INDEX

Name: Priyanka Sahu Sub: DEC & MP (Lecture notes)

Std: 5th sem Div: 1 Roll No.:

Sl. No.	Topic	Page No.	Date	Grade/ Marks	Sign

TH-3 DIGITAL ELECTRONICS & MICROPROCESSOR (5th Sem)

Syllabus

Chapter 1. Basics of Digital Electronics

1.1. Binary, Octal, Hexadecimal number systems and compare with Decimal system.
1.2. Binary addition, subtraction, multiplication and division.
1.3. 1's complement and 2's complement numbers for a binary number.
1.4. Subtraction of binary numbers in 2's complement method.
1.5. Use of weighted and un-weighted codes and write binary equivalent number for a number in 8421, Excess-3 and Gray code and vice-versa.
1.6. Importance of parity bit.
1.7. Logic Gates: AND, OR, NOT, NAND, NOR and EX-OR gates with truth table.
1.8. Realize AND, OR, NOT operations using NAND, NOR gates.
1.9. Different postulates and De-Morgan's theorems in Boolean algebra.
1.10. Use of Boolean algebra for simplification of logic expression.
1.11. Karnaugh map for 2, 3, 4 variables; simplification of SOP and POS logic expression using K-map.

Chapter 2. Combinational Logic Circuits

2.1. Give the concept of combinational logic circuits
2.2. Half adder circuit and verify its functionality using truth table.
2.3. Realize a half-adder using NAND gates only and NOR gate only.
2.4. Full adder circuit and explain its operation with truth table.
2.5. Realize full-adder using two half-adders and an OR gate and write truth table.
2.6. Full subtraction circuit and explain its operation with truth table.
2.7. Operation of 4X1 multiplexers and 1X4 demultiplexer.
2.8. Working of binary-decimal encoder and 3X8 decoder.
2.9. Working of two bit magnitude comparator

Chapter 3. Sequential Logic Circuits

3.1. Give the idea of sequential logic circuits
3.2. State the necessity of clock and give the concept of level clocking and edge triggering.
3.3. Clocked SR flip flop with preset & clear inputs
3.4. Construct level clocked JK flip flop using S-R flip flop and explain with truth table
3.5. Concept of race around condition and study of master slave JK flip flop.
3.6. Give the truth tables of edge triggered D and T flip flops and draw their symbols.

3.7. Applications of flip flops.
3.8. Define modulus of a counter.
3.9. 4-bit asynchronous counter and its timing diagram.
3.10. Asynchronous decade counter.
3.11. 4-bit synchronous counter.
3.12. Distinguish between synchronous and asynchronous counters.
3.13. State the need for a register and list the four types of registers.
3.14. Working of SISO, SIPO, PISO, PIPO register with truth table using flip flop.

Chapter 4. 8085 Microprocessor.

4.1. Introduction to microprocessors, microcomputers
4.2. Architecture of Intel 8085A microprocessor and description of each block.
4.3. Pin diagram and description
4.4. Stack & stack pointer and stack top.
4.5. Interrupt
4.6. Opcode & Operand
4.7. Differentiate between one byte, two byte and 3 byte instruction with example.
4.8. Instruction set of 8085 example.
4.9. Addressing modes.
4.10. Fetch cycle, machine cycle, instruction cycle, T-state.
4.11. Timing diagram for m/m read, m/m write, I/O read and I/O write.
4.12. Timing Diagram for 8085 instruction.

4.13. Counter and Time delay.
4.14. Simple assembly language programming of 8085.

Chapter 5. Interfacing and support chips.
5.1. Basic Interfacing Concepts, Memory mapping and I/O mapping.
5.2. Functional block diagram and description of each block of programmable peripheral interface Intel 8255.
5.3. Application using 8255 : Seven segment LED display, square wave generator, traffic light controller.

Learning Resources (Recommended books)

1. Fundamental of Digital Electronics - Ananda Kumar (author), PHI (publisher)
2. Digital Electronics - Principal & Application - S.K. Mondal, TMH (author) (publisher)
3. Digital Electronics - B.R. Gupta & V. Singhal, S.K. Kataria (author) (publisher)
4. Digital Electronics - P. Raja (author), Sci Tech (publisher),
5. Microprocessor Architecture programming & ~~Microcomputers~~ application with 8085 - R.S. Gaonkar (author), Penram (publisher)
6. Fundamentals of microprocessor & microcomputers - B. Ram (author), Dhanpat rai (publisher)
7. Microprocessor and Interfacing - Surendra Choudhury & S.P. Choudhury (author), Scitech (publisher)

CHAPTER - 1

Basics of Digital Electronics

Introduction to Binary number system

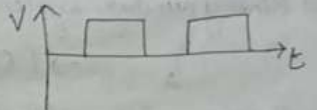
Digital system :- It is an electronic network that process information using only digits (numbers) to implement calculations and operations.

Analog signal

→ A continuously varying signal (current/voltage) is called an analog signal.

Digital signal

→ A signal (current / voltage) which can have only two discrete value is called digital signal



Binary variable :- It represents a binary number that does not have a predefined value. Let A is a binary value that means it can have only two values i.e. A = 0 or A = 1

If $A = 0 \Rightarrow \bar{A} = 1$
If $A = 1 \Rightarrow \bar{A} = 0$

[Not(A) = $\bar{A}$ (complement) Conversion]

→ In an electronics system a single binary digit is called as bit but as using a single digit would seriously limit the maths that could be performed so binary bit are normally used in groups. [8 bits = 1 byte]

1 word = 16 bits = 2 bytes
1 nibble = 4 bit = half byte or half an octet or tetrade or semi-octet

1.1. Binary, Octal, Hexadecimal number systems and compare with decimal number system

Base or Radix:- It is a number that specifies how many digits are available for example in:-

① decimal number system → the base is 10. There are 10 digits to count:- (0, 1, 2, ... 9)

$(\quad)_{10}$ → base or radix

② Binary number system → the base is 2.

$(\quad)_2 \rightarrow (0, 1)$

③ Octal number system → the base is 8.

$(\quad)_8 \rightarrow (0, 1, \ldots\ldots, 7)$

④ Hexadecimal number system → the base is 16.

$(\quad)_{16} \rightarrow (0, 1, 2, 3, \ldots\ldots, 10, 11, 12, 13, 14, 15)$
10 → A, 11 → B, 12 → C, 13 → D, 14 → E, 15 → F

Conversions

① **Decimal to Binary**

$(12)_{10} \rightarrow (\quad)_2$

2 | 12
2 | 6 — 0
2 | 3 — 0
   1 — 1

$= (1100)_2$

$(36)_{10} \rightarrow (\quad)_2$

2 | 36
2 | 18 — 0
2 | 9 — 0
2 | 4 — 1
2 | 2 — 0
   1 — 0

$= (100100)_2$

$(0.6875)_{10} \rightarrow (\quad)_2$

0.6875 × 2 = 1.3750 — 1
0.3750 × 2 = 0.7500 — 0
0.7500 × 2 = 1.5000 — 1
0.5000 × 2 = 1.0000 — 1

$= (.1011)_2$

② **Binary to Decimal**

$(1100)_2 \rightarrow (\quad)_{10}$

$\overset{2^3}{1}\ \overset{2^2}{1}\ \overset{2^1}{0}\ \overset{2^0}{0}$

$(1\times2^3) + (1\times2^2) + (0\times2^1) + 0\times2^0$
$= 8+4 = (12)_{10}$

OR
$2^3\ 2^2\ 2^1\ 2^0$
8 4 2 1
1 1 0 0
= 8+4
= $(12)_{10}$

$(.1011)_2 \rightarrow (\quad)_{10}$

$.\overset{2^{-1}}{1}\ \overset{2^{-2}}{0}\ \overset{2^{-3}}{1}\ \overset{2^{-4}}{1}$

$= 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4}$
$= 1\times0.5 + 0\times0.25 + 1\times0.125 + 1\times0.0625$
$= 0.5 + 0.125 + 0.0625$
$= (0.6875)_{10}$

③ **Binary to Octal**

Octal means 8 $= 2^3$
So, consider 3 bit

4	2	1
0	0	1
	= 1	
0	1	0
	= 2	
1	1	1
= 4+2+1		
= 7		

$(\overline{001}\,\overline{010}\,\overline{111})_2 \rightarrow (\quad)_8$
$= (1\ \ 2\ \ 7)_8$

$\overline{001}\,\overline{010}\,\overline{110}.\overline{010}\,\overline{100} \rightarrow (\quad)_8$
$= (1\ \ 2\ \ 6\ .\ 2\ \ 4)_8$

$\overline{001}\,\overline{010}.\overline{100}\,\overline{101} \rightarrow (\quad)_8$
$(1\ \ 2\ .\ 4\ \ 5)_8$

④ Octal to Binary

$(127)_8 \rightarrow (\quad)_2$

$= (001\ 010\ 111)_2$

	4	2	1
1	0	0	1
2	0	1	0
7	1	1	1

⑤ Binary to Hexadecimal

$(0010\ 1010\ 1011\ 1101)_2 \rightarrow (\quad)_{16}$

$= (2\ A\ B\ D)_{16}$

$(0.0101\ 1111)_2 \rightarrow (\quad)_{16}$

$= (0.5\ F)_{16}$

$16 = 2^4 = 4$ bit

8	4	2	1	
0	0	1	0	= 2
1	0	1	0	= 8+2 = 10(A)
1	0	1	1	= 8+2+1 = 11(B)
1	1	0	1	= 8+4+1 = 13(D)
0	1	0	1	= 4+1 = 5
1	1	1	1	= 8+4+2+1 = 15(F)

⑥ Decimal to Octal

$(378.93)_{10} \rightarrow (\quad)_8$

8	378	
8	47	−2
	5	−7

$0.93 \times 8 = \boxed{7}.44 \rightarrow 7$

$0.44 \times 8 = \boxed{3}.52 \rightarrow 3$

$0.52 \times 8 = \boxed{4}.16 \rightarrow 4$

$0.16 \times 8 = \boxed{1}.28 \rightarrow 1$

$= (572.7341)_8$

→ If you get all 0 after point, then stop

→ If you are not getting ~~term~~ 0, then go by 4 or 5 steps.

⑦ Octal to Decimal

$(572)_8 \rightarrow (\quad)_{10}$

$8^2\ 8^1\ 8^0$

$= 5\times8^2 + 7\times8^1 + 2\times8^0$

$= (5\times64) + (7\times8) + (2\times1)$

$= 320 + 56 + 2$

$= (378)_{10}$

⑧ Hexadecimal to Binary

$(0.5F)_{16} \rightarrow (\quad)_2$

$= (0.0101\ 1111)_2$

$(AB)_{16} \rightarrow (\quad)_2$

$= (10101011)_2$

4 bit

	8	4	2	1
5	0	1	0	1
F	1	1	1	1
(15)				
A(10)	1	0	1	0
B(11)	1	0	1	1
E(14)	1	1	1	0

⑨ Decimal to Hexadecimal

$(58)_{10} \rightarrow (\quad)_{16}$

16	58	
	3	−A (58 − 48 = 10(A))

$= (3A)_{16}$

⑩ Hexadecimal to Decimal

$(3A)_{16} \rightarrow (\quad)_{10}$

$16^1\ 16^0$

$= 3\times16 + A(10)\times1$

$= 48 + 10$

$= (58)_{10}$

(11) Octal to Hexadecimal

→ We cannot convert directly from octal to hexadecimal nor from hexadecimal to octal.

→ First we need to convert to binary or decimal and then to the other form (octal/hexadecimal)

$(127)_8 \rightarrow (\ )_{16}$

$(000001010111)_2$

0 5 7 $= (57)_{16}$

OR

$1\ 2\ 7$ ($8^2\ 8^1\ 8^0$) $= 1\times64 + 2\times8 + 7\times1$

$= 64 + 16 + 7$

$= (87)_{10}$

16 | 87

5 − 7 ↑

$= (57)_{16}$

(12) Hexadecimal to Octal

$(5F)_{16} \rightarrow (\ )_8$

$(00101\ 1111)_2$

1 3 7 $= (137)_8$

OR

$5\ F$ ($16^1\ 16^0$) $= 5\times16 + F(15)\times1 = 80 + 15 = (95)_{10}$

8 | 95

8 | 11 − 7 ↑

1 − 3

$= (137)_8$

1.2. Binary addition, subtraction, multiplication and division

Binary arithmetic :-

Binary addition :-

A B	Sum	Carry
0+0	0	0
0+1	1	0
1+0	1	0
1+1	0	1

Binary subtraction :-

A B	Result (difference)	Borrow
0−0	0	0
0−1	1	1
1−0	1	0
1−1	0	0

Binary Multiplication :-

A B	Result
0×0	0
0×1	0
1×0	0
1×1	1

Binary Division :-

A B	Result
0÷0	0
0÷1	0
1÷0	0
1÷1	1

Examples

Binary addition:-

```
  10101        1111
+   101      +  111
-------      ------
  11010      10110
```

```
 0.001
 0.101
+0.101
-------
 1.011
```

1+1 = 10 i.e Sum 0 with carry 1

1+1+1 = 10+1 = 11 i.e. Sum 1 with carry 1

1+1+1+1 = 11+1 = 100 i.e. Sum 0 with carry 10

Binary Subtraction:-

```
  10101        10001      0.110
-   101      -   100    - 0.001
-------      -------    -------
  10000        10101      0.101
```

```
  10001010
-       111
----------
  10000011
```

Binary Multiplication:-

```
    101          10100
  x  11        x  101
-------      --------
    101          10100
 + 101        00000
-------     10100
   1111      --------
                110100
   1010
 x   10
-------
   0000
+ 1010
-------
  10100
```

Binary Division:-

Step 1: First look at the first two numbers in the dividend and compare with the divisor. Add the number 1 in the quotient place if dividend is greater than or equal to divisor then add 1 in quotient place. on 0

Step 2: Then subtract the value, you get 1 as remainder and then bring down the next number.

Step 3: Repeat the process until the ~~remainder becomes zero by comparing the dividend and the divisor value~~ last number (right) of dividend is compared with the divisor.

```
4|78|19        Divisor) Dividend (Quotient
 -4
 38
-36
 --
  2          Remainder
```

```
100)1001110(10011
 700↓↓
 -----
  00111
   100↓
   ----
   0110
    100
    ---
     10
```

```
10|11101|1110
 -10↓|||
 -----|||
  011 |||
   -10↓||
   ----||
   010 ||
    -10↓|
    ----|
    001 |
     -00|
     ---|
      1 |
```

```
110|110110(1001
  -110↓↓↓
  -------
  000110
     -110
     ----
      000
```

```
11)10010(110
  -11↓|
  ----|
  011 |
   -11↓
   ----
    00
     0
     -
     0
```

```
11)1011.11(11.11
  -11↓
  ----
  101
   11↓
   ---
   101
   -11↓
   ----
   1001
     11
     --
     10
     --
```

1.3. 1's complement and 2's complement numbers for a binary number.

9's Complement

→ To obtain the 9's complement of a decimal number, each digit of the number is subtracted from 9. For example,

9's complement of 45 is (99-45) = 54.

9's complement of 523 is (999-523) = 476

10's complement

10's complement = 9's complement + 1

10's complement of 45 = (99-45) 54+1 = 55

10's complement of 523 = 476+1 = 477

Example 1

```
  45  (Decimal number)
+ 55  (Its 10's complement)
1 00  → Sum
Carry = 1
```

∴ Sum of the number and its 10's complement only upto two digits, it is zero. In other words we are neglecting the carry of the last stage.

