MODULE 1

Module I (11 Hours)

Representation of signed numbers – 1's complement and 2's complement ,Logic gates – AND - OR – NOT - NAND- NOR - XOR , Boolean algebra - Basic laws and theorems , Boolean functions - truth table, Standard forms of Boolean Expressions – Sum of Products and Product of Sums - minimization of Boolean function using Karnaugh map method - Realization using logic gates, Floating point numbers

Combinational Circuits - Half adder - Full Adder- Decoder -Encoder- Multiplexer – Demultiplexer

REPRESENTATION OF SIGNED NUMBERS

There are three basic ways to represent signed numbers:

- Sign-magnitude.
- 1's complement.
- 2's complement.

Sign-Magnitude

- The number consists of two parts:
- the MSB (most significant bit) represents the sign
- The other bits represent the magnitude of the number.
- If the sign bit is 1 the number is negative and if it is 0 the number is positive

Examples

-30 = 1 0011110 (The leftmost 1 indicates that the number is negative. The remaining 7-bits carry the magnitude of 30).

30=0 0011110 (The only difference between–30 and +30 is the sign bit because the magnitude bits are similar in both numbers.).

-121= 1 1111001

99=0 1100011

1's Complement

- Negative numbers are represented in 1's complement format.

- positive numbers are represented as the positive sign-magnitude numbers Examples

- 30 = 00011110
- -30 = 11100001
- the number equals the 1's complement of 30
- 121 = 01111001
- -121 = 10000110
- the number equals the 1's complement of 121
- 99 = 01100011

2's Complement

- The two's complement of a binary integer is the 1's complement of the number plus 1."
- Thus if m is the 2's complement of n, then: m = n + 1.

Examples:

n = 0101 0100, then m = 1010 1011 + 1 = 1010 1100

n = 0101 1111, then m = 1010 0000 + 1 = 1010 0001

n = 0111 1111, then m = 1000 0000 + 1 = 1000 0001

n = 0000 0001, then m = 1111 1110 + 1 = 1111 1111

To convert a negative decimal number to 2 To convert a negative decimal number to 2 s' complement binary: complement binary:

i. Convert the decimal number to a positive binary number.

ii. Take the 1's complement of that binary number and add 1.

•Example

– 50: 50 = 0011 0010; 1's C. = 1100 1101; 2's C. = 1100 1110.

– 127: 127 = 0111 1111; 1's C. = 1000 0000; 2's C. = 1000 0001.

– 1: 1 = 0000 0001; 1's C. = 1111 1110; 2's C. =1111 1111

To convert a negative decimal number to 2 To convert a negative decimal number to 2 s' complement binary: complement binary:

i. Convert the decimal number to a positive binary number.

ii. Take the 1's complement of that binary number and add 1.

• Ex

– 50: 50 = 0011 0010; 1's C. = 1100 1101; 2's C. = 1100 1110.

– 127: 127 = 0111 1111; 1's C. = 1000 0000; 2's C. = 1000 0001.

– 1: 1 = 0000 0001; 1's C. = 1111 1110; 2's C. =1111 1111

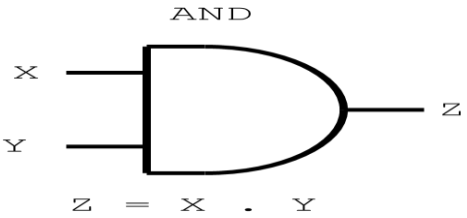
Logic Gates

- Basic logical operators are the logic functions AND, OR and NOT.
- Logic gates implement logic functions.
- The three basic logical operations are:

1. AND
2. OR
3. NOT

1. AND Gates

- Is denoted by a dot (·).

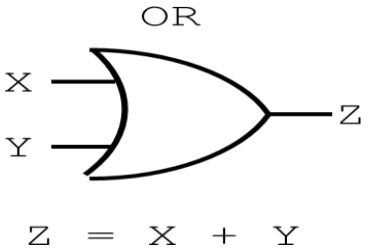


X	Y	Z
0	0	0
0	1	0
1	0	0
1	1	1

Fig 2.1 truth table and logical symbol for AND gate

2. OR Gates

- Is denoted by a plus (+).

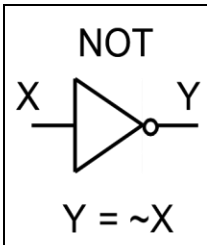


X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	1

Fig 2.2 logical symbol and truth table for OR gate

3. NOT Gates

- is denoted by an over bar (¯),

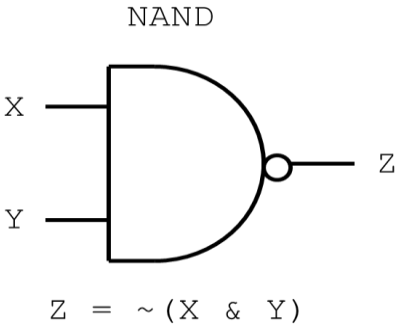


X	Y
0	1
1	0

Fig 2.3 logical symbol and truth table for NOT gate

NAND Gates

- combination of AND and NOT gate

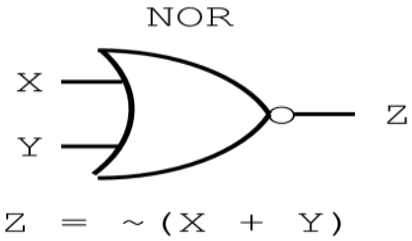


X	Y	Z
0	0	1
0	1	1
1	0	1
1	1	0

Fig 2.6 logical symbol and truth table for NAND gate

NOR Gate

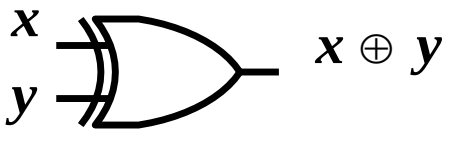
- combination of OR and NOT gate



X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	0

Fig 2.7 logical symbol and truth table for NOR gate

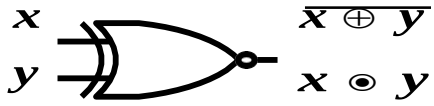
XOR (Exclusive-OR) Gate



X	Y	Z
0	0	0
0	1	1
1	0	1
1	1	0

$$x\,y + x\,y$$

XNOR (Exclusive-NOR Gate)



