0304.1	3	-	3
ACSE0304.2	2	-	3
ACSE0304.3	2	-	3
ACSE0304.4	2	-	3
ACSE0304.5	2	-	3
Average	2.2	-	3

Number System

Topic Objective	Mapping with CO
To understand different number systems.	CO1
To understand number system conversion.	CO1

Number System

Prerequisite:

- Basics of decimal number system.

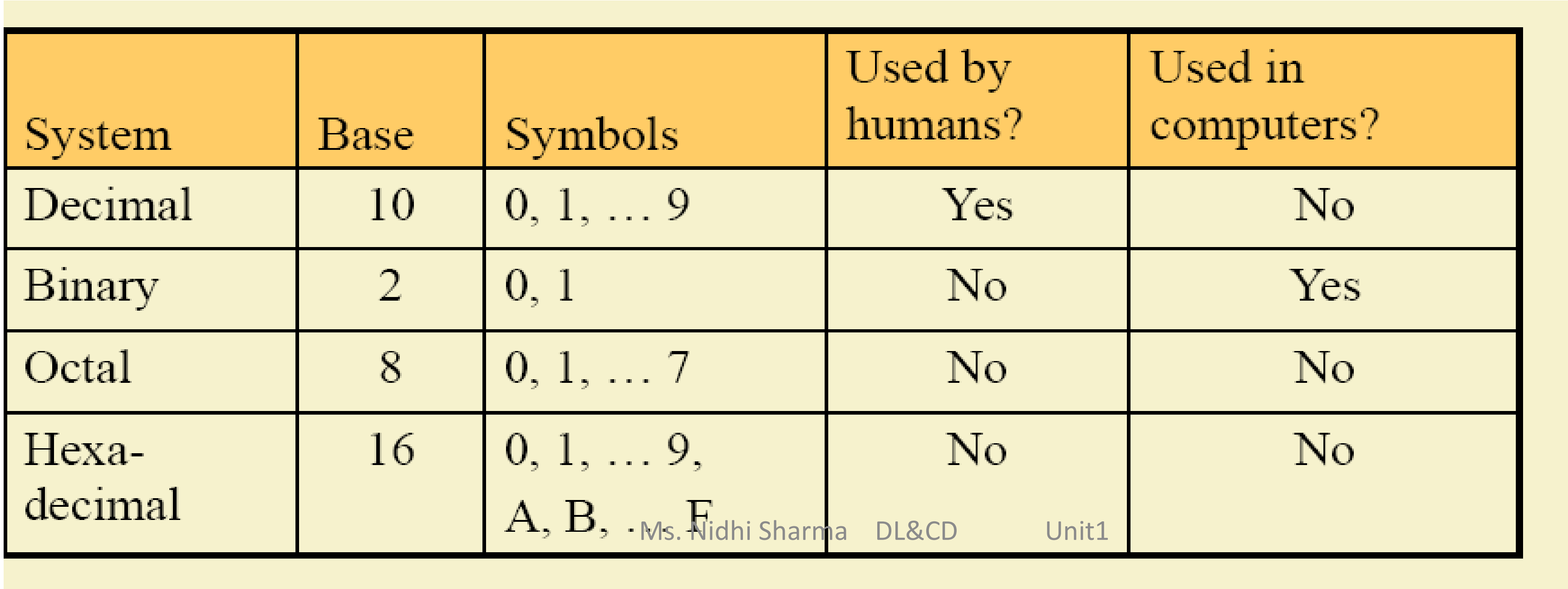
INTRODUCTION TO NUMBER SYSTEMS

In digital electronics, the number system is used for representing the information.

The number system has different bases and the most common of them are the decimal, binary, octal, and hexadecimal.

The **base or radix** of the number system is the total number of the digit used in the number system.

Suppose if the number system representing the digit from 0 – 9 then the base of the system is the 10.



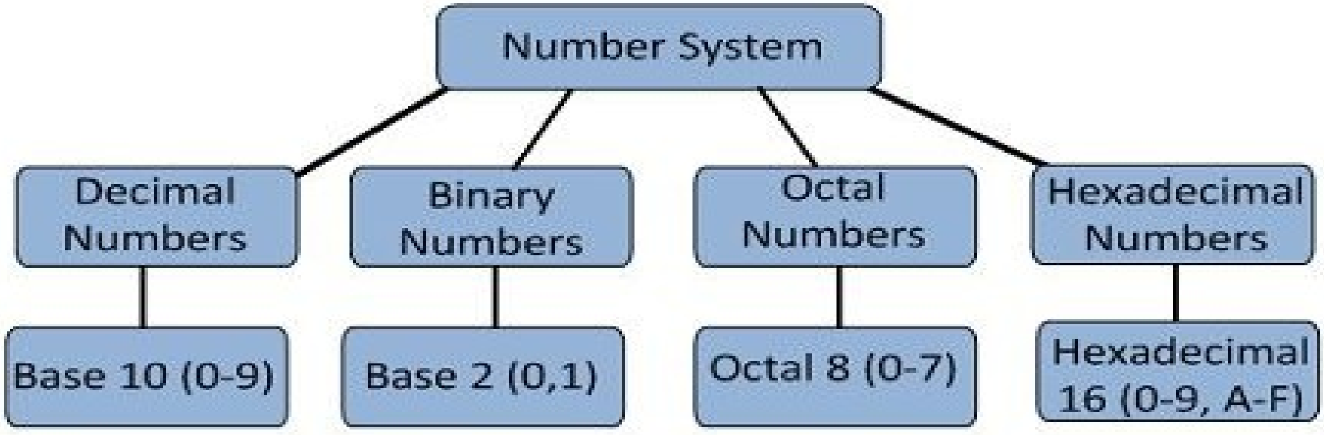
System	Base	Symbols	Used by humans?	Used in computers?
Decimal	10	0, 1, … 9	Yes	No
Binary	2	0, 1	No	Yes
Octal	8	0, 1, … 7	No	No
Hexa-decimal	16	0, 1, … 9, A, B, … F	No	No

NUMBER SYSTEMS

Types of Number Systems

Some of the important types of number system are:

- Decimal Number System
- Binary Number System
- Octal Number System
- Hexadecimal Number System



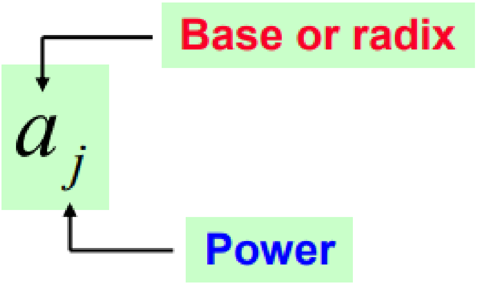
DECIMAL NUMBER SYSTEMS

- The number system is having digit 0, 1, 2, 3, 4, 5, 6, 7, 8, 9;
- this number system is known as a decimal number system because total ten digits are involved.
- The base of the decimal number system is 10.

- Decimal number

$$\dots a_5a_4a_3a_2a_1.a_{-1}a_{-2}a_{-3}\dots$$

Decimal point



$$\Rightarrow \quad \boxed{\dots+10^5a_5+10^4a_4+10^3a_3+10^2a_2+10^1a_1+10^0a_0+10^{-1}a_{-1}+10^{-2}a_{-2}+10^{-3}a_{-3}+\dots}$$

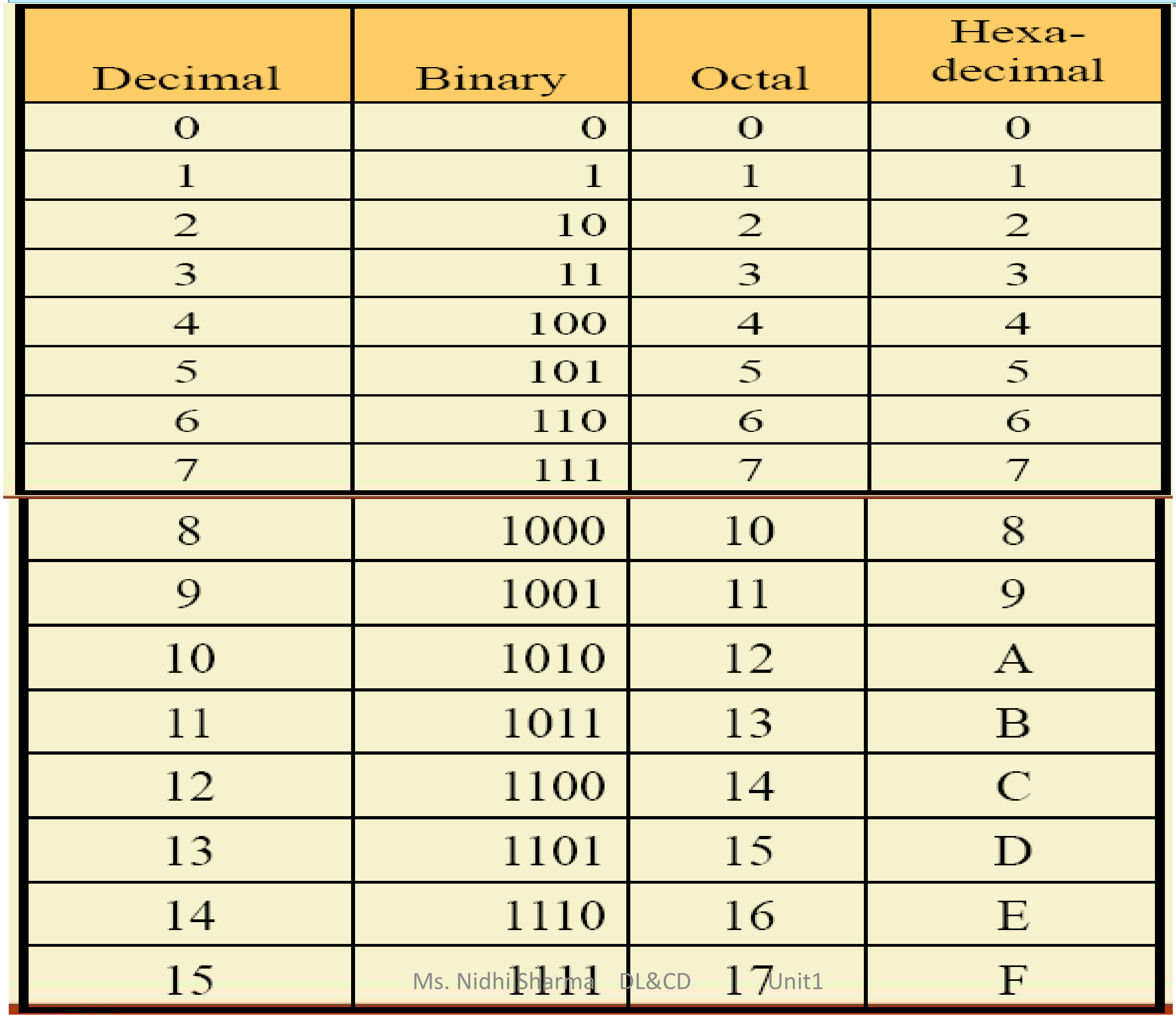
Example:

$$7{,}329=7\times10^3+3\times10^2+2\times10^1+9\times10^0$$

BINARY NUMBER SYSTEMS

- The modern computers do not process decimal number; they work with another number system known as a binary number system which uses only two digits 0 and1.
- The base of binary number system is 2 because it has only two digit 0 and 1.
- The digital electronic equipments work on the binary number system and hence the decimal number system is converted into binary system.
- The table is shown below the decimal, binary, octal, and hexadecimal numbers from 0 to 15 and their equivalent binary number.

BINARY NUMBER SYSTEMS



Decimal	Binary	Octal	Hexa-decimal
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	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

OCTAL NUMBERS

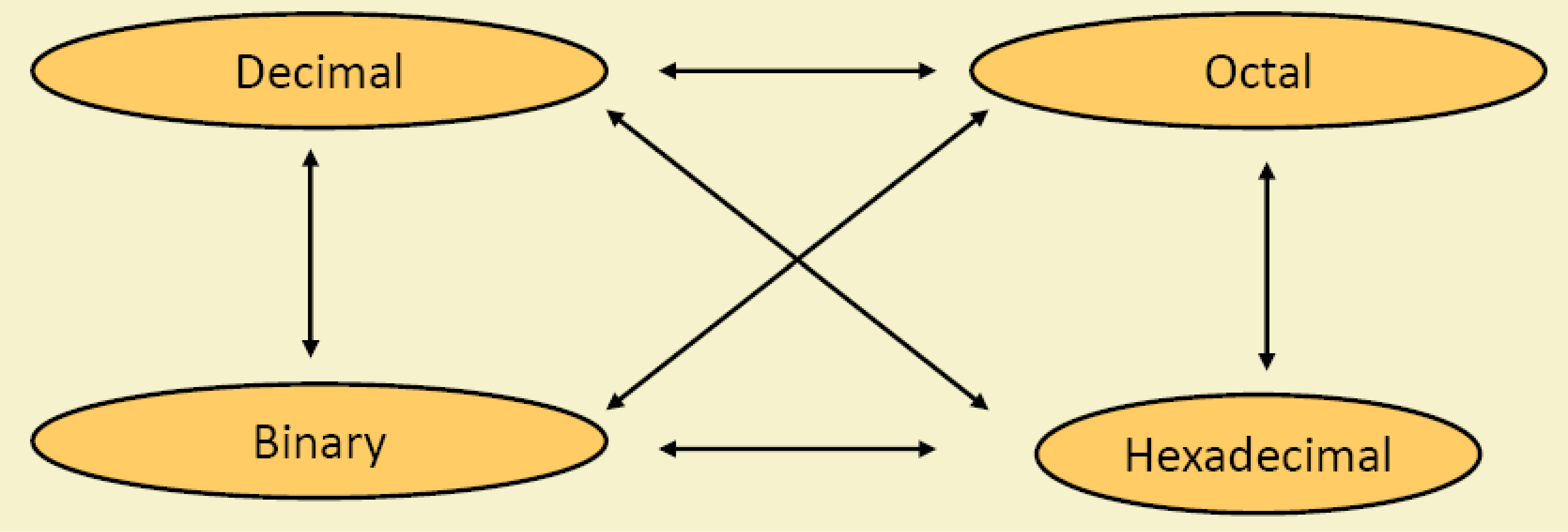
- The base of a number system is equal to the number of digits used, i.e., for decimal number system the base is ten while for the binary system the base is two. The octal system has the base of eight as it uses eight digits 0, 1, 2, 3, 4, 5, 6, 7.
- All these digits from 0 to 7 have the same physical meaning as by decimal symbols, the next digit in the octal number is represented by 10, 11, 12, which are equivalent to decimal digits 8, 9, 10 respectively.
- In this way, the octal number 20 will represent the decimal digit and subsequently, 21, 22, 23.. Octal numbers will represent the decimal number digit 17, 18, 19… etc. and so on.

HEXADECIMAL NUMBERS

- These numbers are used extensively in microprocessor. The hexadecimal number system has a base of 16, and hence it consists of the following sixteen number of digits.
- 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.
- The size of the hexadecimal is much shorter than the binary number which makes them easy to write and remember.
- Let 0000 to 000F representing hexadecimal numbers from zero to fifteen, then 0010, 0011, 0012, …etc. Will represent sixteen, seventeen, eighteen… etc. till 001F which represent thirty open and so on.

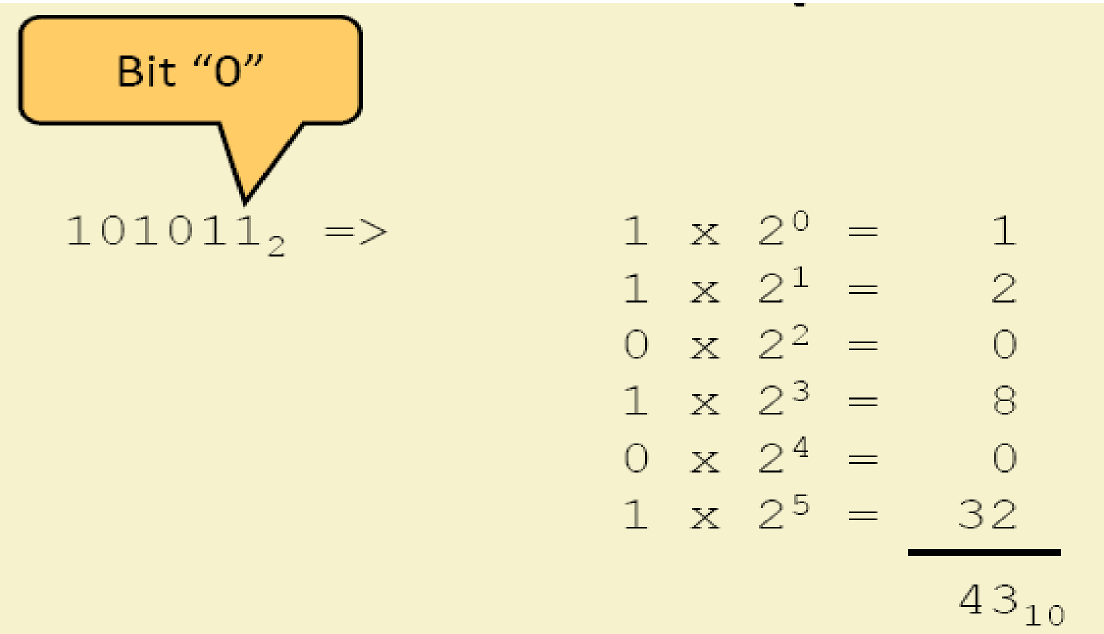
CODE CONVERSION

Conversion among base:



BINARY TO DECIMAL CONVERSION:

- Multiply each bit by 2^n,where n is “weight” of the bits
- The weight is the position of the bit starting form zero from right.
- Add the result
- Example:

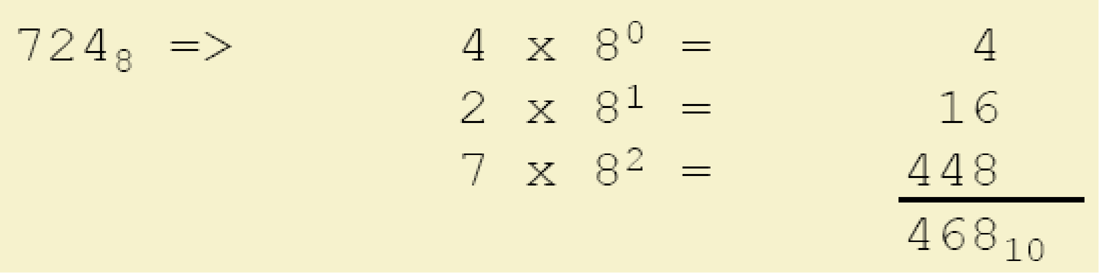


$101011_2 \Rightarrow$

$$\begin{array}{rcrcr}
1 & \times & 2^0 & = & 1 \\
1 & \times & 2^1 & = & 2 \\
0 & \times & 2^2 & = & 0 \\
1 & \times & 2^3 & = & 8 \\
0 & \times & 2^4 & = & 0 \\
1 & \times & 2^5 & = & \underline{32} \\
 & & & & 43_{10}
\end{array}$$

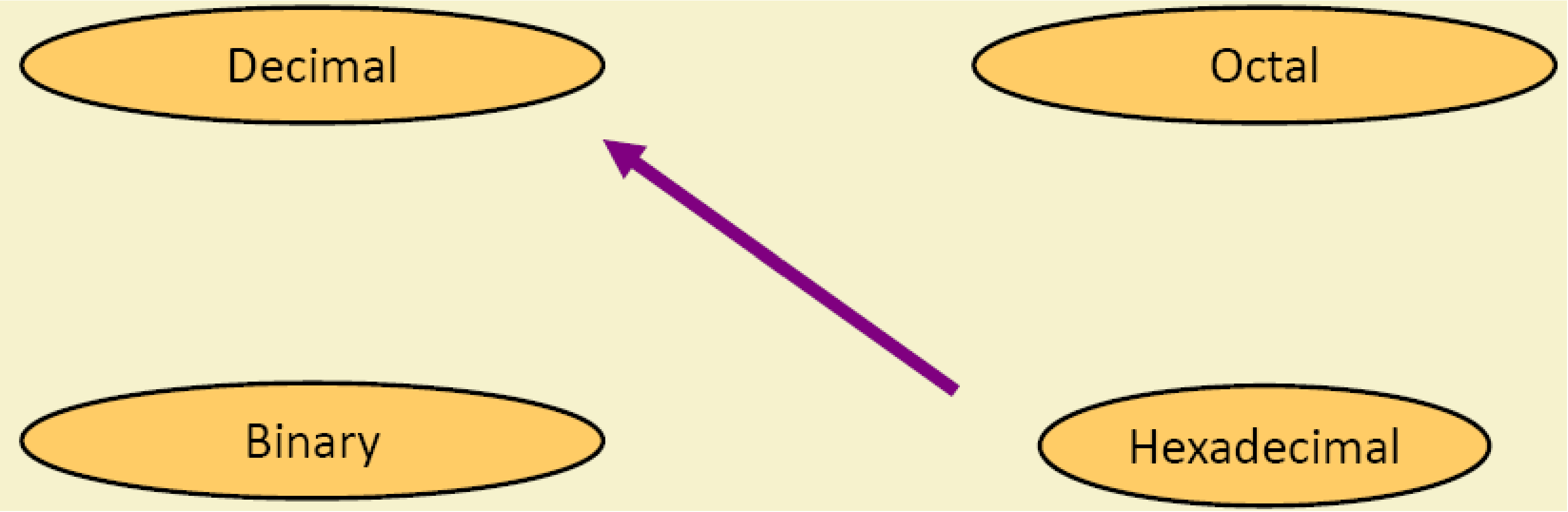
OCTAL TO DECIMAL

- multiplying each digit by 8^n bits where n the weight of the digits.
- the weight is the position of the digit starting from 0 on the right
- add the result
- example:



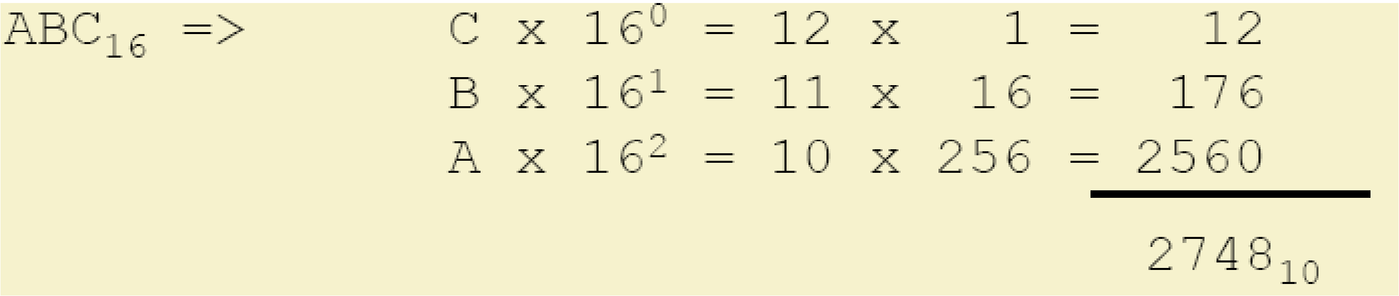
$$
\begin{array}{lrr}
724_8 \Rightarrow & 4 \times 8^0 = & 4 \\
 & 2 \times 8^1 = & 16 \\
 & 7 \times 8^2 = & \underline{448} \\
 & & 468_{10}
\end{array}
$$

HEXADECIMAL TO DECIMAL



HEXADECIMAL TO DECIMAL CONVERSION

- Multiplying each digit by 16^n bits where n the weight of the digits.
- The weight is the position of the digit starting from 0 on the right
- Add the result
- Example:

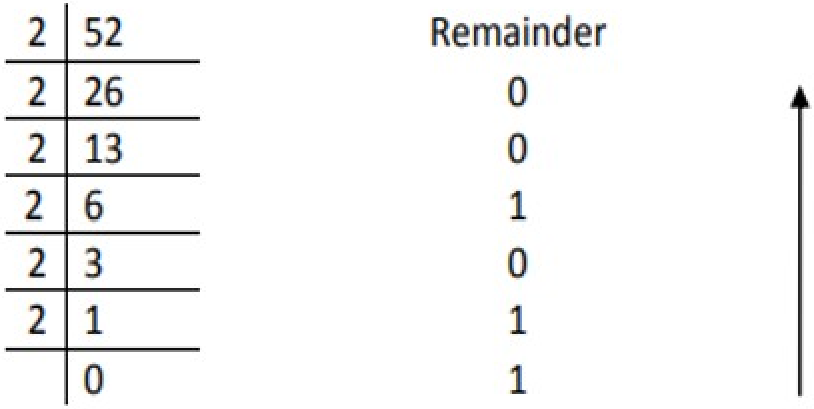


$$
\begin{array}{llcl}
ABC_{16} \Rightarrow & C \times 16^{0} = 12 \times 1 & = & 12 \\
 & B \times 16^{1} = 11 \times 16 & = & 176 \\
 & A \times 16^{2} = 10 \times 256 & = & 2560 \\
\hline
 & & & 2748_{10}
\end{array}
$$

DECIMAL TO BINARY CONVERSION

Example : convert $(52)_{10}$ to binary.

Sol:

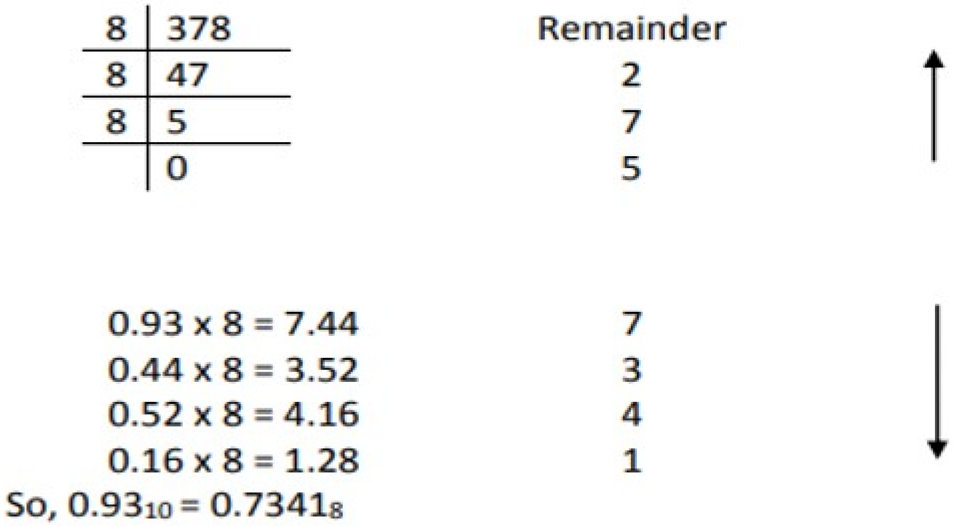


		Remainder
2	52	
2	26	0
2	13	0
2	6	1
2	3	0
2	1	1
	0	1

$(52)_{10} = (110100)_2$

DECIMAL TO OCTAL CONVERSION

Example : Convert $(378.93)_{10}$ to octal Sol:



8	378	Remainder
8	47	2
8	5	7
	0	5

$0.93 \times 8 = 7.44$	7
$0.44 \times 8 = 3.52$	3
$0.52 \times 8 = 4.16$	4
$0.16 \times 8 = 1.28$	1

So, $0.93_{10} = 0.7341_8$

Ans: $(378.93)_{10} = (572.7341)_8$

BINARY TO DECIMAL CONVERSION

Example : Convert $(10101)_2$ to decimal.

$$\begin{aligned} 1\,0\,1\,0\,1 &= (1 \times 2^4) + (0 \times 2^3) + (1 \times 2^2) + (0 \times 2^1) + (1 \times 2^0) \\ &= 16 + 0 + 4 + 0 + 1 \\ &= 21 \end{aligned}$$

$(10101)_2 = (21)_{10}$

Example : Convert $(11011.101)_2$ to decimal.

$$\begin{aligned} 1\,1\,0\,1.1\,0\,1 &= (1 \times 2^4) + (1 \times 2^3) + (0 \times 2^2) + (1 \times 2^1) + (1 \times 2^0) + (1 \times 2^{-1}) + (0 \times 2^{-2}) + \\ &\quad (1 \times 2^{-3}) \\ &= 16 + 8 + 0 + 2 + 1 + 0.5 + 0 + 0.125 \\ &= 27.625 \end{aligned}$$

$(11011.101)_2 = (27.625)_{10}$

Daily Quiz

- What do you mean by radix of any number system?
- Convert $(1101.11)_2$ into decimal.
- Convert $(267.89)_{10}$ into binary.
- The value of $(011010101.110)_2$ in octal and hexadecimal are:

a) (236.6)8 and (D5.B)16

b) (235.6)8 and (D5.C)16

c) (325.6)8 and (D5.C)16

- The value of base x for $(412)_x = (153)_8$ is:

a) 9

b) 5

c) 8

d) 4

- Convert $(ACB.8D)_{16}$ into binary and then convert it into octal number system.

FACULTY VIDEO LINKS, YOUTUBE & NPTEL VIDEO LINKS AND ONLINE COURSES DETAILS

Youtube/other Video Links:

- https://www.youtube.com/watch?v=crSGS1uBSNQ&ab_channel=NesoAcademyNesoAcademyVerified
- https://www.youtube.com/watch?v=crSGS1uBSNQ&list=RDCMUCQYMhOMi_Cdj1CEAU-fv80A&start_radio=1&t=1&ab_channel=NesoAcademyNesoAcademyVerified

Old Questions

- Determine the value of if $(193)_x = (623)_8$
- Convert $(100000011110)_2$ into hexadecimal and octal number system without converting it to decimal.
- The solution to the quadratic equation $k^2 - 11k + 22 = 0$ are $k = 3$ and $k = 6$. What are the base of number systems?
- Convert $(238.99)_{10}$ into binary, hexadecimal and octal.

