Subject: Analog and Digital Electronics
Code:15CS32

Syllabus:

The Basic Gates
: Review of Basic Logic gates, Positive and Negative Logic, Introduction to HDL.

Combinational Logic Circuits:Sum-of-Products Method, Truth Table to Karnaugh Map, Pairs Quads, and Octets, Karnaugh Simplifications, Don't-care Conditions, Product-of-sums Method, Product-of-sums simplifications, Simplification by Quine-McClusky Method, Hazards and Hazard covers, HDL Implementation Models.

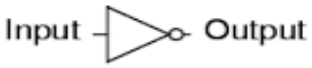
Basic gates

Three logic circuits, the inverter, the OR gate, and the AND gate can be used to produce any digital system.

The inverter gate (NOT gate):

Figure 1 shows the symbol and truth table for an inverter. The NOT gate performs the basic logical function called inversion or complementation. NOT gate is also called inverter. The purpose of this gate is to convert one logic level into the opposite logic level. It has one input and one output. When a HIGH level is applied to an inverter, a LOW level appears on its output and vice versa.

NOT gate truth table



Input	Output
0	1
1	0

Figure 1. Truth Table

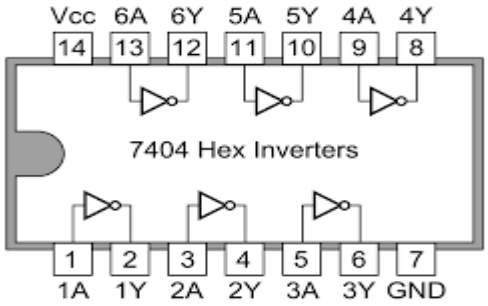


Figure 2. Shows the pinout diagram of a 7404 hex inverter. This IC contains six inverters.

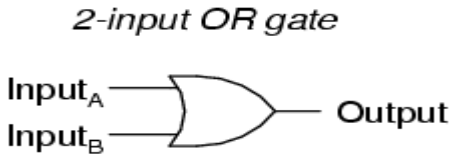
Figure 2.Pinout diagram of INV 7404

OR gates:

An OR gate has two or more input signals but only one output signal. It is called an OR gate because the output voltage is high if any or all of the input voltages are high.
Figure 3 shows the logic symbol and truth table for a 2-input OR gate.

Y= A OR B i.e. Y= A+B

An OR gate can have as many inputs as desired. No matter how many inputs, the action of any OR gate is summarized like this: One or more high inputs produce a high output.



A	B	Output
0	0	0
0	1	1
1	0	1
1	1	1

Figure 3. Truth Table

Figure 4 shows the pinout diagram of a 7432, a TTL quad 2-input OR gate. This digital IC contains four 2-input OR gates inside a 14-pin DIP.

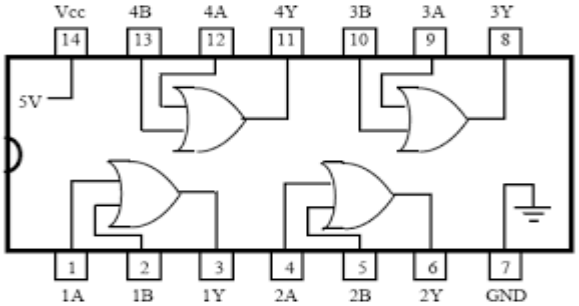
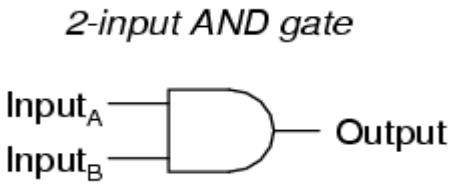


Figure 4.Pin Diagram of 7432

AND gates:

The AND gate has a high output only When all inputs are high. Figure 5 shows logic diagram and truth table of a 2-input AND gate. The AND gate has a high output only when both inputs are high.

Y= A AND B i.e. Y= AB



A	B	Output
0	0	0
0	1	0
1	0	0
1	1	1

Figure 5. Truth Table

Figure 6. shows the pinout diagram of a 7408, a TTL quad 2-input AND gate. This digital IC contains four 2-iput AND gates.