$\overline{x \oplus y}$

$x \odot y$

$\bar{x}\,\bar{y} + x\,y$

X	Y	Z
0	0	1
0	1	0
1	0	0
1	1	1

Fig 2.9 logic symbol and truth table for XNOR gate

Boolean algebra

- a useful mathematical system for specifying and transforming logic functions
- **Boolean Functions:** is an expression formed with binary variables, the 2 two binary operators **OR**

  and **AND**, and unary operator **NOT,** parentheses, and an equal sign**.**
- For a given value of the variables, the function can be either **0** or **1**.
- A Boolean function has:
  - i. At least one Boolean variable,
  - ii. At least one Boolean operator, and
  - iii. At least one input from the set {0, 1}.
- It produces an output that is also a member of the set {0, 1}.
- All Boolean functions through AND and NOT operations

All Boolean functions through NAND function

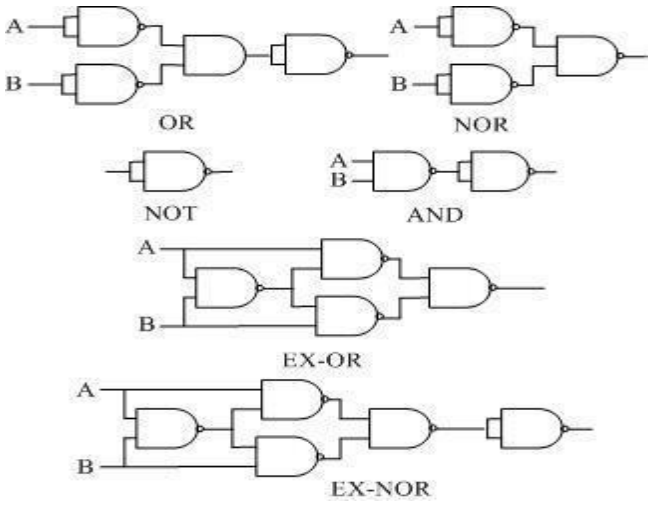


Fig 2.11 Boolean functions through NAND gate

All Boolean functions through NOR function

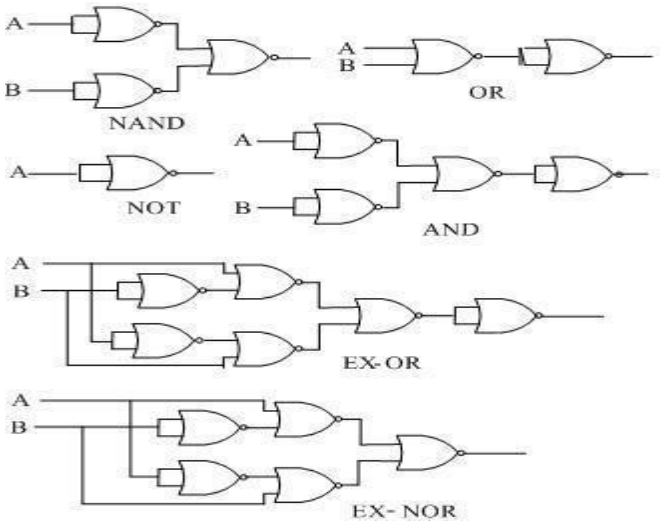


Fig 2.12 Boolean functions through NOR gate

1.	Law of Identity	$\overline{A} = A$ $\overline{\overline{A}} = A$
2.	Commutative Law	$A \cdot B = B \cdot A$ $A + B = B + A$
3.	Associative Law	$A \cdot (B \cdot C) = A \cdot B \cdot C$ $A + (B + C) = A + B + C$
4.	Idempotent Law	$A \cdot A = A$ $A + A = A$
5.	Double Negative Law	$\overline{\overline{A}} = A$
6.	Complementary Law	$A \cdot \overline{A} = 0$ $A + \overline{A} = 1$
7.	Law of Intersection	$A \cdot 1 = A$ $A \cdot 0 = 0$
8.	Law of Union	$A + 1 = 1$ $A + 0 = A$
9.	DeMorgan's Theorem	$\overline{AB} = \overline{A} + \overline{B}$ $\overline{A + B} = \overline{A}\,\overline{B}$
10.	Distributive Law	$A \cdot (B + C) = (A \cdot B) + (A \cdot C)$ $A + (BC) = (A + B) \cdot (A + C)$
11.	Law of Absorption	$A \cdot (A + B) = A$ $A + (AB) = A$
12.	Law of Common Identities	$A \cdot (\overline{A} + B) = AB$ $A + (\overline{A}B) = A + B$

- A Boolean expression can be represented in

i. Minterms

ii. Maxterms

iii. SOP (Sum Of Products)

iv. POS (Product of Sums)

i. Minterms

- Are AND terms with every variable present in either true or complemented form.
- Evaluates to '1' for a specific combination

	A	B	C	Minterm	
0	0	0	0	m_0	$\overline{A}\,\overline{B}\,\overline{C}$
1	0	0	1	m_1	$\overline{A}\,\overline{B}\,C$
2	0	1	0	m_2	$\overline{A}\,B\,\overline{C}$
3	0	1	1	m_3	$\overline{A}\,B\,C$
4	1	0	0	m_4	$A\,\overline{B}\,\overline{C}$
5	1	0	1	m_5	$A\,\overline{B}\,C$
6	1	1	0	m_6	$A\,B\,\overline{C}$
7	1	1	1	m_7	$A\,B\,C$