Recap

- The number system has different bases and the most common of them are the decimal, binary, octal, and hexadecimal.
- The **base or radix** of the number system is the total number of the digit used in the number system.
- Some of the important types of number system are:

Decimal Number System

Binary Number System

Octal Number System

Hexadecimal Number System

SIGNED BINARY NUMBERS

Topic Objective	Mapping with CO
To understand signed binary numbers.	CO1

SIGNED BINARY NUMBERS

- Positive integers including zero can be represented as unsigned numbers. However, to represent negative integers, we need a notation for negative values. In ordinary arithmetic, a negative number is indicated by a minus sign and a positive number by a plus sign. Because of hardware limitations, computers must represent everything with binary digits, commonly referred to as bits.
- It is customary to represent the sign with a bit placed in the leftmost position of the number. The convention is to make the sign bit 0 for positive and 1 for negative.
- If the binary number is signed, then the leftmost bit represents the sign and the rest of the bits represent the number. If the binary number is assumed to be unsigned, then the leftmost bit is the most significant bit of the number.

SIGNED BINARY NUMBERS

- For example, the string of bits 01001 can be considered as 9 (unsigned binary) or a +9 (signed binary) because the leftmost bit is 0. The string of bits 11001 represent the binary equivalent of 25 when considered as an unsigned number or as - 9 when considered as a signed number because of the 1 in the leftmost position, which designates negative, and the other 4 bits represents binary 9.

Example:

- In signed-magnitude representation: 10001001
- In signed-1's-complement representation: 11110110
- In signed-2's-complement representation: 11110111

Arithmetic Addition

Arithmetic operations with numbers in base r follow the same rules as for decimal numbers.

Examples of addition, subtraction, and multiplication of two binary numbers are as follows:

```
augend:     101101     minuend:       101101     multiplicand:      1011
addend:    +100111     subtrahend:   -100111     multiplier:       × 101
           -------                   -------                       -----
sum:       1010100     difference:    000110                        1011
                                                                   0000
                                                                  1011
                                                                  ------
                                                 product:         110111
```

```
                 Quotient
                 101
Divisor        ______
          11 | 1111   Dividend
              -11
              ----
              0011
               -11
              ----
                00
                 Remainder
```

Complements

- Complements are used in digital computers for simplifying the subtraction operation and for logical manipulation.
- There are two types of complements for each base-r system: the radix complement and the diminished radix complement.
- The first is referred to as the r's complement and the second as the (r - 1)'s complement. When the value of the base r is substituted in the name, the two types are referred to as the 2's complement and l's complement for binary numbers, and the 10's complement and 9's complement for decimal numbers.

Complements

Diminished Radix Complement(r-1)’s:

- Given a number N in base r having n digits, the (r - l)'s complement of N is defined as $(r^n - 1) – N$. For decimal numbers, r = 10 and r - 1 = 9, so the 9's complement of N is $(10^n - 1) -N$

  eg: The 9's complement of 546700 is 999999 - 546700 = 453299.

- For binary numbers, r = 2 and r - 1 = 1, so the 1's complement of *N is* $(2^n - 1) - N$.

  eg: The 1's complement of 1011000 is 0100111.

7’s complement

$(16)_8 = 77 -16 =61$

15’s $(AB)_{16} = (16^2 - 1) – AB = FF –AB = 54$

Complements

Radix Complement:

- The r's complement of an n-digit number, N in base r is defined as (r^n– N) for N is not equal to and 0 for N = 0
- Comparing with the (r – 1)'s complement, we note that the r's complement is obtained by adding 1 to the (r - 1)'s complement since r^n - N =[(r^n - 1) – N] + 1.

  Eg: 1. The 10's complement of 012398 is 987602.

  2. The 2's complement of 1101100 is 0010100.
- 56.89
- (012398) = (999999– 012398) + 1= 987602
- (12398)= 99999 -12398 +1= 87601 +1 = 87602

8’s complement

(16) = 77 -16 +1=61 +1 = 62

16’s $(AB)_{16}$ = (16^2 - 1) – AB +1 = FF –AB +1 = 54 +1 =55

Complements

Subtraction using r’s Complements:

The subtraction of two n-digit unsigned numbers M - N in base r can be done as follows:

1. Add the minuend M to the r's complement of the subtrahend N. This performs $M + (r^n - N) = M - N + r^n$

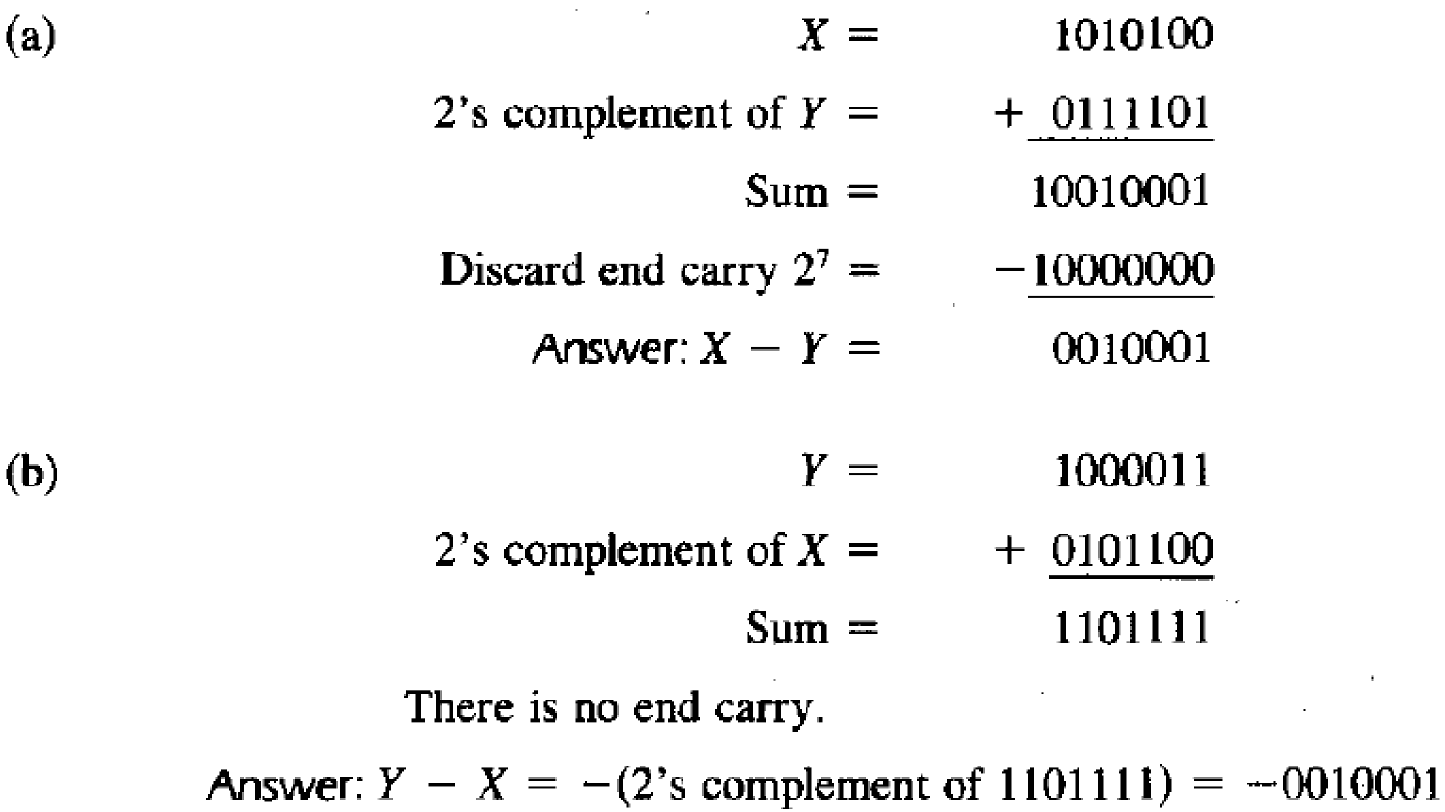
2. If M >= N, the sum will produce an end carry, r^n, which is discarded; what is left is the result M - N.

3. If M < N, the sum does not produce an end carry and is equal to $r^n - (N - M)$, which is the r's complement of (N - M). To obtain the answer in a familiar form, take the r's complement of the sum and place a negative sign in front.

Complements

Eg: Given the two binary numbers X = 1010100 and Y =1000011, perform the subtraction

(a) X – Y and (b) Y - X using 2's complements.



(a)

$$\begin{array}{rr} X = & 1010100 \\ \text{2's complement of } Y = & +\ \underline{0111101} \\ \text{Sum} = & 10010001 \\ \text{Discard end carry } 2^7 = & -\underline{10000000} \\ \text{Answer: } X - Y = & 0010001 \end{array}$$

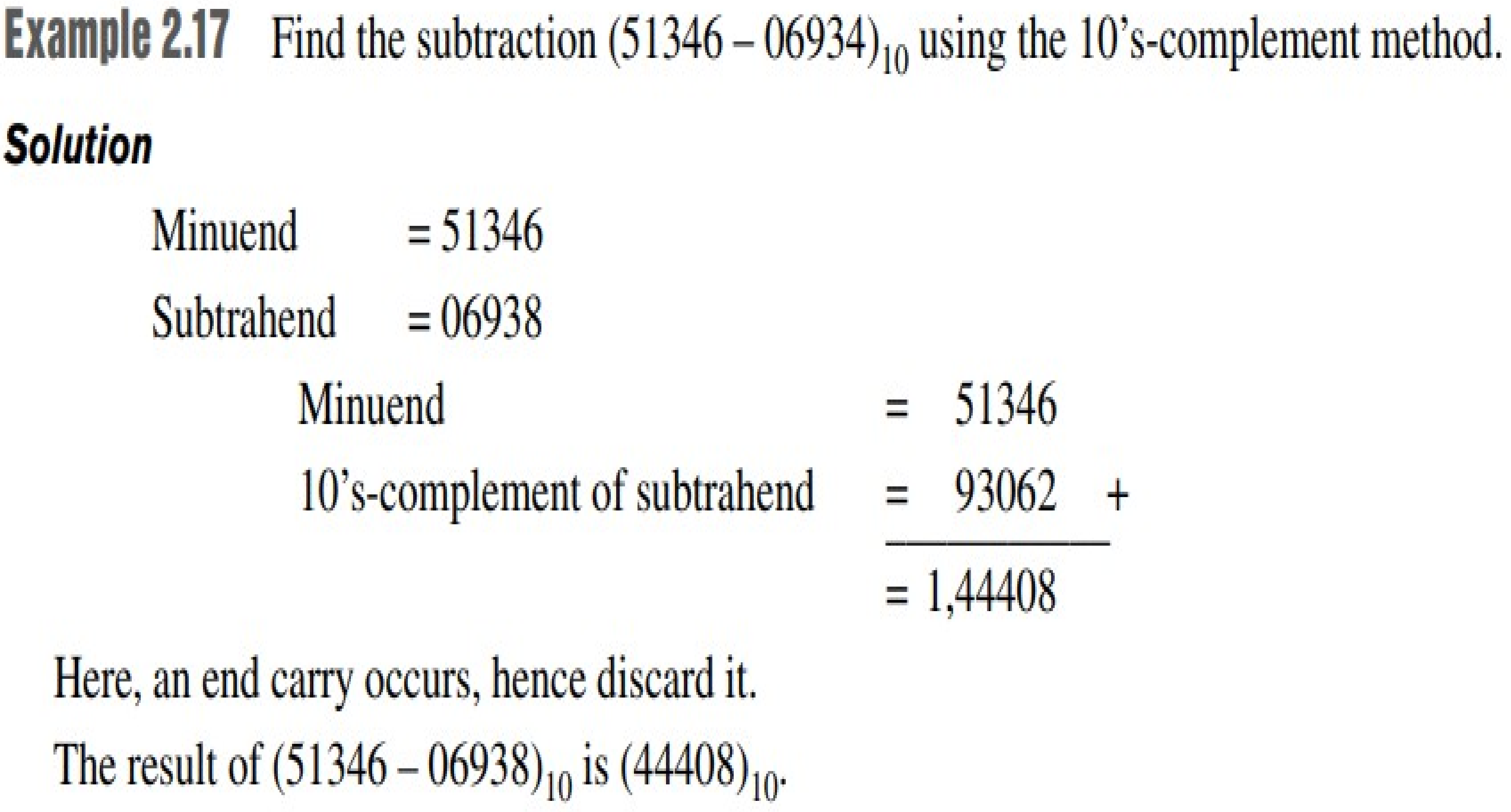
(b)

$$\begin{array}{rr} Y = & 1000011 \\ \text{2's complement of } X = & +\ \underline{0101100} \\ \text{Sum} = & 1101111 \end{array}$$

There is no end carry.

Answer: $Y - X = -(\text{2's complement of } 1101111) = -0010001$

Complements



Example 2.17 Find the subtraction $(51346 - 06934)_{10}$ using the 10's-complement method.

Solution

Minuend = 51346

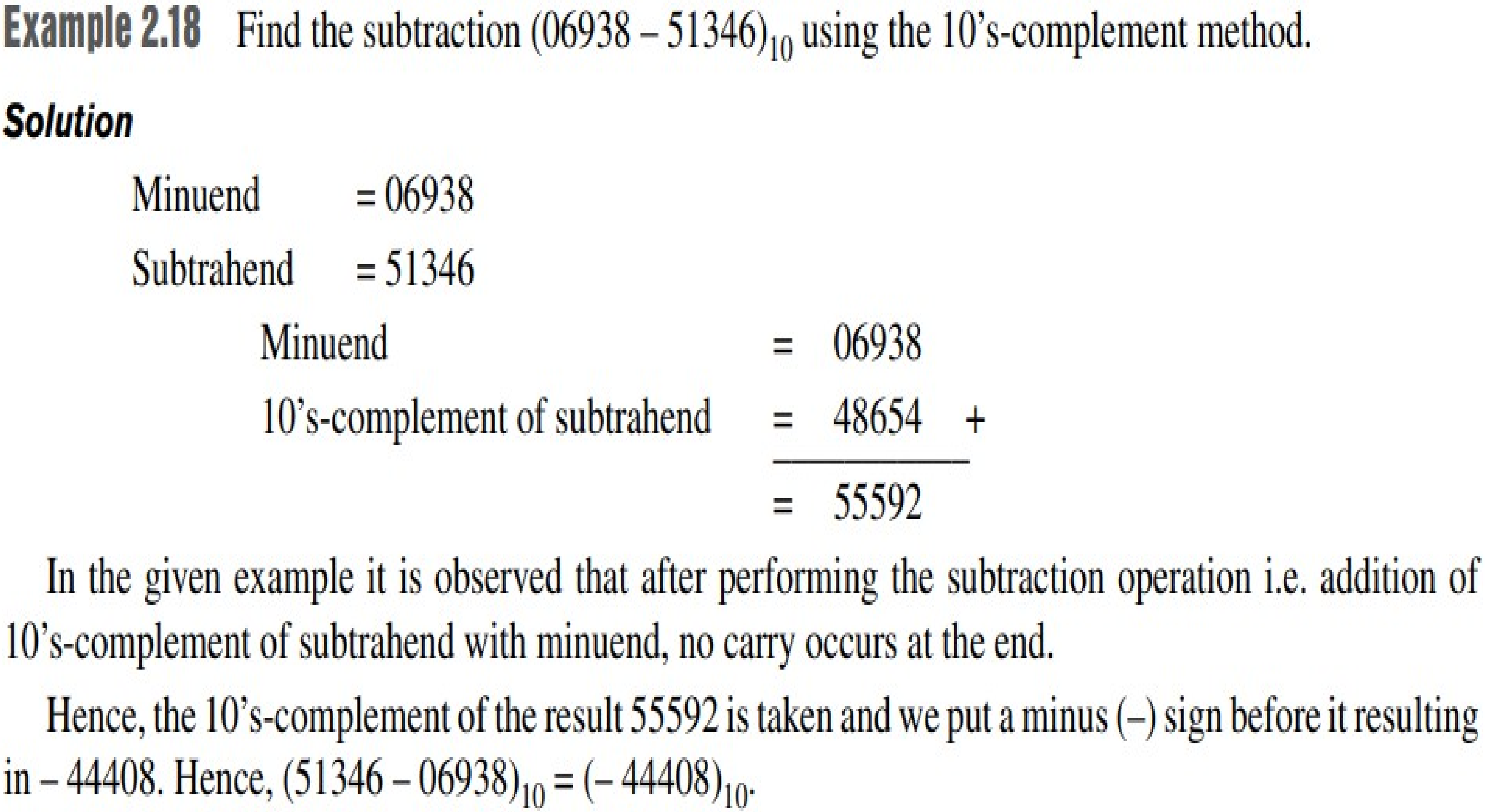
Subtrahend = 06938

$$
\begin{array}{lrl}
\text{Minuend} & = & 51346 \\
\text{10's-complement of subtrahend} & = & 93062 \quad + \\
\hline
 & = & 1,44408
\end{array}
$$

Here, an end carry occurs, hence discard it.

The result of $(51346 - 06938)_{10}$ is $(44408)_{10}$.

Complements



Example 2.18 Find the subtraction $(06938 - 51346)_{10}$ using the 10's-complement method.

Solution

Minuend = 06938

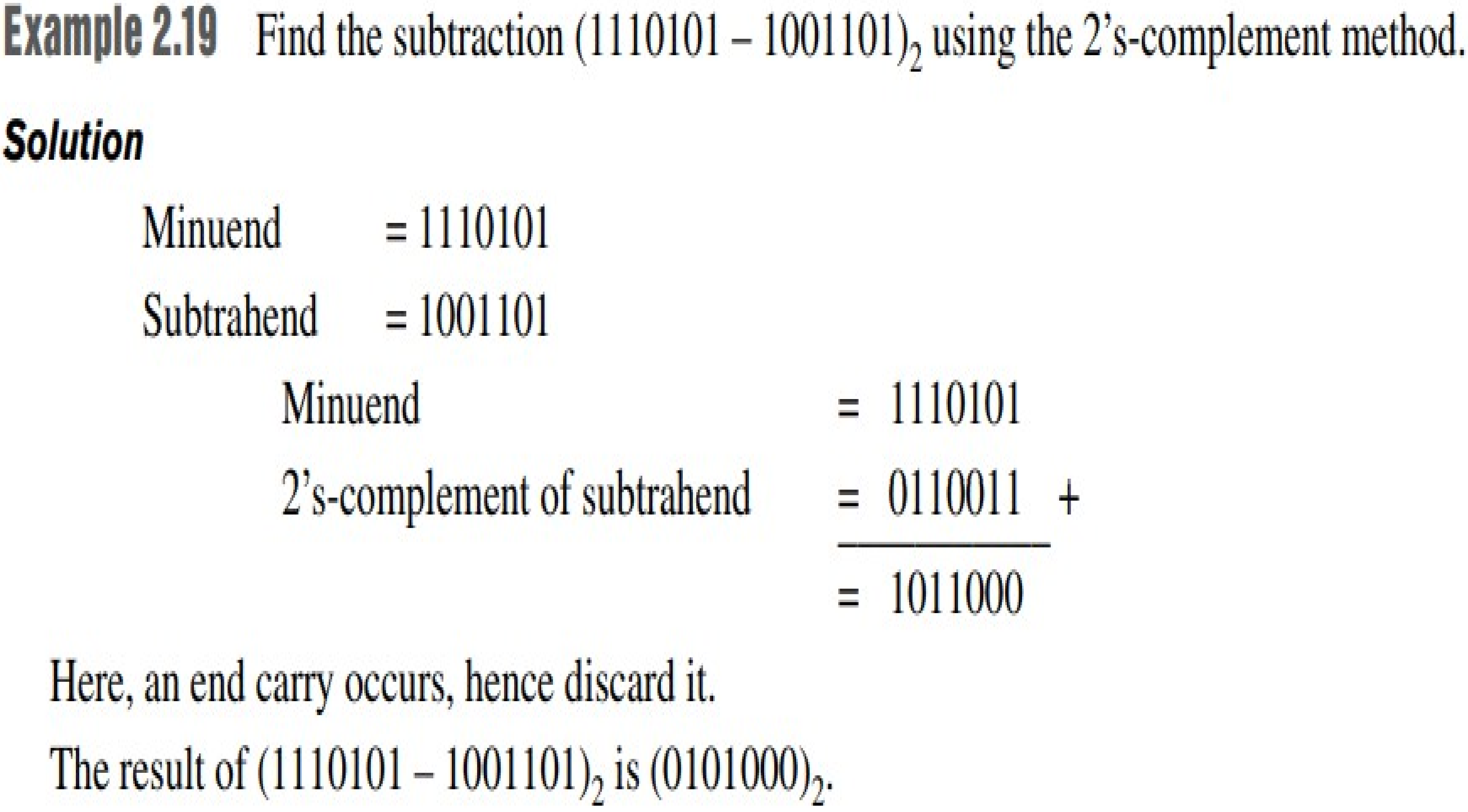
Subtrahend = 51346

$$\begin{array}{lrr} \text{Minuend} & = & 06938 \\ \text{10's-complement of subtrahend} & = & 48654 \quad + \\ \hline & = & 55592 \end{array}$$

In the given example it is observed that after performing the subtraction operation i.e. addition of 10's-complement of subtrahend with minuend, no carry occurs at the end.

Hence, the 10's-complement of the result 55592 is taken and we put a minus (–) sign before it resulting in – 44408. Hence, $(51346 - 06938)_{10} = (-\,44408)_{10}$.

Complements



Example 2.19 Find the subtraction $(1110101 - 1001101)_2$ using the 2's-complement method.

Solution

Minuend = 1110101

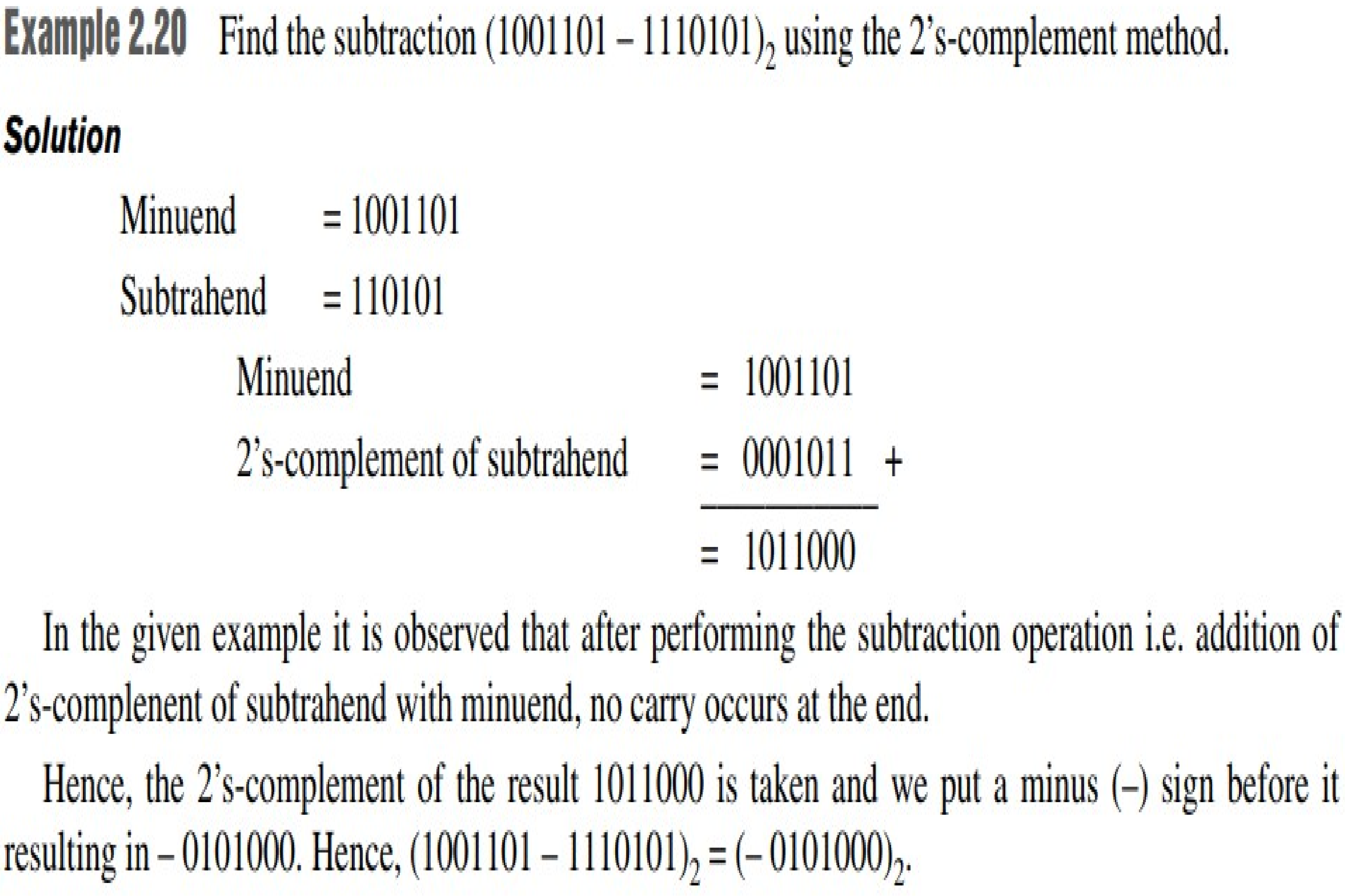
Subtrahend = 1001101

$$\begin{array}{lrl} \text{Minuend} & = & 1110101 \\ \text{2's-complement of subtrahend} & = & 0110011 \quad + \\ \hline & = & 1011000 \end{array}$$

Here, an end carry occurs, hence discard it.

The result of $(1110101 - 1001101)_2$ is $(0101000)_2$.

Complements



Example 2.20 Find the subtraction $(1001101 - 1110101)_2$ using the 2's-complement method.

Solution

Minuend = 1001101

Subtrahend = 110101

$$
\begin{array}{lrl}
\text{Minuend} & = & 1001101 \\
\text{2's-complement of subtrahend} & = & 0001011 \quad + \\
\hline
 & = & 1011000
\end{array}
$$

In the given example it is observed that after performing the subtraction operation i.e. addition of 2's-complenent of subtrahend with minuend, no carry occurs at the end.

Hence, the 2's-complement of the result 1011000 is taken and we put a minus (–) sign before it resulting in – 0101000. Hence, $(1001101 - 1110101)_2 = (-\,0101000)_2$.

Complements

Subtraction using (r-1)’s Complements:

The subtraction of two n-digit unsigned numbers M - N in base r can be done as follows:

1. Add the minuend M to the (r-1)'s complement of the subtrahend N. This performs $M + (r^n - N) = M - N + r^n$.

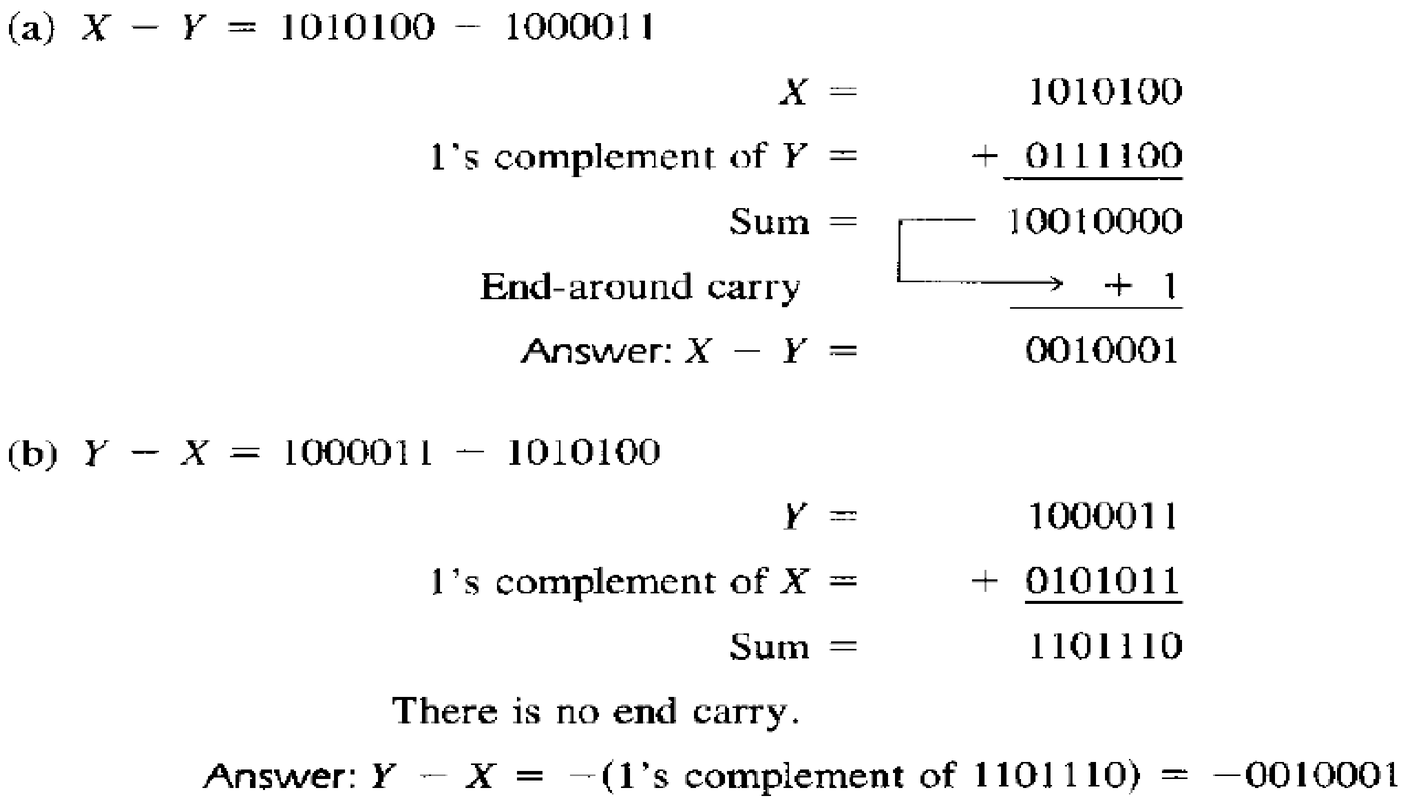
2. If M >= N, the sum will produce an end carry, which is added to the sum gives result M - N.

3. If M < N, the sum does not produce an end carry, To obtain the answer in a familiar form, take the (r-1)'s complement of the sum and place a negative sign in front.

Complements

Eg: Given the two binary numbers X = 1010100 and Y =1000011, perform the subtraction

(a) X – Y and (b) Y - X using1's complements.