Example 2

```
  5 2 3  (Decimal number)
  4 7 7  (Its 10's complement)
1 0 0 0  → Sum
Carry = 1
```

∴ Sum of the number and its 10's complement only upto 3 digits, it is zero.

∴ We see that the 10's complement gives the negative value of the number.

10's complement of a decimal number = − decimal number

1's Complement

→ The 1's complement in the binary number system is similar to the 9's complement in the decimal system.

→ The 1's complement of a binary number is obtained by subtracting each bit of the number from 1. The 1's complement of 01 is 10. The 1's complement of 111 is 000.

∴ We see that the 1's complement of a binary number can be obtained by simply changing each bit 1 to 0 and 0 to 1.

E.g. 1. 1's complement of 100110
= 011001

E.g. 2. 1's complement of 0000
= 1111

E.g. 3 1's complement of 111111
= 000000

2's complement

→ The 2's complement in the binary system is similar to the 10's complement in the decimal number system.

→ **2's complement of binary number = 1's complement + 1**

→ 2's complement can also be obtained by subtracting the number from zero.

E.g. 1. Find 2's complement of 10011.

Soln

1's complement of 10011 + 1

= 01100 + 1

= 01101

E.g. 2. Find 2's complement of 111.

1's comp. = 000

2's comp = 000 + 1 = 001

E.g. 3 Find 2's complement of 0000

1's comp. = 1111

2's comp. = 1111 + 1

carry 1 ← 0000

→ Let us examine the sum when a binary number is added to its 2's complement.

Example 1 (4 bit):- Binary number = 1001

1' complement = 0110

2' complement = 0110 + 1 = 0111

Number + its 2's complement = 1001 + 0111 = 1 0000

carry

8 bit examples

1) Binary number = 00001001

1's complement = 11110110

2's complement = 11110110 + 1 = 11110111

Number = 00001001

1 00000000

carry

→ The last carry will be neglected if the 8 bit sum is considered.

2) +4 (decimal) = 00000100 (binary)

−4 = its 2' comp. = 11111100

+4 = 00000100

−4 = 11111100

1 00000000

carry

1·4 Subtraction of binary numbers using 2's comp method?

Q Add +5 and −7

→ If there is a carry out from MSB, then ignore it
→ Look at the sign bit i.e. if MSB=0, then result is +ve and is in true binary form. If the MSB=1, the result is −ve and is in the 2's complemented form. We have to convert it to its true binary form.

7 = 00000111

1's comp of 7 = 11111000

2's comp of 7 = −7 = 11111001

00000101 → 5

00000111 → 7

+5 = 00000101

−7 = +11111001

−2 = 11111100

Check: 2: 00000010

1's comp of 2 = 11111101

2's comp of 2 = −2 = 11111110

Q Subtract (379)10 from (968)10 using 10's comp

999 − 379 = 620 → 9's comp

620 + 1 = 621 → 10's comp

968 + 621 = 1589 (carry generated)

968 − 379 = 589

∴ We see that the 2's complement of a binary number represents its negative.

Binary subtraction using 2's complement

Ordinary binary subtraction:

5 = 0101

−2 = −0010

3 = 0011

Binary subtraction using 2's comp:

2 = 0010

2' (1's comp) = 1101

+1

−2 = 1110 → 2's comp

5 = 0101

2's comp of 2 = +1110

10011 = 3

↑ Neglect carry of last step

Q Subtract −5 from 2 using 2's complement 4 bit subtraction and 8 bit subtraction

2 − 5 = −3

2 = 0010

2' = 1101

+1

−2 = 1110

5 = 0101

4 bit subtraction: 0011

5 = 0101

5' = 1010

+1

−5 = 1011

2 = +0010

1101

0010

+1

0011 = 3

8 bit subtraction

5 = 00000101

5' = 11111010

+1

−5 = 11111011

2 = +00000010

11111101

00000010

+1

00000011 = 3

Subtraction using 1's complement method

→ If there is a carry out, bring the carry from MSB around and add it to the least significant bit (LSB).

→ Look at the sign bit i.e. if MSB is 0, then result is +ve and is in true binary form.

→ If the MSB is 1, then result is -ve and is in its 1's complement form, so we have to convert it into its true binary form.

Q1 Subtract 14 from 25 using the 1's complement arithmetic.

2|25 → 12-1, 6-0, 3-0, 1-1 ↑
2|14 → 7-0, 3-1, 1-1 ↑

25 − 14 = 11

14 → 00001110
14' → 11110001
25 → 00011001
+
00001010
carry 00001010
+1
00001011 → +ve
8+2+1 = 11

Q2 Add -25 to 14 using 8 bit 1's complement method.

14 − 25 = −11

25 → 00011001
25' → 11100110
14 → +00001110
11110100 → -ve
00001011 = 11

Q3 Add -25 to -14 using 1's complement method.

−14 − 25 = −39

14 → 00001110
14' → 11110001
25 → 00011001
25' → 11100110
14' → 11110001
25' → +11100110
(1)11010111
carry 11010111
+1
11011000 → -ve
00100111 = 32+4+2+1
32 16 8 4 2 1 = 39

2's complement method subtraction

Q1 Subtract 14 from 46 using 8 bit 2's complement method.

46 − 14 = 32

2|46 → 23-0, 11-1, 5-1, 2-1, 1-0 ↑
2|14 → 7-0, 3-1, 1-1 ↑

46 → 00101110
14 → 00001110
14' → 11110001
+1
−14 → 11110010 (2's)

+
100100000
carry (neglected)
= 00100000

Check 32: 2|32 → 16-0, 8-0, 4-0, 2-0, 1-0 ↑
32 = 00100000

Q2 Add -75 and 26 using 8 bit 2's complement arithmetic.

−75 + 26 = −49

2|75 → 37-1, 18-1, 9-0, 4-1, 2-0, 1-0 ↑
2|26 → 13-0, 6-1, 3-0, 1-1 ↑
2|49 → 24-1, 12-0, 6-0, 3-0, 1-1 ↑

26 = 00011010
49 = 00110001

75 = 01001011 , 75' = 10110100
+1
−75 = 10110101
26 = 00011010
11001111 = −49
Check 49' = 11001110
−49 = 11001111
00110000+1 = 00110001
−49

1.5 Use of weighted and unweighted codes and write binary equivalent number for a number in 8421, Excess-3 and Gray Code and vice-versa.

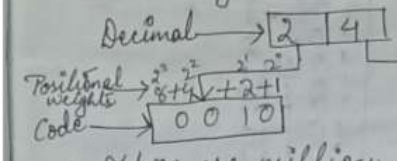
Code :-

→ It is a symbolic representation of discrete information which may be present in the form of numbers letters or physical quantity.
→ The symbols used are the binary digits 0 and 1 which are arranged according to the rule of code.
→ The conversion of the incoming information into the binary format before it can be processed is called encoding.
→ To achieve the reverse of encoding is called decoding.
→ Coding is used to be done :-
i) To represent numeric or alphanumeric or special characters in only binary digits i.e 0 & 1
ii) To check whether a character transmitted in the coded format is correctly received & if not then to correct it that is detecting or correcting errors.

→ Codes are classified into 5 groups :-
i) Weighted binary codes.
ii) Un-weighted codes / Non-weighted codes
iii) Error detecting codes
iv) Error correcting codes
v) Alpha-numeric codes.
vi) Binary Coded Decimal codes / BCD / 8421 Code

Weighted binary codes :- It is of 2 types :- (i) Binary & (ii) BCD

→ These are those binary codes which obey the positional weight principle. Each position of the number represents a specific weight.
→ Several systems of the codes are used to express the decimal digits 0 through 9. (0 to 9)
→ For e.g., in binary number, each bit is assigned a particular weight 2^n, where n is the bit or position for $n = 0, 1, 2, 3, 4, \ldots$
→ The weights are 1, 2, 4, 8, 16 respectively.



→ There are millions of weighted code and the most common one is 8421/BCD code.

Non-weighted codes

→ The non weighted codes are not positionally weighted.

→ In other words codes that are not assigned with any weight to each digit position.

→ 3 types of non-weighted code:- (i) Excess 3 code (ii) Gray code

Examples: Excess-3 (XS-3) and Gray Codes.

Binary Coded Decimal (BCD) code

~~Decimal~~ or 8421 BCD code or Natural BCD code

→ It is a combination of four binary digits that represents decimal numbers.

→ In the BCD numbering system, a decimal number is separated into four bits for each decimal digit within the number. eg 16 → 0001 0110

→ Generally BCD codes mean 8421 code, where 8421 are the weights of four bits respectively.

→ For decimal numbers greater than 9, each digit in decimal can be represented as 4 digit in BCD

Decimal	Binary (8 4 2 1)	BCD
0	0000	0000
1	0001	0001
2	0010	0010
3	0011	0011
4	0100	0100
5	0101	0101
6	0110	0110
7	0111	0111
8	1000	1000
9	1001	1001
10	1010	0001 0000 (10) → 1+0
11	1011	0001 0001 (11) → 1+1
12	1100	0001 0010 (12) → 1+2
13	1101	0001 0011 (13) → 1+3
14	1110	0001 0100 (14) → 1+4
15	1111	0001 0101 (15) → 1+5

Legal or valid codes in BCD i.e from 0000 (0) to 1001 (9)

Illegal codes in BCD i.e. 1010, 1011, 1100, 1101, 1110, 1111 (10 to 15)

Decimal to BCD Conversion

→ It is similar to ~~conversion of hexadecimal~~ to binary.

$(85)_{10}$ = 1000 0101 (BCD)
(8 5)

$(572)_{10}$ = 0101 0111 0010 (BCD)
(5 7 2)

$(8579)_{10}$ = 1000 0101 0111 1001 (BCD)
(8 5 7 9)

BCD to Decimal Conversion

$(1001)_2 = (1001)_{BCD} = (9)_{10}$

8 4 2 1
1 0 0 1
8+1 = 9

$(1000111)_2 = (\overline{0100}\ \overline{0111})_{BCD} = (47)_{10}$

0100 = 4
0111 = 7

$(101001111000.101)_2 = (\overline{0101}\ \overline{0011}\ \overline{1000}.\overline{1010})_{BCD}$

$= (538.625)_{10}$

$(0.1010)_2$
$2^{-1}\ 2^{-3}$
$1 \times 0.5 + 1 \times \frac{1}{8}$
$= 0.5 + 0.125$
$= 0.625$

BCD Addition Rule:-
→ If there is no carry & the sum term is not an illegal code, no correction is needed.
→ If there is a carry out from one group to the next group or if the sum is an illegal code, then 6 (0110) is added to the sum of that group & the resulting carry is added to the next group.

→ The main advantage of binary coded decimal system is that it is a fast and efficient system to convert the decimal numbers into binary numbers as compared to the pure binary system.

BCD Addition:-

(a) 25 + 13

25 → 0010 0101
+13 → 0001 0011
38 → 0011 1000

No carry, no illegal code. So, this is in correct form.

(b) 679.6 + 536.8 = 1216.4

0110 0111 1001 . 0110
+0101 0011 0110 . 1000
1011 1010 1111 . 1110 (All are illegal code)
+0110 +0110 +0110 +0110 (Add 0110 to each)
0001 0000 0101 0100
+1 +1 +1 +1
0001 (1) 0010 (2) 0001 (1) 0101 (5) . 0100 (4) = 1216.4

Excess 3 code (XS-3) :-

→ It is a non-weighted code used to express ~~code to express~~ decimal numbers.

Decimal	BCD 8421	Excess-3 BCD + 0011
0	0000	0011
1	0001	0100
2	0010	0101
3	0011	0110
4	0100	0111
5	0101	1000
6	0110	1001
7	0111	1010
8	1000	1011
9	1001	1100

Gray code

→ The reflected binary code also known as gray code after Frank Gray.

→ It is an ordering of the binary numeral system such that two successive value differ only in one bit.

→ For e.g. the representation of the decimal value "1" in binary would normally be 001 and 2 would be 010. In Gray code : 1 → 0001 but 2 → 011 i.e.

→ So, there is only a bit change from 1 to 2 i.e. 001 to 011

→ Here, we need to understand EX-OR gate mechanism. i.e. when odd number of 1's is present then output is true or '1' otherwise 0/ is false or '0'.

A	B	o/p
0	0	0
0	1	1
1	0	1
1	1	0

→ OR We can follow Binary addition rule i.e. 0+0=0, 0+1=1, 1+0=1, 1+1=0 (sum=0, carry=1 but neglect carry)

Decimal	Binary	Gray code
0	0000	0000
1	0001	0001
2	0010	0011
3	0011	0010
4	0100	0110
5	0101	0111
6	0110	0101
7	0111	0100
8	1000	1100
9	1001	1101
10	1010	1111
11	1011	1110
12	1100	1010
13	1101	1011
14	1110	1001
15	1111	1000

E.g of 8 bit 11001110 → 10101001

→ Gray code is known as unit distance code because of only one bit difference.

Gray to binary conversion

*Use Cross Ex-or mechanism

E.g 1 0 1 1 0 1 1 0 1 (Gray code)

1 1 0 1 1 0 1 1 0 (binary equivalent number)

1 0 1 1 0 1 1 0 1 (Gray code)

Error detection

Error detecting code:-

→ In digital system a word consisting of a group of bits is stated as an unit and move from one unit to another.

→ It is quite possible this word is transmitted from one location to another or to an arithmetic unit that an error may occur.

→ This may happen due to any reason such as noise, intermediate fault, etc.

For example

1100 may be received as 1000 during transmission.

→ For detecting such errors we use a simple method of parity.

Parity

→ An additional bit which is added with the number or data to detect the errors is known as parity bit.

→ It is of 2 types:-

i) Even parity (Even no. of ones)

ii) Odd parity (Odd no. of ones)

Example :① 1 0 1 1 0 1

Odd parity bit required, $P_{od} = 1$

Even parity bit required, $P_{ev} = 0$

② 1 0 1 1 1 1

$P_{odd} = 0$ (as there is already 5 no. of 1)

$P_{ev} = 1$

→ Parity bits are also known as check bits.

Decimal	BCD	Add even parity	Add odd parity
0	0 0 0 0	0	1
1	0 0 0 1	1	0
2	0 0 1 0	1	0
3	0 0 1 1	0	1
4	0 1 0 0	1	0
5	0 1 0 1	0	1
6	0 1 1 0	0	1
7	0 1 1 1	1	0
8	1 0 0 0	1	0
9	1 0 0 1	0	1

For example:- If we want to transmit the data 1100 by using even parity then the data will be 11000 (even parity) and if odd parity then 11001 (odd parity)

Error correcting code

→ The parity check discussed above can detect an error which has been created during the transmission of digital data that it parity bit indicates only the error exist.

→ It doesn't tell which bit is incorrect nor it corrects the incorrect bit.

→ To overcome this problems there is another method known as hamming code method.

→ It detects an error and indicates which bit has error. Then this error bit can be changed into correct bit. So, this otherwise called as self correcting code.

→ This code is used for correcting a single error of a message of any length. For transmitting four bit message (BCD), we use three check or parity bits and the data is in the format

$$\overset{7}{D_7}\ \overset{6}{D_6}\ \overset{5}{D_5}\ \overset{4}{P_4}\ \overset{3}{D_3}\ \overset{2}{P_2}\ \overset{1}{P_1}$$

Parity bit $= 2^n$ for $n = 0, 1, 2, \ldots$

$2^0 = 1,\ 2^1 = 2,\ 2^2 = 4, \ldots$

→ In the above expression, D represents the data bit and P represents the parity or check bits. D_7 is the MSB.