Fig 2.14 Minterm table

Sum of *Minterms(SOM)*

x	y	z	F	Minterm
0	0	0	0	
0	0	1	1	$m_1 = \overline{x}\,\overline{y}\,z$
0	1	0	0	
0	1	1	0	
1	0	0	0	
1	0	1	0	
1	1	0	1	$m_6 = x\,y\,\overline{z}$
1	1	1	1	$m_7 = x\,y\,z$

$$F = m_1 + m_6 + m_7 = \sum(1, 6, 7) = \overline{x}\,\overline{y}\,z + x\,y\,\overline{z} + x\,y\,z$$

Fig 2.15 Example for SOM

ii. Maxterms

- Are OR terms with every variable in true or complemented form.
- Evaluates to '0' for a specific combination

	A	B	C	Maxterm	
0	0	0	0	M_0	$A + B + C$
1	0	0	1	M_1	$A + B + \overline{C}$
2	0	1	0	M_2	$A + \overline{B} + C$
3	0	1	1	M_3	$A + \overline{B} + \overline{C}$
4	1	0	0	M_4	$\overline{A} + B + C$
5	1	0	1	M_5	$\overline{A} + B + \overline{C}$
6	1	1	0	M_6	$\overline{A} + \overline{B} + C$
7	1	1	1	M_7	$\overline{A} + \overline{B} + \overline{C}$

Fig 2.15 maxterm table

Product-Of-Maxterm (POM)

x	y	z	F	Maxterm
0	0	0	1	
0	0	1	1	
0	1	0	**0**	$M_2 = (x + y + z)$
0	1	1	1	
1	0	0	**0**	$M_4 = (x + y + z)$
1	0	1	1	
1	1	0	**0**	$M_6 = (x + y + z)$
1	1	1	1	

$$F = M_2 \cdot M_4 \cdot M_6 = \prod (2, 4, 6) = (x+\bar{y}+z)\ (\bar{x}+y+z)\ (\bar{x}+\bar{y}+z)$$

iii. SOP (Sum of Products)

- equations are written as an OR of AND terms
- Implementation of this form is a two-level network of gates such that:
- The first level consists of *n*-input AND gates
- The second level is a single OR gate
- This form often can be simplified so that the corresponding circuit is simpler.

  - A Simplification Example:
    $F(A, B, C) = \sum (1,4,5,6,7)$
  - Writing the minterm expression:
    $F = \bar{A}\,\bar{B}\,C + A\,\bar{B}\,\bar{C} + A\,B\,\bar{C} + ABC + ABC$
  - Simplifying:
    $F = \bar{A}\,\bar{B}\,C + A\,(\bar{B}\,\bar{C} + \bar{B}\,C + B\,\bar{C} + B\,C)$
    $F = \bar{A}\,\bar{B}\,C + A\,(\bar{B}\,(\bar{C} + C) + B\,(\bar{C} + C))$
    $F = \bar{A}\,\bar{B}\,C + A\,(\bar{B} + B)$
    $F = \bar{A}\,\bar{B}\,C + A$
    $F = \bar{B}\,C + A$
  - Simplified F contains 3 literals compared to 15

Iv.Product of Sum (POS)

- equations are written as an AND of OR terms
- $F = (A + C)(A + \bar{B})(\bar{B} + C)$

Minterms and maxterms are related

- Any minterm m_i is the *complement* of the corresponding maxterm M_i

Minterm	Shorthand	Maxterm	Shorthand
x'y'z'	m_0	x + y + z	M_0
x'y'z	m_1	x + y + z'	M_1
x'yz'	m_2	x + y' + z	M_2
x'yz	m_3	x + y' + z'	M_3
xy'z'	m_4	x' + y + z	M_4
xy'z	m_5	x' + y + z'	M_5
xyz'	m_6	x' + y' + z	M_6
xyz	m_7	x' + y' + z'	M_7

- For example, $m_4' = M_4$ because $(xy'z')' = x' + y + z$

Karnaugh-Map

- A Karnaugh map is a graphical method used to obtain the most simplified form of an expression in a standard form (Sum-of-Products or Product-of-Sums).
- The simplest form of an expression is the one that has the minimum number of terms with the least number of literals (variables) in each term.
- An *n*-variable K-map has 2^n cells with each cell corresponding to a row of an *n*-variable truth table.
- K-map cells are labeled with the corresponding truth-table row.
- K-map cells are arranged such that adjacent cells correspond to truth rows that differ in only one bit position (*logical adjacency*).

For the case of 2 variables, we form a map consisting of 2^2=4 cells as shown in Figure