(a) $X - Y = 1010100 - 1000011$

$$
\begin{array}{rr}
X = & 1010100 \\
\text{1's complement of } Y = & +\ \underline{0111100} \\
\text{Sum} = & 10010000 \\
\text{End-around carry} & \underline{+\ \ 1} \\
\text{Answer: } X - Y = & 0010001
\end{array}
$$

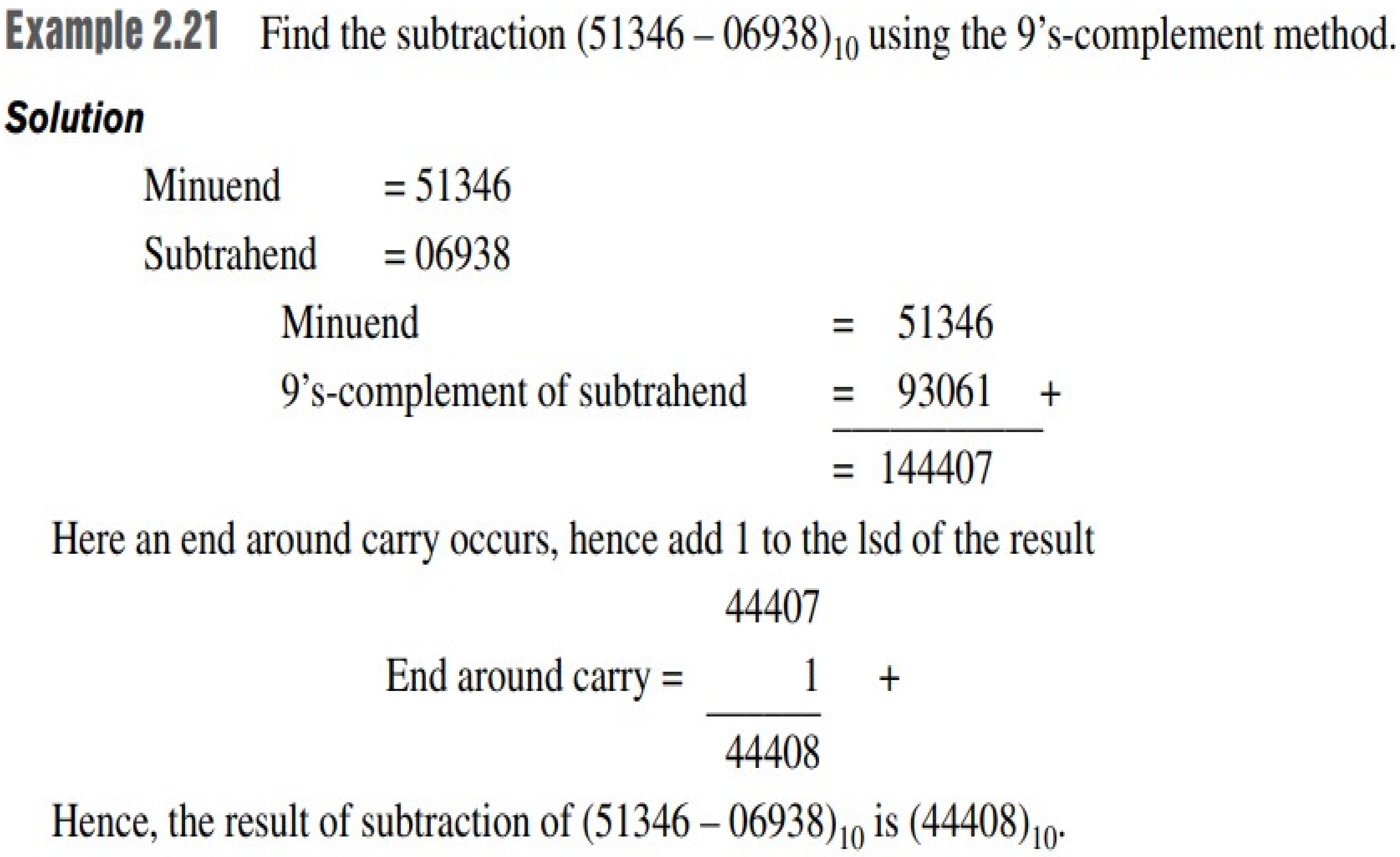
(b) $Y - X = 1000011 - 1010100$

$$
\begin{array}{rr}
Y = & 1000011 \\
\text{1's complement of } X = & +\ \underline{0101011} \\
\text{Sum} = & 1101110
\end{array}
$$

There is no end carry.

Answer: $Y - X = -(\text{1's complement of } 1101110) = -0010001$

Complements



Example 2.21 Find the subtraction $(51346 - 06938)_{10}$ using the 9's-complement method.

Solution

Minuend = 51346

Subtrahend = 06938

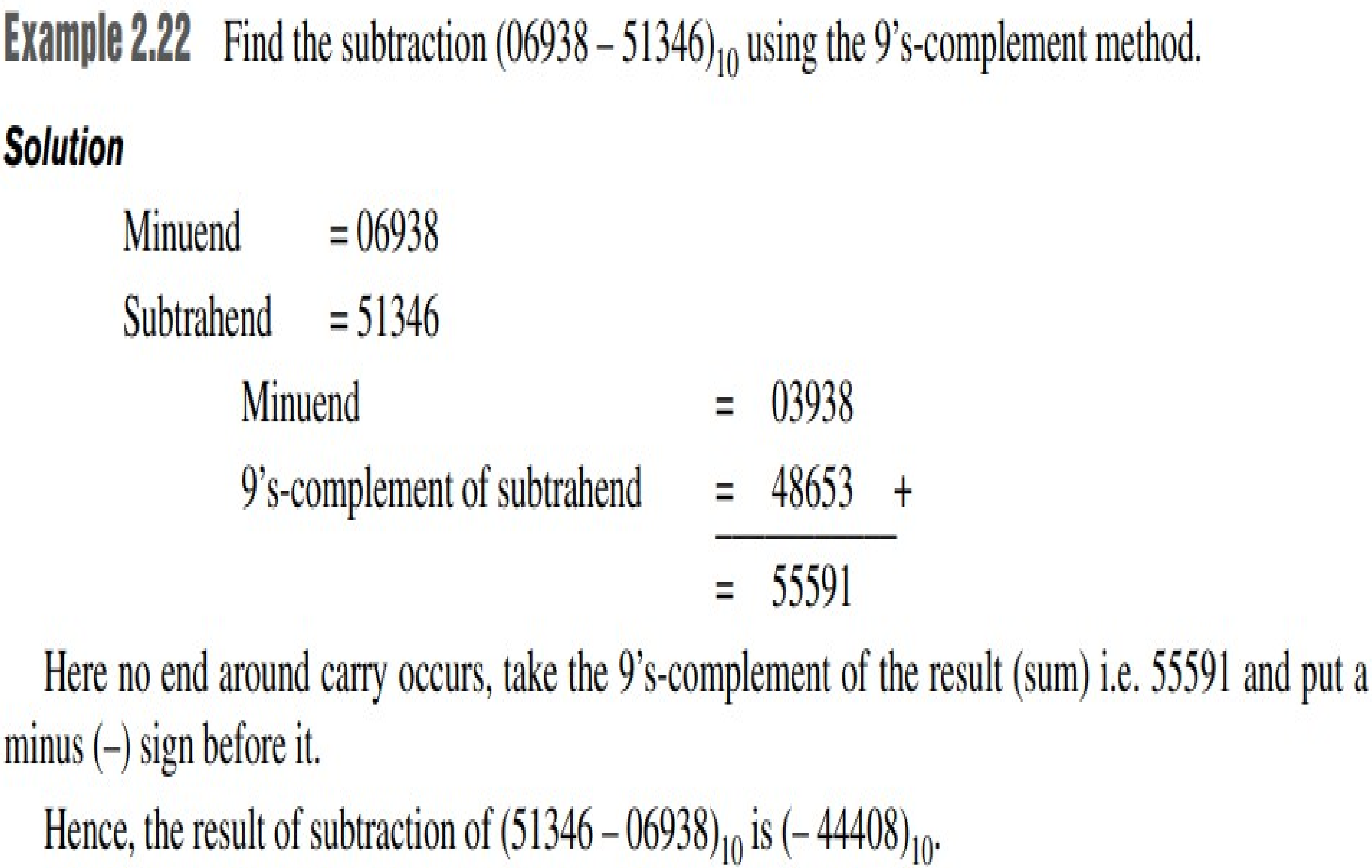
$$\begin{array}{lrl} \text{Minuend} & = 51346 & \\ \text{9's-complement of subtrahend} & = \underline{93061} & + \\ & = 144407 & \end{array}$$

Here an end around carry occurs, hence add 1 to the lsd of the result

$$\begin{array}{lrl} & 44407 & \\ \text{End around carry} = & \underline{1} & + \\ & 44408 & \end{array}$$

Hence, the result of subtraction of $(51346 - 06938)_{10}$ is $(44408)_{10}$.

Complements



Example 2.22 Find the subtraction $(06938 - 51346)_{10}$ using the 9's-complement method.

Solution

Minuend = 06938

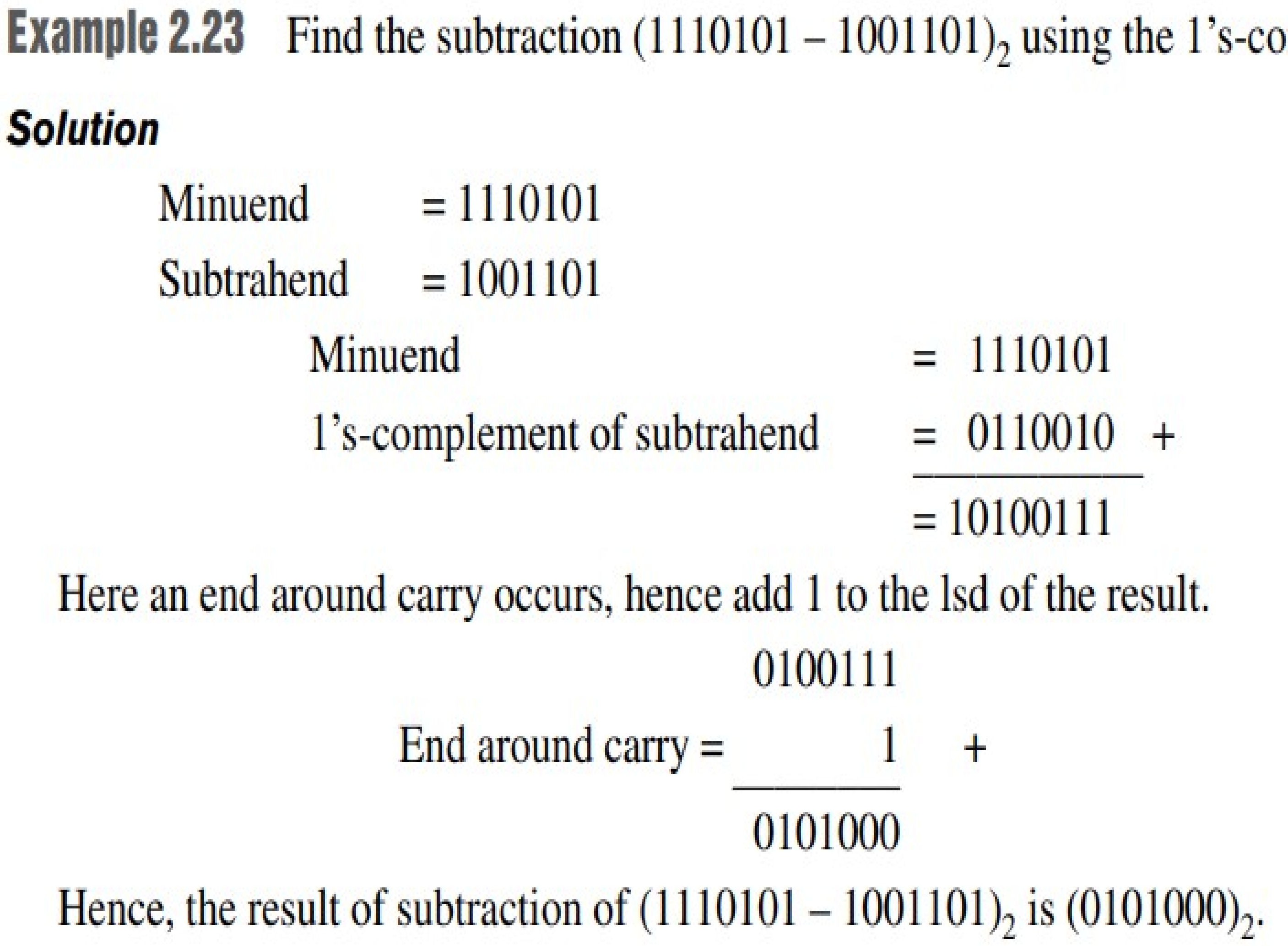
Subtrahend = 51346

$$\begin{array}{lrl} \text{Minuend} & = & 03938 \\ \text{9's-complement of subtrahend} & = & 48653 \quad + \\ \hline & = & 55591 \end{array}$$

Here no end around carry occurs, take the 9's-complement of the result (sum) i.e. 55591 and put a minus (–) sign before it.

Hence, the result of subtraction of $(51346 - 06938)_{10}$ is $(-44408)_{10}$.

Complements



Example 2.23 Find the subtraction $(1110101 - 1001101)_2$ using the 1's-complement method.

Solution

Minuend = 1110101

Subtrahend = 1001101

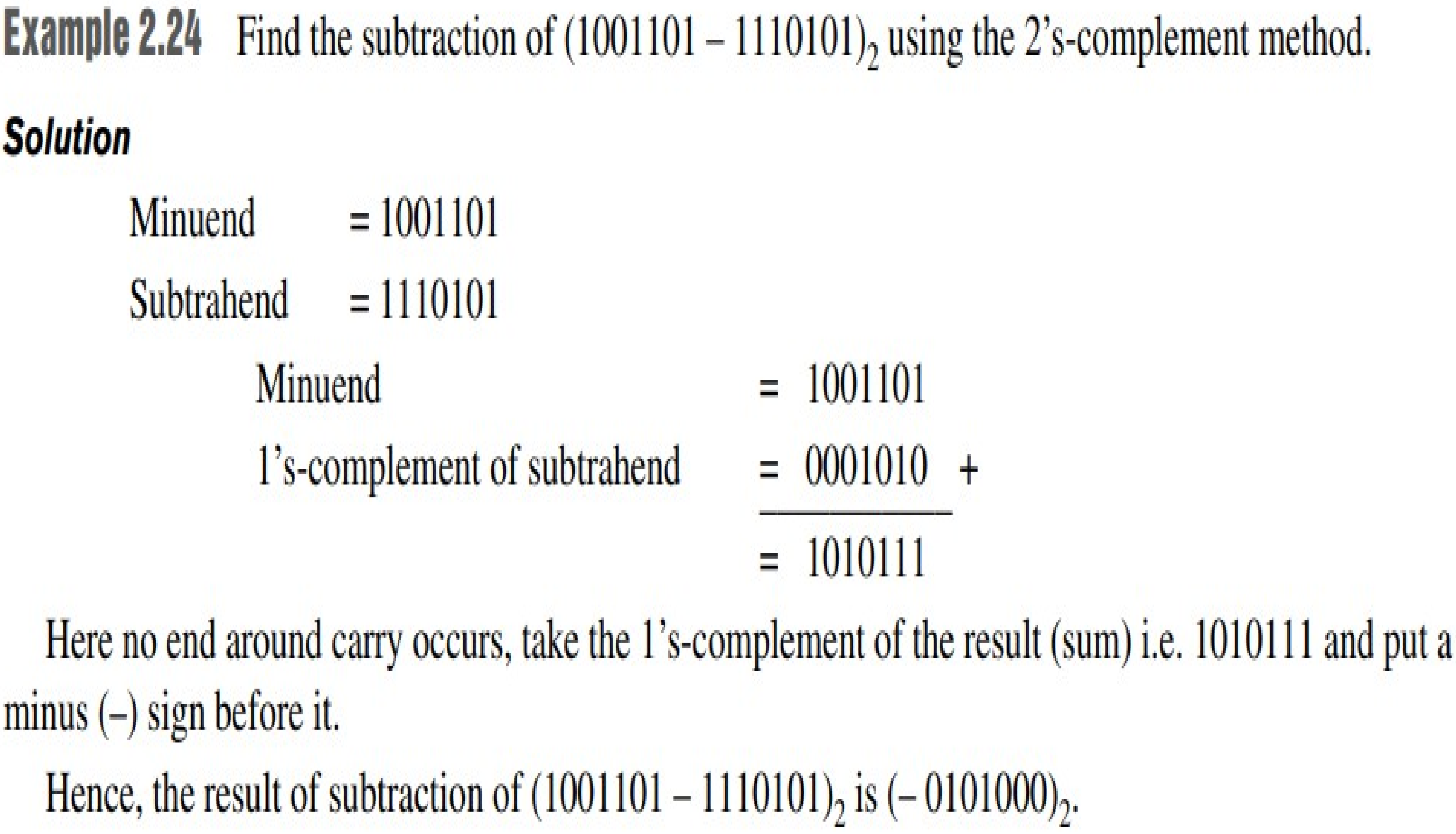
$$\begin{array}{lrl} \text{Minuend} & = & 1110101 \\ \text{1's-complement of subtrahend} & = & \underline{0110010} \quad + \\ & = & 10100111 \end{array}$$

Here an end around carry occurs, hence add 1 to the lsd of the result.

$$\begin{array}{lrl} & & 0100111 \\ \text{End around carry} & = & \underline{1} \quad + \\ & & 0101000 \end{array}$$

Hence, the result of subtraction of $(1110101 - 1001101)_2$ is $(0101000)_2$.

Complements



Example 2.24 Find the subtraction of $(1001101 - 1110101)_2$ using the 2's-complement method.

Solution

Minuend = 1001101

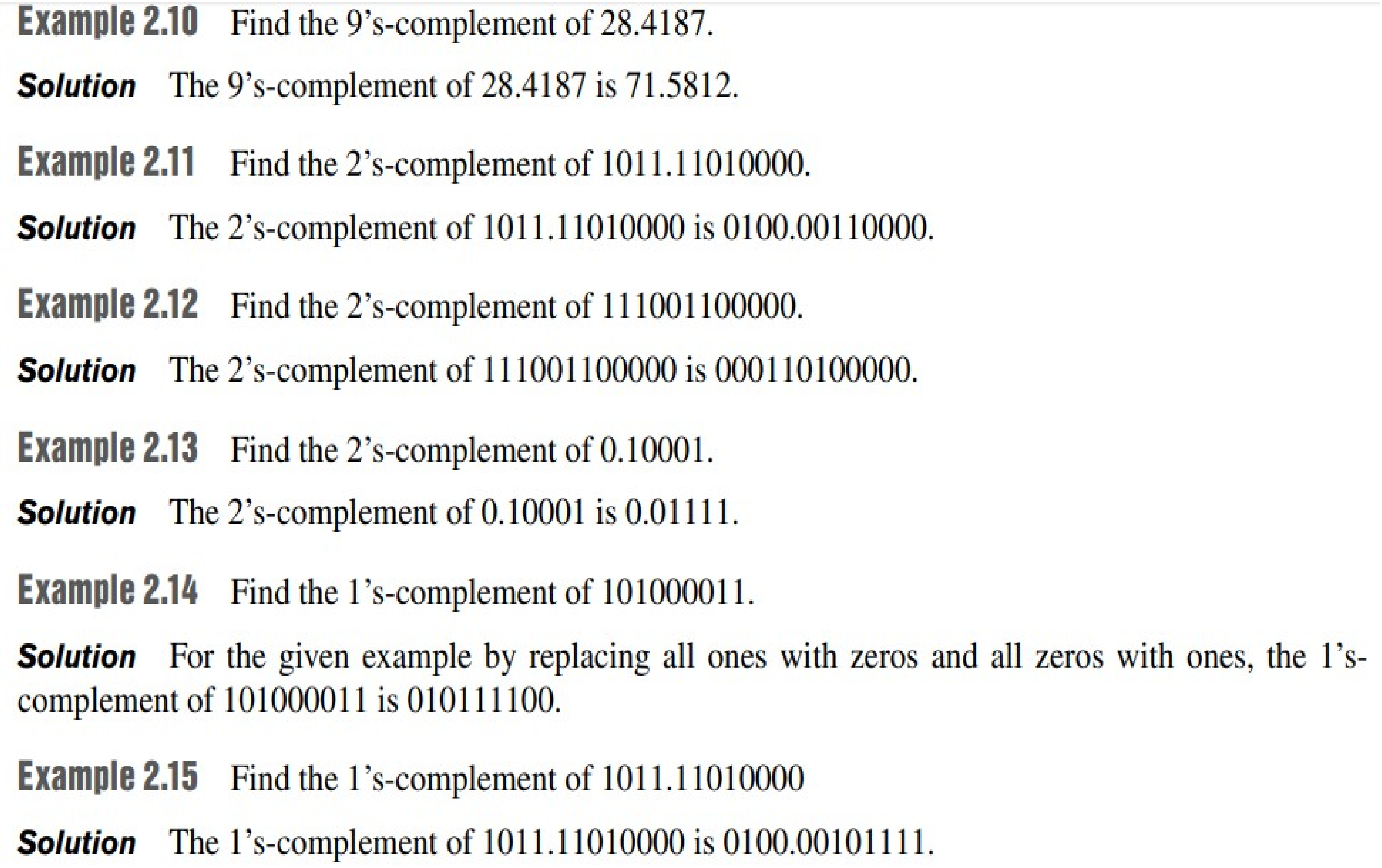
Subtrahend = 1110101

$$\begin{array}{lrl} \text{Minuend} & = & 1001101 \\ \text{1's-complement of subtrahend} & = & 0001010 \; + \\ \hline & = & 1010111 \end{array}$$

Here no end around carry occurs, take the 1's-complement of the result (sum) i.e. 1010111 and put a minus (–) sign before it.

Hence, the result of subtraction of $(1001101 - 1110101)_2$ is $(-\,0101000)_2$.

Complements



Example 2.10 Find the 9's-complement of 28.4187.

Solution The 9's-complement of 28.4187 is 71.5812.

Example 2.11 Find the 2's-complement of 1011.11010000.

Solution The 2's-complement of 1011.11010000 is 0100.00110000.

Example 2.12 Find the 2's-complement of 111001100000.

Solution The 2's-complement of 111001100000 is 000110100000.

Example 2.13 Find the 2's-complement of 0.10001.

Solution The 2's-complement of 0.10001 is 0.01111.

Example 2.14 Find the 1's-complement of 101000011.

Solution For the given example by replacing all ones with zeros and all zeros with ones, the 1's-complement of 101000011 is 010111100.

Example 2.15 Find the 1's-complement of 1011.11010000

Solution The 1's-complement of 1011.11010000 is 0100.00101111.

Binary Codes

Topic Objective	Mapping with CO
To understand classification of binary codes.	CO1

Binary Codes

Perquisite:

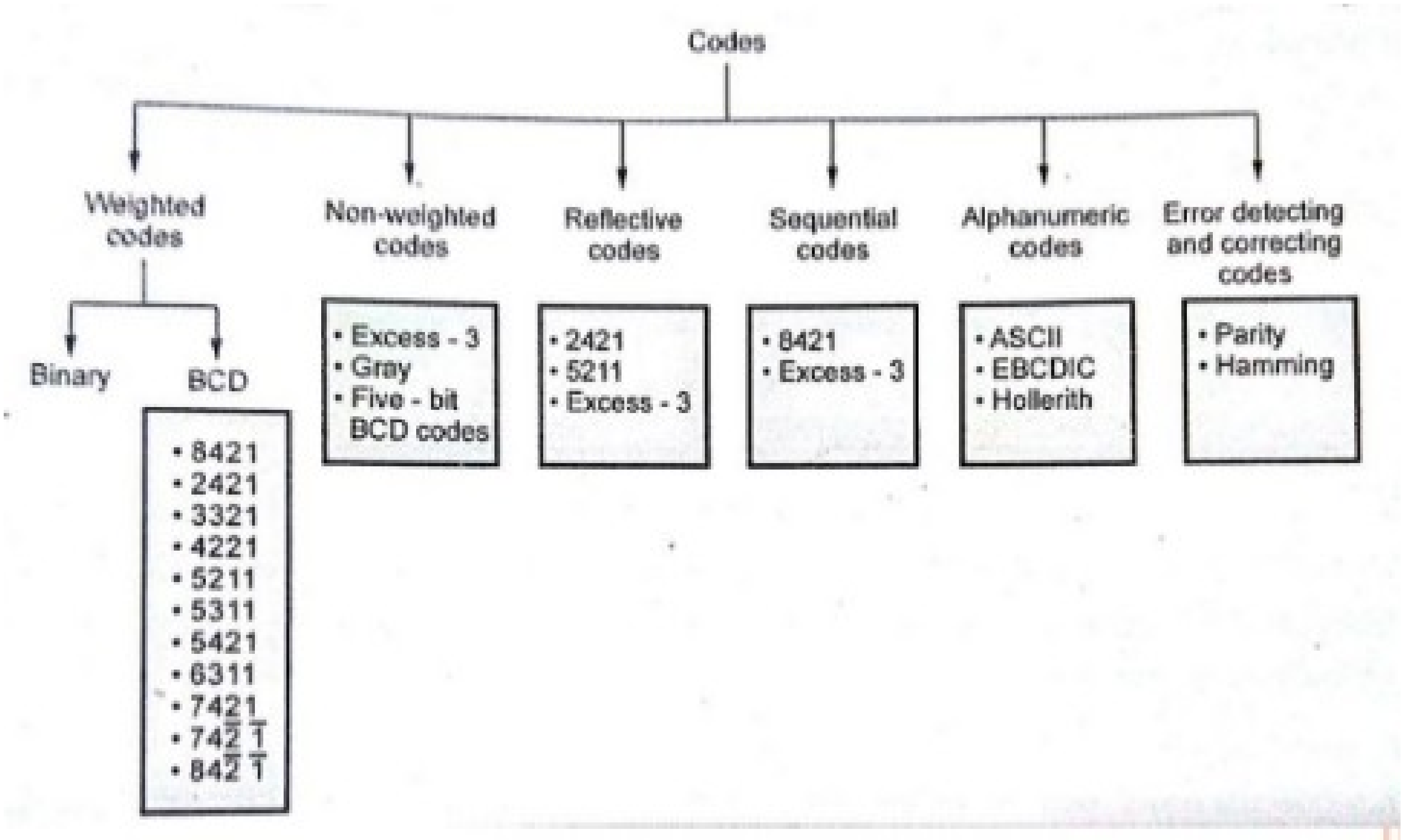
- Knowledge of number system.

Binary Codes

- In the coding, when numbers, letters or words are represented by a specific group of symbols, it is said that the number, letter or word is being encoded. The group of symbols is called as a code.
- The digital data is represented, stored and transmitted as group of binary bits.
- This group is also called as **binary code**. The binary code is represented by the number as well as alphanumeric letter.
- The distinct bit combinations of an n-bit code can be found by counting in binary from 0 to (2^n – 1).

Binary Codes

Classification of Binary Codes



Binary Codes

- **Weighted codes:**
  - In weighted codes, each digit is assigned a specific weight according to its position .
  - Several system of codes are used to express the decimal digits 0 to through 9. These codes have 8421,2421,3321…. All are the weighted codes .
  - In this codes each decimal digit is represented by a group of four bits .

- **Non-weighted codes:**
  - In these codes, positional weights are not assigned.
  - The example of non-weighted codes are excess-3 and gray codes.

- **Reflective codes**:
  - A code is reflective when the code is **self complementing**. In other words, when the code for 9 is the complement the code for 0, 8 for 1, 7 for 2, 6 for 3 and 5 for 4.
  - 2421, 5211 and XS-3 are the examples of reflective codes.

Binary Codes

- **Sequential codes**:
  - In sequential codes, each succeeding 'code is one binary number greater than its preceding code.
  - The 8421 and XS-3 are sequential codes.

- **Alphanumeric codes**:
  - Codes used to represent numbers, alphabetic characters, symbols.
  - Some of these codes are capable to representing some symbols and instructions as well.
  - Example of alphanumeric codes are : ASCII(American Standard Code for Information Interchange), EBCDIC(Extended Binary Coded Decimal Interchange Code).

Binary Codes

- **Error detecting and correcting codes**:
  - When digital data is transmitted from one system to other, unwanted electrical disturbance called "Noise" gets added to it.
  - The noise can force an "error" in digital information. That means a 0 may change to 1 or 1 to 0.
  - To detect and correct such errors we can use some special codes which possess the capacity to detect and correct the error . Such codes are called as error detecting and error correcting codes.

Weighted codes

Binary Coded Decimal (BCD) code:

- BCD codes are weighted codes.
- 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.

Advantages of BCD Codes:

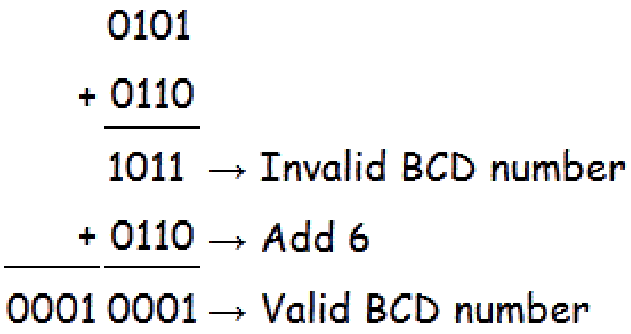
- It is very similar to decimal system.
- We need to remember binary equivalent of decimal numbers 0 to 9 only.

Disadvantages of BCD Codes:

- The addition and subtraction of BCD have different rules.
- The BCD arithmetic is little more complicated.
- BCD needs more number of bits than binary to represent the decimal number. So BCD is less efficient than binary.