→ The parity bit P_1, P_2, P_4 are assigned values by making the following three parity relations, each involve four out of 7 bits.

→ The 3 combinations are :-

i) $P_1\ D_3\ D_5\ D_7$

ii) $P_2\ D_3\ D_6\ D_7$

iii) $P_4\ D_5\ D_6\ D_7$

Q Construct the even parity 7 bit hamming code for a word 1011.

Ans [1] [0] [1] [1]

$D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$

i) $P_1\ D_3\ D_5\ D_7$
[1] 1 1 1
$P_1 \rightarrow$ even parity $= 1$

ii) $P_2\ D_3\ D_6\ D_7$
[0] 1 0 1
$P_2 \rightarrow$ even parity $= 0$

iii) $P_4\ D_5\ D_6\ D_7$
[0] 1 0 1
$P_4 \rightarrow$ even parity $= 0$

code → 1 0 1 0 1 0 1

Q A 7 bit hamming code is received as 1000010 what was the code transmitted or what is the correct code by using even parity bit?

If actual P_1 = given P_1 (satisfied) then $x = 0$, otherwise 1

$D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$
1 0 0 0 0 1 0

i) $P_1\ D_3\ D_5\ D_7$ actual $P_1 = 1$ but given $P_1 = 0$ (not satisfied) $x = 1$
1 0 0 0

ii) $P_2\ D_3\ D_6\ D_7$ actual $P_2 = 1$ = given P_2 (satisfied) $y = 0$
1 0 0 1

iii) $P_4\ D_5\ D_6\ D_7$ actual $P_4 = 1$ but given $P_4 = 0$ (not satisfied) $z = 1$
1 0 0 1

1 0 0 0 0 1 0 → corrected data

$D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$

z y x → 1 0 1 → 4 + 1 = 5th bit should be corrected

Q Construct odd parity 7 bit hamming code for a word 1101.

$D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$

i) $P_1\ D_3\ D_5\ D_7$ $\quad P_1 = 1 \quad x = 1$
[1] 1 0 1

ii) $P_2\ D_3\ D_6\ D_7$ $\quad P_2 = 0 \quad y = 0$
[0] 1 1 1

iii) $P_4\ D_5\ D_6\ D_7$ $\quad P_4 = 1 \quad z = 1$
[1] 0 1 1

1 1 0 1 1 0 1 → Code

I.6 Importance of parity bit

→ A parity bit is a check bit, which is added to a block of data for error detection purposes.

→ It is used to validate the integrity of the data.

→ The value of the parity bit is assigned either 0 or 1 in the message block either even or odd depending upon the type of parity.

→ If even parity, then there should be even number of 1's and if odd parity, then there should be odd number of 1's.

→ A parity check is the process that ensures accurate data transmission between nodes during communication.

→ A parity bit is appended to the original data bits to create an even or odd bit number, the number of bits with value one.

→ A parity bit is a bit with a value of 0 or 1, that is added to a block of data for error detection purposes.

→ Parity bits are often used in data transmission to ensure that the data is not corrupted during the transfer process.

Q A 7 bit hamming code coming out of transmission line is 0010100 (even parity). If there is any error, if yes in which data bit and what was the actually transmitted data?

Ans 0010100
$D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$

i) $P_1\ D_3\ D_5\ D_7$ $\quad$ actual $P_1 = 0$ = given P_1 (satisfied)
0 1 1 0 $\quad$ So, $x = 0$

ii) $P_2\ D_3\ D_6\ D_7$ $\quad$ actual $P_2 = 1$ but given $P_2 = 0$ (not satisfied)
1 1 0 0 $\quad$ So, $y = 1$

iii) $P_4\ D_5\ D_6\ D_7$ $\quad$ actual $P_4 = 1$ but given $P_4 = 0$ (not satisfied)
1 1 0 0 $\quad$ So, $z = 1$

z y x
1 1 0

4 2 1 = 4+2 = 6th bit should be corrected

$D_7\ D_6\ D_5\ P_4\ D_3\ P_2\ P_1$
0 1 1 0 1 0 0 → Actually transmitted Data
↳ 6th bit

1.7 Logic Gates : AND, OR, NOT, NAND, NOR and EXOR gates with truth table

→ Logic gates are the fundamental building blocks of digital systems.

→ There are just three basic types of gates - AND, OR and NOT.

→ The interconnection of gates to perform a variety of logical operations is called logic design.

→ Logic gates are electronic circuits because they are made up of a no. of electronic devices & components

→ Inputs and outputs of logic gates can occur only in two levels.

→ These two levels are termed high and low, or true and false, or on and off, or simply 1 and 0.

→ A table which ~~contain~~ lists all the possible combinations of input variables and the corresponding outputs is called a truth table. It shows how the logic circuit output responds to various combinations of logic levels at the inputs.

→ Level logic is a logic in which the voltage levels represent logic 1 and logic 0. Level logic may be positive logic or negative logic.

→ A positive logic system is the one in which the higher of the two voltage levels represents the logic 1 and the lower of the two voltage levels represents the logic 0.

→ A negative logic system is the one in which the lower of the two voltage levels represents the logic 1 and the higher of the two voltage levels represents the logic 0.

→ In transistor-transistor logic (TTL, the most widely used logic family), the voltage levels are +5V and 0V. Logic 1 corresponds to +5V and logic 0 to 0V.

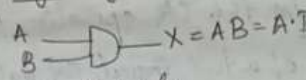
→ Advantages: (i) It can perform complex math calculation (ii) Remember data.

1. AND Gate

→ The output is 1 only when all the inputs are 1. Hence the AND gate is also called an all or nothing gate.

→ The symbol for the AND operation is '.', or we use no symbol at all.

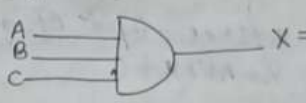
Logic symbol of two input AND gate



Truth table

Inputs		Output
A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

Logic symbol of a three input AND gate



Truth table

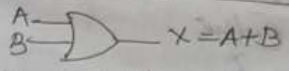
Inputs			Output
A	B	C	X
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

2. OR Gate

→ The output is 1 even if one of its inputs is 1. Hence an OR gate is also called an any or all gate. It can also be called an inclusive OR gate because it includes the condition 'both the inputs can be present'.

→ The symbol for the OR operation is '+'

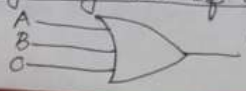
Logic symbol of a two input OR gate



Truth table

Inputs		Output
A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

Logic symbol of a three-input OR gate



Truth Table

Inputs			Output
A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

3. NOT Gate

→ The output is always the complement of its input. So, a NOT gate is also called an inverter and it has only one input and only one output.

→ The symbol for NOT operation is '-' (bar)

Logic symbol of NOT gate

Truth table

Input	Output
A	X
0	1
1	0

*Universal gates (NAND and NOR)

→ Though logic circuits of any complexity can be realized using only the three basic gates (AND, OR & NOT), there are two universal gates (NAND and NOR), each of which can also realize the logic circuits single-handedly. The NAND and NOR gates are therefore called universal building blocks.

→ Both NAND and NOR gates can perform all three basic logic functions, i.e. AND, OR and NOT.

4. NAND Gate

→ NAND means NOT AND i.e. the AND output is NOTed. So, a NAND gate is a combination of an AND gate & a NOT gate.

→ The output is logic 0 level, only when each of the inputs assumes a logic 1 level. For any other combination of inputs, the output is a logic 1 level.

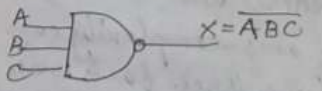
→ The expression for the output of the NAND gate can be written as $X = \overline{ABC...}$

Logic Symbol of two i/p NAND gate

Truth Table

Inputs		Output
A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

Logic Symbol of three i/p NAND gate



Truth Table

Inputs			Output
A	B	C	X
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

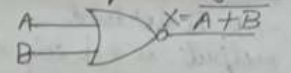
5. NOR Gate

→ NOR means NOT OR i.e. the OR output is NOTed. So, a NOR gate is a combination of an OR gate and a NOT gate.

→ The output is logic 1 level when each one of its inputs assumes a Logic 0 level. For any other combination of inputs, the output is a logic 0 level.

→ The expression for the output of the NOR gate is $X = \overline{A+B+C+...}$

Logic Symbol for a two i/p NOR gate



Truth Table

Inputs		Output
A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

Logic Symbol for a three i/p NOR gate

Truth Table

Inputs			Output
A	B	C	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	0

6. Exclusive-OR (X-OR or EXOR) gate

→ The output assumes a logic 1 when one and only one of its two inputs assumes a logic 1 state.

→ When both the inputs assume the logic 0 state, or when both the inputs assume the logic 1 state, the output assumes a logic 0 state.

→ Since an X-OR gate produces an output 1 only when the inputs are not equal. It is called an anti-coincidence gate or inequality detector.

→ The name exclusive-OR is derived from the fact that its output is a 1 only when exclusively one of its inputs is a 1. (It excludes the condition when both the inputs are 1).

→ The expression for the output of this gate is written as $X = A \oplus B = A\bar{B} + \bar{A}B$.

Logic Symbol of a two i/p EX-OR gate

Truth Table

Inputs		Output
A	B	X = A⊕B
0	0	0
0	1	1
1	0	1
1	1	0

→ In general, an EX-OR gate will give an output value of logic "1" only when there are an "ODD" number of 1's on the inputs to the gate, if the two numbers are equal, the output is "0".

→ When an EX-OR function with more than two inputs is called an "odd function" or modulo-2-sum (Mod-2-sum), not an EX-OR.

Logic Symbol of 3 i/p EX-OR gate

$X = A \oplus B \oplus C$

$\Rightarrow X = A\bar{B}\bar{C} + \bar{A}B\bar{C} + \bar{A}\bar{B}C + ABC$

Truth Table

Inputs			Output
A	B	C	X
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

(Any odd no. of i/ps gives X)

7. Exclusive-NOR (X-NOR) Gate or EX-NOR Gate

→ An XNOR gate is a combination of an X-OR gate and a NOT gate.

→ The output is a 1 only when both the inputs assumes a 0 state or when both the inputs assume a 1 state.

→ The o/p is a 0 ~~or~~ when one of the inputs assumes a 0 state and the other a 1 state.

→ It is also called a coincidence gate, because its output is 1 only when its input coincide.

→ It can be used as an equality detector because its outputs are "1" only when its inputs are equal.

→ The expression for the output of this gate is written as

$$X = A \odot B = AB + \bar{A}\bar{B}$$
$$= \overline{A \oplus B}$$
$$= \overline{A\bar{B} + \bar{A}B}$$

Logic Symbol of two i/p XNOR gate

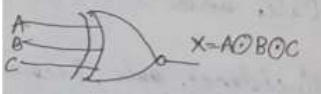
Truth Table

Inputs		Output
A	B	X=A⊙B
0	0	1
0	1	0
1	0	0
1	1	1

→ In general, an EX-NOR gate will give an output value of logic "1" only when there are an even number of 1's on the inputs to the gate (the inverse of the Ex-OR gate) except when all its inputs are low (output is "1").

→ Then an Ex-NOR function with more than two inputs is called an "even function" or modulo-2-sum (Mod-2-SUM), not an EX-NOR.

Logic Symbol of 3 input EX-NOR gate



$$Q = A \odot B \odot C = \overline{A \oplus B \oplus C}$$
$$= \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C + \bar{A}BC$$

Truth Table

Inputs			Output
C	B	A	X
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

(Any even no. of 1's i/p's gives X')

1.8 Realize AND, OR, NOT operations using NAND, NOR gates

→ NAND and NOR gates are called universal gates because they can be used to perform the function of other gates (i.e. AND, OR, NOT, etc.

→ OR, AND and NOT gates are called as basic digital gates.

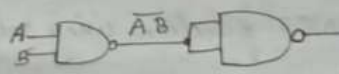
→ A universal gate is a gate which alone can be used to build any logic circuit.

→ So, to show that the NAND gate and the NOR gate are universal gates, we have to show that all the three basic logic gates can be realized using only NAND gates or using only NOR gates.

NAND gate as universal gate

(a) Realization of AND function using only NAND gates.

$$A \cdot B = \overline{A\bar{B}} = \overline{(A\bar{B})}$$

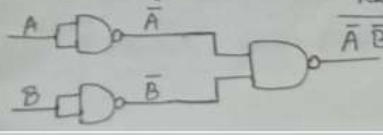


(b) Realization of OR function using only NAND gates.

$$A + B = \overline{\overline{A+B}} = \overline{\bar{A} \cdot \bar{B}}$$ (according to Demorgan's theorem)

Demorgan's theorem: Law 1: $\overline{A+B} = \bar{A} \cdot \bar{B}$

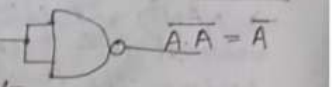
Law 2: $\overline{A \cdot B} = \bar{A} + \bar{B}$



(c) Realization of NOT function using only NAND gates

$\bar{A} = A \cdot A$

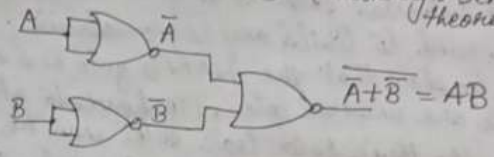
(∵ 1·1 = 1 and 0·0 = 0, so A·A = A)



NOR gate as universal gate :-

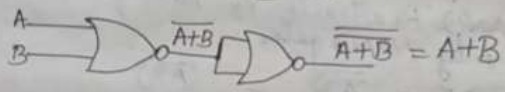
d) Realization of AND function using only NOR gates

$AB = \overline{\overline{AB}} = \overline{\bar{A} + \bar{B}}$ (according to Demorgan's theorem)



e) Realization of OR function using only NOR gates

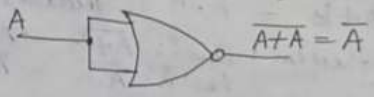
$A + B = \overline{\overline{A+B}} = \overline{(A+B)}$



f) Realization of NOT function using only NOR gate

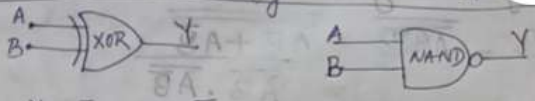
$\bar{A} = \overline{A+A}$

(∵ 0+0 = 0 and 1+1 = 1, so A+A = A)



48 / 98

EX-OR Gate Using 4 NAND Ga



$Y = \bar{A} \cdot B + A \cdot \bar{B} + 0 + 0$

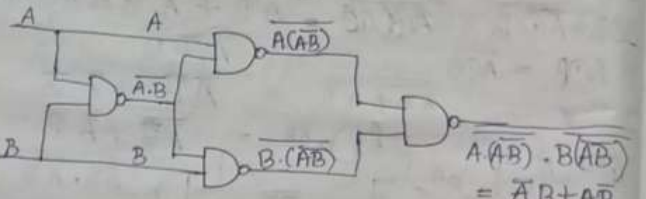
$= \bar{A} \cdot B + A\bar{B} + A\bar{A} + B\bar{B}$ [∵ $A\bar{A} = 0$, $B\bar{B} = 0$]

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

$Y = A \cdot \overline{A \cdot B} + B \cdot \overline{A \cdot B}$ [∵ $\overline{A \cdot B} = \bar{A} + \bar{B}$]