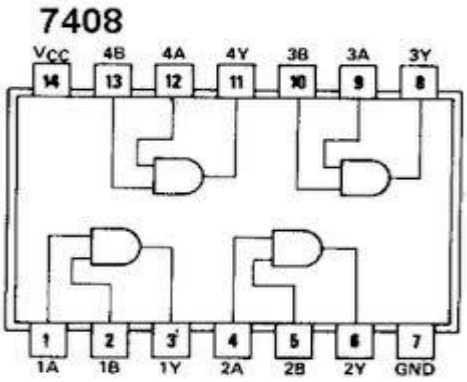


Figure 6. Pin diagram 7408

Universal Logic Gates- NOR, NAND:

A universal logic gate is a logic gate that can be used to construct all other logic gates.

NOR gate:

NOR gate is cascade of OR gate and NOT gate, as shown in the figure 7. The bubble on the output is a remainder of the inversion that takes place after the ORing.

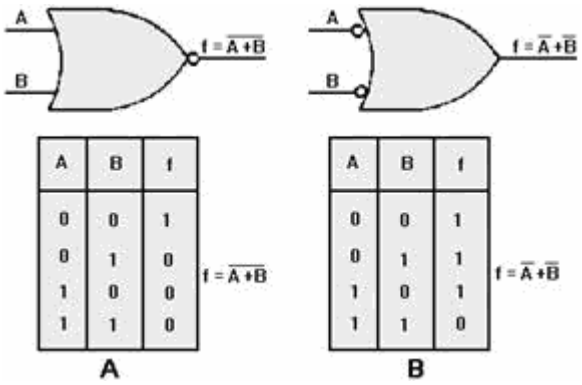


Figure 7. Symbol and Truth Table

The 7402 is a quad 2-input NOR gate in a 14-pin DIP as illustarted in figure 8.

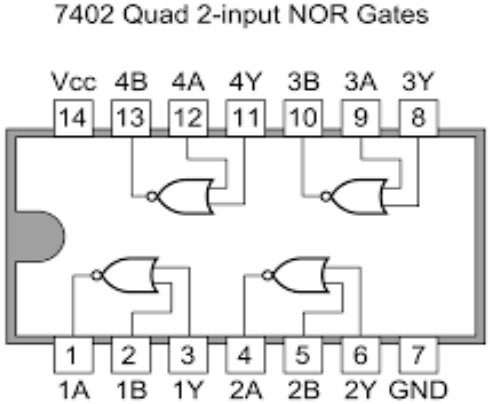


Figure 8. Pin diagram of 7402

De Morgam's first theorem:

The De-Morgan's theorem states that the complement of a sum equals the product of the complements.

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

Proof

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

Figure 9. Truth table

Universality of NOR gate:

Figure shows how all other logic gates can be obtained from NOR gates.

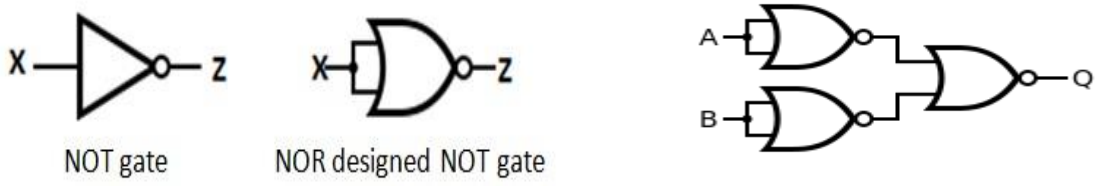
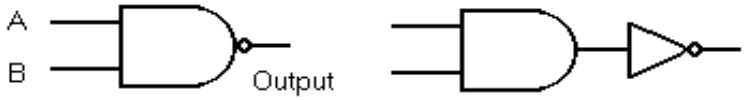


Figure 10. Realisation of basic gates using NOR gate

NAND gate:

NAND gate is a cascade of AND gate and NOT gate, as shown in the figure below. It has two or more inputs and only one output. The output of NAND gate is HIGH when any one of its input is LOW (i.e. even if one input is LOW, output will be HIGH). The logic symbol and truth table is shown in figure 11.



A	B	Output
0	0	1
0	1	1
1	0	1
1	1	0

Output = $\overline{A \cdot B}$

Figure 11.Symbol and Truth Table

Figure 12 shows the pin diagram for quad NAND gate.