BCD Codes

- The **addition of BCD numbers** is slightly different from **binary addition**. Here, the rules of binary addition are partially applicable only to the individual 4-bit groups.
- The **BCD addition**, is thus carried out by individually adding the corresponding 4-bit groups starting from the LSB side.
- If there is a carry to the next group and if the result belongs to any of the 6 illegal states than we add **6_{10}(0110)** to the sum term of that group and resulting carry is added in the next group.
- **Example: Perform BCD Addition of 5 and 6.**



```
          0101
        + 0110
          ----
          1011  → Invalid BCD number
        + 0110  → Add 6
     ---- ----
     0001 0001  → Valid BCD number
```

BCD Codes

Example: Perform BCD Addition of 184 and 576

	1	1		
• BCD	0001	1000	0100	184
	+ 0101	0111	0110	+576
Binary sum		10000	1010	
Add 6		0110	0110	
BCD sum	0111	0110	0000	760

BCD Codes

Example: Perform BCD Addition of 184 and 976

	1	1		
• BCD	0001	1000	0100	184
	+ 1001	0111	0110	+976
Binary sum	1011	10000	1010	
Add 6	0110	0110	0110	
BCD sum	1 0001	0110	0000	1160

Non-weighted codes

1. Excess-3 code 2. Gray code

Excess-3 Code:

- The excess-3 code is a non-weighted code used to express decimal numbers.
- Excess-3 codes are non-weighted and can be obtained by adding 3 to each decimal digit then it can be represented by using 4 bit binary number for each digit.
- An Excess-3 equivalent of a given binary number is obtained using the following steps:
  1. Find the decimal equivalent of the given binary number.
  2. Add +3 to each digit of decimal number.
  3. Convert the newly obtained decimal number back to binary number to get required excess-3 equivalent.
- Excess-3 code is non-weighted and self complementary code. A self complementary binary codes are always compliment themselves.

Excess-3 Code

- In other words, the 1's complement of an excess-3 code is the excess-3 code for the 9's complement of the corresponding decimal number.
- For example, the excess-3 code for decimal number 5 is 1000 and 1's complement of 1000 is 0111, which is excess-3 code for decimal number 4, and it is 9's complement of number 5.

These are following advantages of Excess-3 codes,

- These are self-complementary codes.
- The codes 0000 and 1111 are not used for any digit which is an advantage for memory organization as these codes can cause fault in transmission line.
- It has no limitation, and it considerably simplifies arithmetic operations.
- It is particularly significant for arithmetic operations as it overcomes shortcoming encountered while using 8421 BCD code to add two decimal digits whose sum exceeds 9.

Excess-3 Code

Example-1 –Convert decimal number 23 to Excess-3 code.

So, according to excess-3 code we need to add 3 to both digit in the decimal number then convert into 4-bit binary number for result of each digit. Therefore,

= 23+33=56 =0101 0110 which is required excess-3 code for given decimal number 23.

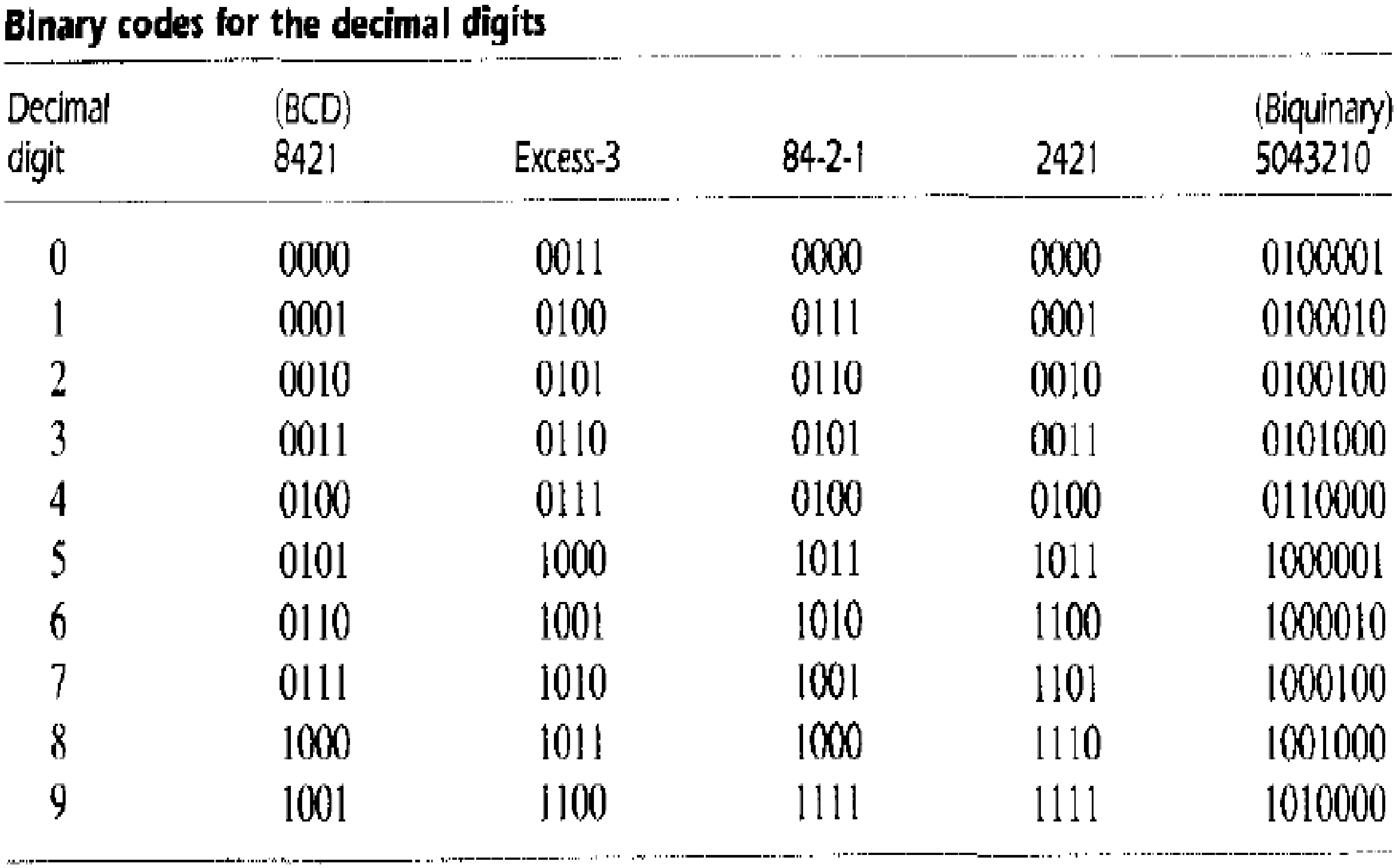
Example 2− Convert Excess-3 code 1001001 into BCD and decimal number.

So, grouping 4-bit for each group, i.e., 0100 1001 and subtract 0011 0011 from given number. Therefore,

= 0100 1001 -0011 0011 =0001 0110

So, binary coded decimal number is 0001 0110 and decimal number will be 16.

Binary Codes



Binary codes for the decimal digits

Decimal digit	(BCD) 8421	Excess-3	84-2-1	2421	(Biquinary) 5043210
0	0000	0011	0000	0000	0100001
1	0001	0100	0111	0001	0100010
2	0010	0101	0110	0010	0100100
3	0011	0110	0101	0011	0101000
4	0100	0111	0100	0100	0110000
5	0101	1000	1011	1011	1000001
6	0110	1001	1010	1100	1000010
7	0111	1010	1001	1101	1000100
8	1000	1011	1000	1110	1001000
9	1001	1100	1111	1111	1010000

Non-weighted codes

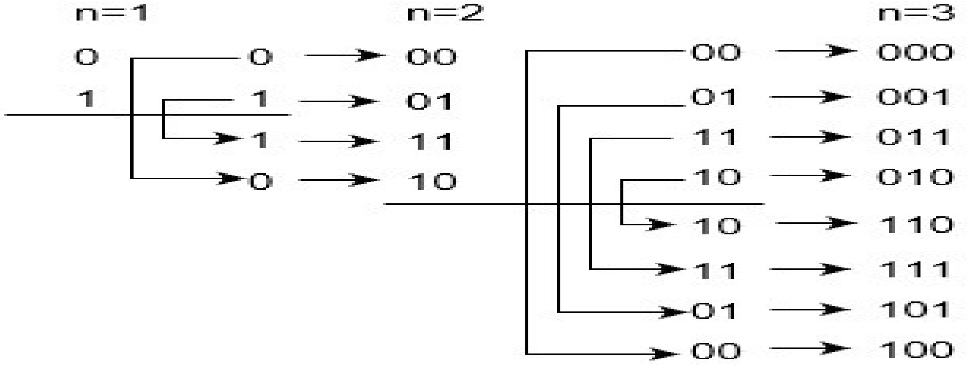
Gray codes:

- The reflected binary code or Gray code is an ordering of the binary numeral system such that two successive values differ in only one bit (binary digit).
- Gray code also known as reflected binary code, because the first (n/2) values compare with those of the last (n/2) values, but in reverse order.
- Gray codes are used in the general sequence of hardware-generated binary numbers.
- The numbers cause ambiguities or errors when the transition from one number to its successive is done. This code simply solves this problem by changing only one bit when the transition is between numbers is done.

Gray codes

How to generate Gray code?

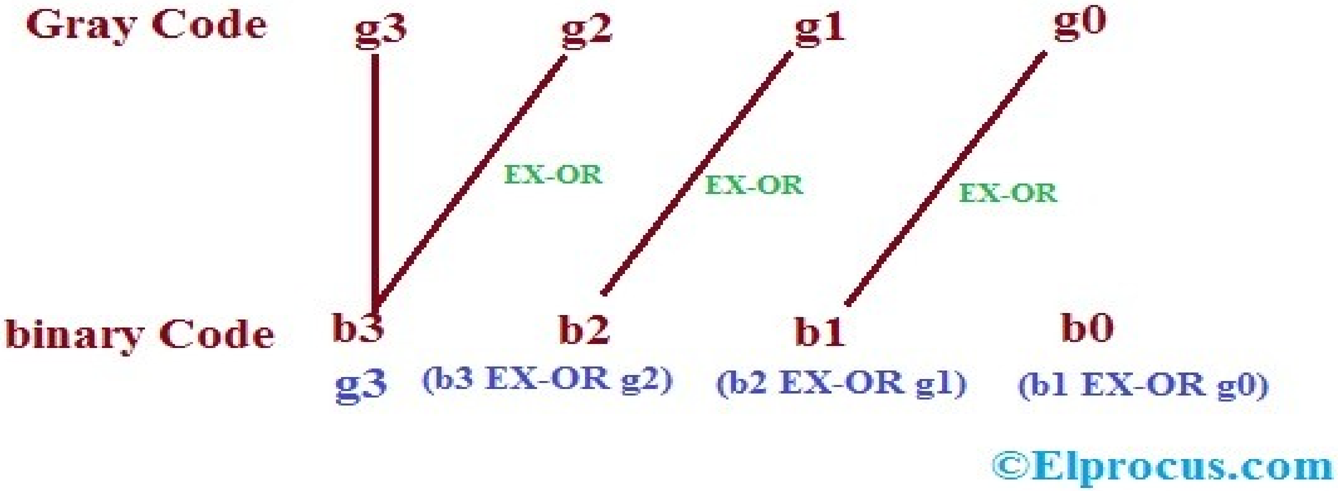
- The prefix and reflect method are recursively used to generate the Gray code of a number.
- For generating gray code:
  - Generate code for n=1: 0 and 1 code.
  - Take previous code in sequence: 0 and 1.
  - Add reversed codes in the following list: 0, 1, 1 and 0.
  - Now add prefix 0 for original previous code and prefix 1 for new generated code: 00, 01, 11, and 10



Gray codes

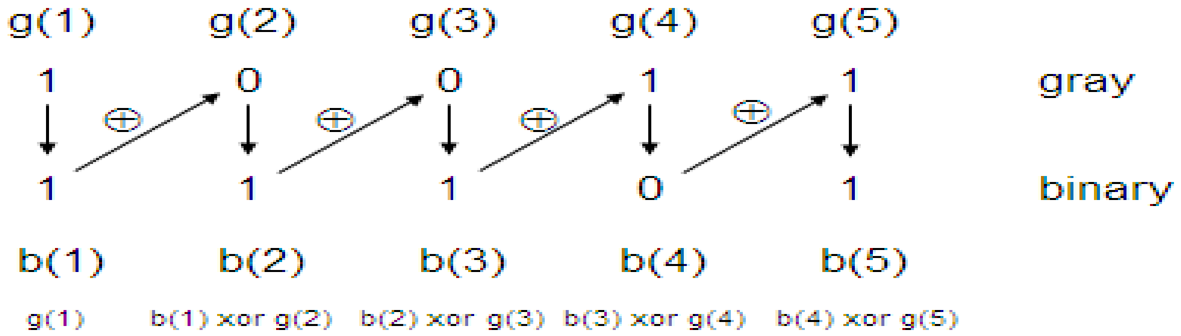
Gray to Binary Code:

- To change gray to binary code, take down the MSB digit of the gray code number, as the primary digit or the MSB of the gray code is similar to the binary digit.
- To get the next straight binary bit, it uses the XOR operation among the primary bit or MSB bit of binary to the next bit of the gray code.



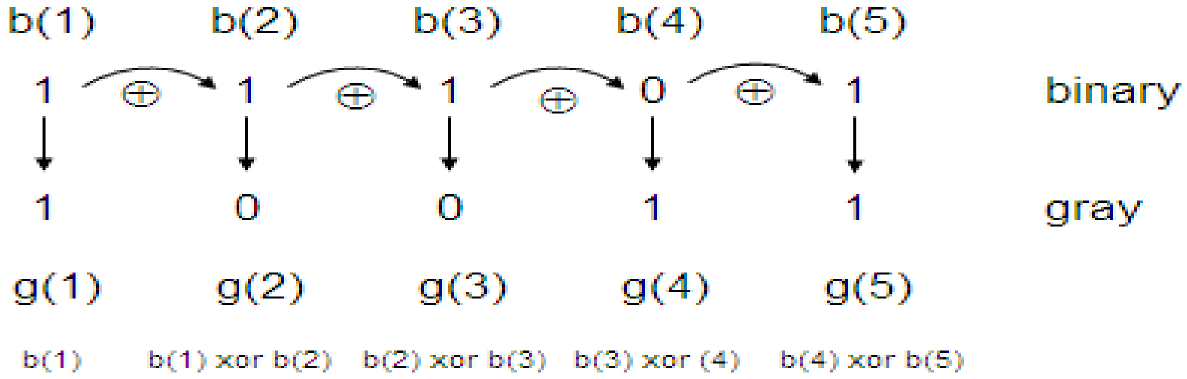
Gray codes

- **Convert the 10011 gray code into binary code**.



g(1)	g(2)	g(3)	g(4)	g(5)	
1	0	0	1	1	gray
1	1	1	0	1	binary
b(1)	b(2)	b(3)	b(4)	b(5)	
g(1)	b(1) xor g(2)	b(2) xor g(3)	b(3) xor g(4)	b(4) xor g(5)	

- **Convert the 10011 binary code into gray code.**



b(1)	b(2)	b(3)	b(4)	b(5)	
1	1	1	0	1	binary
1	0	0	1	1	gray
g(1)	g(2)	g(3)	g(4)	g(5)	
b(1)	b(1) xor b(2)	b(2) xor b(3)	b(3) xor (4)	b(4) xor b(5)	

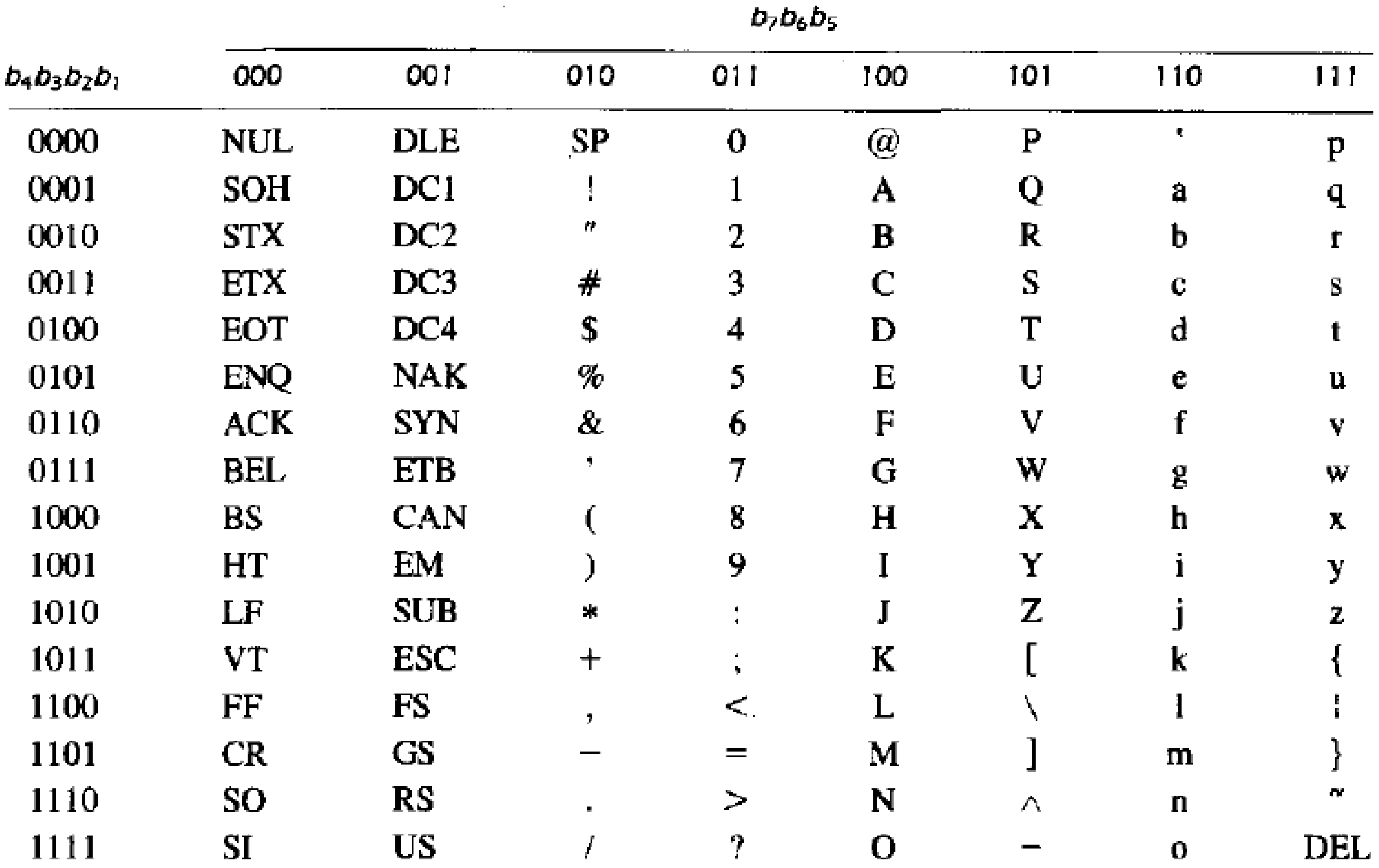
Alphanumeric Codes

ASCII Character Code:

- Many applications of digital computers require the handling of data not only of numbers, but also of letters.
- An alphanumeric character set is a set of elements that includes the 10 decimal digits, the 26 letters of the alphabet, and a number of special characters.
- The standard binary code for the alphanumeric characters is ASCII (American Standard Code for Information Interchange).
- It uses 7 bits to code 128 characters, but it can be considered as an 8-bit code with MSB = 0 always.
- The first 32 ASCII characters are non graphic commands ,these are used only for command purpose.
- The ASCII code set consist of 94 printable characters, SPACE and DELETE characters, and 32 control symbols.

Alphanumeric Codes

American Standard Code for Information Interchange (ASCII)



	$b_7b_6b_5$							
$b_4b_3b_2b_1$	000	001	010	011	100	101	110	111
0000	NUL	DLE	SP	0	@	P	`	p
0001	SOH	DC1	!	1	A	Q	a	q
0010	STX	DC2	″	2	B	R	b	r
0011	ETX	DC3	#	3	C	S	c	s
0100	EOT	DC4	$	4	D	T	d	t
0101	ENQ	NAK	%	5	E	U	e	u
0110	ACK	SYN	&	6	F	V	f	v
0111	BEL	ETB	’	7	G	W	g	w
1000	BS	CAN	(	8	H	X	h	x
1001	HT	EM	)	9	I	Y	i	y
1010	LF	SUB	*	:	J	Z	j	z
1011	VT	ESC	+	;	K	[	k	{
1100	FF	FS	,	<	L	\	l	¦
1101	CR	GS	−	=	M	]	m	}
1110	SO	RS	.	>	N	^	n	~
1111	SI	US	/	?	O	—	o	DEL

Daily Quiz

- Classify binary codes.
- Differentiate between weighted and non weighted codes.
- Binary equivalent of gray code 101011 is:

a) 110010

b) 101010

c) 110110

d) 110011

- Classify binary codes.
- Differentiate between weighted and non weighted codes.
- Binary equivalent of gray code 101011 is:

a) 110010

b) 101010

c) 110110

d) 110011

Daily Quiz

- Excess-3 code of 739 is:

a) 1010 0110 1100

b) 1101 1001 1000

c) 1101 0110 1100

d) 1010 1001 1000

- What do you mean by binary codes?
- Classify binary codes.
- Excess-3 codes are also called:

a) Non-weighted codes.

b) Alphanumeric codes.

c) Sequential codes.

d) Reflective codes.

FACULTY VIDEO LINKS, YOUTUBE & NPTEL VIDEO LINKS AND ONLINE COURSES DETAILS

Youtube/other Video Links:

- https://www.youtube.com/watch?v=VBM5XTvJSRQ&ab_channel=JAESCompanyJAESCompany
- https://www.youtube.com/watch?v=1A_NcXxdoCc&ab_channel=NesoAcademyNesoAcademyVerified

Old Questions

- Convert 1110110 binary code into gray code.
- What do you understand by self complementing codes?
- Construct the hamming code, if 4 bit data 1001 is transformed.
- A receiver receives the hamming code 1110101, What is the correct code for even parity is?
- Find binary equivalent of gray code 101011.

Recap

- The digital data is represented, stored and transmitted as group of binary bits, This group is also called as **binary code.**
- In weighted codes, each digit is assigned a specific weight according to its position .
- In weighted codes, positional weights are not assigned.
- A code is reflective when the code is **self complementing**.

Hamming Codes

Topic Objective	Mapping with CO
To understand hamming codes.	CO1

Hamming Codes

Perquisite:

- Knowledge of number system.

Hamming Codes

Hamming Code: When data is transferred from one location to another location, there is always a possibility that an error may occur. Error result in changes to the content of data transferred.

- Hamming code is an error-detection method that can detect some errors, but it is only capable of single error correction.
- Hamming Codes are named after Richard Hamming, the American mathematician.
- It has fix format, there are reserved bit for the parity at the position $2^{0,} 2^{1,} 2^{2,}$ 2^3 and so on.
- The **minimum value of 'k'** for which the following relation is correct valid is nothing but the required number of parity bits.

$\mathbf{2^k \geq n+k+1}$ Where,

‘n’ is the number of bits in the binary code information

‘k’ is the number of parity bits

Therefore, the number of bits in the Hamming code is equal to n + k.

Hamming Codes

- Let n = 4, bits present is data

$$2^k \geq 4+k+1$$

k must be equal to 3.

- Let n = 8, bits present is data

$$2^k \geq 8+k+1$$

k =4

- **Table for different data bits and check bits:**

Number of information bits	Number of parity bits
2 to 4	3
5 to 11	4
12 to 26	5
27 to 57	6
58 to 120	7

Parity Bits	Bits to be checked
P1	1,3,5,7,9,11,13,15,…..
P2	2,3,6,7,10,11,14,15,….
P4	4,5,6,7,12,13,14,15,…..
P8	8,9,10,11,12,13,14,15,…

Hamming Codes

Format of Hamming Code for 4 bit:

If D3, D5, D6 and D7 are data bits, (for n =4, k= 3)

7	6	5	4	3	2	1
D7	D6	D5	P4	D3	P2	P1

P1 = (1, 3, 5, 7) should have even number of 1’s.
P2 = (2, 3, 6, 7) should have even number of 1’s.
P4 = (4, 5, 6, 7) should have even number of 1’s.

Format of Hamming Code for 8 bit:

If D3, D5, D6, D7, D9, D10,D11, D12 are data bits, (for n =8, k= 4)

12	11	10	9	8	7	6	5	4	3	2	1
D12	D11	D10	D9	P8	D7	D6	D5	P4	D3	P2	P1

P1 = (1, 3, 5, 7, 9, 11) should have even number of 1’s.
P2 = (2, 3, 6, 7, 10,11) should have even number of 1’s.
P4 = (4, 5, 6, 7, 12) should have even number of 1’s.
P8 = (8, 9, 10, 11, 12) should have even number of 1’s.

Hamming Codes

1. Construct the hamming code, if 4 bit data 1100 is transformed.

Sol: for n = 4, k= 3

7	6	5	4	3	2	1
1	1	0	P4	0	P2	P1

P1 = (1, 3, 5, 7) (p1,0,0,1) should have even number of 1's. P1 = 1

P2 = (2, 3, 6, 7) (P2, 0,1,1) should have even number of 1's. P2 = 0

P4 = (4, 5, 6, 7) (P4, 0,1,1) should have even number of 1's. P4 = 0

Hamming Code : **1100001**

2. Construct the hamming code, if 8 bit data 10101011 is transformed.

Sol: for n = 8, k= 4

12	11	10	9	8	7	6	5	4	3	2	1
1	0	1	0	P8	1	0	1	P4	1	P2	P1

P1 = (1, 3, 5, 7, 9, 11) should have even number of 1's. P1 = 1

P2 = (2, 3, 6, 7, 10,11) should have even number of 1's. P2 = 1

P4 = (4, 5, 6, 7, 12) should have even number of 1's. P4 = 1

P8 = (8, 9, 10, 11, 12) should have even number of 1's. P8 = 0

Hamming Code: **101001011111**

Hamming Codes

- **Correction for the checking of error in 4 bit data with even parity received at the receiver:**

7	6	5	4	3	2	1
D7	D6	D4	P4	D3	P2	P1

- P1 = (1, 3, 5, 7) should have even number of 1's.
  If P1 is even means no error so C1 = 0
  If P1 is not even means error so C1 = 1

- P2 = (2, 3, 6, 7) should have even number of 1's.
  If P2 is even means no error so C2 = 0
  If P2 is not even means error so C2 = 1

- P4 = (4, 5, 6, 7) should have even number of 1's.
  If P4 is even means no error so C4 = 0
  If P4 is not even means error so C4 = 1

So, error bit C = $C_4C_2C_1$

Hamming Codes

- **<u>Correction for the checking of error in 8 bit data with even parity received at the receiver:</u>**

12	11	10	9	8	7	6	5	4	3	2	1
D12	D11	D10	D9	P8	D7	D6	D5	P4	D3	P2	P1

- P1 = (1, 3, 5, 7, 9, 11) should have even number of 1's.
  If P1 is even means no error so C1 = 0
  If P1 is not even means error so C1 = 1
- P2 = (2, 3, 6, 7, 10,11) should have even number of 1's.
  If P2 is even means no error so C2 = 0
  If P2 is not even means error so C2 = 1
- P4 = (4, 5, 6, 7, 12) should have even number of 1's.
  If P4 is even means no error so C4 = 0
  If P4 is not even means error so C4 = 1
- P8 = (8, 9, 10, 11, 12) should have even number of 1's.
  If P8 is even means no error so C8 = 0
  If P8 is not even means error so C8 = 1
  So, error bit C = $C_8C_4C_2C_1$

Hamming Codes

1. Receiver receives the hamming code 1101001, check whether the data received is correct if not, check the error and correct it.

Sol:

7	6	5	4	3	2	1
1	1	0	1	0	0	1

- P1 = (1, 3, 5, 7) should have even number of 1's.
  P1 is already even, so no error, C1 = 0
- P2 = (2, 3, 6, 7) should have even number of 1's.
  P2 is already even, so no error, C2 = 0
- P4 = (4, 5, 6, 7) should have even number of 1's.
  P4 is odd , so there is error, C4 = 1
  So, error bit C = $C_4C_2C_1$, C = 100 = 4
  C = 4 so the error in 4th bit. The correct code is 11**0**0001

Hamming Codes

2. Receiver receives the hamming code 1111001, check whether the data received is correct if not, check the error and correct it.

Sol:

7	6	5	4	3	2	1
1	1	1	1	0	0	1

- P1 = (1, 3, 5, 7) should have even number of 1's.
  P1 is not even, so error, C1 = 1
- P2 = (2, 3, 6, 7) should have even number of 1's.
  P2 is even, so no error, C2 = 0
- P4 = (4, 5, 6, 7) should have even number of 1's.
  P4 is already even, so no error, C4 = 0
  So, error bit C = $C_4C_2C_{1\,,}$ C = 001 = 1
  C = 1 so error is there in the 1st bit. So correct data is 111100**0**.

Hamming Codes

3. Receiver receives the hamming code 110010001110, check whether the data received is correct if not, check the error and correct it.

Sol:

12	11	10	9	8	7	6	5	4	3	2	1
1	1	0	0	1	0	0	0	1	1	1	0

- P1 = (1, 3, 5, 7,9,11) should have even number of 1’s.
  P1 is even, so no error, C1 = 0

Click to add text

- P2 = (2, 3, 6, 7,10, 11) should have even number of 1’s.
  P2 is not even, so error, C2 = 1
- P4 = (4, 5, 6, 7, 12) should have even number of 1’s.
  P4 is not even, so error, C4 = 0
- P8 = (8, 9, 10, 11, 12) should have even number of 1’s.
  P8 is not even, so error, C8 = 1

So, error bit C = $C_8C_4C_2C_1$ C = 1010 = 10

C = 10 so error is there in the 10th bit correct code is: 11**1**010001110

Hamming Codes

4. Receiver receives the hamming code 1101001, check whether the data received is correct if not, check the error and correct it. Use odd parity.

Sol: **Odd Parity:**

In the case of odd parity, for a given set of bits, the number of 1's are counted. If that count is even, the parity bit value is set to 1, making the total count of occurrences of 1's an odd number. If the total number of 1's in a given set of bits is already odd, the parity bit's value is 0.

7	6	5	4	3	2	1
1	1	0	1	0	0	1

- P1 = (1, 3, 5, 7) should have odd number of 1's.

  P1 is even, so error, C1 = 1
- P2 = (2, 3, 6, 7) should have odd number of 1's.