$Y = A\overline{(AB)} + B\overline{(AB)}$ (∵ $\bar{\bar{A}} = A$)

$Y = \overline{A\overline{(AB)} \cdot B\overline{(AB)}}$ [∵ $\overline{A+B} = \bar{A} \cdot \bar{B}$]



$\overline{A\overline{(AB)} \cdot B\overline{(AB)}}$

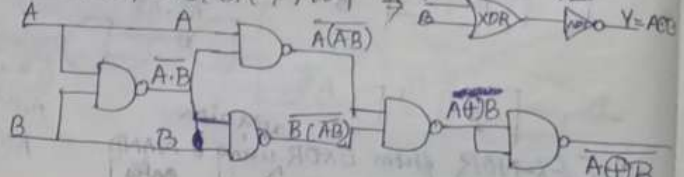
$= \bar{A}B + A\bar{B}$

$= A \oplus B$

EX-NOR Gate from EX-OR expression using 5 NAND gates

$Y = A \cdot B + \bar{A} \cdot \bar{B} = \overline{A \oplus B}$

EXNOR = EXOR + NOT ⇒



EX-OR Gate Using 5 NOR Gates (NOR as XOR gate)

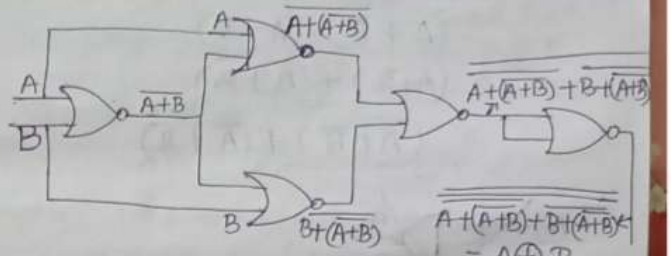
$Y = A\bar{B} + \bar{A}B = A\bar{B} + \bar{A}B + 0 + 0$

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

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

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

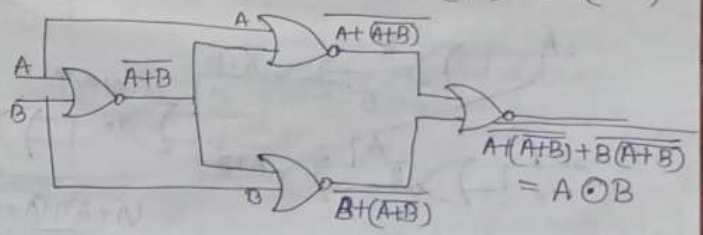
$Y = A + (\overline{A+B}) + B + (\overline{A+B})$



EX-NOR Gate from EX-OR expression using 4 NOR Gates (NOR as XNOR gate)

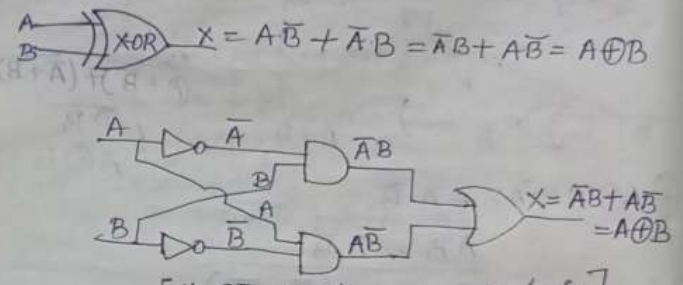
$Y = AB + \bar{A}\bar{B} = \overline{\bar{A}B + A\bar{B}} = \overline{A+(A+B)} + \overline{B+(A+B)}$

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



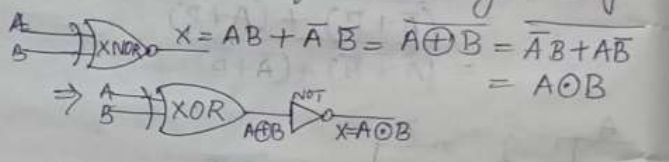
Q1 Realize the X-OR function using AOI

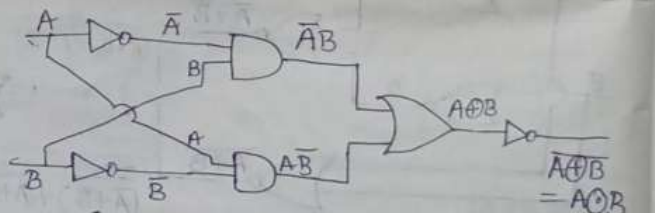
AOI
- AND gate
- OR gate
- NOT gate (INVERTER)



[X-OR function using AOI logic]

Q2 Realize X-NOR function using AOI logic

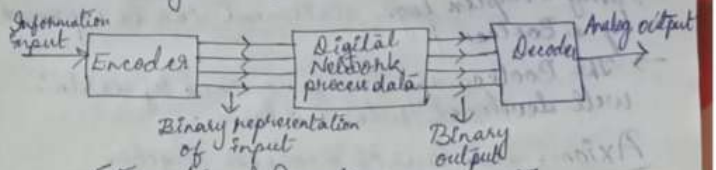




[X-OR function using AOI logic]

1.9 Different postulates and De-Morgan's theorem in Boolean Algebra

Block Diagram of data representation



[Encoding & Decoding process]

Operation

→ The process of giving meaning to a group of bits is called encoding. Encoder converts original signal to binary digits. Numbers and letters are the items most commonly encoded. The reverse process is known as decoding.

→ Information is transformed into binary form using the encoder unit. The digital network uses binary to process the data resulting in a binary output. The o/p passes through the decoder which translates it back into a form that can be recognized by the user.

Boolean Algebra

→ Switching circuits are also called logic circuits, gate circuits and digital circuits.

→ Switching algebra is also called Boolean algebra.

→ Boolean algebra is a system of mathematical logic. It is the basic mathematical tool in the analysis and synthesis of switching circuits. It is a way to express logic functions algebraically.

→ It is an algebraic system consisting of the set of elements (0, 1), two binary operations OR and AND and one unary operation NOT.

→ Any complex logic statement can be expressed by a Boolean function.

→ The Boolean algebra is governed by certain well-developed rules and laws.

Axioms and laws of Boolean Algebra

→ Axioms or postulates of Boolean algebra are a set of logical expressions that we accept without proof and upon which we can build a set of useful theorems.

→ Axioms are nothing more than the definitions of the three basic logic operations that we have already discussed: AND, OR and INVERT. (NOT)

AND operation	OR operation	NOT operation
Axiom 1: 0.0 = 0	Axiom 5: 0+0 = 0	Axiom 9: $\overline{1} = 0$
Axiom 2: 0.1 = 0	Axiom 6: 0+1 = 1	Axiom 10: $\overline{0} = 1$
Axiom 3: 1.0 = 0	Axiom 7: 1+0 = 1	
Axiom 4: 1.1 = 1	Axiom 8: 1+1 = 1	

Complementation Laws

→ The term complement simply means to invert i.e to change 0s to 1s and 1s to 0s.

Law 1: $\overline{0} = 1$

Law 2: $\overline{1} = 0$

Law 3: If $A=0$, then $\bar{A}=1$
Law 4: If $A=1$, then $\bar{A}=0$
Law 5: $\bar{\bar{A}} = A$ (double complementation law)

AND Laws

Law 1: $A \cdot 0 = 0$
Law 2: $A \cdot 1 = A$ (Identity Law)
Law 3: $A \cdot A = A$
Law 4: $A \cdot \bar{A} = 0$

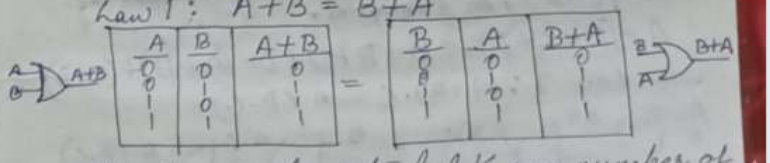
OR Laws

Law 1: $A+0 = A$ (Null Law)
Law 2: $A+1 = 1$ (Identity Law)
Law 3: $A+A = A$
Law 4: $A+\bar{A} = 1$

Commutative Laws

→ This law allow change in position of AND or OR variables. There are two commutative laws.

Law 1: $A+B = B+A$



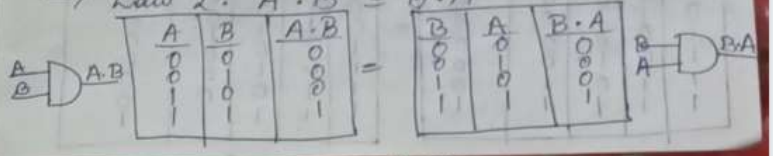
A	B	A+B
0	0	0
0	1	1
1	0	1
1	1	1

=

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

→ This law can be extended to any number of variables. For example,
$A+B+C = B+C+A = C+A+B = B+A+C$

→ Law 2: $A \cdot B = B \cdot A$



A	B	A·B
0	0	0
0	1	0
1	0	0
1	1	1

=

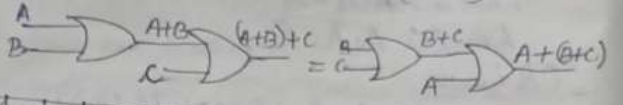
B	A	B·A
0	0	0
0	1	0
1	0	0
1	1	1

→ This law can be extended to any number of variables. For example,
$A \cdot B \cdot C = B \cdot C \cdot A = C \cdot A \cdot B = B \cdot A$

Associative Laws

→ This law allows grouping of variables. There are two associative laws.

Law 1: $(A+B)+C = A+(B+C)$



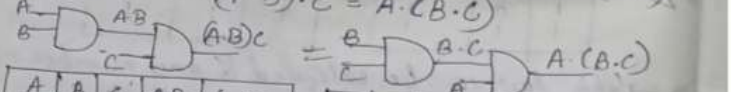
A	B	C	A+B	(A+B)+C
0	0	0	0	0
0	0	1	0	1
0	1	0	1	1
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	1	1
1	1	1	1	1

=

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

→ This law can be extended to any number of variables. For examples,
$A+(B+C+D) = (A+B+C)+D = (A+B)+(C+D)$

→ Law 2: $(A \cdot B) \cdot C = A \cdot (B \cdot C)$



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

=

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

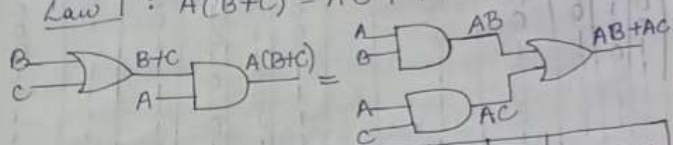
→ This law can be extended to any number of variables. For example,

$A(BCD) = (ABC)D = (AB)(CD)$

Distributive Laws

The distributive laws allow factoring or multiplying out of expressions.

Law 1 : $A(B+C) = AB + AC$



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

=

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

→ This law applies to single variables as well as combinations of variables. For example,

$ABC(D+E) = ABCD + ABCE$

$AB(CD+EF) = ABCD + ABEF$

→ The distributive property is often used in reverse i.e. ~~AB+AC~~ $= A(B+C)$

$ABC + ABD = AB(C+D)$

→ Law 2 : $A + BC = (A+B)\cdot(A+C)$

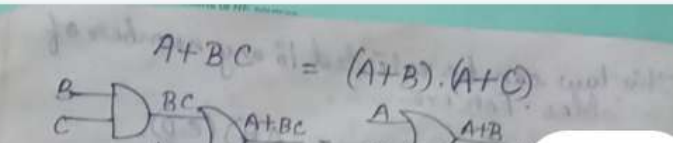
$RHS = (A+B)(A+C) = AA + AC + BA + BC$ [$\because A \cdot A = A$

$= A + AC + AB + BC = A(1+C) + AB + BC$

$= A \cdot 1 + BC$ [$\because 1+C+B = 1+B = 1$

$= A + BC$

$= LHS$

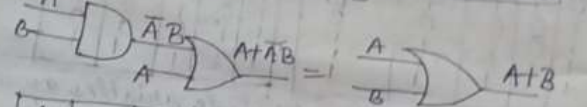


A	B	C	BC	A+BC
0	0	0	0	0
0	0	1	0	0
0	1	0	0	0
0	1	1	1	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

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

Redundant Literal Rule :- Complement of a term appearing in another term is redundant.

Law 1: $A + \bar{A}B = A + B$



A	B	Ā	ĀB	A+ĀB
0	0	1	0	0
0	1	1	1	1
1	0	0	0	1
1	1	0	0	1

=

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

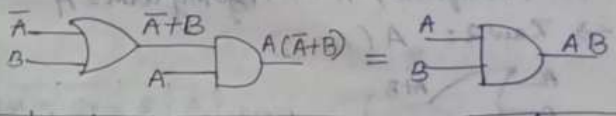
LHS $= A + \bar{A}B$

$= (A+\bar{A})\cdot(A+B)$ [$\because$ Distributive law

$= 1 \cdot (A+B)$

$= A + B = R.H.S$

→ Law 2: $A(\bar{A}+B) = AB$



A	B	Ā	Ā+B	A(Ā+B)
0	0	1	1	0
0	1	1	1	0
1	0	0	0	0
1	1	0	1	1

=

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

L.H.S = $A(\bar{A}+B)$
= $A\bar{A}+AB$ (∵ Distributive Law)
= $0+AB$
= AB

Idempotence Laws :- This law states that ANDing or ORing of a variable with itself is equal to that variable only.

→ Law 1: $A \cdot A = A$

→ Idempotence means the same value.