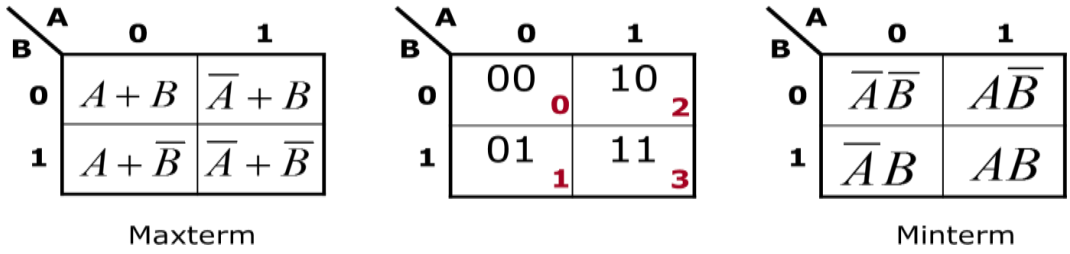


Fig 2.17 2 variable K-Map

3 variables Karnaugh map

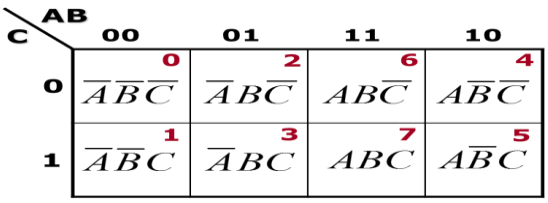


Fig 2.18 3 variable K-Map

4 variables Karnaugh map

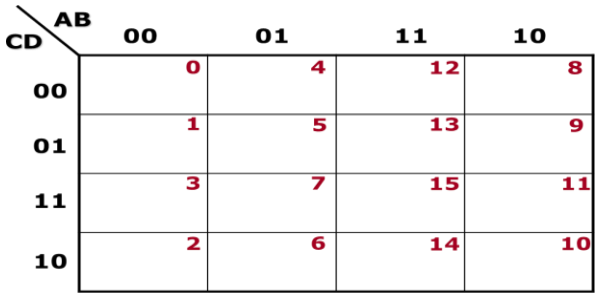


Fig 2.19 4 variable K-Map

- The Karnaugh map is completed by entering a '1'(or '0') in each of the appropriate cells.

- Within the map, adjacent cells containing 1's (or 0's) are grouped together in twos, fours, or eights.

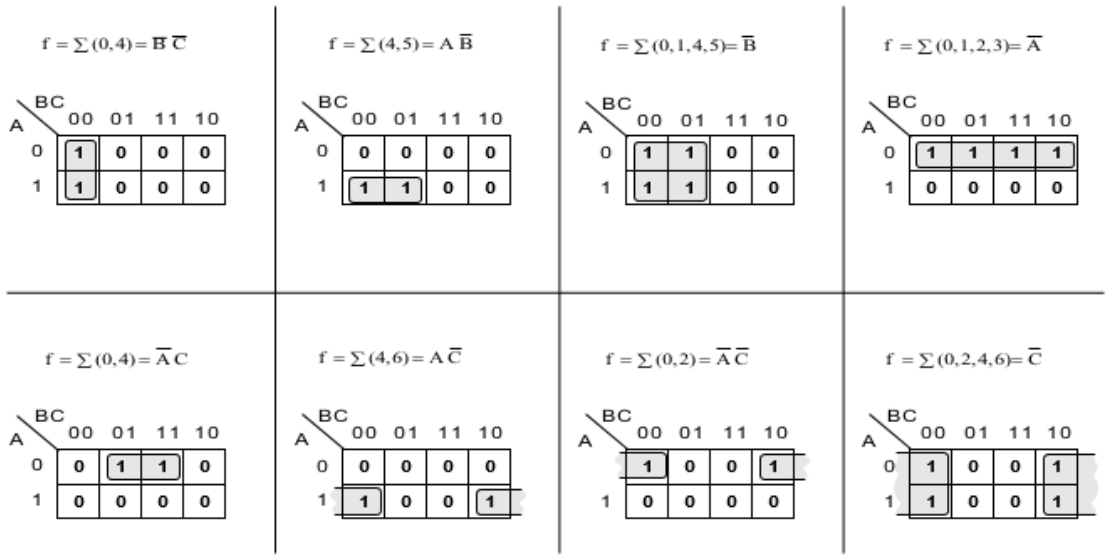


Fig 2.20 examples of simplication of 3 variable k-map

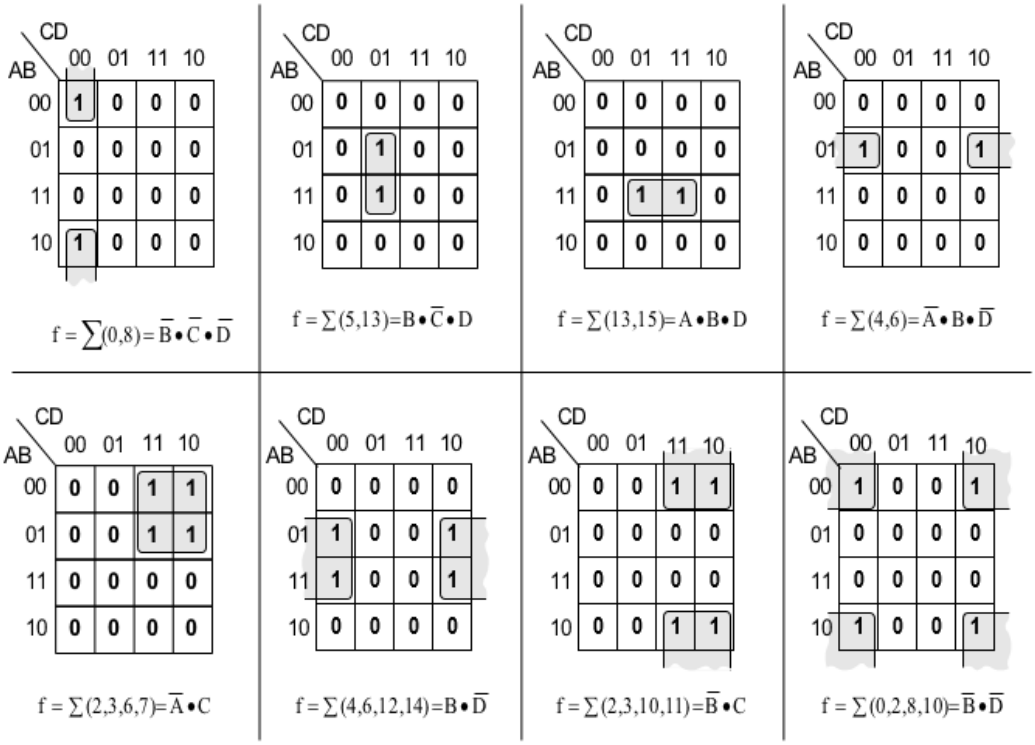


Fig 2.21 examples for 4 variable K-Map

Don't Cares

- In certain cases some of the minterms may never occur or it may not matter what happens if they do.In such cases we fill in the Karnaugh map with and X meaning don't care. When minimizing an X is like a "joker" X can be 0 or 1 - whatever helps best with the minimization.