  P2 is even, so error, C2 = 1
- P4 = (4, 5, 6, 7) should have odd number of 1's.

  P4 is odd , so there is no error, C4 = 0

  So, error bit $C = C_4C_2C_1$, C = 011 = 3

  C = 3 so the error in 3rd bit. The correct code is 1101**1**01

Daily Quiz

- Classify binary codes.
- Differentiate between weighted and non weighted codes.
- Binary equivalent of gray code 101011 is:

a) 110010

b) 101010

c) 110110

d) 110011

- Construct the hamming code, if 4 bit data 1100 is transformed.(Use odd parity)

a) 1101011

b) 1101010

c) 1101110

d) 1011010

Daily Quiz

- Excess-3 code of 739 is:

a) 1010 0110 1100

b) 1101 1001 1000

c) 1101 0110 1100

d) 1010 1001 1000

- What do you mean by binary codes?
- Classify binary codes.
- Excess-3 codes are also called:

a) Non-weighted codes.

b) Alphanumeric codes.

c) Sequential codes.

d) Reflective codes.

FACULTY VIDEO LINKS, YOUTUBE & NPTEL VIDEO LINKS AND ONLINE COURSES DETAILS

Youtube/other Video Links:

- https://nptel.ac.in/courses/117/106/117106086/
- https://www.youtube.com/watch?v=kAezzdEGJ8A

Old Questions

- Construct the hamming code, if 4 bit data 1001 is transformed.
- A receiver receives the hamming code 1110101, What is the correct code for even parity is?
- Generate the hamming code for the word 11011. Assume that a single error occurs while storing the generated hamming code. Explain how this single error is detected.

Weekly Assignment

- Perform BCD Addition of 999 and 989.
- For 989 and 674, Find BCD Subtraction using 9's complement and 10's complement method.
- Explain hamming code. A receiver receives the hamming code 10101101, check whether the data received is correct if not, check the error and correct it.
- Construct the hamming code, if 4 bit data 1001 is transformed.
- Write a short note on binary codes.

Recap

- Hamming code is an error-detection method that can detect some errors, but it is only capable of single error correction.
- Hamming Codes are named after Richard Hamming, the American mathematician.
- It has fix format, there are reserved bit for the parity at the position $2^{0,}$ $2^{1,}$ $2^{2,}$ 2^3 and so on.
- The **minimum value of 'k'** for which the following relation is correct valid is nothing but the required number of parity bits.

$\mathbf{2^k \geq n+k+1}$ Where,

‘n’ is the number of bits in the binary code information

‘k’ is the number of parity bits

Therefore, the number of bits in the Hamming code is equal to n + k.

Boolean algebra & Logic Gates

Topic Objective	Mapping with CO
To understand basic rules of Boolean algebra.	CO1
To understand SOP and POS form of Boolean function	CO1

Boolean algebra & Logic Gates

Prerequisite:

- Knowledge of basic logic gates.

Boolean algebra & Logic Gates

- Boolean Algebra is used to analyze and simplify the digital (logic) circuits. It uses only the binary numbers i.e. 0 and 1. It is also called as **Binary Algebra** or **logical Algebra**. Boolean algebra was invented by **George Boole** in 1854.
- Binary logic consists of binary variables and a set of logical operations. The variables are designated by letters of the alphabet, such as A, B, C, x, y, z, etc, with each variable having two and only two distinct possible values: 1 and 0, There are three basic logical operations: AND, OR, and NOT.

Boolean algebra & Logic Gates

Following are the important rules used in Boolean algebra.

- Variable used can have only two values. Binary 1 for HIGH and Binary 0 for LOW.
- Complement of a variable is represented by an overbar (-).
- ORing of the variables is represented by a plus (+) sign between them. For example ORing of A, B, C is represented as A + B + C.
- Logical ANDing of the two or more variable is represented by writing a dot between them such as A.B.C. Sometime the dot may be omitted like ABC.

Boolean algebra & Logic Gates

There are six types of Boolean Laws.

1) Commutative law:

- Any binary operation which satisfies the following expression is referred to as commutative operation.

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

Commutative law states that changing the sequence of the variables does not have any effect on the output of a logic circuit.

2) Associative law:

This law states that the order in which the logic operations are performed is irrelevant as their effect is the same.

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

Boolean algebra & Logic Gates

- **Distributive law:**

  Distributive law states the following condition.

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

AND law

- These laws use the AND operation. Therefore they are called as **AND** laws.

(i) $A.0 = 0$ (ii) $A.1 = A$

(iii) $A.A = A$ (iv) $A.\overline{A} = 0$

OR law

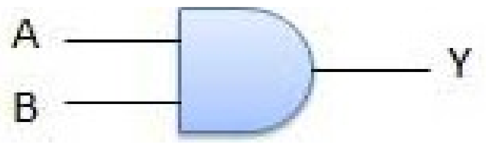
- These laws use the OR operation. Therefore they are called as **OR** laws

(i) $A + 0 = A$ (ii) $A + 1 = 1$

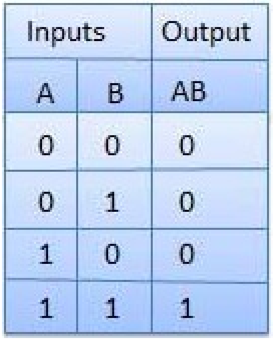
(iii) $A + A = A$ (iv) $A + \overline{A} = 1$

Boolean algebra & Logic Gates

- **AND Gate**
- A circuit which performs an AND operation is shown in figure. It has n input (n >= 2) and one output.
- **Logic diagram**



- **Truth Table**



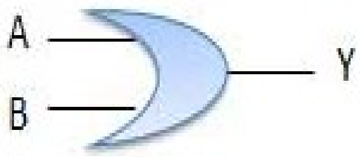
Inputs		Output
A	B	AB
0	0	0
0	1	0
1	0	0
1	1	1

Boolean algebra & Logic Gates

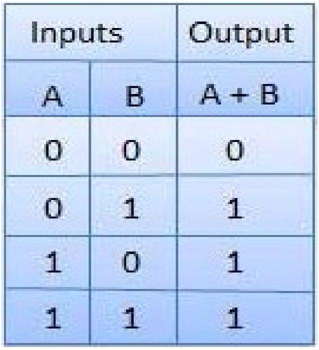
OR Gate

- A circuit which performs an OR operation is shown in figure. It has n input (n >= 2) and one output.

Logic diagram



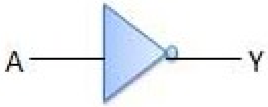
Truth Table



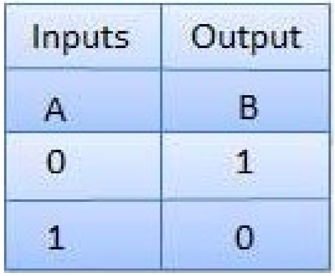
Inputs		Output
A	B	A + B
0	0	0
0	1	1
1	0	1
1	1	1

Boolean algebra & Logic Gates

- **NOT Gate**
- NOT gate is also known as **Inverter**. It has one input A and one output Y.
- Symbol



- Truth Table



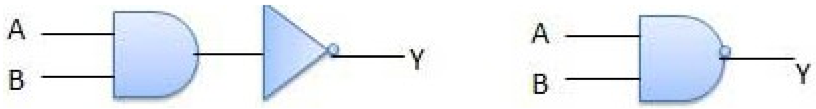
Inputs	Output
A	B
0	1
1	0

Boolean algebra & Logic Gates

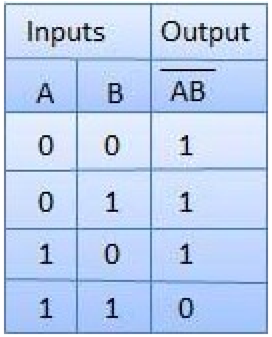
NAND Gate

A NOT-AND operation is known as NAND operation. It has n input (n >= 2) and one output.

- **Logic diagram**



- **Truth Table**

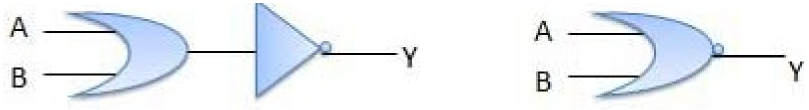


Inputs		Output
A	B	$\overline{AB}$
0	0	1
0	1	1
1	0	1
1	1	0

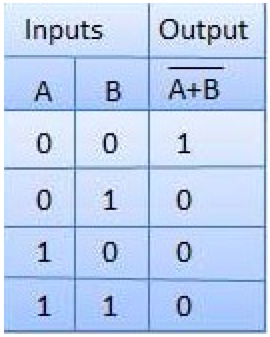
Boolean algebra & Logic Gates

- **NOR Gate**
- A NOT-OR operation is known as NOR operation. It has n input (n >= 2) and one output.

- Logic diagram



Truth Table

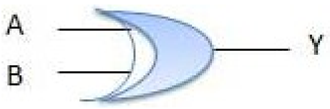


Inputs		Output
A	B	$\overline{A+B}$
0	0	1
0	1	0
1	0	0
1	1	0

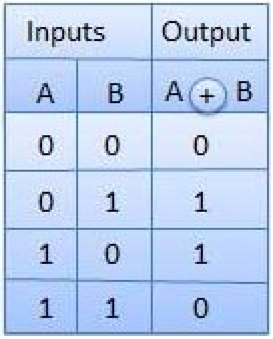
Boolean algebra & Logic Gates

- **XOR Gate:**
- XOR or Ex-OR gate is a special type of gate. 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**

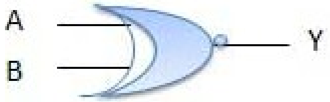


Inputs		Output
A	B	$A \oplus B$
0	0	0
0	1	1
1	0	1
1	1	0

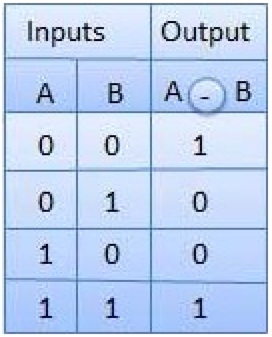
Boolean algebra & Logic Gates

- **XNOR Gate**
- XNOR gate is a special type of gate. It can be used in the half adder, full adder and subtractor. The exclusive-NOR gate is abbreviated as EX-NOR gate or sometime as X-NOR gate. It has n input (n >= 2) and one output.

- **Logic diagram**



- **Truth Table**



Inputs		Output
A	B	A ⊙ B
0	0	1
0	1	0
1	0	0
1	1	1

Daily Quiz

Q1. The NOR gate output will be high if the two inputs are ____

a) 00

b) 01

c) 10

d) 11

Q2. A universal logic gate is one which can be used to generate any logic function. Which of the following is a universal logic gate?

a) OR

b) AND

c) XOR

d) NAND

Q3. Which of following are known as universal gates?

a) NAND & NOR

b) AND & OR

c) XOR & OR

d) EX-NOR & XOR

FACULTY VIDEO LINKS, YOUTUBE & NPTEL VIDEO LINKS AND ONLINE COURSES DETAILS

Youtube/other Video Links:

- https://www.youtube.com/watch?v=VBM5XTvJSRQ&ab_channel=JAESCompanyJAESCompany
- https://www.youtube.com/watch?v=1A_NcXxdoCc&ab_channel=NesoAcademyNesoAcademyVerified

Old Questions

- What do you understand by universal gates?
- Realize a Ex-or gate with the help of NAND gate.
- Realize a OR gate using NAND gates.
- How an Ex-or gate can work as an inverter?
- What do understand by the term SOP and POS?

Recap

- Boolean Algebra is used to analyze and simplify the digital (logic) circuits.
- It uses only the binary numbers i.e. 0 and 1.
- It is also called as **Binary Algebra** or **logical Algebra**.
- Binary logic consists of binary variables and a set of logical operations.
- Here variable having two and only two distinct possible values: 1 and 0.
- There are three basic logical operations: AND, OR, and NOT.

KARNAUGHMAP (K-MAP) REPRESENTATION

Topic Objective	Mapping with CO
To understand rules of K-map simplification	CO1
To understand the logic minimization using K-map.	CO1

KARNAUGHMAP (K-MAP) REPRESENTATION

Prerequisite:

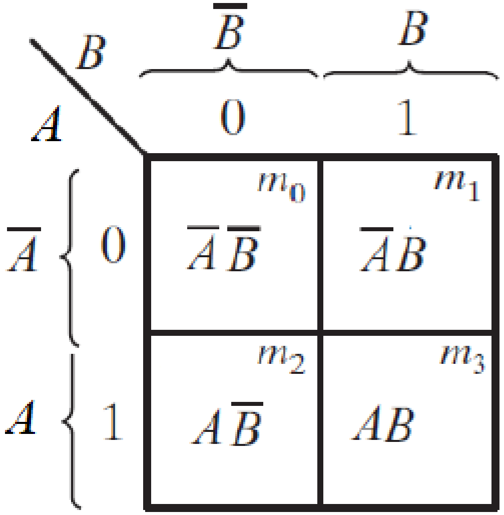
- Knowledge of Boolean algebra.
- Knowledge of SOP and POS forms.

KARNAUGHMAP (K-MAP) REPRESENTATION

- The complexity of the digital logic gates that implement a Boolean function is directly related to the complexity of the algebraic expression from which the function is implemented.
- The map method provides a simple straightforward procedure for minimizing Boolean functions.
- The map method, first proposed by Veitch and modified by Karnaugh, is also known as the "Veitch diagram" or the "Karnaugh map."
- The map is a diagram made up of squares. Each square represents one minterm.
- Since any Boolean function can be expressed as a sum of minterms, it follows that a Boolean function is recognized graphically in the map from the area enclosed by those squares whose min terms are included in the function.

KARNAUGHMAP (K-MAP) REPRESENTATION

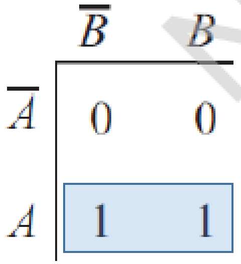
- **Two-Variable Karnaugh Map:**



$A \backslash B$	$\overline{B}$ 0	B 1
$\overline{A}$ 0	m_0 $\overline{A}\,\overline{B}$	m_1 $\overline{A}B$
A 1	m_2 $A\overline{B}$	m_3 AB

$Y = F(A,B) = \sum m(2,3)$

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

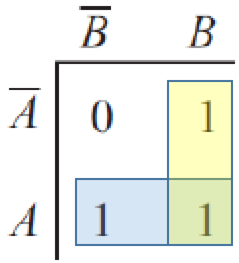


	$\overline{B}$	B
$\overline{A}$	0	0
A	1	1

$Y = A$

$Y = F(A,B) = \sum m(1,2,3)$

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



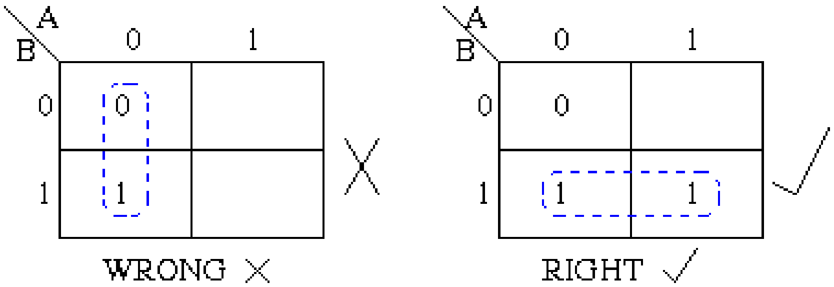
	$\overline{B}$	B
$\overline{A}$	0	1
A	1	1

$Y = A + B$

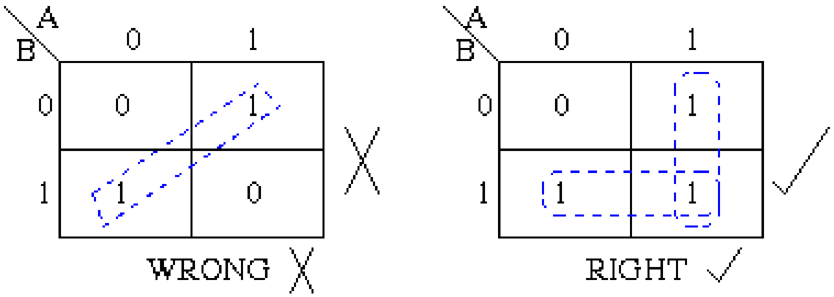
KMAP

KMAP Simplifictaion Rules:

- Groups may not include any cell containing a zero.

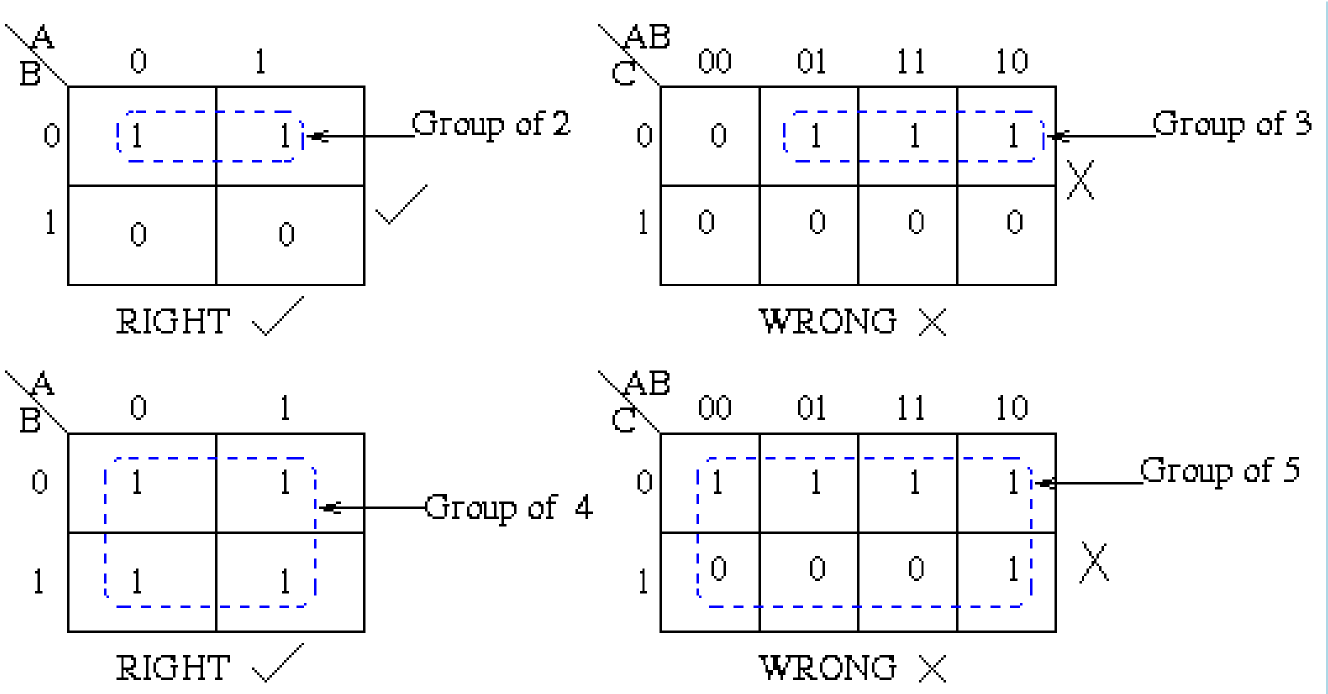


- Groups may be horizontal or vertical, but not diagonal.



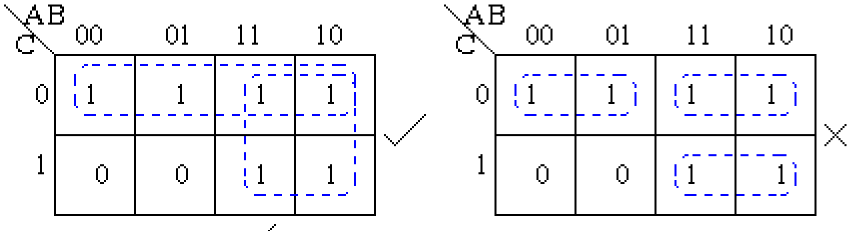
KMAP

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



KMAP

- Each group should be as large as possible.

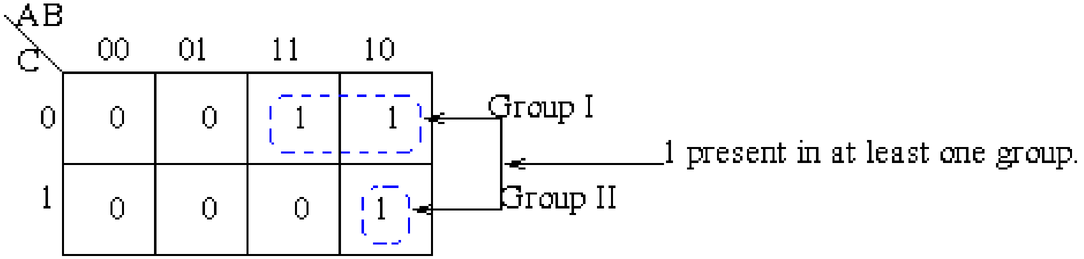


RIGHT ✓

WRONG ✕

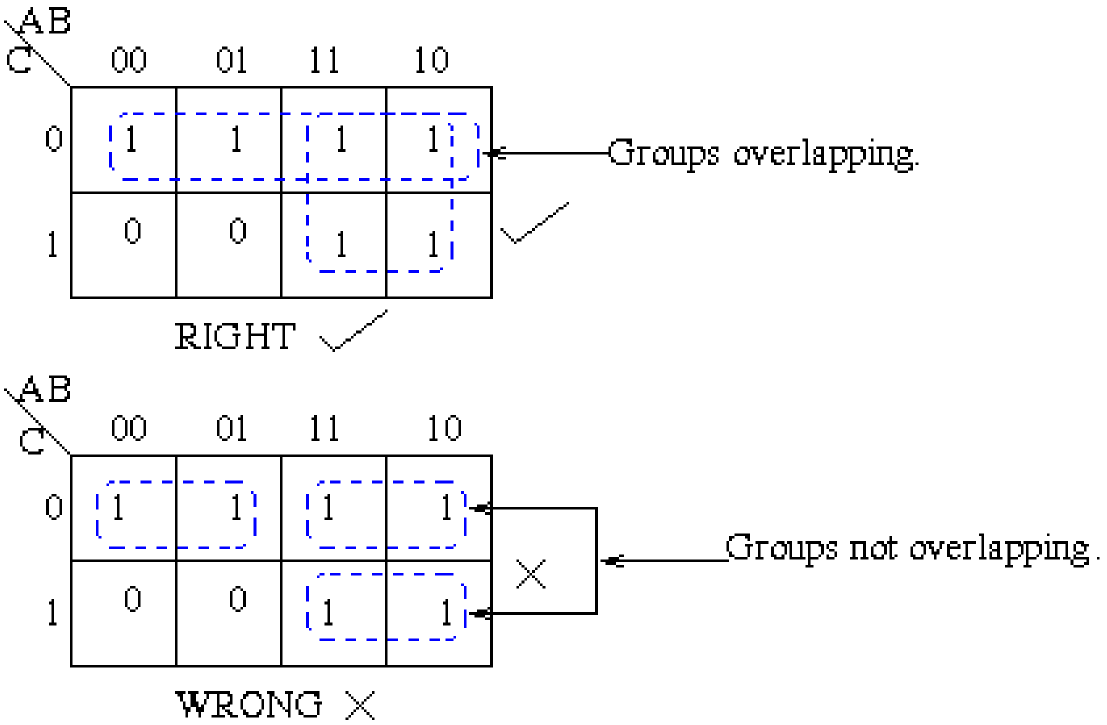
(Note that no Boolean laws broken, but not sufficiently minimal)

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



KMAP

- Groups may overlap.



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

Groups overlapping. ✓

RIGHT ✓

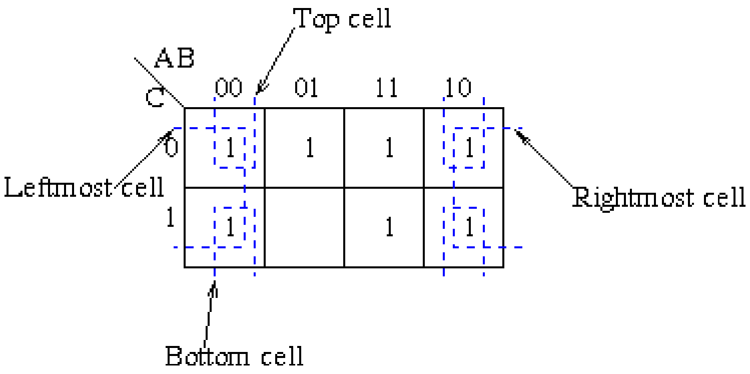
C \ AB	00	01	11	10
0	1	1	1	1
1	0	0	1	1

Groups not overlapping. ×

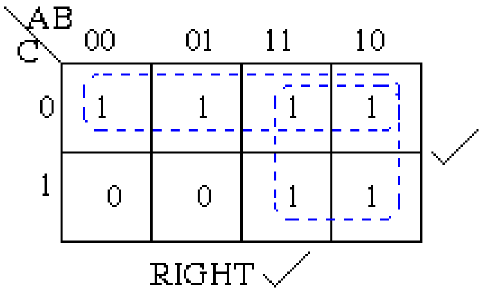
WRONG ×

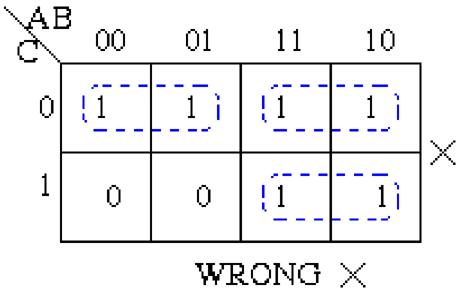
KMAP

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



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





KMAP

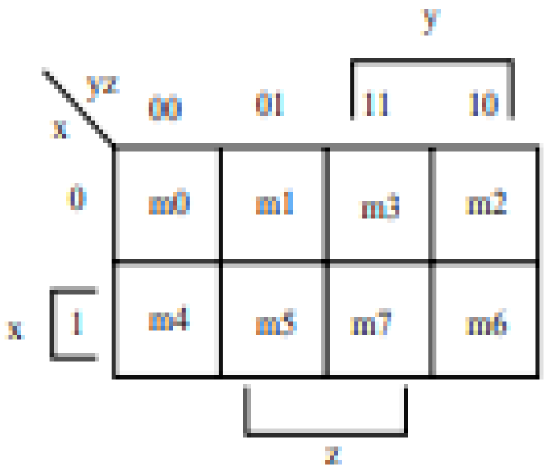
Summary:

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

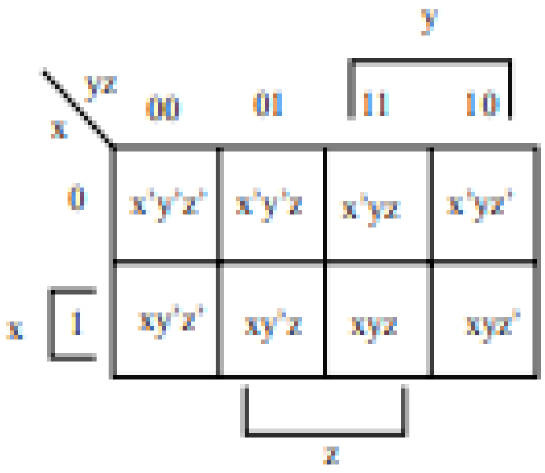
THREE VARIABLE KARNAUGH MAP

- **Three-Variable Karnaugh Map:**

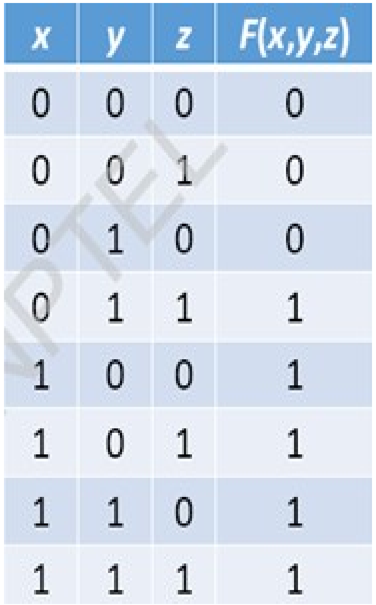
  3 variables, F = f(x, y, z)



x \ yz	00	01	11	10
0	m0	m1	m3	m2
1	m4	m5	m7	m6



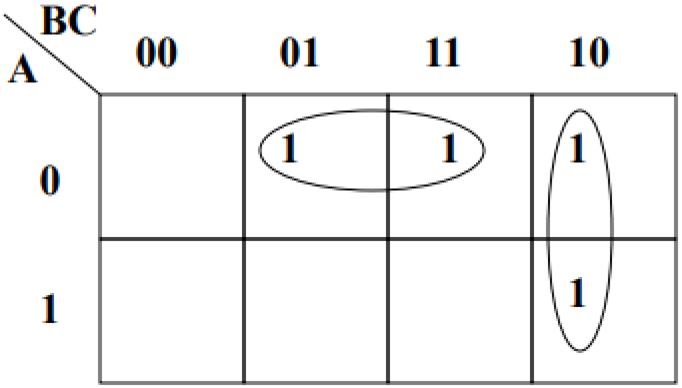
x \ yz	00	01	11	10
0	x'y'z'	x'y'z	x'yz	x'yz'
1	xy'z'	xy'z	xyz	xyz'



x	y	z	F(x,y,z)
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

THREE VARIABLE KARNAUGH MAP

Eg: F(A,B,C)= ∑(1,2,3,6) minimize the given using K-MAP.