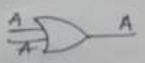
If $A=0$, then $A \cdot A = 0 \cdot 0 = 0 = A$

If $A=1$, then $A \cdot A = 1 \cdot 1 = 1 = A$

→ Law 2: $A+A = A$

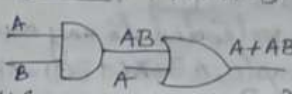
If $A=0$, then $A+A = 0+0 = 0 = A$

If $A=1$, then $A+A = 1+1 = 1 = A$



Absorption Laws

→ Law 1: $A+A \cdot B = A$

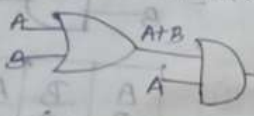


A	B	AB	A+AB
0	0	0	0
0	1	0	0
1	0	0	1
1	1	1	1

L.H.S = $A+A \cdot B = A(1+B)$
$= A \cdot 1$ (∵ $1+B=1$)
$= A$ = R.H.S

→ Therefore, $A + A \cdot$ Any term $= A$

→ Law 2: $A(A+B) = A$



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

Algebraically, we have

L.H.S = $A(A+B) = A \cdot A + AB$ (∵ Distributive law)
$= A + AB$ ($A \cdot A = A$)
$= A(1+B)$ ($1+B=1$)
$= A$ = R.H.S

$A(A+$ Any term$) = A$

Consensus Theorem (Included Factor Theorem)

Theorem 1: $AB + \bar{A}C + BC = AB + \bar{A}C$

Proof: L.H.S $= AB + \bar{A}C + BC = AB + \bar{A}C + BC \cdot 1$
$= AB + \bar{A}C + BC(A+\bar{A})$ (∵ $A+\bar{A}=1$)
$= AB + \bar{A}C + BCA + BC\bar{A}$
$= AB + ABC + \bar{A}C + \bar{A}CB$
$= ABC(1+C) + \bar{A}C(1+B)$
$= AB \cdot 1 + \bar{A}C \cdot 1$ (∵ $1+C=1+B=1$)
$= AB + \bar{A}C$ = R.H.S ∴ L.H.S = R.H.S

→ This theorem can be extended to any number of variables. For example, $AB+\bar{A}C+BCD = AB+\bar{A}C$

L.H.S $= AB + \bar{A}C + BCD$
$= AB + \bar{A}C + BCD(A+\bar{A})$
$= AB + \bar{A}C + ABCD + \bar{A}BCD$
$= AB(1+CD) + \bar{A}C(1+BD)$

$= AB \cdot 1 + \bar{A}C \cdot 1 \quad (\because B+\bar{B}=1,\ C+\bar{C}=1)$

$= AB + \bar{A}C = R.H.S \quad \therefore L.H.S = R.H.S$

Theorem 2: $(A+B)(\bar{A}+C)(B+C) = (A+B)(\bar{A}+C)$

Proof: $L.H.S = (A+B)(\bar{A}+C)(B+C)$

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

$= (0+AC+BC+\bar{A}B)(B+C)$

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

$= ABC + B \cdot BC + \bar{A}BB + AC \cdot C + BC \cdot C + \bar{A}BC$

$(\because B \cdot B = B,\ C \cdot C = C)$

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

$(\because A + A = A)$

$(\text{Say } BC + BC = BC)$

$(\because A + \bar{A} = 1)$

$(\because BC + BC = BC)$

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

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

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

$R.H.S = (A+B)(\bar{A}+C)$

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

$(\because A \cdot \bar{A} = 0)$

$= 0 + AC + BC + \bar{A}B$

$= AC + BC + \bar{A}B = L.H.S$

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

→ This theorem can be extended to a number of variables. For example,

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

$L.H.S = (A+B)(\bar{A}+C)(B+C+D)$

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

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

$= ABC + B \cdot B \cdot C + \bar{A}B \cdot B + ACC + BC \cdot C + \bar{A}BC + ACD + BCD + \bar{A}BD$

$(\because B \cdot B = B,\ C \cdot C = C)$

$= ABC + \bar{A}BC + BCD + BC + BC + \bar{A}B + \bar{A}BD + AC + ACD$

$= ABC(A+\bar{A}) \cdot 0 \cdot 0$

$= ABC + \bar{A}BC + BCD + BC + \bar{A}BC(1+D) + ACC(1+D) \quad (\because BC + BC = BC)$

$= BC(A+\bar{A}+D+1) + \bar{A}B + AC \quad (\because 1+D=1)$

$= BC(1+1) + \bar{A}B + AC$

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

$R.H.S = (A+B)(\bar{A}+C)$

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

$[\because \bar{A} \cdot \bar{A} = 0]$

$= 0 + AC + BC + \bar{A}B$

$= AC + BC + \bar{A}B = L.H.S$

$\therefore L.H.S = R.H.S$

Transposition Theorem

Theorem: $AB + \bar{A}C = (A+C)(\bar{A}+B)$

$R.H.S = (A+C)(\bar{A}+B)$

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

$(\because A\bar{A} = 0)$

$= 0 + \bar{A}C + AB + BC \cdot 1$

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

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

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

$(\because 1+C = 1+B = 1)$

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

$= AB + \bar{A}C = L.H.S$

$\therefore L.H.S = R.H.S$

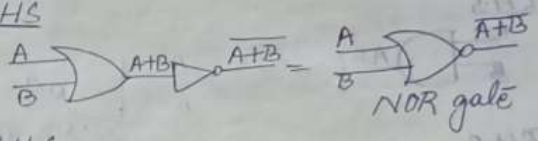
De Morgan's Theorem

→ This theorem presents two of the most powerful laws in Boolean algebra.

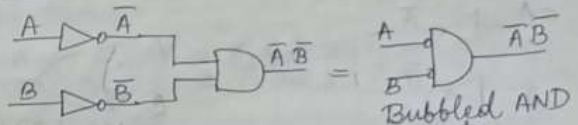
Law 1: $\overline{A+B} = \bar{A}\bar{B}$

→ This law states that the complement of a sum of variables is equal to the product of their individual complements.

LHS



R.H.S



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

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

→ It shows that the NOR gate is equivalent to bubbled AND gate.

→ This law can be extended to any number of variables or combination of variables. For example,

① $\overline{A+B+C+D+\ldots} = \overline{A}\cdot\overline{B}\cdot\overline{C}\cdot\overline{D}\ldots$

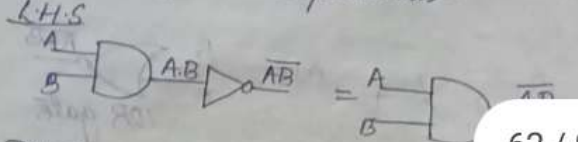
② $\overline{AB+CD+EFG+\ldots} = (\overline{AB})\cdot(\overline{CD})\cdot(\overline{EFG})\ldots$
$= (\bar{A}+\bar{B})(\bar{C}+\bar{D})(\bar{E}+\bar{F}+\bar{G})\ldots$

→ This law permits transformation from a sum-of-products (SOP) form to product-of-sums (POS) form

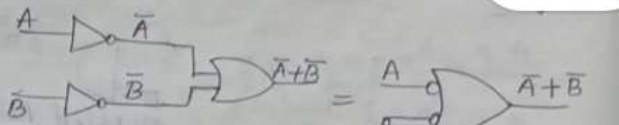
→ Law 2 : $\overline{AB} = \bar{A}+\bar{B}$

→ This law states that the complement of the product of variables is equal to the sum of their individual complements.

LHS



RHS



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

=

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

→ It shows that the NAND gate is equivalent to a bubbled OR gate.

→ This law can be extended to any number of variables or combination of variables. For e.g.

① $\overline{ABCD\ldots} = \bar{A}+\bar{B}+\bar{C}+\bar{D}+\ldots$

② $\overline{(AB)(CD)(EFG)\ldots} = \overline{AB}+\overline{CD}+\overline{EFG}+\ldots$

③ $\overline{(A+B)(C+D)(E+F+G)} = \overline{(A+B)}+\overline{(E+F+G)}+\overline{(C+D)}$
$= \bar{A}\bar{B}+\bar{C}\bar{D}+\bar{E}\bar{F}\bar{G}$

→ This law permits transformation from a product of sums form to a sum-of-products (SOP) form

1.10 Use of Boolean algebra for simplification of logic expression

→ Every Boolean expression must be reduced to as simple a form as possible before realization

→ The techniques used for these reductions are similar to those used in ordinary algebra. The procedure is :-

(a) Multiply all variables necessary to remove parantheses.

(b) Look for identical terms. Only one of those terms be retained and all others dropped. For example, $AB+AB+AB+AB=AB$

(c) Look for a variable and its negation in the same term. This term can be dropped. For example,

$$A\cdot B\bar{B} = A\cdot 0 = 0 \; ; \; ABC\bar{C} = AB\cdot 0 = 0$$

(d) Look for pairs of terms that are identical except for one variable which may be missing in one of the terms. The larger term can be dropped. For example,

$$AB\bar{C}\bar{D} + AB\bar{C} = AB\bar{C}(\bar{D}+1) = AB\bar{C}\cdot 1 = AB\bar{C}$$

(e) Look for pairs of terms which have the same variables, with one or more variables complemented. If a variable in one term of such a pair is complemented while in the second term it is not, then such terms can be combined into a single term with that variable dropped. For example,

$$AB\bar{C}\bar{D} + AB\bar{C}D = AB\bar{C}(\bar{D}+D) = AB\bar{C}\cdot 1 = AB\bar{C}$$

$$AB(C+D) + AB(C+\bar{D}) = AB[(C+D)+(C+\bar{D})] = AB\cdot 1 = AB$$

Example 1

Reduce the expression $f = A[B+\bar{C}(\overline{AB}+A\bar{C})]$.

Solution:

$f = A[B+\bar{C}(\overline{AB}+A\bar{C})]$

$= A[B+\bar{C}(\bar{A}\bar{B}\cdot\overline{A\bar{C}})]$

$[\because \bar{\bar{C}} = C]$ $= A[B+\bar{C}(A+B)(\bar{A}+C)]$

$[\because A\cdot\bar{A}=\bar{A}]$ $= A[B+\bar{C}(\bar{A}A+\bar{A}C+\bar{A}\bar{B}+\bar{B}C)]$

$(\because \bar{A}\cdot A = A)$ $= A[B+\bar{C}\bar{A}+\bar{A}C\cdot C+\bar{A}\bar{B}\bar{C}+\bar{B}C\cdot\bar{C}]$

$(\because C\cdot\bar{C}=0)$ $(\because A\cdot\bar{A}=0)$ $= AB+A\bar{A}\bar{C}+A\bar{A}\bar{B}\bar{C}$

$= AB$

Example 2

Reduce the expression $f = A+B[AC+(B+\bar{C})D]$

Solution:

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

$= A+B[AC+BD+\bar{C}D]$

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

$= AC(1+B)+BD+B\bar{C}D$

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

$= A+BD\cdot 1$

$= A+BD$

Example 3

Reduce the expression $f = \overline{(A+B\bar{C})}(A\bar{B}+ABC)$

Solution:

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

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

$= [\bar{A}(\bar{B}+C)](A\bar{B}+ABC)$

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

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

$= 0+0+0+0 = 0$

Example 4

Reduce the expression $f = (B+BC)(B+\bar{B}C)(B+D)$

Soln: $f = (B+BC)(B+\bar{B}C)(B+D)$

$= (BB + B\bar{B}C + B\cdot BC + B\bar{B}\,C\cdot C)(B+D)$

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

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

$= [B(C+C)](B+D)$

$= B\cdot 1\,(B+D) = B\cdot B + BD$

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

$= B\cdot 1 = B$

Example 5

~~Reduce~~ Show that $AB + A\bar{B}C + B\bar{C} = AC + B\bar{C}$

L.H.S $= AB + A\bar{B}C + B\bar{C}$

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

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

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

$= A\cdot 1\,(B+C) + B\bar{C}$

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

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

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

~~A~~ $AC(B+1) + B\bar{C}(A+1)$

$= AC\cdot 1 + B\bar{C}$

$= AC + B\bar{C} = R.H.S$

$\therefore L.H.S = R.H.S$

Hence proved

Example 6

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

Soln: $A\bar{B}C + B + B\bar{D} + AB\bar{D} + \bar{A}C = B + C$

L.H.S $A\bar{B}C + B + B\bar{D} + AB\bar{D} + \bar{A}C$

$= A\bar{B}C + \bar{A}C + B(1 + \bar{D} + A\bar{D})$

$= A\bar{B}C + \bar{A}C + B\cdot 1$

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

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

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

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

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

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

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

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

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

$= B + C\cdot 1$

$= B + C = R.H.S$

Example 7

Simplify the function

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

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

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

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

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

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

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

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

$= 1.(A+\bar{B}) + \bar{C}$

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

1.11 Karnaugh map for 2, 3, 4 variable, simplification of SOP and POS logic expression using K-map

Canonical Logic form:-

All the boolean expression can be converted into 2 forms.

① Sum-of-products (SOP) form

② Product-of-sums (POS) form

① Sum-of-products (SOP) form:

→ This form is also called the Disjunctive Normal Form (DNF).

→ In SOP, a boolean function is expressed as several products as min terms which are sum or ORed together i.e. SOP is a OR-AND function.

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

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

① $f = AB + A\bar{B} + \bar{A}\bar{B}$

$\bar{A}=0$, $A=1$ → m_3 (2+1), m_2, m_0

$= \Sigma m(0,2,3)$

② Here, $\bar{A}=0$, $A=1$

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

(1+2+1 = 7) (1+0+0 = 4) (1+0+1 = 5)

m_7 m_4 m_5

$= \Sigma m(4,5,7)$

② Product-of-sum (POS) form:

→ This form is also called the Conjunctive Normal Form (CNF).

→ In POS, a boolean function is expressed as max terms which are multiplied or ANDed together i.e. POS is AND-OR function.

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

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

(0+0 = 0) (0+1 = 1) (2+0 = 2)

M_0 M_1 M_2

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

68 / 98

Standard forms of boolean functions/expressions

There are two standard forms :-

① Standard sum-of-products form (SOP) / Canonical SOP

② Standard product-of-sums form (POS) / Canonical POS

① Standard / Canonical SOP form :-

→ This form is also called Disjunctive Canonical form (DCF) or Expanded sum of products form or canonical sum-of-products form.

→ In this form, the function is the sum of a number of product terms where each product term contains all the variables of the function either in complemented or uncomplemented form.

→ If all the variables are present in each (min) term, then the expression is canonical SOP or standard SOP.

Examples

① $f(A,B,C) = \bar{A}B + \bar{B}C$ → SOP form

$= \bar{A}B \cdot 1 + \bar{B}C \cdot 1$

$= \bar{A}B(C+\bar{C}) + \bar{B}C(A+\bar{A})$ [∵ Distributive law]

$= \bar{A}BC + \bar{A}B\bar{C} + ABC + \bar{A}\bar{B}C$ → Standard or Canonical SOP form

$(0+2+1)\ (0+2+0)\ (4+0+1)\ (0+0+1)$ [Here $\bar{A}=0$, $A=1$]

$= m_3 + m_2 + m_5 + m_1$

$= \Sigma m(1,2,3,5)$

② $f(X,Y,Z) = XY + X\bar{Y} + YZ$

$= XY \cdot 1 + X\bar{Y} \cdot 1 + YZ \cdot 1$

$= XY(Z+\bar{Z}) + X\bar{Y}(Z+\bar{Z}) + YZ(X+\bar{X})$

$= XYZ + XY\bar{Z} + X\bar{Y}Z + X\bar{Y}\bar{Z} + XYZ + \bar{X}YZ$

$= XYZ + XY\bar{Z} + X\bar{Y}Z + X\bar{Y}\bar{Z} + \bar{X}YZ$ [∵ A+A=A]

$(4+2+1)\ (4+2+0)\ (4+0+1)\ (4+0+0)\ (0+2+1)$

$= m_4 + m_6 + m_5 + m_4 + m_3$

$= \Sigma m(3,4,5,6,7)$

② Standard / Canonical POS form:-

→ This form is also called Conjunctive Canonical form (CCF) or expanded product-of-sums form or canonical product-of-sums form.

→ In this form, the function is the product of a number of sum terms where each sum term contains all the variables of the function either in complemented or uncomplemented form.

→ If all the variables are present in each max term, then the expression is canonical SOP or standard SOP.

Examples,

① $f(A,B,C) = (\bar{A}+B)(A+B)$ Here $\bar{A}=1$, $A=0$

$= (\bar{A}+B+0)(A+B+0)$

$= (\bar{A}+B+C\cdot\bar{C})(A+B+C\cdot\bar{C})$ [∵ $C\cdot\bar{C}=0$]

$= (\bar{A}+B+C)(\bar{A}+B+\bar{C})(A+B+C)(A+B+\bar{C})$ [∵ Distributive law]

$(4+2+0)\ (4+2+1)\ (0+2+0)\ (0+0+1)$

$= M_6 \cdot M_7 \cdot M_0 \cdot M_1$

$= \pi M(0,1,6,7)$

② $f(X,Y,Z) = (X+Y)(X+\bar{Y})(Y+Z)(X+\bar{Z})$

$= (X+Y+0)(X+\bar{Y}+0)(Y+Z+0)(X+\bar{Z}+0)$

$= (X+Y+Z\cdot\bar{Z})(X+\bar{Y}+Z\cdot\bar{Z})(Y+Z+X\cdot\bar{X})(X+\bar{Z}+Y\cdot\bar{Y})$

$= (X+Y+Z)(X+Y+\bar{Z})(X+\bar{Y}+Z)(X+\bar{Y}+\bar{Z})(X+Y+Z)(\bar{X}+Y+Z)(X+Y+\bar{Z})(X+\bar{Y}+\bar{Z})$

$= (X+Y+Z)(X+Y+\bar{Z})(X+\bar{Y}+Z)(X+\bar{Y}+\bar{Z})(\bar{X}+Y+Z)$ [∵ A·A=A]

$(0+0+0)\ (0+0+1)\ (0+2+0)\ (0+2+1)\ (4+0+0)$

$= M_0 \cdot M_1 \cdot M_2 \cdot M_3 \cdot M_4$

$= \pi M(0,1,2,3,4)$

*NOTE :- SOP & POS are complement of each other.