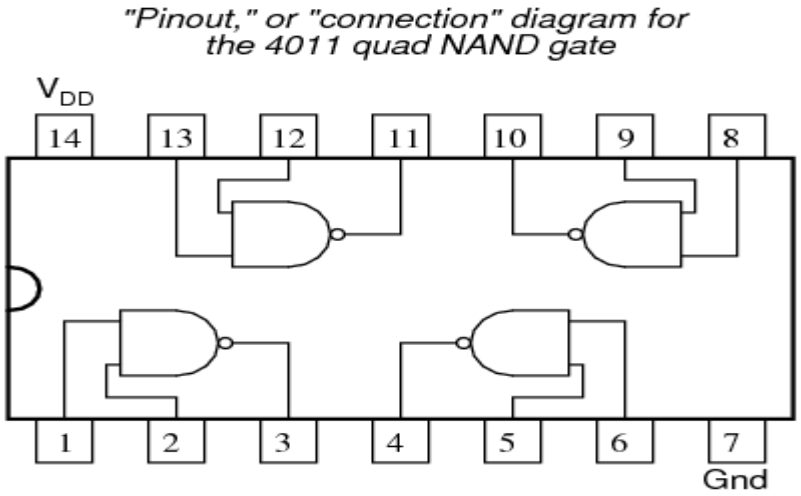


Figure 12. Pin diagram NAND gate

De Morgan's second theorem:

The De Morgan's theorem states that the complement of a product equals the product of the sum.

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

Proof

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

Universality of NAND gate:

Figure shows how all other logic gates can be obtained using NAND gate.

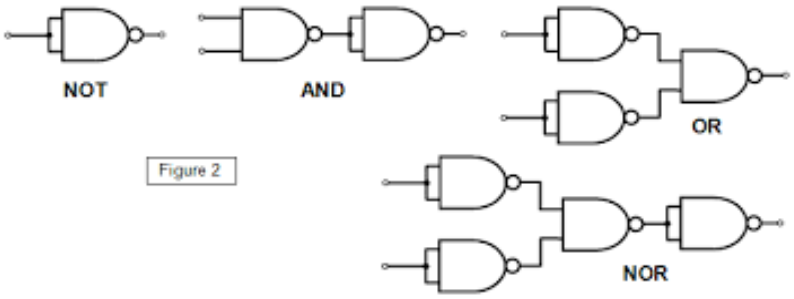
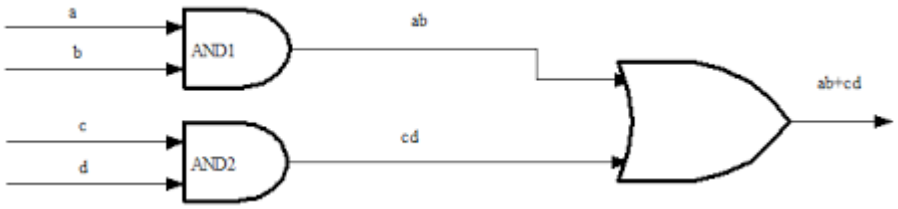
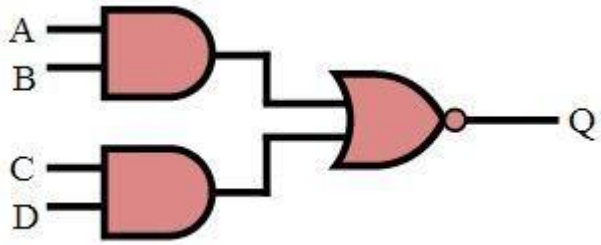


Figure 2

AND-OR-Invert gates:
Figure shows an AND-OR circuit.