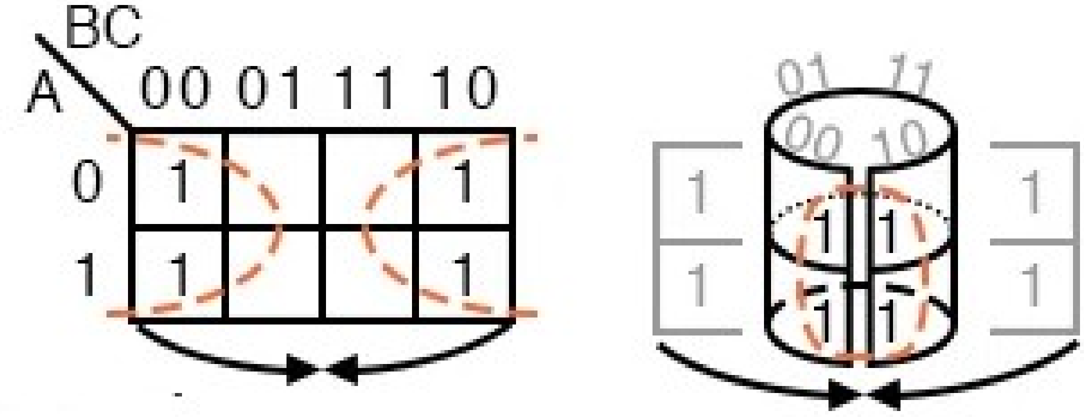


BC A	00	01	11	10
0		1	1	1
1				1

Ans: F = A'C + BC’

THREE VARIABLE KARNAUGH MAP

Eg: F= A'B'C ' + AB'C ' + A'BC ' + ABC ' minimize the given using K-MAP.

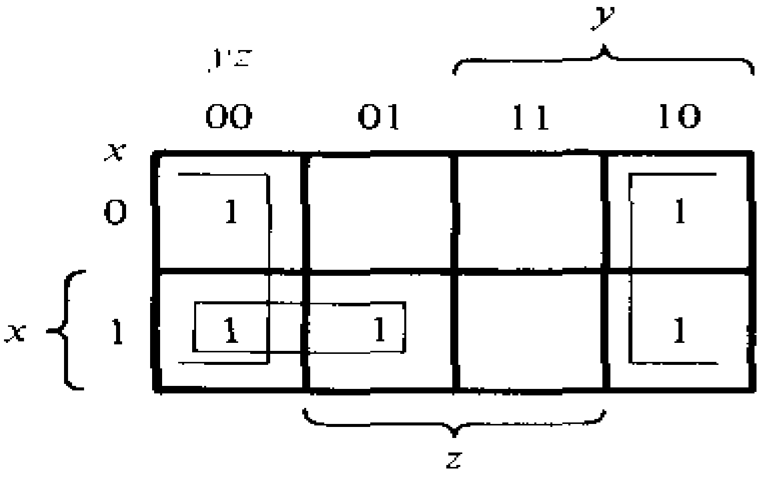


A \ BC	00	01	11	10
0	1			1
1	1			1

Ans: F = C '

THREE VARIABLE KARNAUGH MAP

Eg: F(x, y, z)= ∑(0,2,4,5,6) minimize the given using K-MAP.

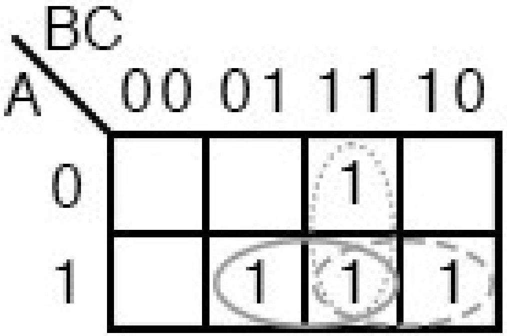


x \ yz	00	01	11	10
0	1			1
1	1	1		1

Ans: F = z' + x.y'

THREE VARIABLE KARNAUGH MAP

Eg: F= ∑A,B,C(3, 5,6,7) minimize the given using K-MAP.

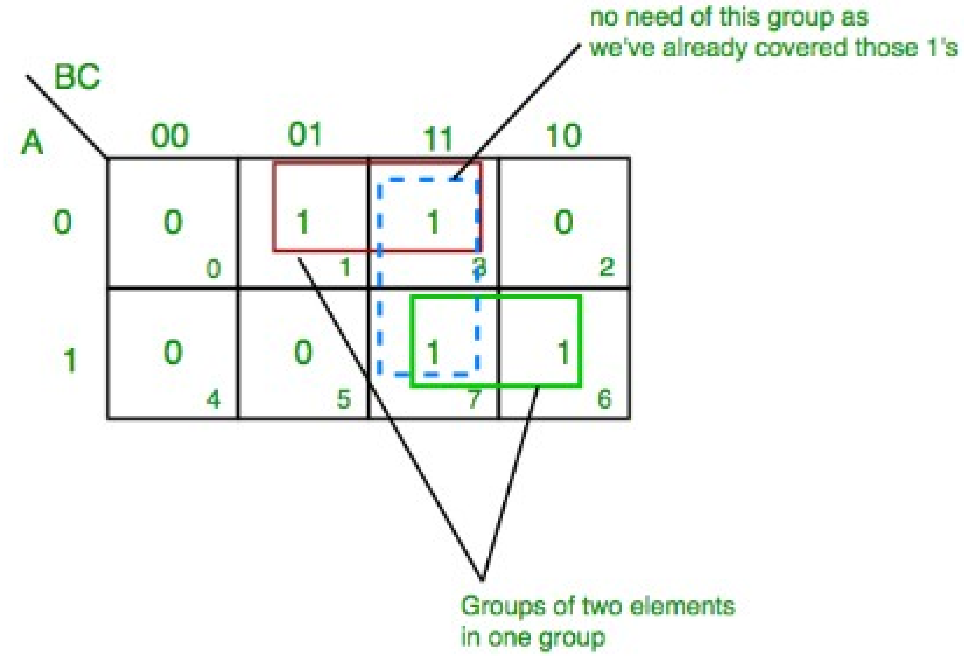


A \ BC	00	01	11	10
0			1	
1		1	1	1

Ans: F = BC + AC + AB

THREE VARIABLE KARNAUGH MAP

Eg: F(A,B,C)= ∑(1,3,6,7) minimize the given using K-MAP.



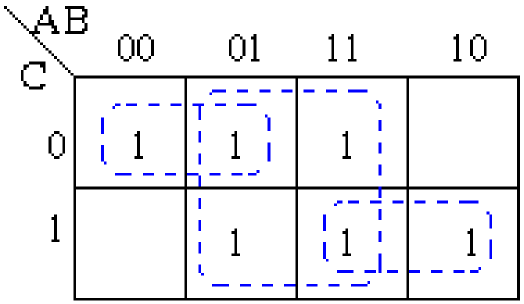
no need of this group as we've already covered those 1's

Groups of two elements in one group

Ans: F = A’C + AB

THREE VARIABLE KARNAUGH MAP

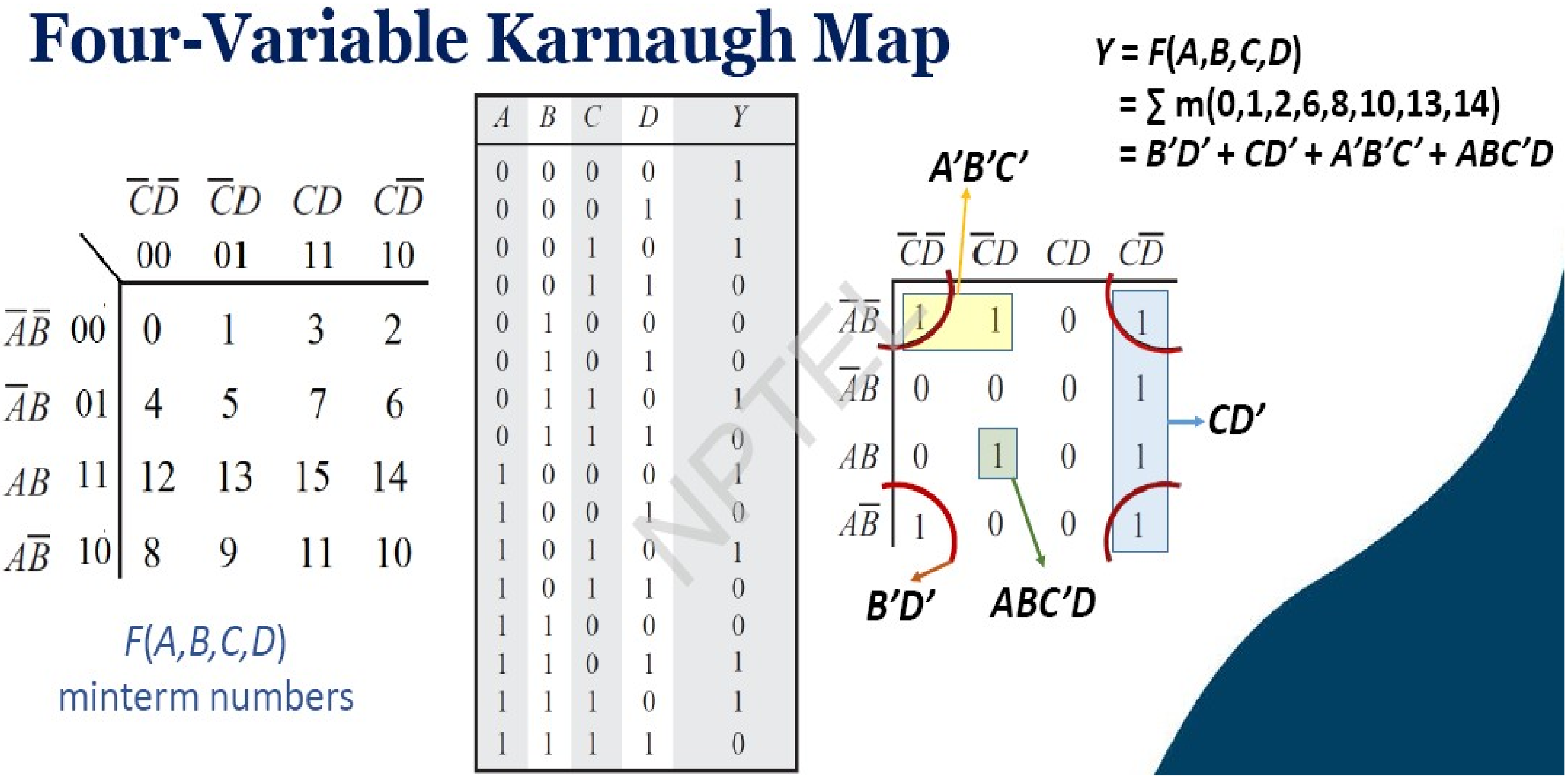
Eg: F(A,B,C) = A'B'C ' + A ' B + ABC ' + AC minimize the given using K-MAP.



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

Ans: F = B + AC + A'C '

FOUR-VARIABLE KARNAUGH MAP



Four-Variable Karnaugh Map

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

F(*A*,*B*,*C*,*D*)
minterm numbers

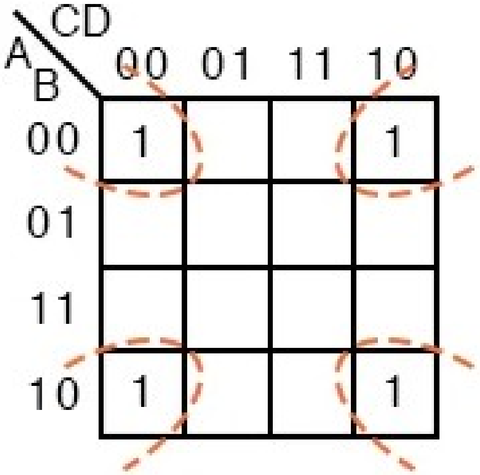
A	*B*	*C*	*D*	*Y*
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

$$Y = F(A,B,C,D)$$
$$= \sum m(0,1,2,6,8,10,13,14)$$
$$= B'D' + CD' + A'B'C' + ABC'D$$

	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	1	1	0	1
$\overline{A}B$	0	0	0	1
AB	0	1	0	1
$A\overline{B}$	1	0	0	1

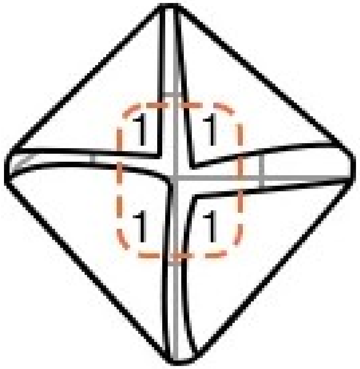
FOUR-VARIABLE KARNAUGH MAP

Eg: F = $\sum m(0, 2, 8, 10)$ minimize the given using K-MAP in SOP form.



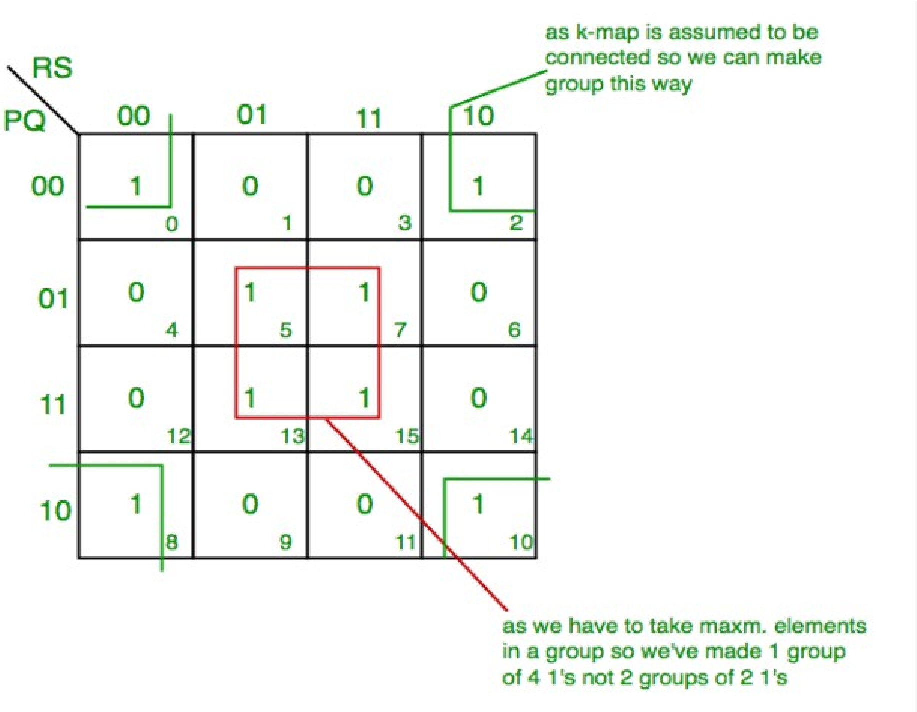
AB \ CD	00	01	11	10
00	1			1
01				
11				
10	1			1

Out = $\overline{B}\,\overline{D}$



FOUR-VARIABLE KARNAUGH MAP

- F(P,Q,R,S)=∑(0,2,5,7,8,10,13,15) minimize the given using K-MAP.



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

as k-map is assumed to be connected so we can make group this way

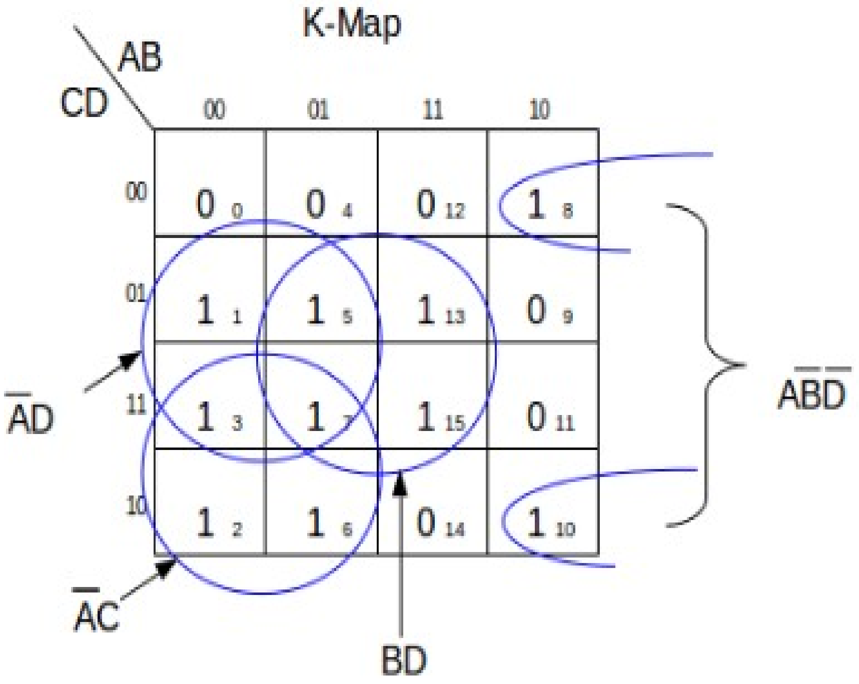
as we have to take maxm. elements in a group so we've made 1 group of 4 1's not 2 groups of 2 1's

F = (QS+Q’S’)

FOUR-VARIABLE KARNAUGH MAP

- F(A,B,C,D)=∑(1,2,3,5,6,7,8,10,13,15) minimize the given using K-MAP.

cd \ ab	00	01	11	10
00	0	4	12	8
01	1	4	13	9
11	3	6	15	11
10	2	7	14	10



K-Map

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

$F = \bar{A}D + \bar{A}C + BD + A\bar{B}\bar{D}$

DON’T CARE IN KARNAUGH MAP

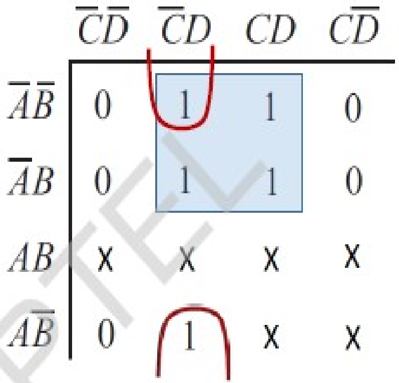
Design: *Y* is H when BCD (Binary Coded Decimal) input is odd.

A
B
C
D
Digital Circuit
Y

$Y = F(A,B,C,D)$
$= \sum m(1,3,5,7,9) +$
$d(10,11,12,13,14,15)$

A	*B*	*C*	*D*	*Y*
0	0	0	0	0
0	0	0	1	1
0	0	1	0	0
0	0	1	1	1
0	1	0	0	0
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	0
1	0	0	1	1
1	0	1	0	X
1	0	1	1	X
1	1	0	0	X
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

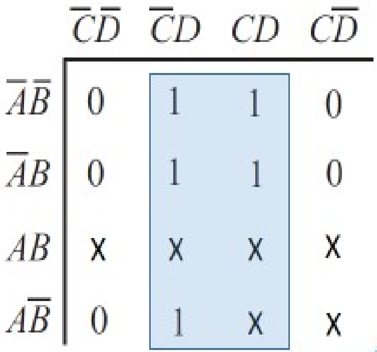
Not considering X



	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	0	1	1	0
$\overline{A}B$	0	1	1	0
AB	X	X	X	X
$A\overline{B}$	0	1	X	X

$Y = A'D + B'C'D$

Considering X

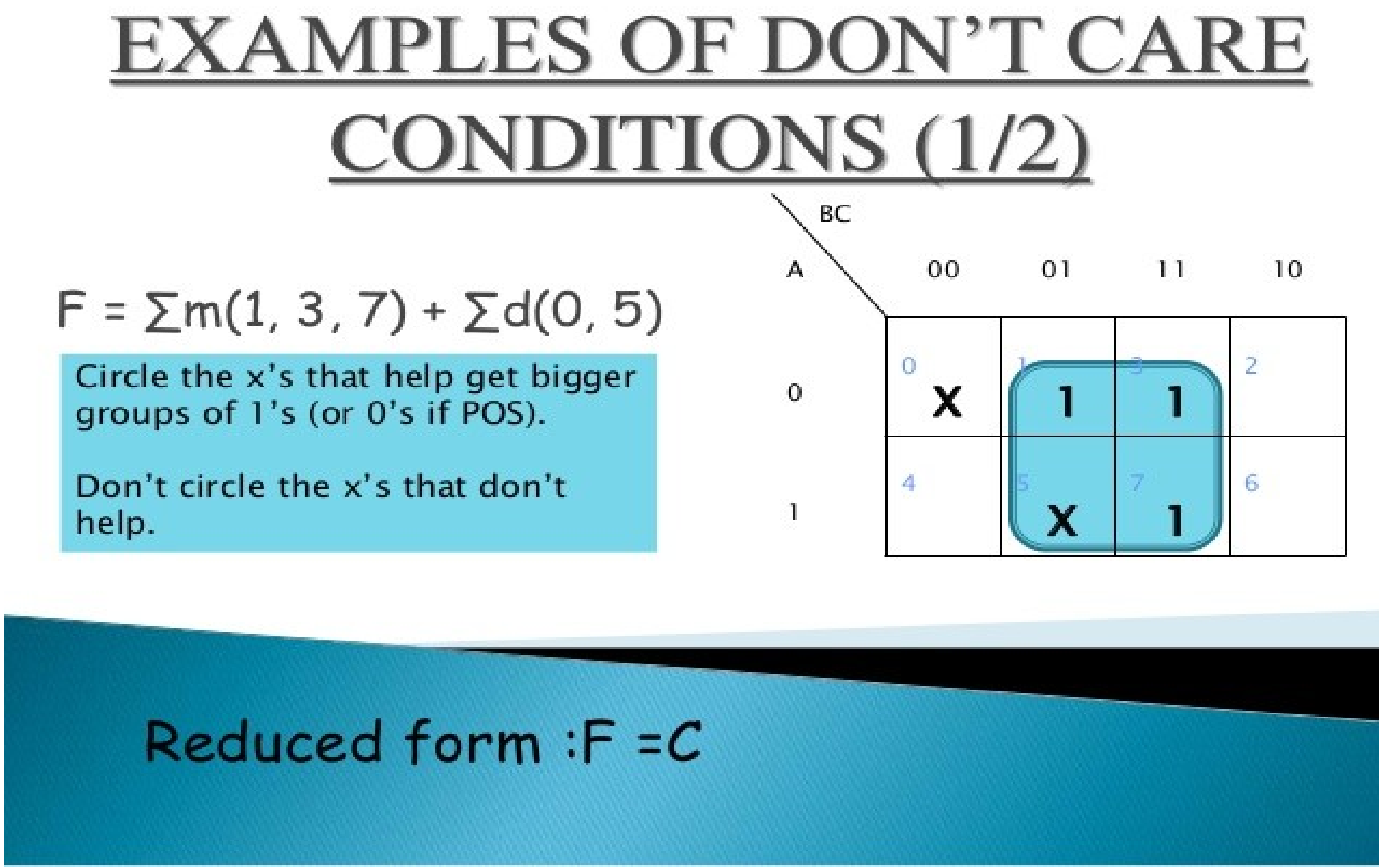


	$\overline{C}\overline{D}$	$\overline{C}D$	CD	$C\overline{D}$
$\overline{A}\overline{B}$	0	1	1	0
$\overline{A}B$	0	1	1	0
AB	X	X	X	X
$A\overline{B}$	0	1	X	X

$Y = D$

- If not considered, X = 0.
- If considered, X = 1.
- Consideration wherever helps.

DON’T CARE IN KARNAUGH MAP



EXAMPLES OF DON’T CARE CONDITIONS (1/2)

$F = \sum m(1, 3, 7) + \sum d(0, 5)$

Circle the x’s that help get bigger groups of 1’s (or 0’s if POS).

Don’t circle the x’s that don’t help.

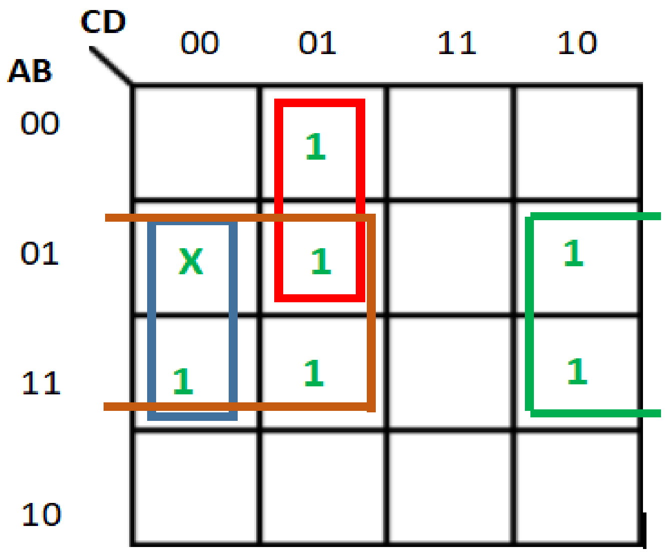
A \ BC	00	01	11	10
0	0 X	1 1	3 1	2
1	4	5 X	7 1	6

Reduced form :F =C

DON’T CARE IN KARNAUGH MAP

1. Minimise the following function in SOP minimal form using K-Maps: f(A,B,C,D) = m(1, 5, 6, 12, 13, 14) + d(4)

- From **green & blue** group we find terms : BD'
- From **red** group we find terms : A'C'D
- From **brown** group we find terms : BC'
- Therefore, SOP minimal is, f = BC' + BD' + A'C'D

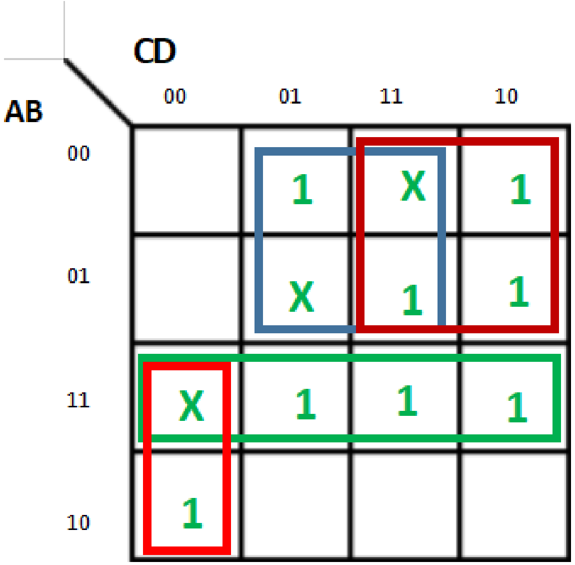


AB \ CD	00	01	11	10
00		1		
01	X	1		1
11	1	1		1
10				

DON’T CARE IN KARNAUGH MAP

2. Minimise the following function in SOP minimal form using K-Maps: F(A, B, C, D) = m(1, 2, 6, 7, 8, 13, 14, 15) + d(3, 5, 12)

- From **green** group we find terms : AB
- From **red** group we find terms : AC'D'
- From **brown** group we find terms : A'D
- From **blue** group we find terms : A'C
- f = AC'D' + A'D + A'C + AB

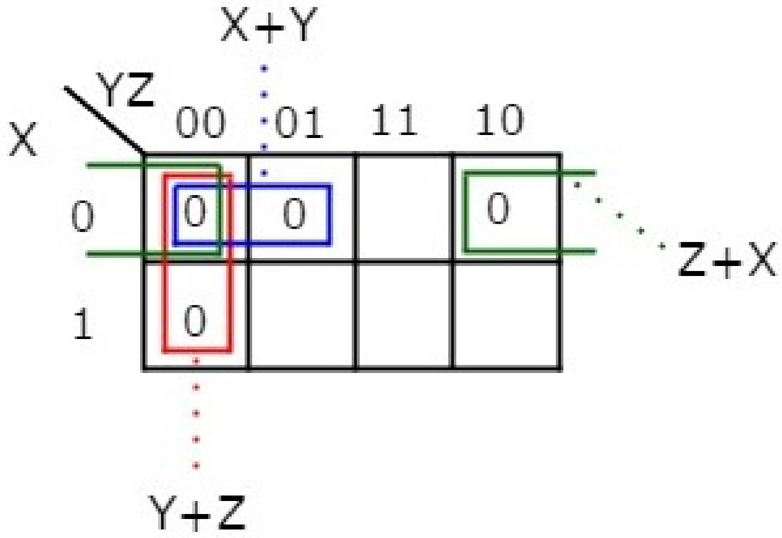


CD

AB	00	01	11	10
00		1	X	1
01		X	1	1
11	X	1	1	1
10	1			

KARNAUGH MAP: POS

Eg: F(X,Y,Z)=∏M(0,1,2,4) minimize the given using K-MAP in POS form.