→ Now we will discuss the truth table of SOP and POS form.

Truth Table for SOP

A	B	C	Y	SOP
0	0	0	$\bar{A}\bar{B}\bar{C}$	m_0
0	0	1	$\bar{A}\bar{B}C$	m_1
0	1	0	$\bar{A}B\bar{C}$	m_2
0	1	1	$\bar{A}BC$	m_3
1	0	0	$A\bar{B}\bar{C}$	m_4
1	0	1	$A\bar{B}C$	m_5
1	1	0	$AB\bar{C}$	m_6
1	1	1	ABC	m_7

Truth Table for POS

A	B	C	Y	POS
0	0	0	$A+B+C$	M_0
0	0	1	$A+B+\bar{C}$	M_1
0	1	0	$A+\bar{B}+C$	M_2
0	1	1	$A+\bar{B}+\bar{C}$	M_3
1	0	0	$\bar{A}+B+C$	M_4
1	0	1	$\bar{A}+B+\bar{C}$	M_5
1	1	0	$\bar{A}+\bar{B}+C$	M_6
1	1	1	$\bar{A}+\bar{B}+\bar{C}$	M_7

Q1 Find the standard SOP of $f = C+\bar{A}B+ABC$

Solu^n $f = C.1.1 + \bar{A}B.1 + ABC$

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

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

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

$= ABC + A\bar{B}\bar{C} + \bar{A}BC + \bar{A}B\bar{C} + A\bar{B}C$ $(4+2+1=7)$ $(4+0+0=4)$ $(0+2+1=3)$ $(0+2+0=2)$ $(4+0+1=5)$

$= m_7 + m_4 + m_3 + m_2 + m_5$

$= \Sigma m(1,2,3,5,7)$

Q2 Find the standard SOP of $F = A+\bar{B}C$ & draw the truth table.

Solu^n $F = A+\bar{B}C$

$= A.1.1 + \bar{B}C.1$

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

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

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

$= ABC + AB\bar{C} + A\bar{B}C + A\bar{B}\bar{C} + \bar{A}BC$ $(4+2+1=7)$ $(4+2+0=6)$ $(4+0+1=5)$ $(4+0+0=4)$ $(0+2+1=3)$

$= m_7 + m_6 + m_5 + m_4 + m_1$

$= \Sigma m(1,4,5,6,7)$

Truth Table for $f(A,B,C) = \Sigma m(1,4,5,6,7)$

Decimal value	A	B	C	$A+\bar{B}C$
0	0	0	0	0
1	0	0	1	1
2	0	1	0	0
3	0	1	1	0
4	1	0	0	1
5	1	0	1	1
6	1	1	0	1
7	1	1	1	1

Q3 If $f = \Sigma m(2,3,4,6)$, then write down the expression

Solu^n $f = m_2 + m_3 + m_4 + m_6$

$= \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + AB\bar{C}$ $(0,1,0)$ $(0,1,1)$ $(1,0,0)$ $(1,1,0)$

	4	2	1
	A	B	C
2→	0	1	0
3→	0	1	1
4→	1	0	0
6→	1	1	0

Q4 If $f = \Sigma m(0, 4, 5, 9, 11, 13)$, then find the SOP expression.

Solu^ $f = m_0 + m_4 + m_5 + m_9 + m_{11} + m_{13}$

$= 0000 + 0100 + 0101 + 1001 + 1011 + 1101$

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

Q5 $F = (A+B)(A+\bar{C})(A+B+\bar{C})$. Convert F into POS or standard POS form.

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

$= (A+B+0)(A+\bar{C}+B\bar{B})(A+B+\bar{C})$

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

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

$(0+0+0 = 0)$ $(0+0+1 = 1)$ $(0+0+1 = 1)$ $(0+2+1 = 3)$ $(0+0+1 = 1)$

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

$= M_0 + M_1 + M_3$

$= \pi M(0, 1, 3)$

Q6 Convert $F = (\bar{A}+B)(\bar{A}+B+C)(\bar{A}+\bar{B}+\bar{C}+D)$ into POS form

Solu^ $F = (\bar{A}+B+0)(\bar{A}+B+C+0)(\bar{A}+\bar{B}+\bar{C}+D)$

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

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

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

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

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

$= M_8 + M_9 + M_{10} + M_{11} + M_{14}$

$= \pi M(8, 9, 10, 11, 14)$

SOP to POS or POS to SOP conversion

→ To convert one canonical form to another, interchange the symbols Σ and π and list those numbers missing from the original form.

Q1 Find the POS of $F(A, B, C) = \Sigma m(2, 4, 6, 7)$

Solu^ $F(A, B, C) = \Sigma m(2, 4, 6, 7)$ → SSOP form

$= \pi M(0, 1, 3, 5)$ → terms which are missing from 0-7

$= M_0 \cdot M_1 \cdot M_3 \cdot M_5$ → SPOS form

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

Q2 Find the SOP of $F(A, B, C) = \pi M(0, 1, 3, 5)$

Solu^ $F(A, B, C) = \pi M(0, 1, 3, 5)$ → ~~SSOP~~ SPOS form

$= \Sigma m(2, 4, 6, 7)$ → ~~SPOS~~ SSOP form

$= m_2 + m_4 + m_6 + m_7$

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

Q3 Convert the SOP to POS: $F = AB\bar{C} + \bar{A}B\bar{C} + AB$

Solu^ $F = AB\bar{C} + \bar{A}B\bar{C} + AB\cdot 1$

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

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

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

$= m_6 + m_1 + m_7$

$= \Sigma m(1, 6, 7)$ → SSOP form

$= \pi M(0, 2, 3, 4, 5)$ → SPOS form

$= M_0 \cdot M_2 \cdot M_3 \cdot M_4 \cdot M_5$

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

Q4 Convert the POS to SOP: $F=(A+B+C)(A+\bar{B}+C)(A+\bar{C})$

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

$=(A+B+C)(A+\bar{B}+C)(A+\bar{C}+0)$

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

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

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

$=M_0\cdot M_2\cdot M_1\cdot M_3$

$=\pi M(0,1,2,3)$ → SPOS form

$=\Sigma m(4,5,6,7)$ → SSOP form

$=m_4+m_5+m_6+m_7$

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

Q5 Expand $\bar{A}+\bar{B}$ to minterms and maxterms.

Soln: The given expression is a two-variable function. It is in SOP form, so we have to convert it into standard/canonical SOP form.

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

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

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

$=m_1+m_0+m_2$

$=\Sigma m(0,1,2)$ → SSOP

$=\pi M(3)$ → SPOS

$=M_3$

$=AB$ → maxterm

$2^2-1=4-1=3$ (max value)

0 1→1, 0 0→0, 1 0→2

∴ $\bar{A}B$, $\bar{A}\bar{B}$, $A\bar{B}$ are minterms and AB is the maxterm.

Q6 Expand $A+B\bar{C}+AB\bar{D}+ABCD$ to minterms and maxterms.

Soln: The given expression is a four-variable function.

$2^4-1=16-1=15$ (max value)

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

$=A(B+\bar{B})(C+\bar{C})(D+\bar{D})+B\bar{C}(A+\bar{A})(D+\bar{D})+AB\bar{D}(C+\bar{C})+ABCD$

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

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

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

$=m_{15}+m_{14}+m_{13}+m_{12}+m_{11}+m_{10}+m_9+m_8+m_4+m_8$

$=\Sigma m(4,5,8,9,10,11,12,13,14,15)$ → SSOP

$=\pi M(0,1,2,3,6,7)$ → SPOS

$=M_0\cdot M_1\cdot M_2\cdot M_3\cdot M_6\cdot M_7$

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

Q7 Expand $A(\bar{B}+A)B$ to maxterms and minterms

Soln The given expression is a two variable func and is in the POS form.

$(A+0)(A+\bar{B})(B+0)$

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

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

$= (A+B)(A+\bar{B})(\bar{A}+B)$
$\quad 0\ 0 \quad 0\ 1 \quad 1\ 0$

$= M_0 \cdot M_1 \cdot M_2$

$= \pi M(0,1,2) \rightarrow$ SPOS form

$= \Sigma m(3) \rightarrow$ SSOP form

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

$= A \cdot B$

Karnaugh map (K-map) :-

→ The simplification of boolean expression using the theorems of boolean algebra is very inconvinient because of lack of specific rules

→ The final simplified expression is not granted that it is the most simplified form.

→ It is also a time consuming process, so to overcome this problem the K-map method is used.

→ In this representation we put the truth table in a compact form by labelling the rows and columns of a map. This is very useful in the minimization of functions of 2, 3, 4, 5 or 6 variables.

→ The rows and columns are assigned a binary code such that two adjacent rows or columns differ in one bit only.

→ The column on the extreme left is adjacent to the column on the extreme right.

→ Likewise, the top row and bottom row are adjacent.

→ The Karnaugh map consists of a no. of squares. Each one of the squares represents a minterm or a maxterm. Each square is called a cell.

Advantages

→ Easy and convenient to implement

→ Data representation is simple

→ Reduces the cost and quantity of logical gates.

→ Less no. of steps when compared to algebraic minimization technique.

→ Does not demand for the knowledge of boolean algebraic theorems.

Disadvantages

→ Complexity of K-map simplification process increases with the increase in the number of variables.

→ It is limited to 5 or 6 variables.

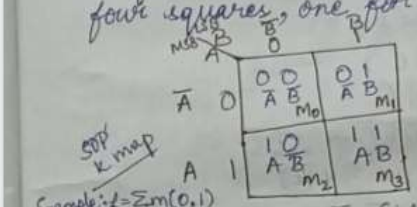
Format of the map :-

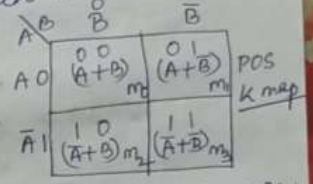
→ The map is a diagram consist of 2^n number of squares for n variables with each square represent 1 min term or max term of the func.

→ The squares are arranged with gray codes sequence so that only 1 variable change between that two adjacent squares.

Two variable k-map :-

→ Let the two variables be A and B. So, there are four min terms hence the map consist of four squares, one for each min term.





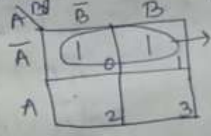
Example: $f = \Sigma m(0,1)$

$= m_0 + m_1 = \bar{A}B + \bar{A}\bar{B} = \bar{A}$ (always take the common term)

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

$= \bar{A} \cdot 1$

$= \bar{A}$

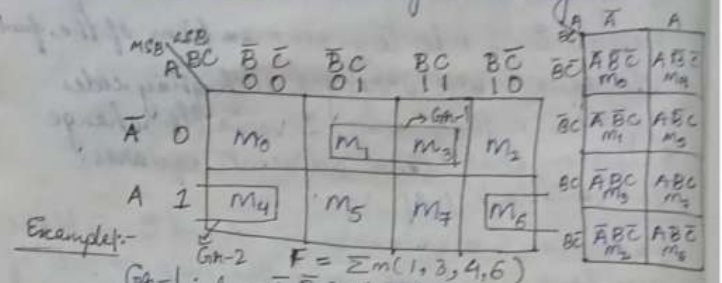


Three variable k-map

→ There are 8 min terms. Let three variables be A, B, C, therefore there are 8 ~~states~~ cells (2^3) in the three variable k-map.

→ In 3-variable k-map, the min term in the adjacent cell only has one literal difference because the shells are arranged with gray code in state of binary sequence.

→ Each cell in three variable k-map contain 3 adjacent neighbours. Generally in k-map, N variable contain N adjacent neighbours.



Example:- $F = \Sigma m(1,3,4,6)$

Grp-1: $f_1 = \bar{A}\bar{B}C + \bar{A}BC$

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

$= \bar{A}C \cdot 1$

$= \bar{A}C$

Grp-2: $f_2 = A\bar{B}\bar{C} + AB\bar{C}$

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

$= A\bar{C}$

$\therefore F = f_1 + f_2$

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

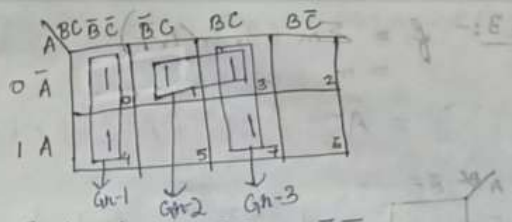
~~$= (\bar{A}C + A)(\bar{A}C + \bar{C})$~~

~~$= \bar{A} \cdot C + \bar{A}$~~

Example 2:- Let the function $f = \bar{A}\bar{B}\bar{C} + \bar{A}BC + ABC + A\bar{B}\bar{C} + \bar{A}\bar{B}C$

$= m_0 + m_3 + m_7 + m_4 + m_1$

$= \Sigma m(0,1,3,4,7)$



Gr-1 : $f_1 = \bar{A}\bar{B}\bar{C} + A\bar{B}\bar{C}$

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

Gr-2 : $f_2 = \bar{A}\bar{B}C + \bar{A}BC$

$= \bar{A}C$

Gr-3 : $f_3 = \bar{A}BC + ABC$

$= BC$

$\therefore f = f_1 + f_2 + f_3 = \bar{B}\bar{C} + \bar{A}C + BC$

Grouping :-

→ **Pair :-** A group of two adjacent cells in a K-map is a pair.

→ **Quad :-** A group of 4 adjacent cells in a k-map is a quad.

→ **Octet :-** A group of 8 adjacent cells in a k-map is an octet.

→ Then get the simplified value for each group which gives the following information :-

i) Group of 2 adjacent square cells will eliminate one variable.

ii) Group of 4 adjacent cells will eliminate two variables.

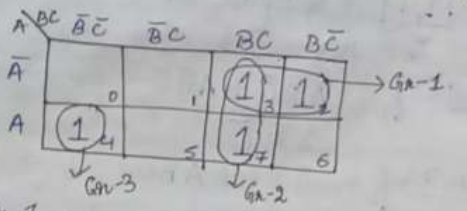
iii) Group of 8 adjacent cells will eliminate three variables.

* So, 2^n adjacent cells will eliminate n variables.