Ex:

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

Simplifies to B if x=1

Ex: simplify the expression using K-map for the truth table given below

M	F1	F2	F3	OPEN
0	0	0	0	0
0	0	0	1	1
0	0	1	0	1
0	0	1	1	X
0	1	0	0	1
0	1	0	1	X
0	1	1	0	X
0	1	1	1	X
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
1	0	1	1	X
1	1	0	0	0
1	1	0	1	X
1	1	1	0	X
1	1	1	1	X

	$\overline{F2}\,\overline{F3}$	$\overline{F2}F3$	$F2F3$	$F2\overline{F3}$
$\overline{M}\,\overline{F1}$	0	1	X	1
$\overline{M}F1$	1	X	X	X
$M\,F1$	0	X	X	X
$M\,\overline{F1}$	0	0	X	0

$$= \overline{M}F_1 + \overline{M}F_3 + \overline{M}F_2$$

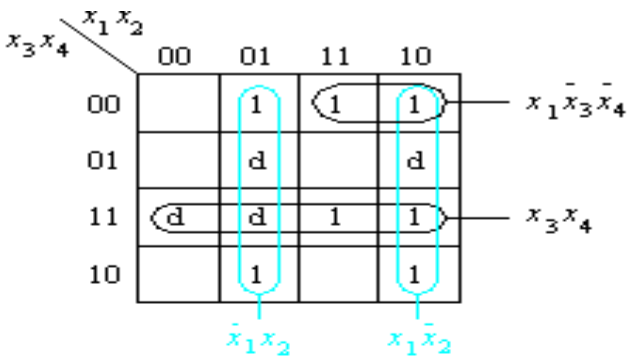
Ex: Find a minimal SOP expression from the following 4 variable K-Map

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

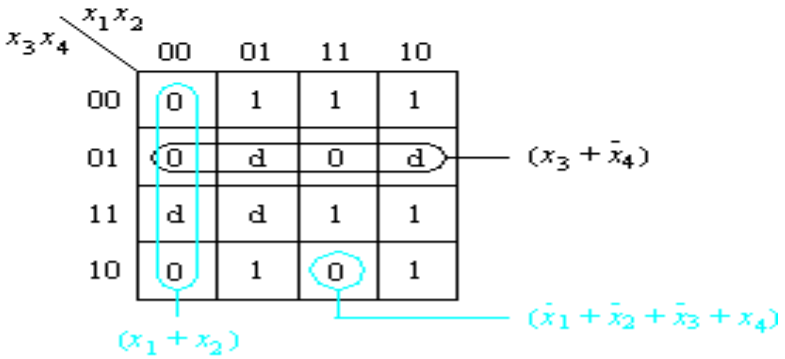
M(A,B,C,D) = B D + A' B'

Problem 1: Determine the minimum-cost SOP and POS expressions for the function *f*(*x*1, *x*2, *x*3, *x*4) =

_∑*m*(4, 6, 8, 10, 11, 12, 15) + *D*(3, 5, 7, 9).



(a) Determination of the SOP expression



(b) Determination of the POS expression

Fig 2.22 Karnaugh maps for problem 1

Floating Point Numbers

There are several ways to represent floating point number but IEEE 754 is the most efficient in most cases.

IEEE 754 has 3 basic components:

The Sign of Mantissa –

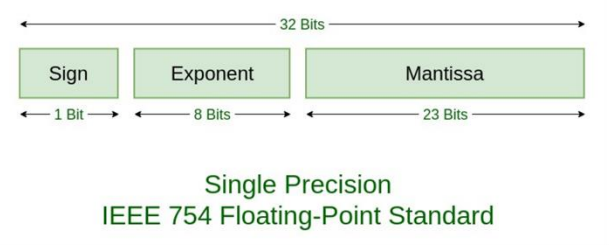
This is as simple as the name. 0 represents a positive number while 1 represents a negative number.

The Biased exponent –

The exponent field needs to represent both positive and negative exponents. A bias is added to the actual exponent in order to get the stored exponent.

The Normalised Mantissa –

The mantissa is part of a number in scientific notation or a floating-point number, consisting of its significant digits. Here we have only 2 digits, i.e. O and 1. So a normalised mantissa is one with only one 1 to the left of the decimal.



Single Precision
IEEE 754 Floating-Point Standard

Double Precision
IEEE 754 Floating-Point Standard

Example:

85.125

85 = 1010101

0.125 = 001

85.125 = 1010101.001

=1.010101001 x 2^6

sign = 0

Single precision:

biased exponent 127+6=133

133 = 10000101

Normalised mantisa = 010101001

we will add 0's to complete the 23 bits

The IEEE 754 Single precision is:

= 0 10000101 01010100100000000000000

Combinational Circuits

Combinational Logic

- Logic circuits for digital systems may be combinational or sequential.
- A combinational circuit consists of input variables, logic gates, and output variables.

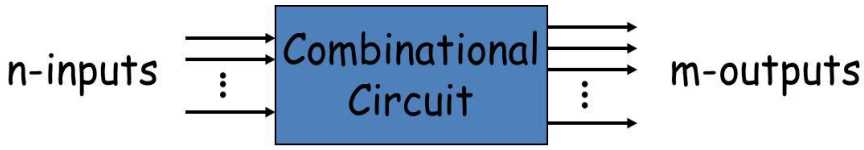


Fig 3.1 combinational circuit

- Hence, a combinational circuit can be described by:

1. A truth table that lists the output values for each combination of the input variables, or
2. *m* Boolean functions, one for each output variable.