Ans: F’ = X’Z’ + Y’Z’ + X’Y’

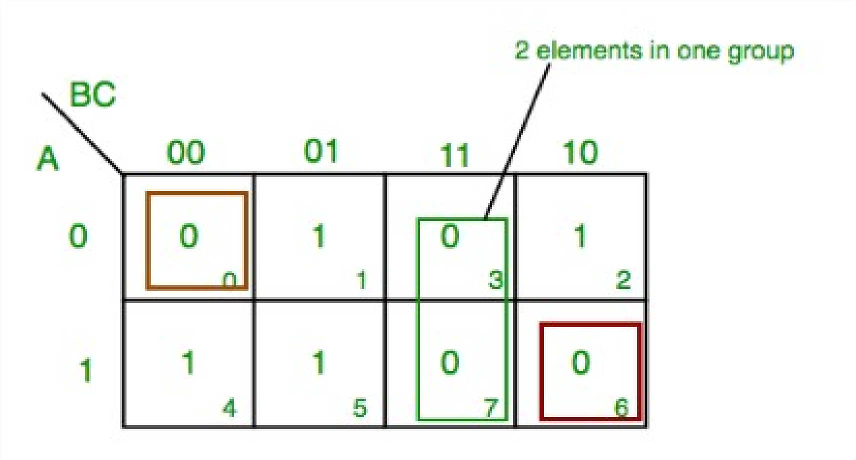
(F’)’ = F= (X’Z’ + Y’Z’ + X’Y’)’

F = (X + Z) . (Y + Z) . (X + Y)

KARNAUGH MAP: POS

1. **K-map of 3 variables-**

F(A,B,C)=π(0,3,6,7)



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

- F = B.C + A’B’C’ + AB
  (F’)’ = F = (B.C + A’B’C’ + ABC’)’ = (B' + C') (A + B + C) (A'+ B')
- From **red** group we find terms : **(B'+ C')**
- From **brown** group we find terms : **(A' + B' + C) & (A + B)**
- **Final expression (A'+ B') (B' + C') (A + B + C)**

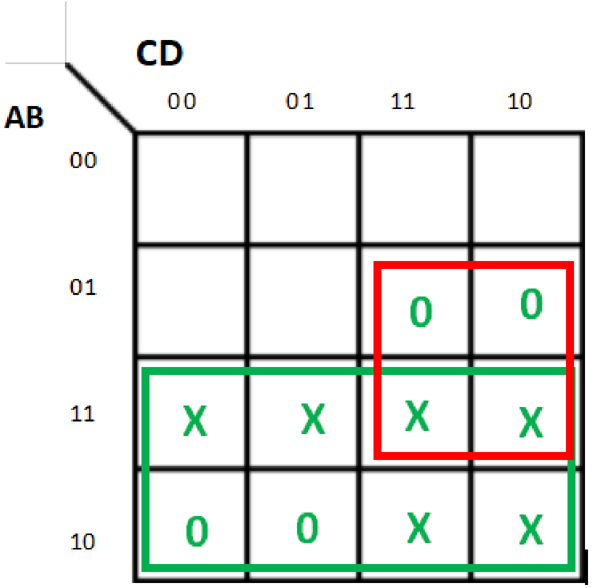
KARNAUGH MAP: POS

Eg: Minimise the following function in POS minimal form using K-Maps: F(A, B, C, D) = M(6, 7, 8, 9) + d(10, 11, 12, 13, 14, 15)

- For **red** group we find terms: (B' + C')
- For **green** group we find terms: A'
- POS minimal is, F’ = A + BC

  (F’)’= (A +BC)’

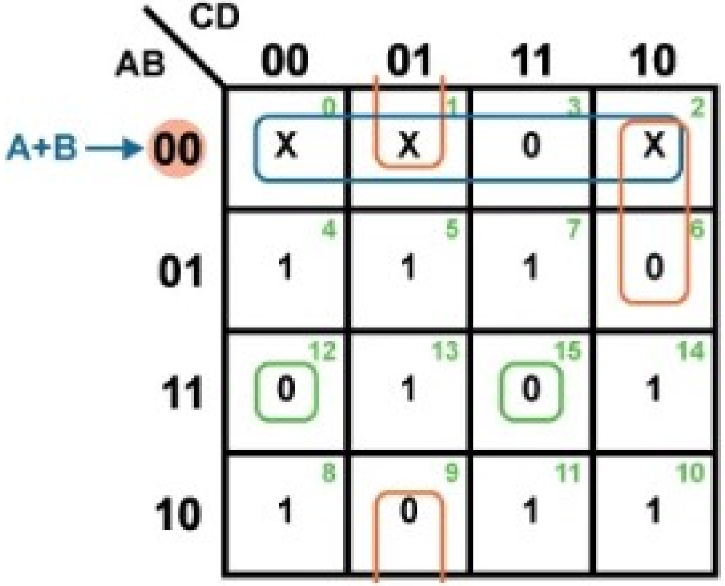
  = A'(B' + C')



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

DON’T CARE IN KARNAUGH MAP

Eg: F(A,B,C,D) = ∑*m*(4,5,7,8,10,11,13,14) + ∑*d* (0,1,2) minimize the given using K-MAP in POS form.



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

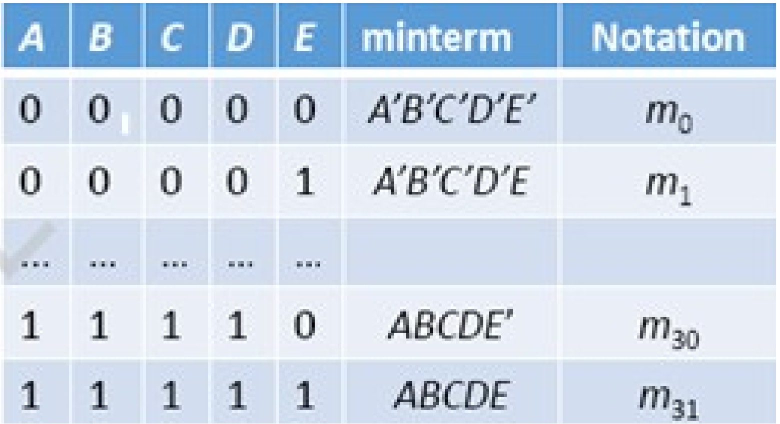
Ans: F’ = ABCD + ABC’D’ + A’CD’ + B’C’D + A’B’

$F = (A+B)\,(B+C+\overline{D})\,(A+\overline{C}+D)\,(\overline{A}+\overline{B}+C+D)\,(\overline{A}+\overline{B}+\overline{C}+D)$

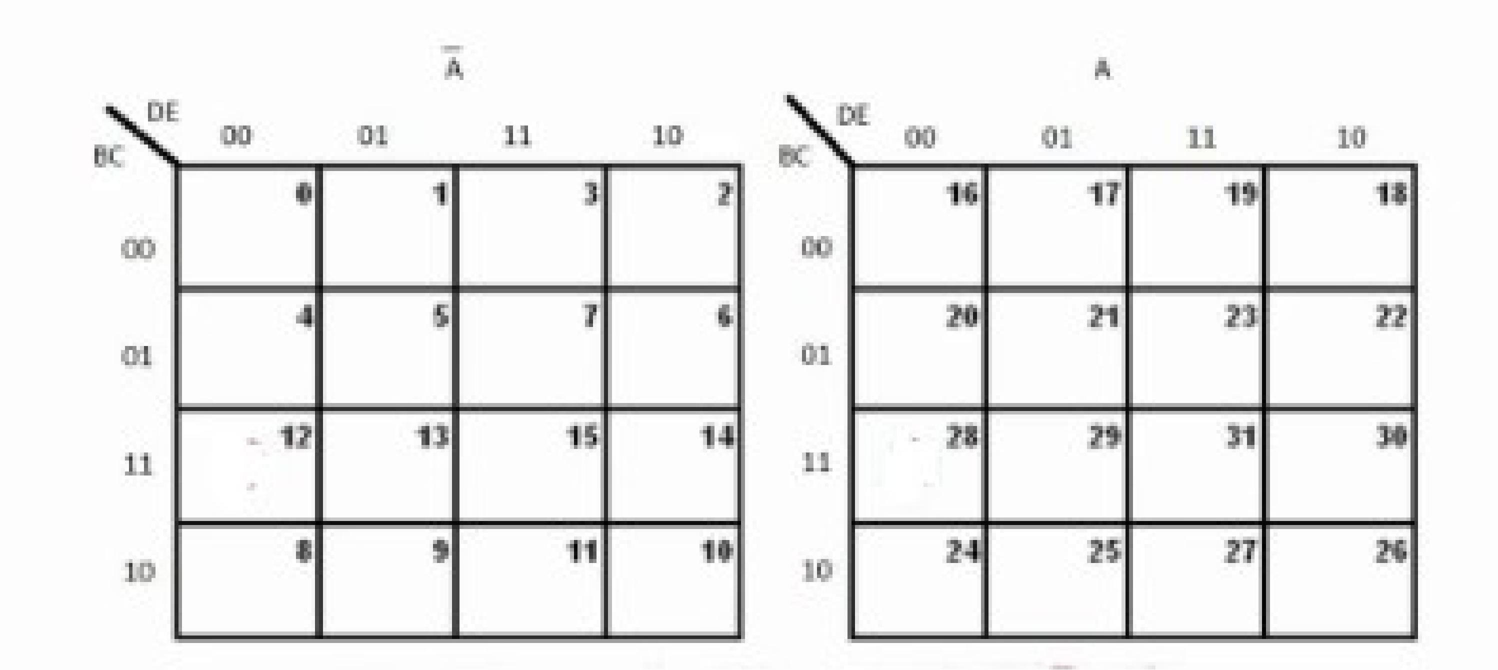
FIVE-VARIABLE KARNAUGH MAP

- Maps for more than four variables are not as simple to use. A five-variable map needs 32 squares and a six-variable map needs 64 squares.
- When the number of variables becomes large, the number of squares becomes excessively large and the geometry for combining adjacent squares becomes more involved.
- The five-variable map is shown in Figure. It consists of 2 four-variable maps with variables A, B, C, D, and E.
- Variable A distinguishes between the two maps, as indicated on the top of the diagram.
- The left-hand four-variable map represents the 16 squares where A = 0, and the other four-variable map represents the squares where A = 1. Minterms 0 through 15 belong with A = 0 and minterms 16 through 31 with A = 1.

FIVE-VARIABLE KARNAUGH MAP



A	*B*	*C*	*D*	*E*	minterm	Notation
0	0	0	0	0	*A'B'C'D'E'*	m_0
0	0	0	0	1	*A'B'C'D'E*	m_1
…	…	…	…	…		
1	1	1	1	0	*ABCDE'*	m_{30}
1	1	1	1	1	*ABCDE*	m_{31}



$\bar{A}$

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

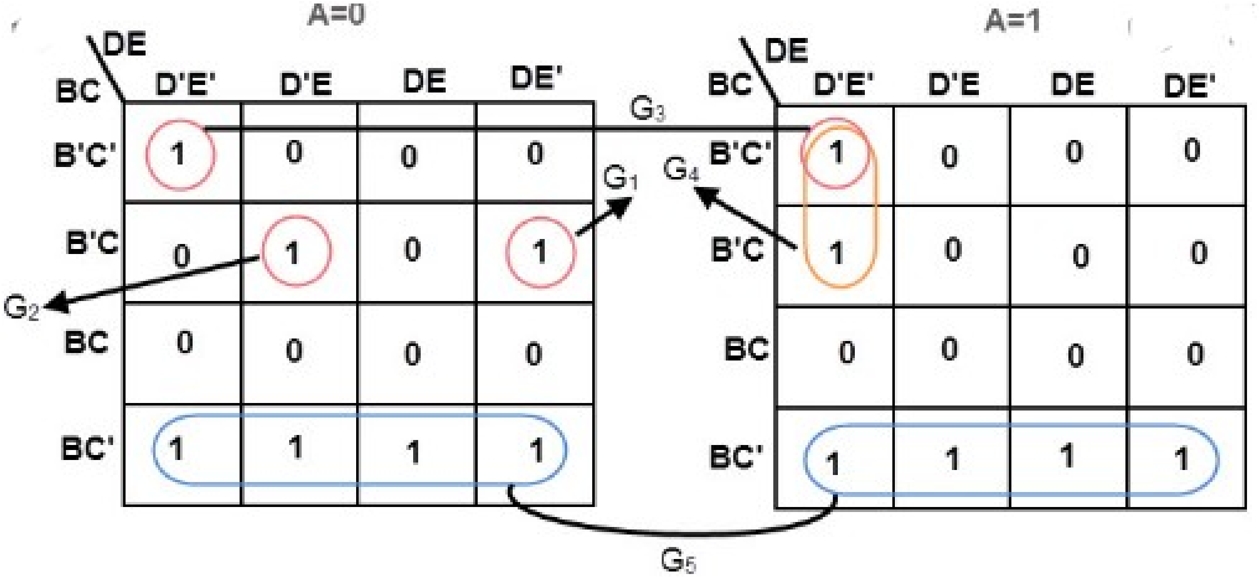
A

BC \ DE	00	01	11	10
00	16	17	19	18
01	20	21	23	22
11	28	29	31	30
10	24	25	27	26

Kmap for 5 variables

FIVE-VARIABLE KARNAUGH MAP

Eg: F(A,B,C,D,E) = ∑*m*(0, 5, 6, 8, 9, 10, 11, 16, 20, 24, 25, 26, 27) minimize the given using K-MAP in SOP form.



A=0

BC \ DE	D'E'	D'E	DE	DE'
B'C'	1	0	0	0
B'C	0	1	0	1
BC	0	0	0	0
BC'	1	1	1	1

A=1

BC \ DE	D'E'	D'E	DE	DE'
B'C'	1	0	0	0
B'C	1	0	0	0
BC	0	0	0	0
BC'	1	1	1	1

Sol: F = A'B'CDE' + A'B'CD'E + B'C'D'E' + AB'D'E' + BC'

Daily Quiz

- Complement of the function F = x'y'z + xy'z' is:

a) (x + y + z'). (x' + y + z)

b) (x' + y' + z). (x + y' +z')

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

- Minimized function for F(A, B, C) = Σ (2,3,6) + d(0,7) using k-map is: (a) A'C' + BC + BC‘ (b) B (c) AB
- What are the selective prime implicants for F(w,x,y,z) = Σ (1,3,6,7,14)

a) w'x'z , xyz‘

b) w'x'z , xyz‘

c) w'x'z, w'xy

FACULTY VIDEO LINKS, YOUTUBE & NPTEL VIDEO LINKS AND ONLINE COURSES DETAILS

Youtube/other Video Links:

- https://www.youtube.com/watch?v=FPrcIhqNPVo&ab_channel=NesoAcademyNesoAcademyVerified
- https://www.youtube.com/watch?v=wjM2RDG5yTI&ab_channel=NesoAcademyNesoAcademyVerified
- https://www.youtube.com/watch?v=BPBiyzc0OBw&ab_channel=IITKharagpurJuly2018IITKharagpurJuly2018

Old Questions

- F(A,B,C) = $\sum m(1,4,5,7)$ minimize the given using K-MAP in SOP form.
- F(A,B,C,D) = $\sum m(0,2,4,6,8,10,12,14)$ minimize the given using K-MAP in SOP form.
- F(w,x,y,z) = $\sum m(4,5,7,8,10,14)$ minimize the given using K-MAP in POS form.
- F(A,B,C,D,E) = $\sum m(4,5,7,8,10,14,27,31) + \sum d\,(0,1,2,11,19,25)$ minimize the given using K-MAP in SOP form.
- F(w,x,y,z) = $\sum m(0,2,4,5,8,12,14) + \sum d\,(1,3,5,7,9)$ minimize the given using K-MAP in POS form.

Recap

- The map method provides a simple straightforward procedure for minimizing Boolean functions.
- The map is a diagram made up of squares. Each square represents one minterm.
- K map rule summary:

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

QM ALGORITHM

Topic Objective	Mapping with CO
To understand the logic minimization using QM algorithm.	CO1

QM ALGORITHM

Prerequisite:

- Knowledge of Boolean algebra.
- Knowledge of SOP and POS forms.

QM ALGORITHM

- The map method of simplification is convenient as long as the number of variables does not exceed five or six. As the number of variables increases, the excessive number of squares prevents a reasonable selection of adjacent squares.

- For functions of six or more variables, it is difficult to be sure that the best selection has been made

- The tabulation method overcomes this difficulty. It is a specific step-by-step procedure that is guaranteed to produce a simplified standard-form expression for a function.

- The tabulation method was first formulated by Quine and later improved by McCluskey. It is also known as the Quine-McCluskey method.

QM ALGORITHM

Procedure of Quine-McCluskey Tabular Method

- Follow these steps for simplifying Boolean functions using Quine-McCluskey tabular method.
- **Step 1** − Arrange the given min terms in an **ascending order** and make the groups based on the number of ones present in their binary representations. So, there will be **at most ‘n+1’ groups** if there are ‘n’ Boolean variables in a Boolean function or ‘n’ bits in the binary equivalent of min terms.
- **Step 2** − Compare the min terms present in **successive groups**. If there is a change in only one-bit position, then take the pair of those two min terms. Place this symbol ‘_’ in the differed bit position and keep the remaining bits as it is.
- **Step 3** − Repeat step2 with newly formed terms till we get all **prime implicants**.

QM ALGORITHM

- **Step 4** − Formulate the **prime implicant table**. It consists of set of rows and columns. Prime implicants can be placed in row wise and min terms can be placed in column wise. Place ‘1’ in the cells corresponding to the min terms that are covered in each prime implicant.

- **Step 5** − Find the essential prime implicants by observing each column. If the min term is covered only by one prime implicant, then it is **essential prime implicant**. Those essential prime implicants will be part of the simplified Boolean function.

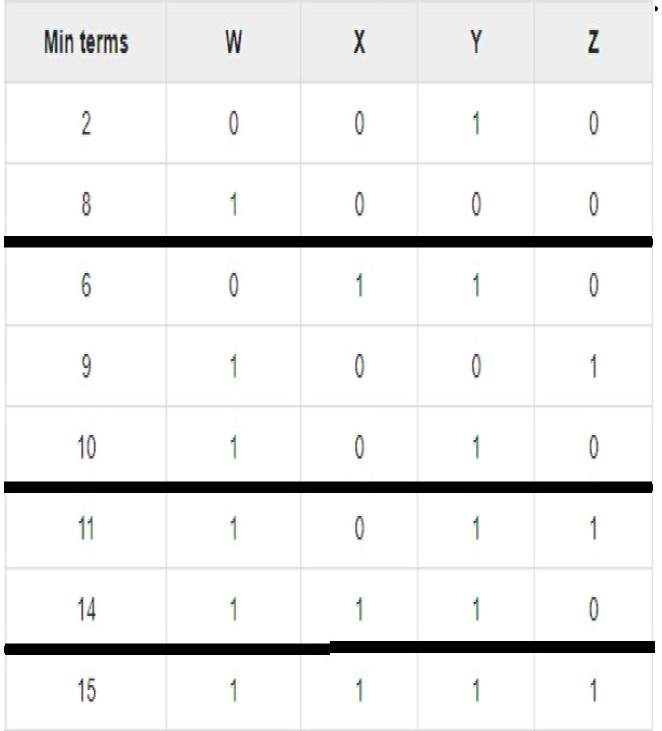
- **Step 6** − Reduce the prime implicant table by removing the row of each essential prime implicant and the columns corresponding to the min terms that are covered in that essential prime implicant. Repeat step 5 for Reduced prime implicant table. Stop this process when all min terms of given Boolean function are over.

- .

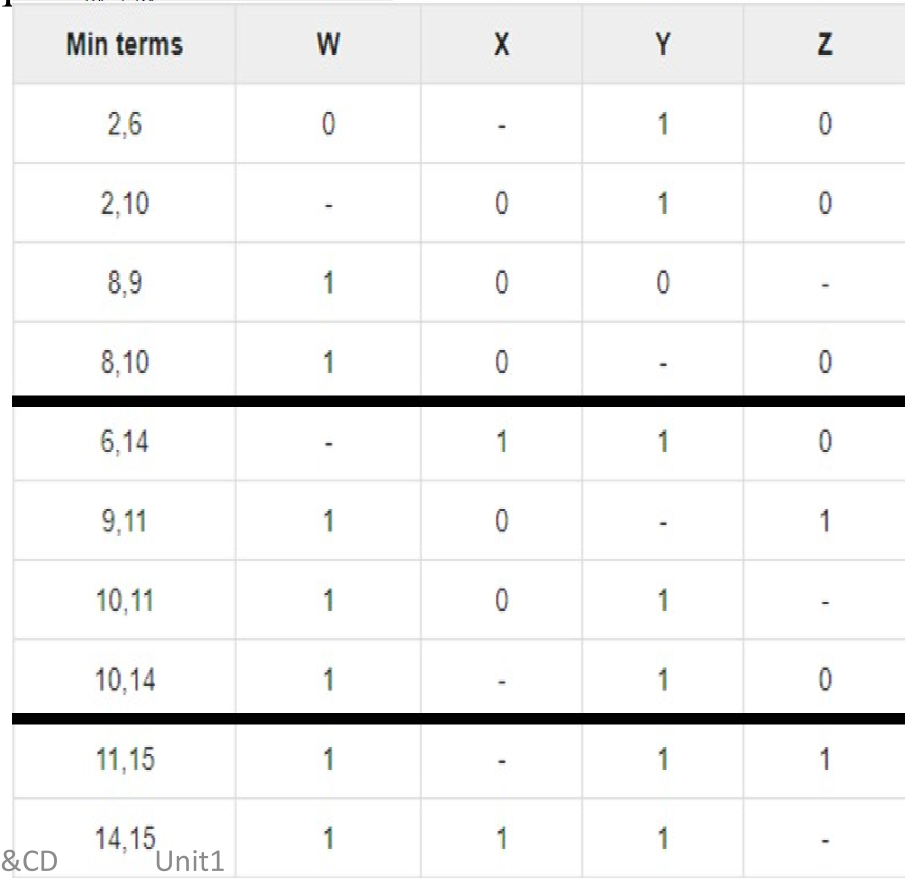
QM ALGORITHM

E.g Minmize the F(*W*,*X*,*Y*,*Z*)=∑*m*(2,6,8,9,10,11,14,15) using Quine-McCluskey method

- Step1 : The given min terms are 2, 6, 8, 9, 10, 11, 14 and 15. The ascending order of these min terms based on the number of ones present in their binary equivalent is 2, 8, 6, 9, 10, 11, 14 and 15.



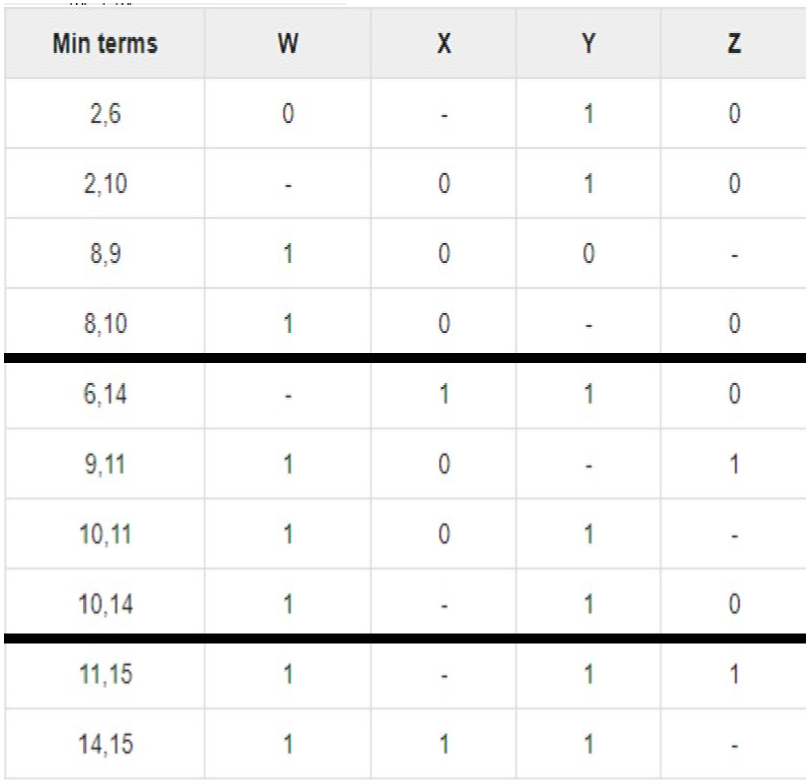
Min terms	W	X	Y	Z
2	0	0	1	0
8	1	0	0	0
6	0	1	1	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
14	1	1	1	0
15	1	1	1	1



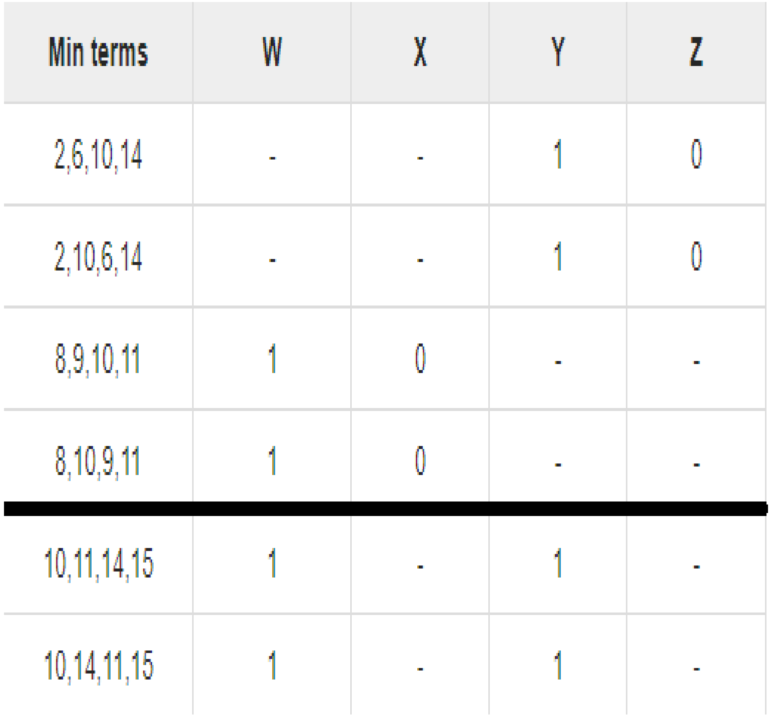
Min terms	W	X	Y	Z
2,6	0	-	1	0
2,10	-	0	1	0
8,9	1	0	0	-
8,10	1	0	-	0
6,14	-	1	1	0
9,11	1	0	-	1
10,11	1	0	1	-
10,14	1	-	1	0
11,15	1	-	1	1
14,15	1	1	1	-

QM ALGORITHM

Step 2 : The given min terms are arranged into 4 groups based on the number of ones present in their binary equivalents. The following table shows the possible **merging of min terms** from adjacent groups.



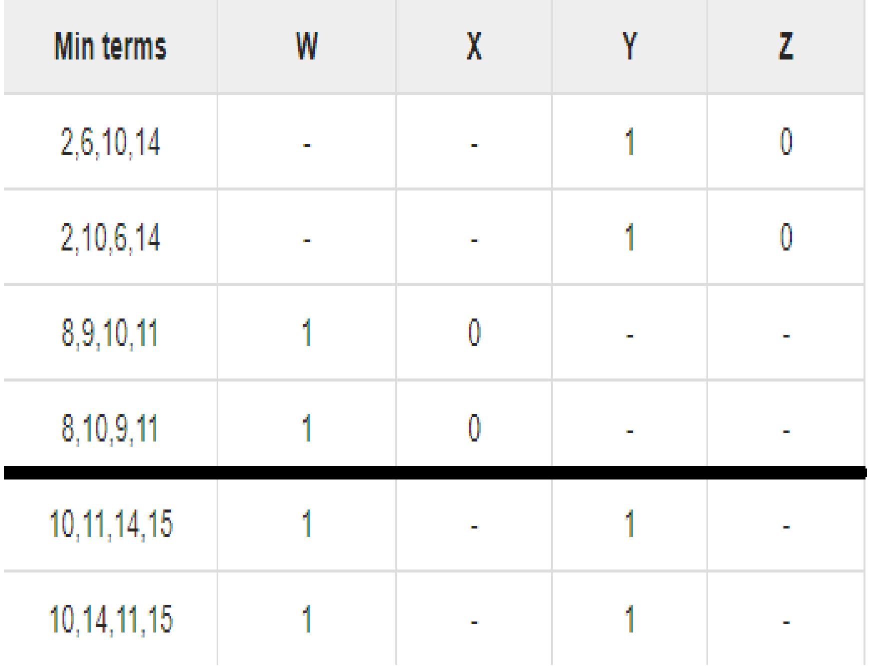
Min terms	W	X	Y	Z
2,6	0	-	1	0
2,10	-	0	1	0
8,9	1	0	0	-
8,10	1	0	-	0
6,14	-	1	1	0
9,11	1	0	-	1
10,11	1	0	1	-
10,14	1	-	1	0
11,15	1	-	1	1
14,15	1	1	1	-



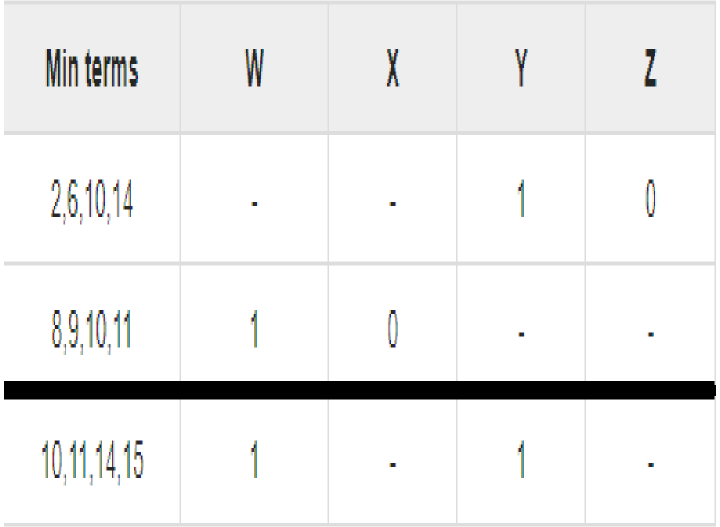
Min terms	W	X	Y	Z
2,6,10,14	-	-	1	0
2,10,6,14	-	-	1	0
8,9,10,11	1	0	-	-
8,10,9,11	1	0	-	-
10,11,14,15	1	-	1	-
10,14,11,15	1	-	1	-

QM ALGORITHM

Step 3 : The min terms, which are differed in only one-bit position from adjacent groups are merged. That differed bit is represented with this symbol, ‘-‘. In this case, there are three groups and each group contains combinations of two min terms. The following table shows the possible **merging of min term pairs** from adjacent groups.