Example 3 :- $f = \Sigma m(3, 2, 4, 7)$

$= m_3 + m_2 + m_4 + m_7$

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



Gr-1 : $f_1 = \bar{A}B$ (Always take the common terms)

Gr-2 : $f_2 = BC$

Gr-3 : $f_3 = A\bar{B}\bar{C}$

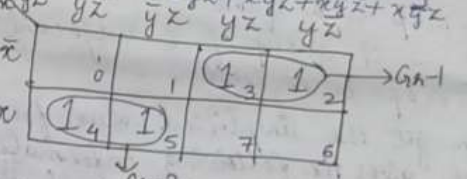
$f = f_1 + f_2 + f_3 = \bar{A}B + BC + A\bar{B}\bar{C}$

Example 4 : Solve using k-map : $f(x,y,z) = \Sigma m(2,3,4,5)$

$f(x,y,z) = \Sigma m(2,3,4,5)$

$= m_2 + m_3 + m_4 + m_5$

$= x\bar{y}z + \bar{x}yz + x\bar{y}\bar{z} + x\bar{y}z$



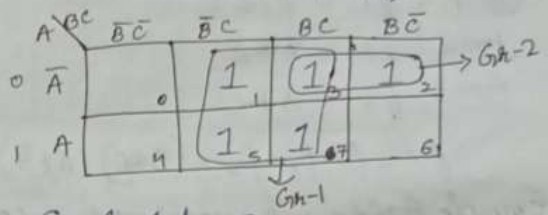
Gr-1 : $f_1 = \bar{x}y$

Gr-2 : $f_2 = x\bar{y}$

$f(x,y,z) = \bar{x}y + x\bar{y}$

Example 5: Solve using k-map: $f(A,B,C) = \bar{A}C + \bar{A}B + ABC + BC$

$$f(A,B,C) = \bar{A}C \cdot 1 + \bar{A}B \cdot 1 + A\bar{B}C + BC \cdot 1$$
$$= \bar{A}C(B+\bar{B}) + \bar{A}B(C+\bar{C}) + A\bar{B}C + BC(A+\bar{A})$$
$$= \bar{A}BC + \bar{A}B\bar{C} + \bar{A}\bar{B}C + \bar{A}BC + A\bar{B}C + ABC + ABC$$
$$= \underset{011}{\bar{A}BC} + \underset{010}{\bar{A}B\bar{C}} + \underset{001}{\bar{A}\bar{B}C} + \underset{011}{\bar{A}BC} + \underset{101}{A\bar{B}C} + \underset{111}{ABC}$$
$$\therefore m_3 + m_2 + m_1 + m_5 + m_7$$
$$= \Sigma m(1,2,3,5,7)$$

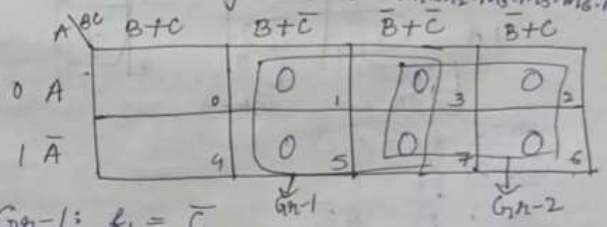


Gr-1 : $f_1 = C$

Gr-2 : $f_2 = \bar{A}B$

$\therefore F = f_1 + f_2 = \bar{A}B + C.$

Example 6: Solve using k-map: $f = \pi M(1,2,3,5,6,7)$
$= M_1 \cdot M_2 \cdot M_3 \cdot M_5 \cdot M_6 \cdot M_7$

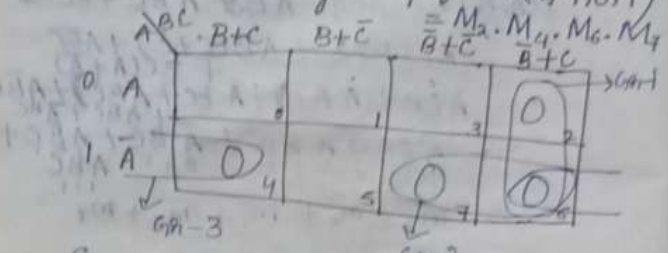


Gr-1: $f_1 = \bar{C}$

Gr-2: $f_2 = \bar{B}$

$\therefore f = f_1 \cdot f_2 = \bar{C} \cdot \bar{B} = (\bar{C})(\bar{B})$

Example 7: Solve using k-map: $f = \pi M(2,4,6,7)$
$= M_2 \cdot M_4 \cdot M_6 \cdot M_7$



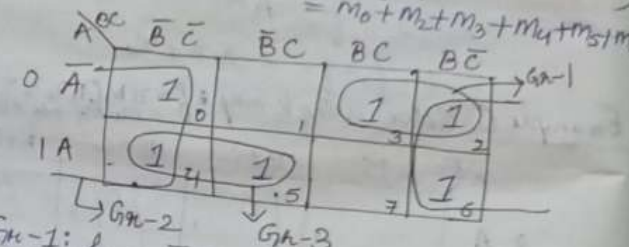
Gr-1 : $f_1 = B + C$

Gr-2 : $f_2 = \bar{A} + \bar{B}$

Gr-3 : $f_3 = \bar{A} + C$

$\therefore f = f_1 \cdot f_2 \cdot f_3 = (\bar{B}+C)(\bar{A}+\bar{B})(\bar{A}+C)$

Example 8: Solve using mapping: $f = \Sigma m(0,2,3,4,5,6)$
$= m_0 + m_2 + m_3 + m_4 + m_5 + m_6$



Gr-1: $f_1 = \bar{A}B$

Gr-2: $f_2 = \bar{C}$

Gr-3: $f_3 = A\bar{B}$

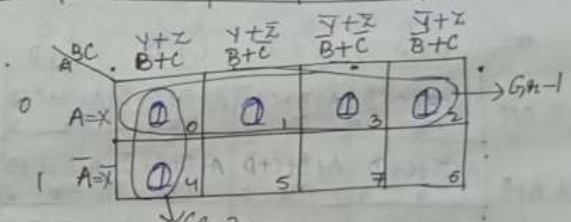
$f = f_1 + f_2 + f_3 = \bar{A}B + \bar{C} + A\bar{B}$

Example 9: Show the truth table of $f(X,Y,Z) = XY + XZ$. Find its simplest POS form using k-map.

Solution: Truth Table

	X	Y	Z	XY	XZ	XY+XZ
0	0	0	0	0	0	0
1	0	0	1	0	0	0
2	0	1	0	0	0	0
3	0	1	1	0	0	0
4	1	0	0	0	0	0
5	1	0	1	0	1	1
6	1	1	0	1	0	1
7	1	1	1	1	1	1

$= \pi M(0,1,2,3,4,6)$



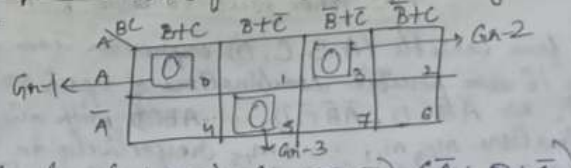
Gp-1: $f_1 = A = X$

Gp-2: $f_2 = B+C = (Y+Z)$

$\therefore f = f_1 \cdot f_2 = X(Y+Z)$

Example 10: Solve $f = \pi M(0,3,5)$ using k-map.

Solution:



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

4-Variable K-Map

AB \ CD	00 $\bar{C}\bar{D}$	01 $\bar{C}D$	11 CD	10 $C\bar{D}$
00 $\bar{A}\bar{B}$	$\bar{A}\bar{B}\bar{C}\bar{D}$ m_0 0	$\bar{A}\bar{B}\bar{C}D$ m_1 1	$\bar{A}\bar{B}CD$ m_3 3	$\bar{A}\bar{B}C\bar{D}$ m_2 2
01 $\bar{A}B$	$\bar{A}B\bar{C}\bar{D}$ m_4 4	$\bar{A}B\bar{C}D$ m_5 5	$\bar{A}BCD$ m_7 7	$\bar{A}BC\bar{D}$ m_6 6
11 AB	$AB\bar{C}\bar{D}$ m_{12} 12	$AB\bar{C}D$ m_{13} 13	$ABCD$ m_{15} 15	$ABC\bar{D}$ m_{14} 14
10 $A\bar{B}$	$A\bar{B}\bar{C}\bar{D}$ m_8 8	$A\bar{B}\bar{C}D$ m_9 9	$A\bar{B}CD$ m_{11} 11	$A\bar{B}C\bar{D}$ m_{10} 10

SOP form four-variable k-map

AB \ CD	00 $C+D$	01 $C+\bar{D}$	11 $\bar{C}+\bar{D}$	10 $\bar{C}+D$
00 $A+B$	$A+B+C+D$ M_0 0	$A+B+C+\bar{D}$ M_1 1	$A+B+\bar{C}+\bar{D}$ M_3 3	$A+B+\bar{C}+D$ M_2 2
01 $A+\bar{B}$	$A+\bar{B}+C+D$ M_4 4	$A+\bar{B}+C+\bar{D}$ M_5 5	$A+\bar{B}+\bar{C}+\bar{D}$ M_7 7	$A+\bar{B}+\bar{C}+D$ M_6 6
11 $\bar{A}+\bar{B}$	$\bar{A}+\bar{B}+C+D$ M_{12} 12	$\bar{A}+\bar{B}+C+\bar{D}$ M_{13} 13	$\bar{A}+\bar{B}+\bar{C}+\bar{D}$ M_{15} 15	$\bar{A}+\bar{B}+\bar{C}+D$ M_{14} 14
10 $\bar{A}+B$	$\bar{A}+B+C+D$ M_8 8	$\bar{A}+B+C+\bar{D}$ M_9 9	$\bar{A}+B+\bar{C}+\bar{D}$ M_{11} 11	$\bar{A}+B+\bar{C}+D$ M_{10} 10

POS form four-variable k-map

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

respectively in POS form.

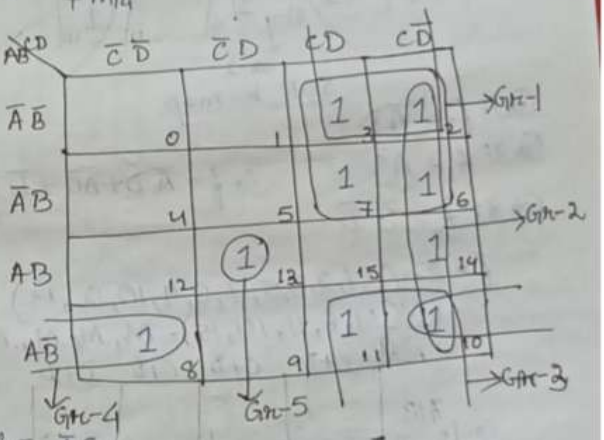
→ A four variable k map has $2^4 = 16$ squares or cells and each square on the map represents either a minterm or a maxterm.

→ The binary number designations of the rows and columns are in the gray code. Here 11 follows 01 and 10 follows 11. This is called adjacency ordering.

Example-1: Reduce using mapping the expression:

$f = \Sigma m(2, 3, 6, 7, 8, 10, 11, 13, 14)$

Solu^n $f = m_2 + m_3 + m_6 + m_7 + m_8 + m_{10} + m_{11} + m_{13} + m_{14}$



Gr1: $f_1 = \bar{A}C$

Gr2: $f_2 = C\bar{D}$

Gr3: $f_3 = \bar{B}C$

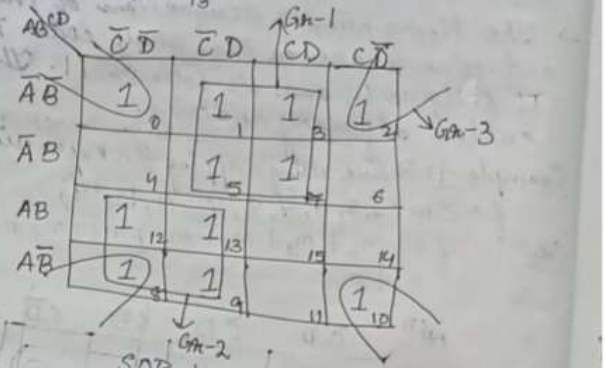
Gr4: $f_4 = A\bar{B}\bar{D}$

Gr5: $f_5 = AB\bar{C}D$

$\therefore f = \bar{A}C + C\bar{D} + \bar{B}C + A\bar{B}\bar{D} + AB\bar{C}D$

Example 2: Find the SOP k-map and POS k-map of the expression: $f = \Sigma m(0, 1, 2, 3, 5, 7, 8, 9, 10, 12, 13)$. Implement the real minimal expression in universal logic.

Solu^n: $f = m_0 + m_1 + m_2 + m_3 + m_5 + m_7 + m_8 + m_9 + m_{10} + m_{12} + m_{13}$



SOP k-map

Gr1: $f_1 = \bar{A}\bar{D}$

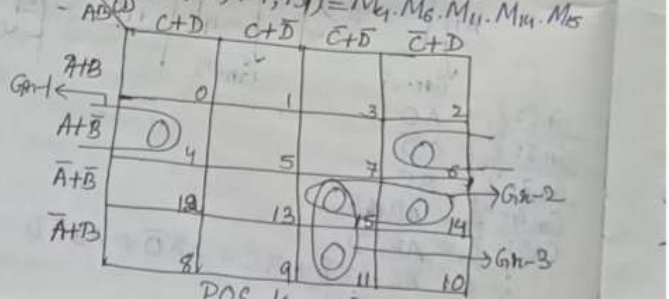
Gr2: $f_2 = A\bar{C}$

Gr3: $f_3 = \bar{B}\bar{D}$

$\therefore f = \bar{A}\bar{D} + A\bar{C} + \bar{B}\bar{D}$

$f = \Sigma m(0, 1, 2, 3, 5, 7, 8, 9, 10, 12, 13)$

$= \pi M(4, 6, 11, 14, 15) = M_4 \cdot M_6 \cdot M_{11} \cdot M_{14} \cdot M_{15}$



POS

Here, minimal expression is $f = \bar{A}D + A\bar{C} + B\bar{D}$

Gate 1: $f_1 = A + \bar{B} + D$

Gate 2: $f_2 = \bar{A} + B + \bar{C}$

Gate 3: $f_3 = \bar{A} + \bar{C} + \bar{D}$

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

$f = \overline{\bar{A}D} \cdot \overline{A\bar{C}} \cdot \overline{B\bar{D}}$

Example 3: Find the POS k-map and SOP k-map of the expression: $f = \pi M(2,8,9,10,11,12,14)$ and implement the minimal expression using universal logic.

Solu^n:- $f = M_2 \cdot M_8 \cdot M_9 \cdot M_{10} \cdot M_{11} \cdot M_{12} \cdot M_{14}$

POS K-map

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

Gr-3 → Gr-2

Gr-1: $f_1 = \bar{A}+D$

Gr-2: $f_2 = \bar{A}+B$

Gr-3: $f_3 = B+\bar{C}+D$

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

$f = \pi M(2,8,9,10,11,12,14) = \Sigma m(0,1,3,4,5,6,7,13,15)$

$f = m_0+m_1+m_3+m_4+m_5+m_6+m_7+m_{13}+m_{15}$

SOP K-map

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

Gr-1 ←

Gr-1: $f_1 = \bar{A}\bar{C}$

Gr-2: $f_2 = \bar{A}D$

Gr-3: $f_3 = \bar{A}B$

Gr-4: $f_4 = B\bar{D}$

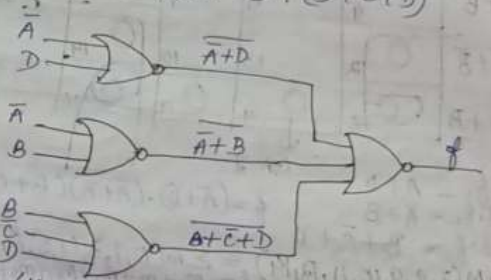
$f = \bar{A}\bar{C} + \bar{A}D + \bar{A}B + B\bar{D}$

Here the minimal expression is obtained from the POS k-map as it contains the minimum number of terms.