Combinational vs. Sequential Circuits

- Combinational circuits are memory-less. Thus, the output value depends ONLY on the current input values.
- Sequential circuits consist of combinational logic as well as memory elements (used to store certain circuit states). Outputs depend on BOTH current input values and previous input values (kept in the storage elements).

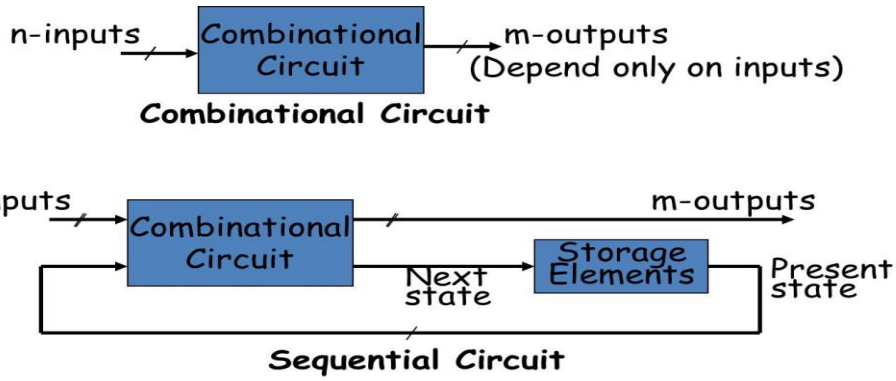


Fig 3.2 combinational circuits vs sequential circuits

Half Adder

A combinational circuit that performs the addition of two bits is called a half adder.

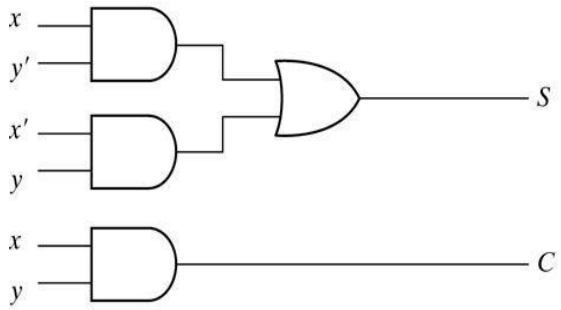
Fig 3.3 logical symbol of half adder

- The truth table for the half adder is listed below:

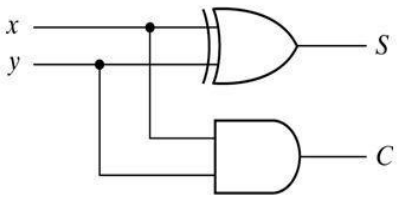
x y	*C S*
0 0	**0 0**
0 1	**0 1**
1 0	**0 1**
1 1	**1 0**

S = x'y + xy'

C = xy



(a) $S = xy' + x'y$
$C = xy$



(b) $S = x \oplus y$
$C = xy$

Full Adder

One that performs the addition of three bits (two significant bits and a previous carry) is a full adder.

Inputs			Outputs	
A	B	C_{in}	S (Sum)	C_{out} (Carry)
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

Fig 3.5 truth table for full adder

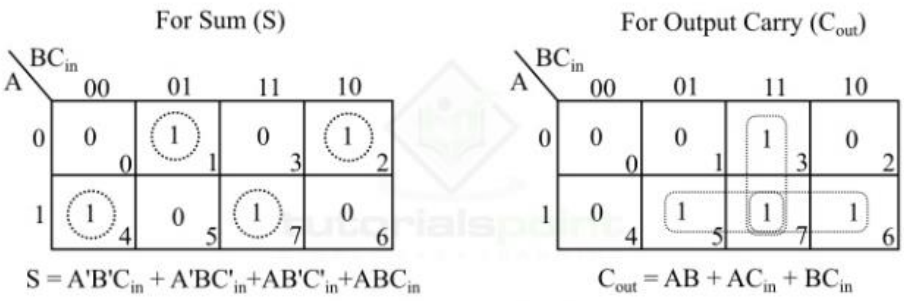
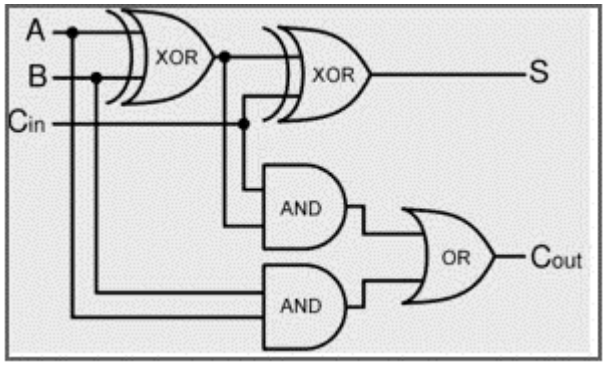


Figure 2 - K Map for Full Adder

- S = A ⊕ B ⊕ Cin.
- Cout = (A · B) + (Cin · (A ⊕ B)).



Decoder

- A decoder is a logic circuit that accepts a set of inputs that represents a binary number and activates only the output that corresponds to the input number.
- A decoder has
  - N inputs
  - 2^N outputs
  - Exactly one output will be active for each combination of the inputs.