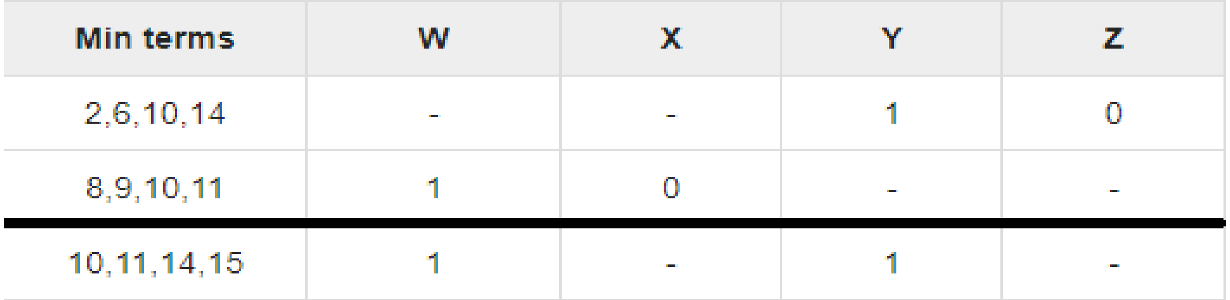
Min terms	W	X	Y	Z
2,6,10,14	-	-	1	0
2,10,6,14	-	-	1	0
8,9,10,11	1	0	-	-
8,10,9,11	1	0	-	-
10,11,14,15	1	-	1	-
10,14,11,15	1	-	1	-



Min terms	W	X	Y	Z
2,6,10,14	-	-	1	0
8,9,10,11	1	0	-	-
10,11,14,15	1	-	1	-

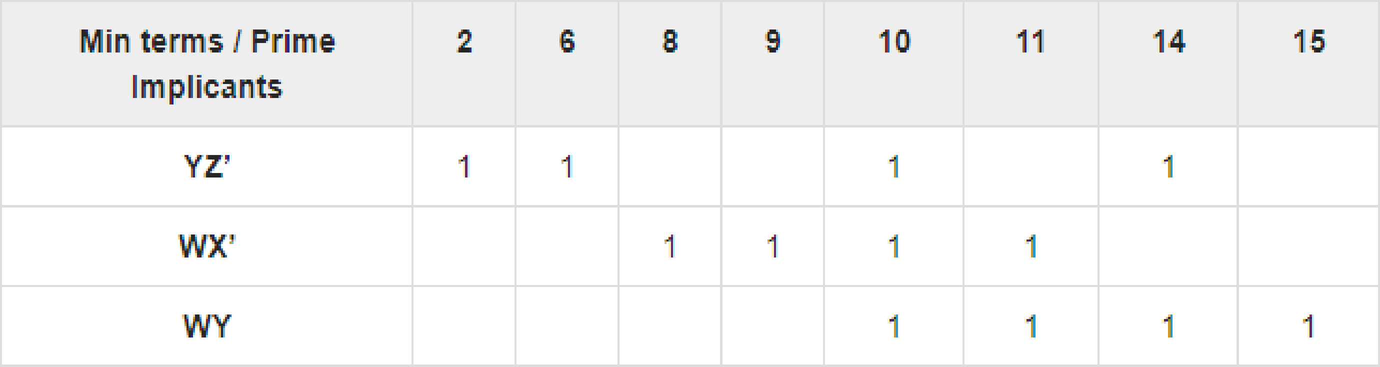
QM ALGORITHM

- These combinations of 4 min terms are available in two rows. So, we can remove the repeated rows. The reduced table after removing the redundant rows is shown below.



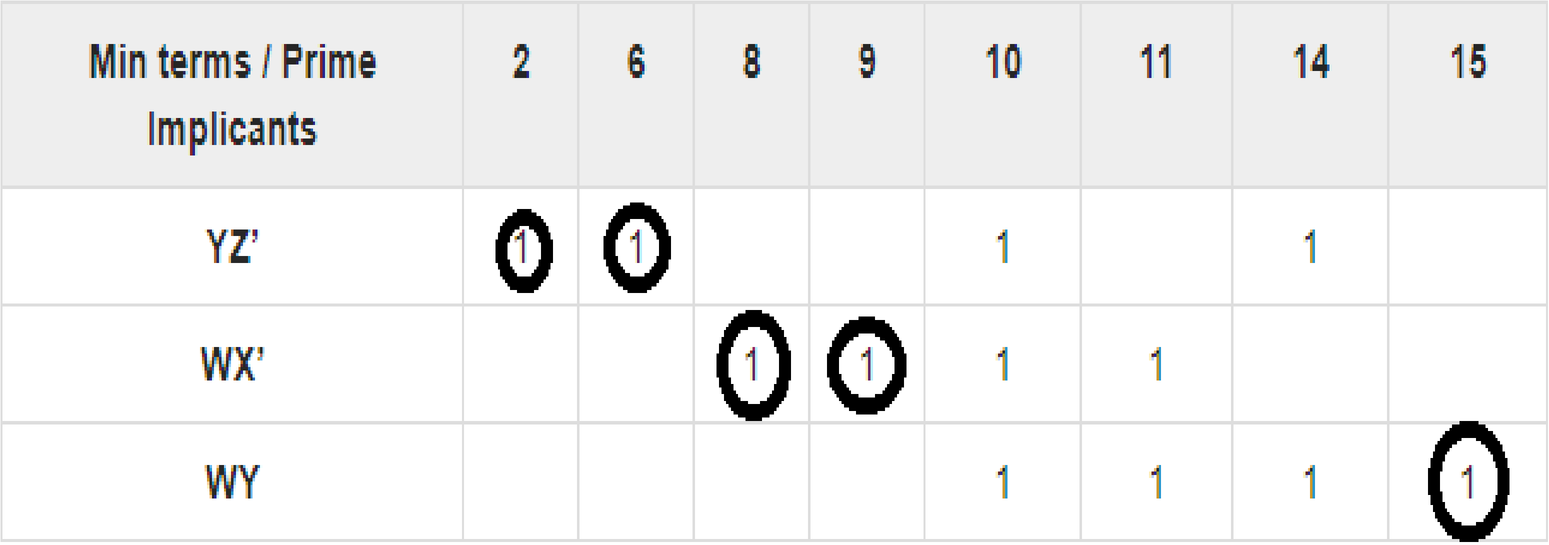
Min terms	W	X	Y	Z
2,6,10,14	-	-	1	0
8,9,10,11	1	0	-	-
10,11,14,15	1	-	1	-

- There are three rows in the above table. So, each row will give one prime implicant. Therefore, the **prime implicants** are YZ’, WX’ & WY.



Min terms / Prime Implicants	2	6	8	9	10	11	14	15
YZ’	1	1			1		1	
WX’			1	1	1	1		
WY					1	1	1	1

QM ALGORITHM



Min terms / Prime Implicants	2	6	8	9	10	11	14	15
YZ'	①	①			1		1	
WX'			①	①	1	1		
WY					1	1	1	①

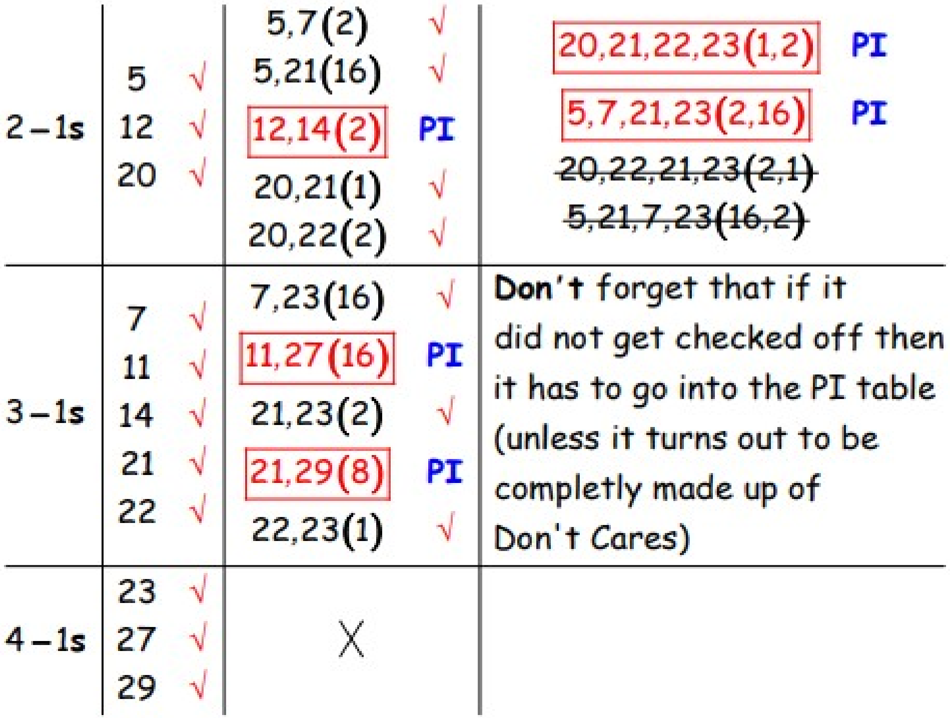
- In this example problem, we got three prime implicants and all the three are essential. Therefore, the **simplified Boolean function** is

  f(*W,X,Y,Z)* = YZ' + WX' + WY.

QM ALGORITHM

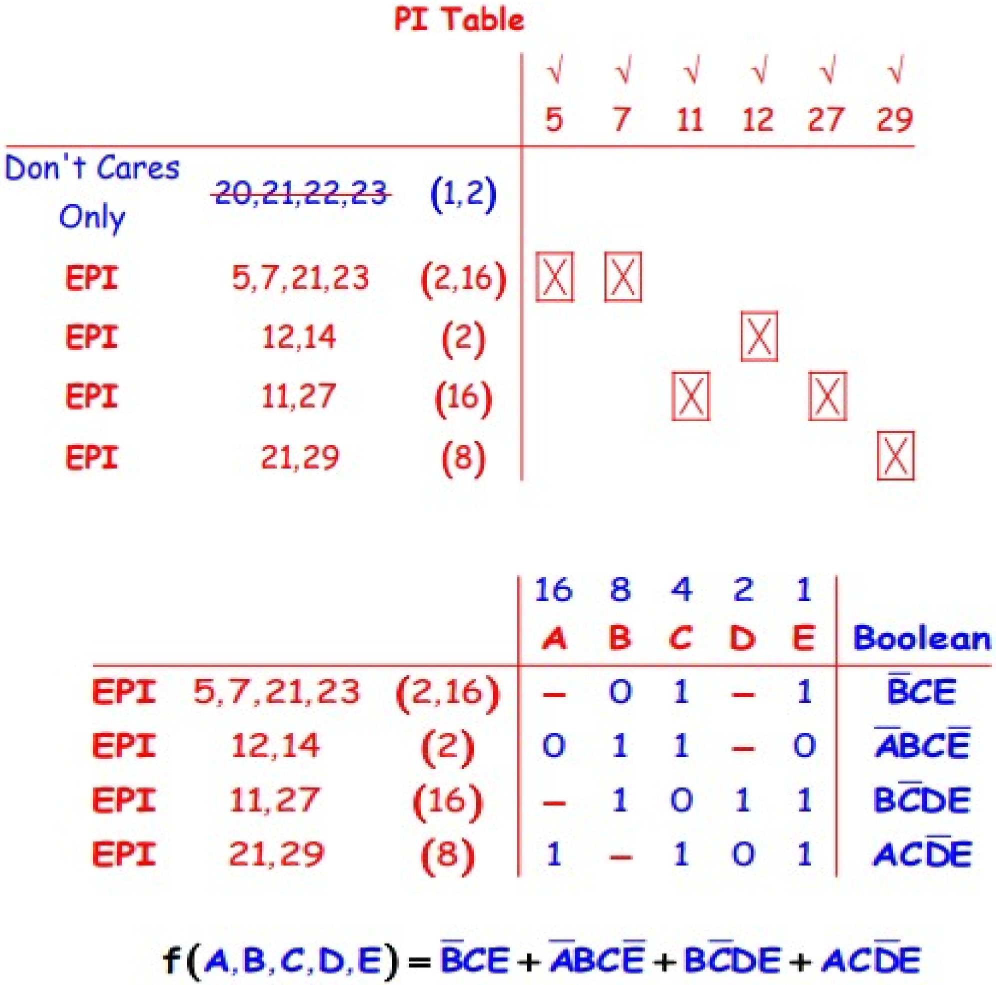
Eg: F(A,B,C,D) = $\sum m(5,7,11,12,27,29) + \sum d\,(14,20,21,22,23)$ minimize the given using QM method.

Sol: At step 1 we have to search for prime Implicants.



2−1s	5 √ 12 √ 20 √	5,7(2) √ 5,21(16) √ 12,14(2) PI 20,21(1) √ 20,22(2) √	20,21,22,23(1,2) PI 5,7,21,23(2,16) PI ~~20,22,21,23(2,1)~~ ~~5,21,7,23(16,2)~~
3−1s	7 √ 11 √ 14 √ 21 √ 22 √	7,23(16) √ 11,27(16) PI 21,23(2) √ 21,29(8) PI 22,23(1) √	Don't forget that if it did not get checked off then it has to go into the PI table (unless it turns out to be completly made up of Don't Cares)
4−1s	23 √ 27 √ 29 √	X	

QM ALGORITHM



PI Table

			✓	✓	✓	✓	✓	✓
			5	7	11	12	27	29
Don't Cares Only	~~20,21,22,23~~	(1,2)						
EPI	5,7,21,23	(2,16)	☒	☒				
EPI	12,14	(2)				☒		
EPI	11,27	(16)			☒		☒	
EPI	21,29	(8)						☒

			16	8	4	2	1	
			A	B	C	D	E	Boolean
EPI	5,7,21,23	(2,16)	–	0	1	–	1	$\bar{B}CE$
EPI	12,14	(2)	0	1	1	–	0	$\bar{A}BC\bar{E}$
EPI	11,27	(16)	–	1	0	1	1	$B\bar{C}DE$
EPI	21,29	(8)	1	–	1	0	1	$AC\bar{D}E$

$$f(A,B,C,D,E) = \bar{B}CE + \bar{A}BC\bar{E} + B\bar{C}DE + AC\bar{D}E$$

Recap

- The tabulation method is a specific step-by-step procedure that is guaranteed to produce a simplified standard-form expression for a function.
- **Step 1** − Arrange the given min terms in an **ascending order** and make the groups based on the number of ones present in their binary representations.
- **Step 2** − Compare the min terms present in **successive groups**.
- **Step 3** − Repeat step2 with newly formed terms till we get all **prime implicants**.
- **Step 4** − Formulate the **prime implicant table**.
- **Step 5** − Find the essential prime implicants by observing each column.

FACULTY VIDEO LINKS, YOUTUBE & NPTEL VIDEO LINKS AND ONLINE COURSES DETAILS

Youtube/other Video Links:

- https://www.youtube.com/watch?v=l1jgq0R5EwQ&t=2s&ab_channel=NesoAcademy
- https://www.youtube.com/watch?v=Shj-u66gdE8&ab_channel=EkeedaEkeedaVerified

Daily Quiz

- What are the advantages of tabulation methods over K-Map method?
- Write the steps to minimize function using QM method.
- $F(A,B,C,D) = \sum m(1,3,7,11,15) + \sum d\,(0,2,5)$ minimize the given using QM method.
- What minimization methods are used when number of variables are greater than 6:

a) K-Map method

b) QM method

c) (a) or (b) both

Old Questions

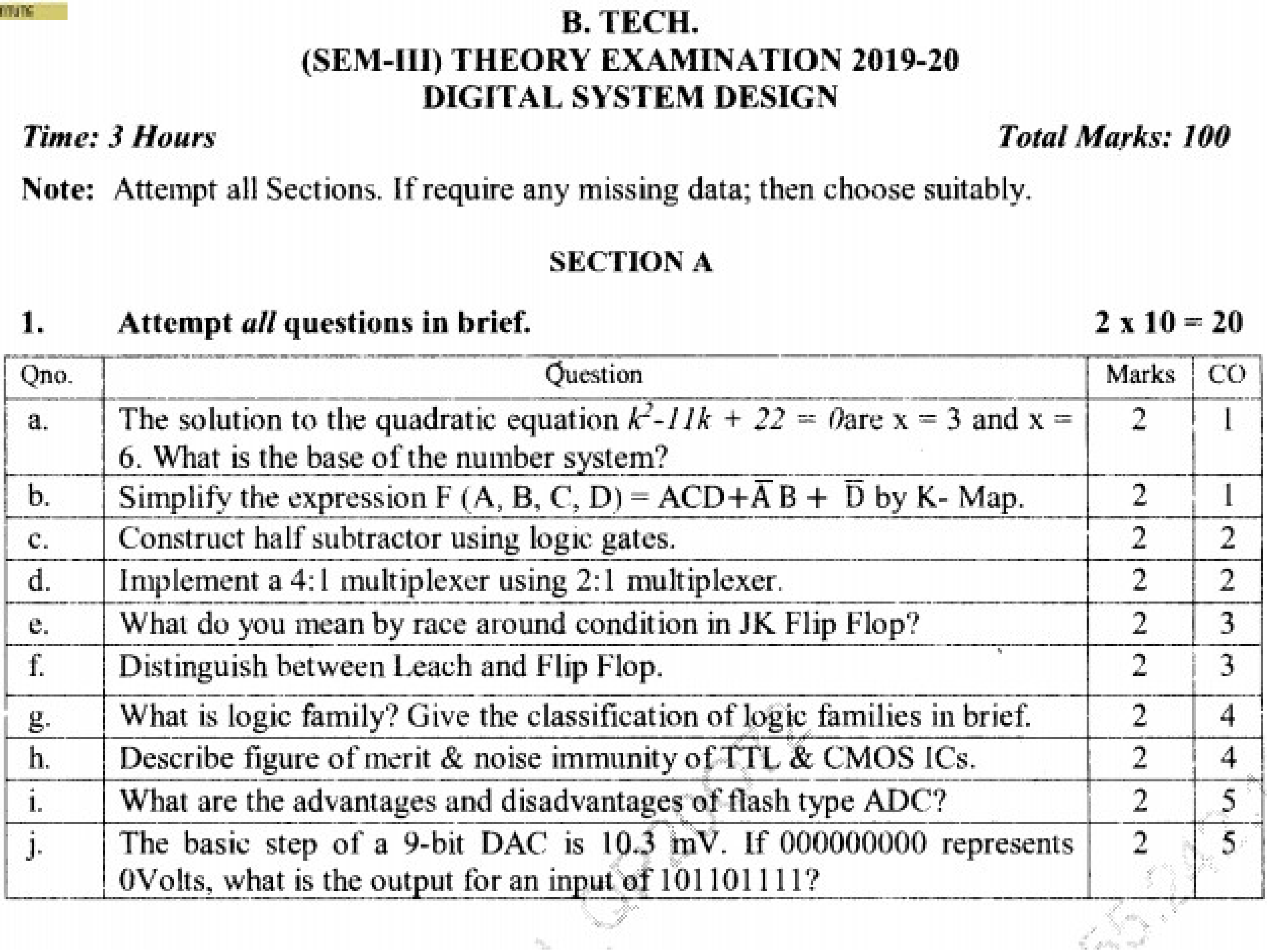
- Minimize the following function by Quine McClusky method and also perform the NAND implementation of the simplified function.

  $F(w,x,y,z) = \sum m(1,4,8,9,13,14,15) + d(2,3,11,12)$
- $F(A,B,C,D) = \sum m(0,1,2,3,10,11,12,13,14,15)$ minimize the given using QM method.
- $F(A,B,C,D) = \sum m(1,3,7,11,15) + \sum d(0,2,5)$ minimize the given using QM method.

Weekly Assignment

- Minimize the function using k-map in SOP form: F(A,B,C,D) = Σ (3,4,6,15) + d(5,7,11,14)
- Minimize the function using k-map in POS form: F(A,B,C,D) = Π (1,9) + d(0,2,3,4,5,6,7,8)
- F(A,B,C,D)= ∑(0,1,9,15, 24,29,30) + d (8,11,31) minimize using Quine-McCluskey method.
- Simplify the following Boolean function using K-map
  Y = ∑m(0,1,3,5,6,7,9,11,16,18,19,20,21,22,24,26)
- F(A,B,C,D)= ∑(0,1,9,15, 24,29,30) + d (8,11,31) minimize using Quine-McCluskey method.

OLD QUESTION PAPERS



B. TECH.
(SEM-III) THEORY EXAMINATION 2019-20
DIGITAL SYSTEM DESIGN

Time: 3 Hours ***Total Marks: 100***

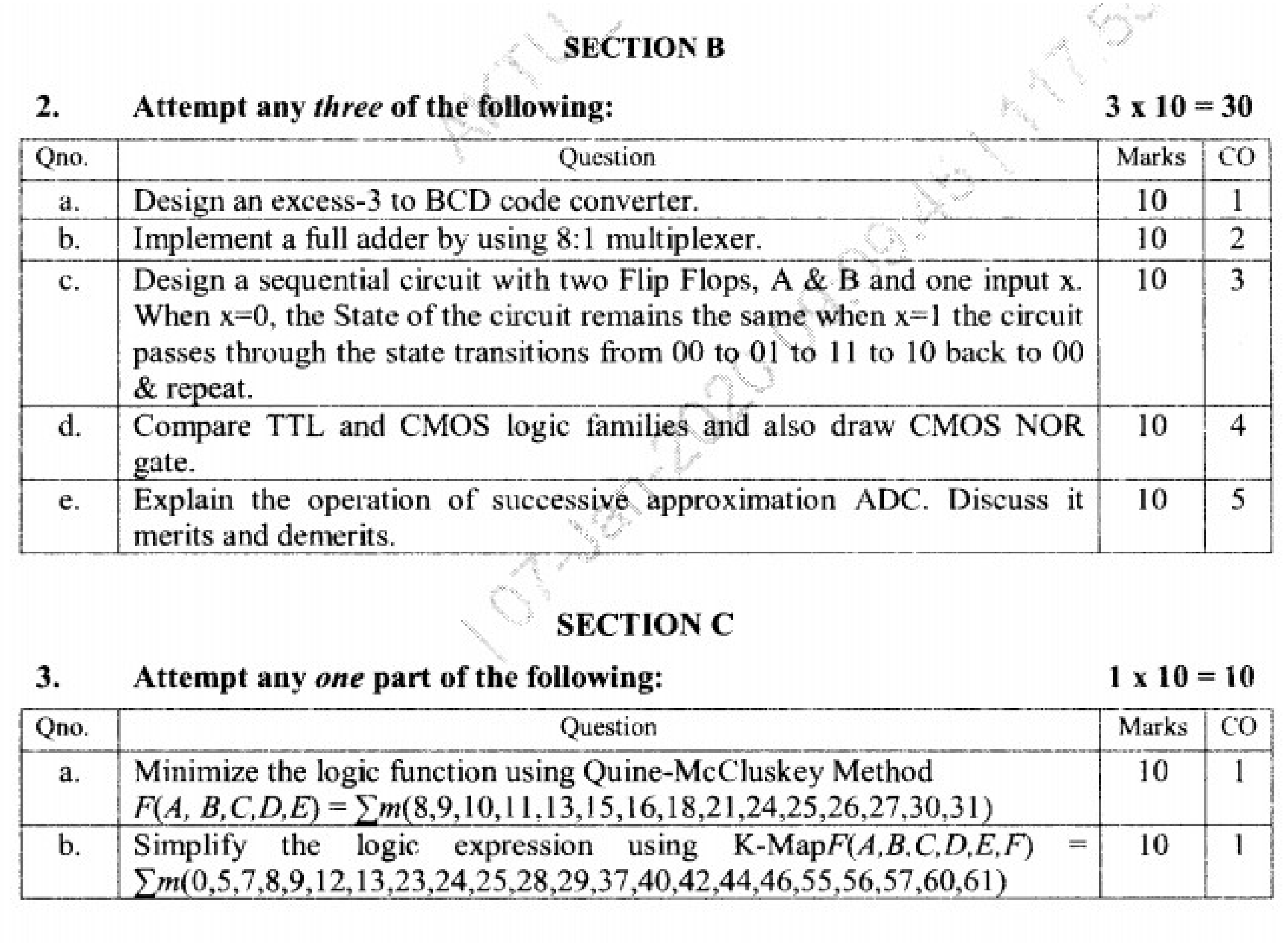
Note: Attempt all Sections. If require any missing data; then choose suitably.

SECTION A

1. Attempt *all* questions in brief. **2 x 10 = 20**

Qno.	Question	Marks	CO
a.	The solution to the quadratic equation k^2-*11k* + *22* = *0*are x = 3 and x = 6. What is the base of the number system?	2	1
b.	Simplify the expression F (A, B, C, D) = ACD+$\bar{A}$B + $\bar{D}$ by K- Map.	2	1
c.	Construct half subtractor using logic gates.	2	2
d.	Implement a 4:1 multiplexer using 2:1 multiplexer.	2	2
e.	What do you mean by race around condition in JK Flip Flop?	2	3
f.	Distinguish between Leach and Flip Flop.	2	3
g.	What is logic family? Give the classification of logic families in brief.	2	4
h.	Describe figure of merit & noise immunity of TTL & CMOS ICs.	2	4
i.	What are the advantages and disadvantages of flash type ADC?	2	5
j.	The basic step of a 9-bit DAC is 10.3 mV. If 000000000 represents 0Volts, what is the output for an input of 101101111?	2	5

OLD QUESTION PAPERS



SECTION B

2. **Attempt any *three* of the following:** **3 x 10 = 30**

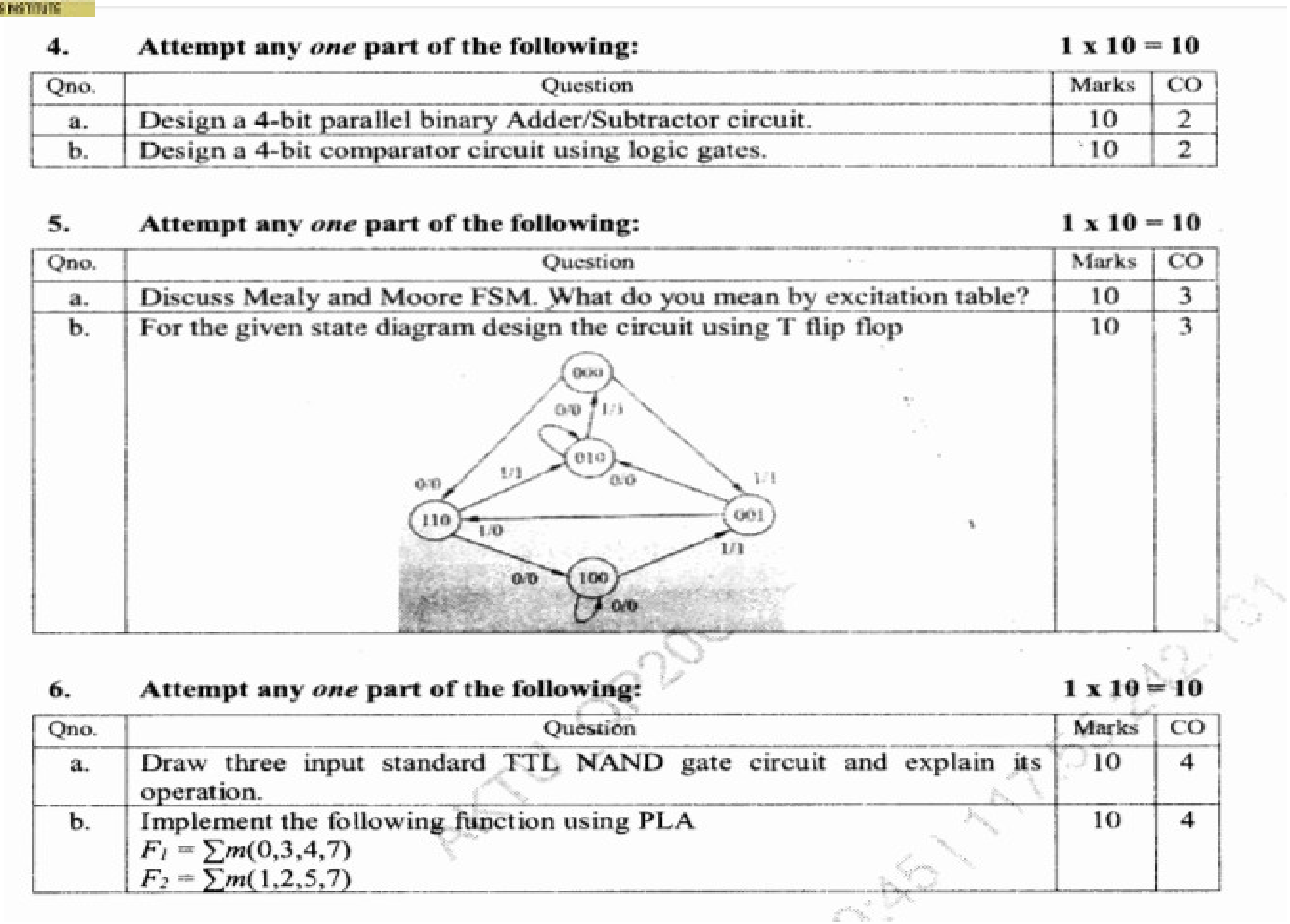
Qno.	Question	Marks	CO
a.	Design an excess-3 to BCD code converter.	10	1
b.	Implement a full adder by using 8:1 multiplexer.	10	2
c.	Design a sequential circuit with two Flip Flops, A & B and one input x. When x=0, the State of the circuit remains the same when x=1 the circuit passes through the state transitions from 00 to 01 to 11 to 10 back to 00 & repeat.	10	3
d.	Compare TTL and CMOS logic families and also draw CMOS NOR gate.	10	4
e.	Explain the operation of successive approximation ADC. Discuss it merits and demerits.	10	5

SECTION C

3. **Attempt any *one* part of the following:** **1 x 10 = 10**

Qno.	Question	Marks	CO
a.	Minimize the logic function using Quine-McCluskey Method $F(A, B, C, D, E) = \sum m(8,9,10,11,13,15,16,18,21,24,25,26,27,30,31)$	10	1
b.	Simplify the logic expression using K-Map$F(A,B,C,D,E,F) = \sum m(0,5,7,8,9,12,13,23,24,25,28,29,37,40,42,44,46,55,56,57,60,61)$	10	1

OLD QUESTION PAPERS



4. Attempt any *one* part of the following: 1 x 10 = 10

Qno.	Question	Marks	CO
a.	Design a 4-bit parallel binary Adder/Subtractor circuit.	10	2
b.	Design a 4-bit comparator circuit using logic gates.	10	2

5. Attempt any *one* part of the following: 1 x 10 = 10

Qno.	Question	Marks	CO
a.	Discuss Mealy and Moore FSM. What do you mean by excitation table?	10	3
b.	For the given state diagram design the circuit using T flip flop	10	3

6. Attempt any *one* part of the following: 1 x 10 = 10

Qno.	Question	Marks	CO
a.	Draw three input standard TTL NAND gate circuit and explain its operation.	10	4
b.	Implement the following function using PLA $F_1 = \sum m(0,3,4,7)$ $F_2 = \sum m(1,2,5,7)$	10	4

REFERENCE

- R.P. Jain, “Modern Digital Electronics,” Tata McGraw Hill, 4th edition, 2009.
- A. Anand Kumar, “Fundamental of Digital Circuits,” PHI 4th edition, 2018.
- W.H. Gothmann, “Digital Electronics- An Introduction to Theory and Practice,” PHI, 2nd edition, 2006.
- D.V. Hall, “Digital Circuits and Systems,” Tata McGraw Hill, 1989.
- A. K. Singh, “Foundation of Digital Electronics & Logic Design,” New Age Int. Publishers.
- Subrata Ghosal, “Digital Electronics,” Cengage publication, 2nd edition, 2018

Thank You