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

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

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



Example 4: Minimize the function:-

$f(A,B,C,D) = \Sigma m(12,13,15,14,8,9,10,11,2)$

Solu^n: $f = \Sigma m(2,8,9,10,11,12,13,14,15)$

AB \ $\bar{C}\bar{D}$	$\bar{C}\bar{D}$	$\bar{C}$D	CD	C$\bar{D}$
A$\bar{B}$	0	1	3	1 (2) → Gr-2
$\bar{A}\bar{B}$	4	5	7	6
AB	1 (12)	1 (13)	1 (15)	1 (14) → Gr-1
A$\bar{B}$	1 (8)	1 (9)	1 (11)	1 (10)

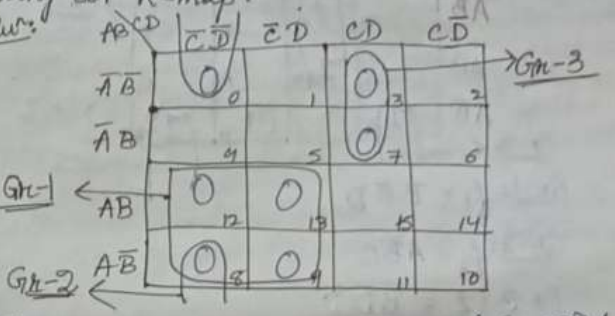
Gr-1: $f_1 = A$

Gr-2: $f_2 = \bar{B}C\bar{D}$

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

Example 5: Minimize the function $f = \pi M(0,3,7,8,9,12,13)$ using SOP k-map.

Solu:-



* Plot the 0's (as 1's are plotted in SOP form)
→ Make the squares with adjacent 0's as groups.
→ Then get the result in SOP form.
→ Then convert SOP form to POS form by taking the complement i.e. by using De-Morgan's law.

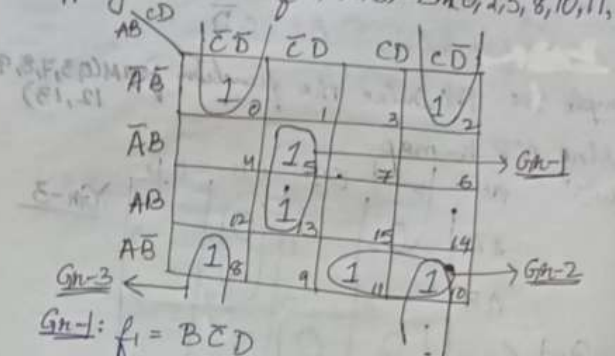
Gr-1: $f_1 = A\bar{C}$ (SOP form)
In POS form:- $f_1 = \overline{A\bar{C}} = \bar{A} + \bar{\bar{C}} = \bar{A} + C$

Gr-2: $f_2 = \bar{B}\bar{C}\bar{D}$ (SOP form)
In POS form:- $f_2 = \overline{\bar{B}\bar{C}\bar{D}} = B + C + D$

Gr-3: $f_3 = \bar{A}CD$ (SOP form)
In POS form:- $f_3 = \overline{\bar{A}CD} = A + \bar{C} + \bar{D}$

$\therefore$ In POS form: $f = (\bar{A}+C)(B+C+D)(A+\bar{C}+\bar{D})$

Example 6: Minimize the function using mapping method: $f(A,B,C,D) = \Sigma m(0,2,5,8,10,11,13)$



Gr-1: $f_1 = B\bar{C}D$

Gr-2: $f_2 = A\bar{B}C$

Gr-3: $f_3 = \bar{B}\bar{D}$

$\therefore f = B\bar{C}D + A\bar{B}C + \bar{B}\bar{D}$

Don't care combinations / Don't care condition:-

→ For some input combinations, output is '1' and for remaining combinations, output is '0'.
→ Sometimes some input combination may not be present that is known as don't care condition. For e.g., for a practical value, the output is 'zero' or 'one' (both are possible) i.e. don't care condition means it is denoted by 'x' or 'd' where 'd' is '0' or '1'.
→ The combinations for which the values of the expression are not specified are called don't care combinations / conditions or optional combinations, therefore, stand incompletely specified.

→ The output is a don't care for these invalid combinations.

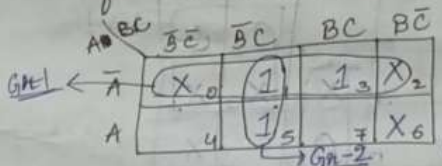
* → A standard SOP expression with don't cares can be converted into a standard POS form by keeping the don't cares as they are, and writing the missing minterms of the SOP form as the maxterms of the POS form.

* → Similarly, to convert a POS expression with don't cares into an SOP expression, keep the don't cares of the POS expression as they are and write the missing maxterms of the POS expression as the minterms of the SOP expression.

→ Don't care condition used in further simplification of logic expression in minimized form.

Example 1:- Reduce the expression:

$f = \Sigma m(1,3,5) + d(0,2,6)$



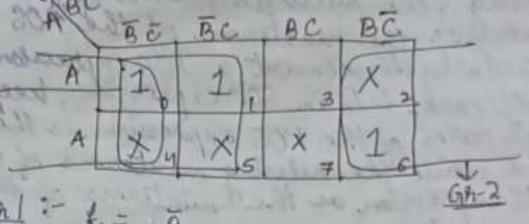
∴ We can't make group with only don't care and don't care. ~~If there~~ Situations by which ~~don't~~ don't care

Gr-1 : $f_1 = \bar{A}$

Gr-2 : $f_2 = \bar{B}C$ ∴ $f = \bar{A} + \bar{B}C$

Point to remember :- While designing k-map using SOP form, don't care condition (X) are considered as 1, if it helps form the largest group, otherwise it is considered as 0 ~~are~~ and are left during encircling. ⇒ On the contrary, while designing a k-map using POS form, don't care conditions (X) are considered as a 0, if it helps form the largest group, otherwise it is considered as 1 and are left during encircling.

Example 2:- Minimize the function using mapping: $f = \Sigma m(0,1,6) + d(4,5,2,7)$

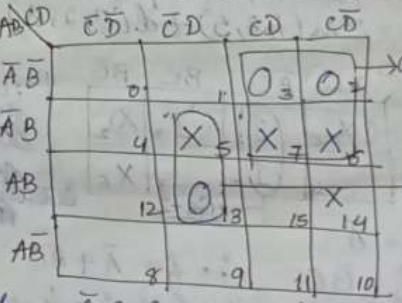


Gr 1 :- $f_1 = \bar{B}$

Gr 2 :- $f_2 = \bar{C}$ $f = \bar{B}\bar{C}$

Example 3 :- Minimize the function using k-map:

$f(A,B,C,D) = \pi M(2,3,13) + d(5,6,7,14)$



Gr 1 : $f_1 = \bar{A}C$ (SOP form)

POS form : $f_1 = \bar{A}C = A + \bar{C}$

Gr 2 : $f_2 = B\bar{C}D$ (SOP form)

POS form : $f_2 = \overline{B\bar{C}D} = \bar{B} + C + \bar{D}$

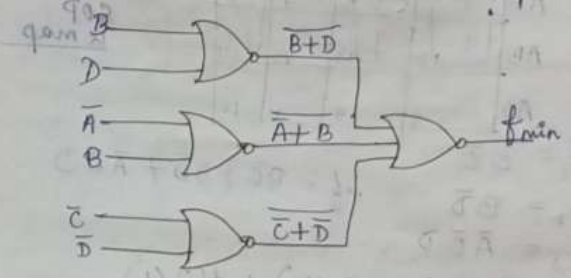
$f = (A + \bar{C})(\bar{B} + C + \bar{D})$

$$\therefore f_{min} = \overline{(B+D)(\bar{A}+B)(\bar{C}+\bar{D})}$$

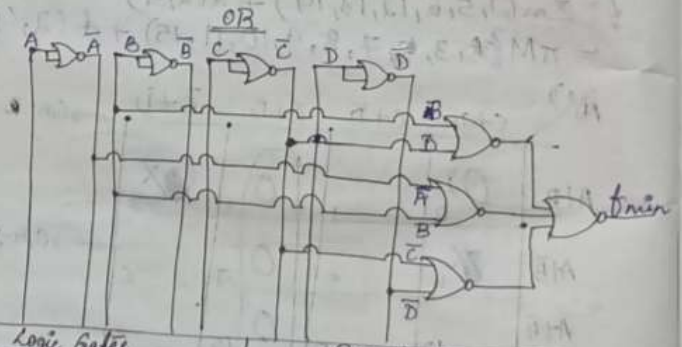
$$= \overline{(B+D)}+\overline{(\bar{A}+B)}+\overline{(\bar{C}+\bar{D})}$$

$$f_{min} = \overline{(B+D)}+\overline{(\bar{A}+B)}+\overline{(\bar{C}+\bar{D})}$$

NOR Logic



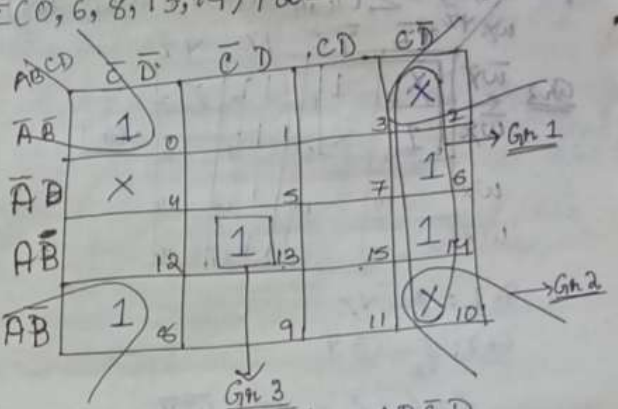
OR



Logic Gates	IC Number (two i/p)	(three i/p)	(four i/p)	(eight i/p)
1) AND gate	IC 7408	IC 7411	IC 7421	
2) OR gate	IC 7432			
3) NOT gate	IC 7404			
4) NAND gate	IC 7400	IC 7410	IC 7420	IC 7430
5) NOR gate	IC 7402	IC 7427	IC 7425	
6) EXOR gate	IC 7486			
7) EXNOR gate	IC 4077			

Example 5: Simplify the boolean function (A,B,C,D)

$= \Sigma(0,6,8,13,14) + d\Sigma(2,4,10)$

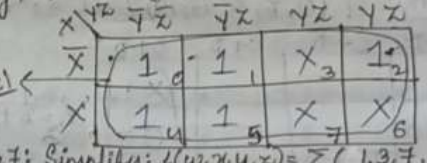


$f_1 = C\bar{D}$, $f_2 = \bar{B}\bar{D}$, $f_3 = AB\bar{C}D$

$f = C\bar{D} + \bar{B}\bar{D} + AB\bar{C}D$

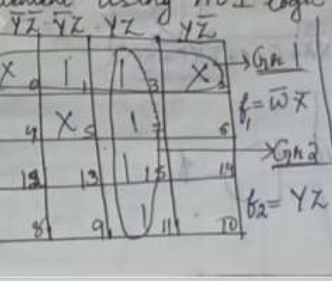
Example 6: Simplify the boolean expression:

$f(x,y,z) = \Sigma(0,1,2,4,5) + d(3,6,7)$



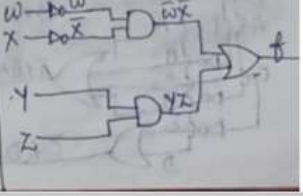
$\therefore f = 1$

Example 7: Simplify: $f(w,x,y,z) = \Sigma(1,3,7,11,15) + d(0,2,5)$ and implement using AOI logic



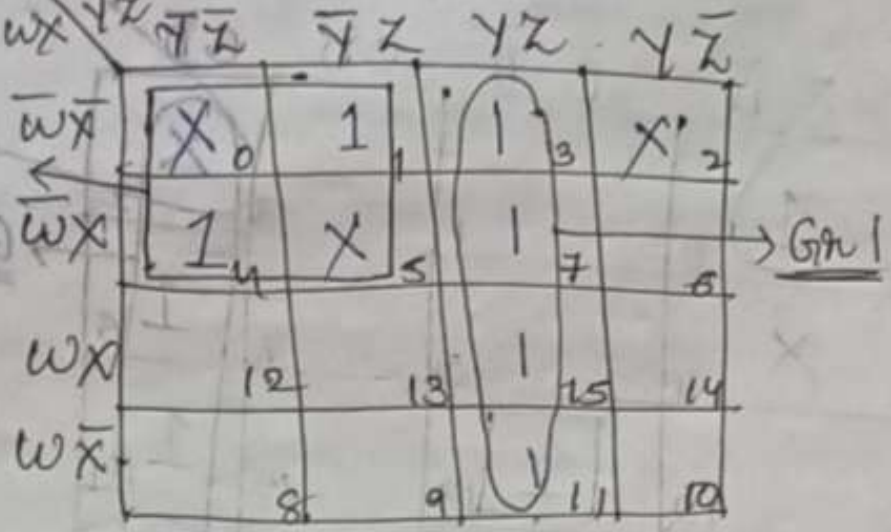
$\therefore f = \bar{w}\bar{x} + YZ$

AOI logic Implementation



Example 8: Simplify the boolean function:

$f(W,X,Y,Z) = \sum(1,3,4,7,11,15) + d(0,2,5)$



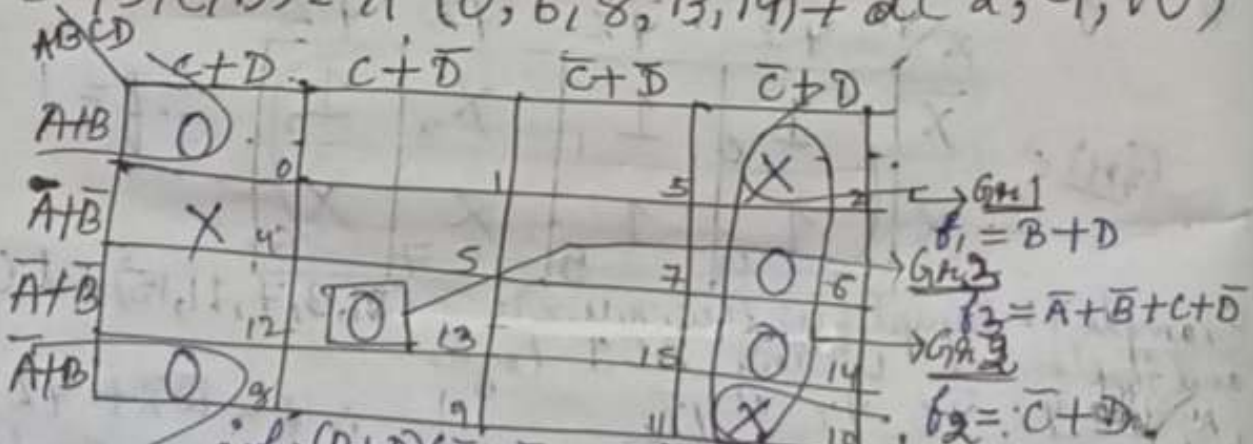
Gr 1: $f_1 = YZ$

Gr 2: $f_2 = \bar{W}\bar{Y}$

$f = f_1 + f_2 = YZ + \bar{W}\bar{Y}$

Example 9: Simplify the boolean function and implement using AOI logic:

$f(A,B,C,D) = \pi(0,6,8,13,14) + d(2,4,10)$



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