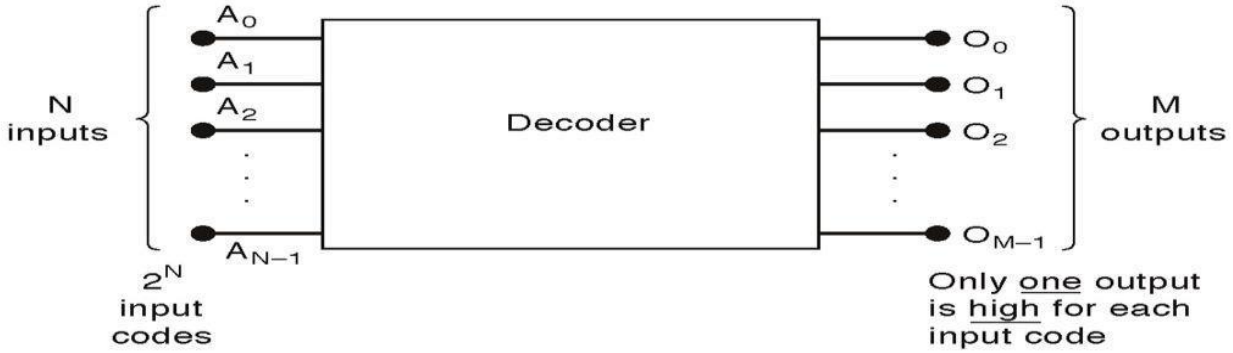
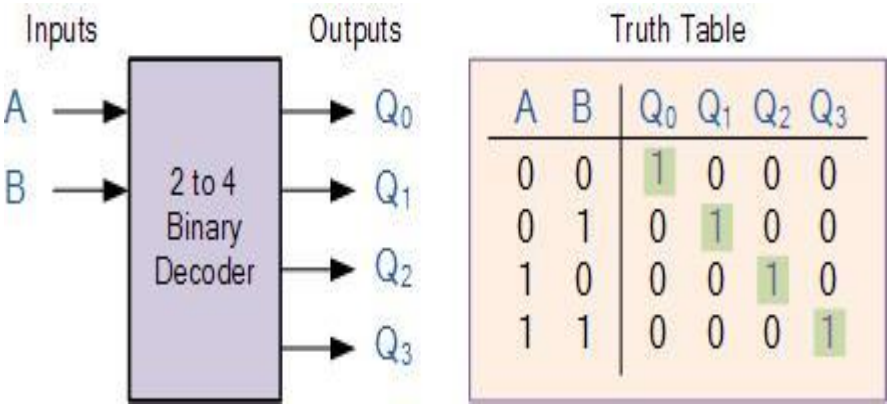


Fig 3.17 logical symbol of N-M decoder

- For each of these input combinations only one of the *M* outputs will be active *HIGH* (1), all the other outputs are *LOW* (0). An AND gate can be used as the basic decoding element because it produces a HIGH output only when all inputs are HIGH.



Truth Table

A	B	Q_0	Q_1	Q_2	Q_3
0	0	1	0	0	0
0	1	0	1	0	0
1	0	0	0	1	0
1	1	0	0	0	1

Encoder

- Encoders typically have 2N inputs and N outputs.
- These are called 2N–to–N encoders.
- Encoders can also be devised to encode various symbols and alphabetic characters.
- The process of converting from familiar symbols or numbers to a coded format is called encoding.

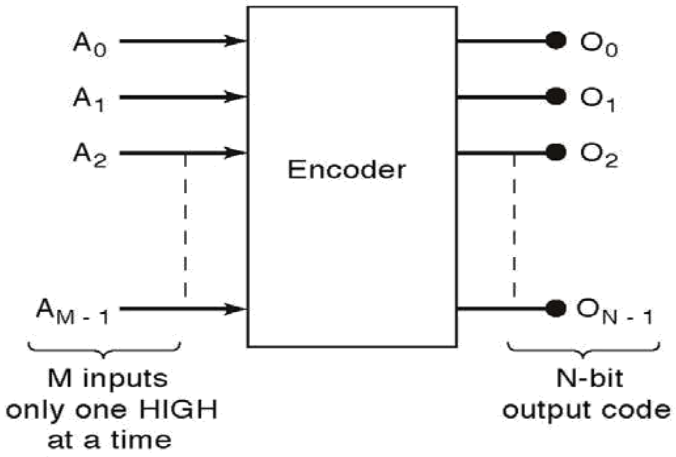


Fig 3.25 Logical diagram of Encoder

8-to-3 encoder Implementation

- Octal-to-Binary
- An octal to binary encoder has $2^3 = 8$ input lines D_0 to D_7 and 3 output lines Y_0 to Y_2. Below is the **truth table for an octal to binary encoder.**

Input								Output		
D_0	D_1	D_2	D_3	D_4	D_5	D_6	D_7	X	Y	Z
1	0	0	0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	0	0	0	1
0	0	1	0	0	0	0	0	0	1	0
0	0	0	1	0	0	0	0	0	1	1
0	0	0	0	1	0	0	0	1	0	0
0	0	0	0	0	1	0	0	1	0	1
0	0	0	0	0	0	1	0	1	1	0
0	0	0	0	0	0	0	1	1	1	1

Fig 3. 26 truth table for 8-3 encoder

From the truth table, the outputs can be expressed by following Boolean Function.

$Z = D_1 + D_3 + D_5 + D_7$

$Y = D_2 + D_3 + D_6 + D_7$

$X = D_4 + D_5 + D_6 + D_7$

Note: Above boolean functions are formed by ORing all the input lines for which output is 1. For instance Z is 1 for D_1, D_3, D_5, D_7 input lines. The encoder can therefore be implemented with OR gates whose inputs are determined directly from truth table as shown in the image below:

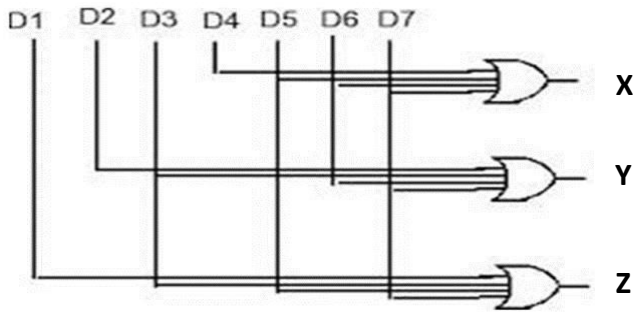


Fig 3.27 logic circuit for 8-3 encoder

Multiplexer

•A multiplexer or mux is a device that selects one of several analog or digital input signals and forwards the selected input into a single line. A multiplexer of 2n inputs has n select lines, which are used to select which input line to send to the output."

•The select lines determine which input is connected to the output.

•MUX Types

2-to-1 (1 select line)

4-to-1 (2 select lines)

8 -to-1 (3 select lines)

16-to-1 (4 select lines)

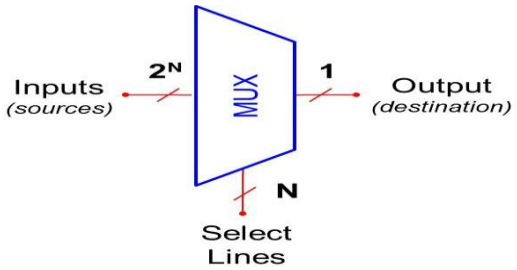
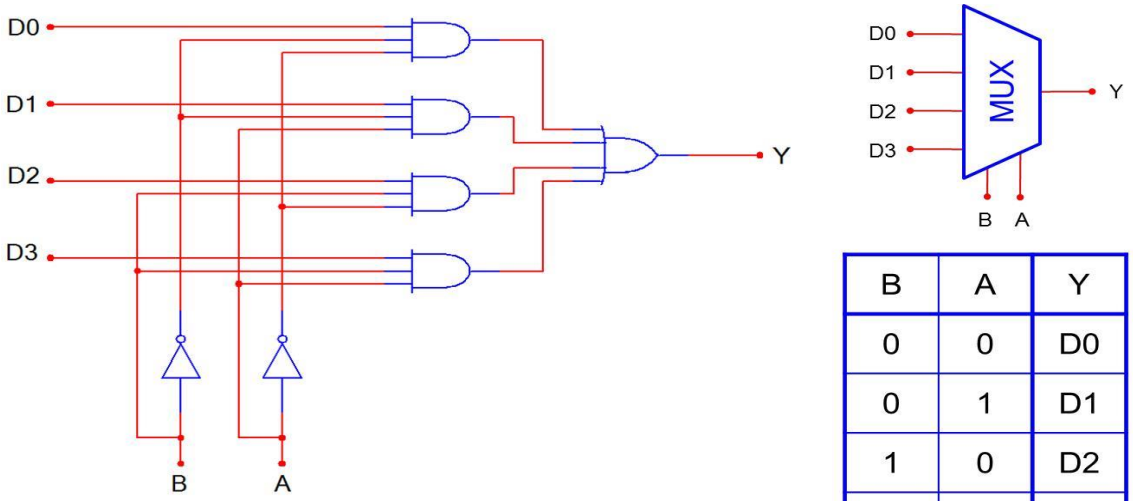


Fig 3.30 multiplexer block diagram

4-to-1 Multiplexer (MUX)



B	A	Y
0	0	D0
0	1	D1
1	0	D2
1	1	D3

Fig 3.31 logic circuit, symbol and table for 4-1 multiplexer

The Boolean expression for this 4-to-1 **Multiplexer** above with inputs D_0 to D_3 and data select lines A, B is given as:

$Y = \mathbf{A'B'D_0 + A'B\ D_1 + AB'D_2} + ABD_3$

Demultiplexer

- A DEMUX is a digital switch with a single input (source) and a multiple outputs (destinations).
- The select lines determine which output the input is connected to.
- DEMUX Types

1-to-2 (1 select line)

1-to-4 (2 select lines)

1-to-8 (3 select lines)

1-to-16 (4 select lines

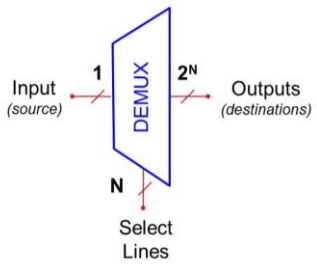
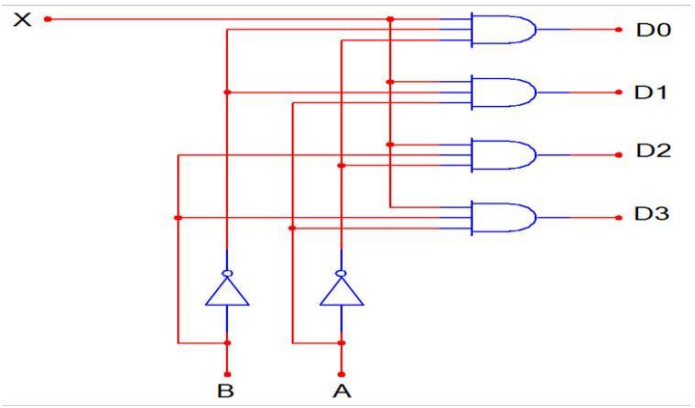
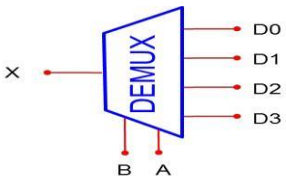


Fig 3.32 Demultiplexer circuit symbol

1-to-4 De-Multiplexer (DEMUX)





B	A	D0	D1	D2	D3
0	0	X	0	0	0
0	1	0	X	0	0
1	0	0	0	X	0
1	1	0	0	0	X

Fig 3.33 logic circuit for 1-4 demultiplexer, symbol and table