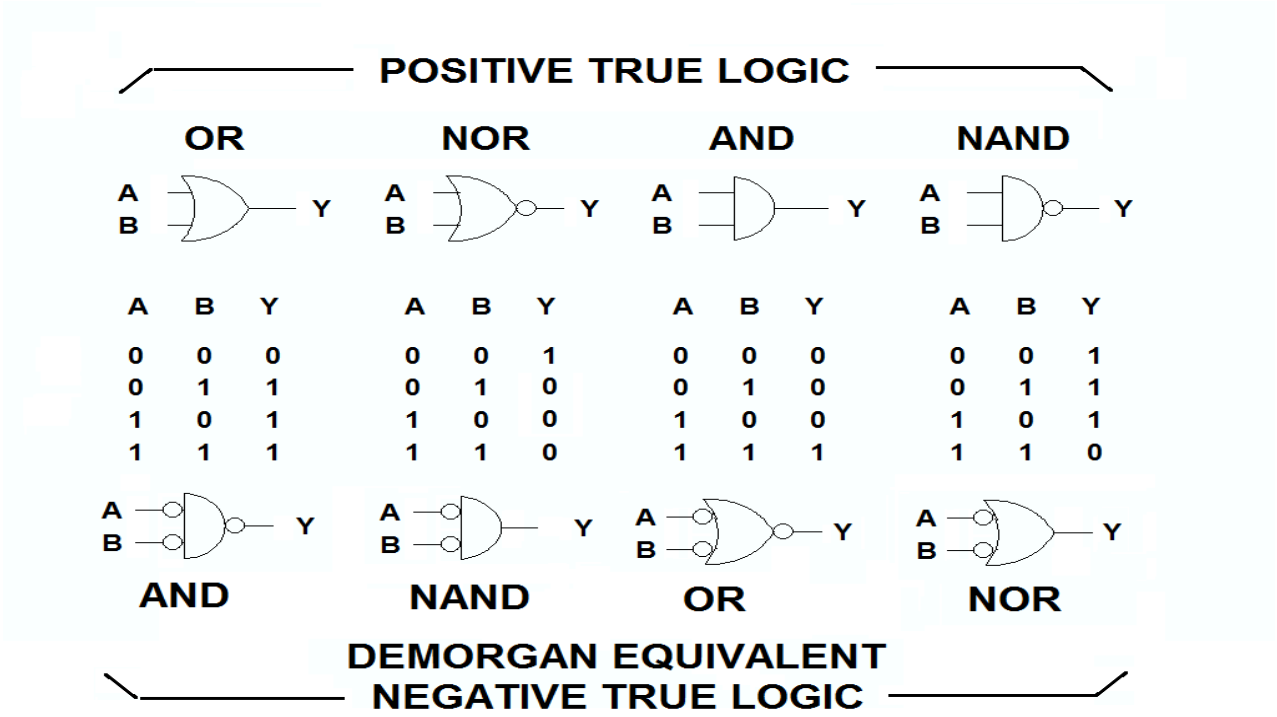
The figure below shows the AND-OR-Invert circuit.



Positive and Negative logic:

In positive logic, the lower voltage level is assigned binary 0 & higher voltage level is assigned binary 1. In negative logic, the lower voltage level is assigned binary 1 & higher voltage level is assigned binary 0.

Positive and negative gates:



An OR gate in a positive logic system becomes an AND gate in a negative logic system.

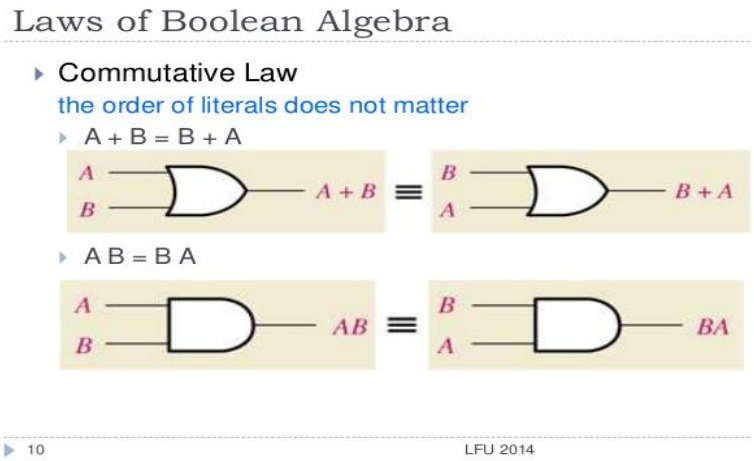
In a positive logic system, binary 0 stands for low and binary 1 for high. In a negative logic system, binary 1 stands for low and binary 0 stands for high.

Positive OR	↔	Negative AND
Positive AND	↔	Negative OR
Positive NOR	↔	Negative NAND
Positive NAND	↔	Negative NOR

Combinational Logic circuits

Boolean laws and theorems:
The commutative laws are

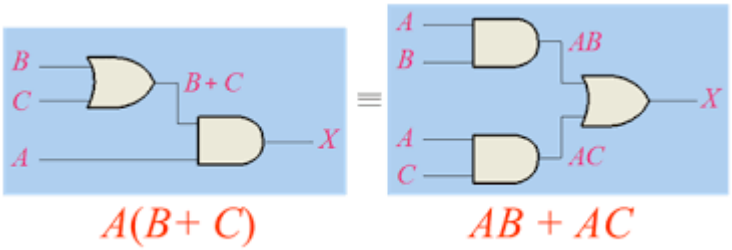
A +B = B+A
AB=BA



The associative laws are

A+(B+C) = (A+B)+C
A(BC) = (AB)C

The distributive law is
A(B+C) = AB+AC



OR operation, AND operations:

Boolean Theorem

- Basic Rules

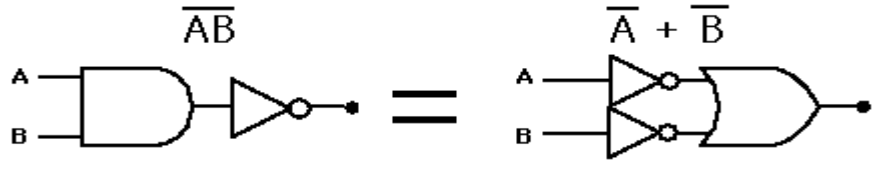
1. $A + 0 = A$
2. $A + 1 = 1$
3. $A + A = A$
4. $A + \overline{A} = 1$
5. $A \cdot 0 = 0$
6. $A \cdot 1 = A$
7. $A \cdot A = A$
8. $A \cdot \overline{A} = 0$
9. $\overline{\overline{A}} = A$
10. $A + AB = A$
11. $A + \overline{A}B = A + B$
12. $(A + B)(A + C) = A + BC$

Double inversion and De Morgan's theorems:

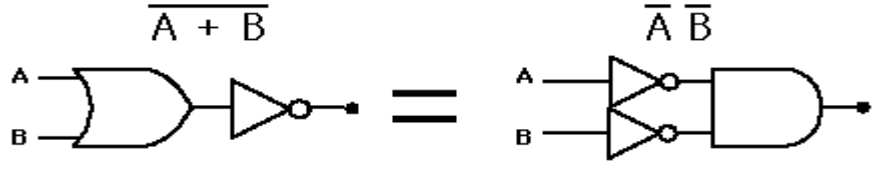
The double inversion rule is

$$\overline{\overline{A}} = A$$

De Morgan's theorem is



A NAND gate is equivalent to an inversion followed by an OR



A NOR gate is equivalent to an inversion followed by an AND

Duality Theorem:

The duality theorem states, starting with a boolean relation, you can derive another boolean relation by,

1. Changing each OR sign to an AND sign.
2. Changing each AND sign to an OR sign.
3. Complementing any 0 or 1 appearing in the expression.

For example:
A+0=A
The dual relation is
A.1=A

Covering and Combination:
The covering rule, where one term covers the condition of the other term so that the other term becomes redundant, can be represented in dual form as

A+AB=A

and A(A+B)=A

A+AB= A.1+AB==A(1+B) =A

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

The combining rules are,

AB+AB' = A

and its dual form (A+B)(A+B')=A

Consensus theorem:
The consensus theorem finds a redundant term which is a consensus of two other terms. This can be expressed in dual form as,

AB+A'C+BC=AB+A'C

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

Sum of Products method:
The outputs are fundamental products. Table lists each fundamental product next to the input conditions producing a high output. For example, A'B' is high when A and B are low. The fundamental products are also called minterms. They can be represented as m_0,m_1,m_2 and m_3.

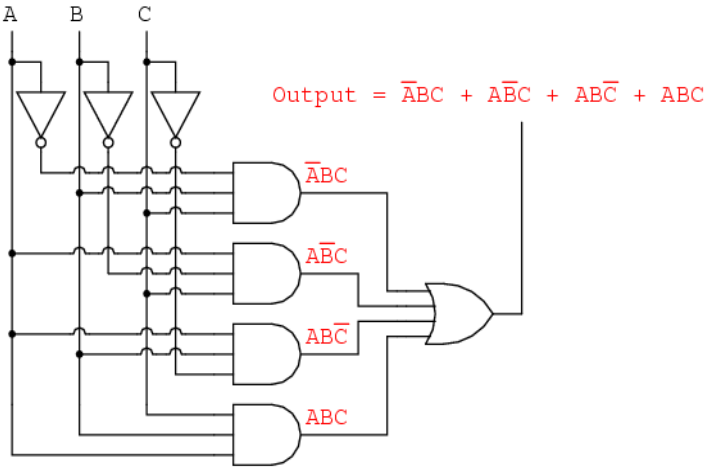
A	B	F	Minterm
0	0	0	A'B'
0	1	1	A'B
1	0	1	AB'
1	1	1	AB

sensor inputs

A	B	C	Output	
0	0	0	0	
0	0	1	0	
0	1	0	0	
0	1	1	1	$\bar{A}BC = 1$
1	0	0	0	
1	0	1	1	$A\bar{B}C = 1$
1	1	0	1	$AB\bar{C} = 1$
1	1	1	1	$ABC = 1$

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

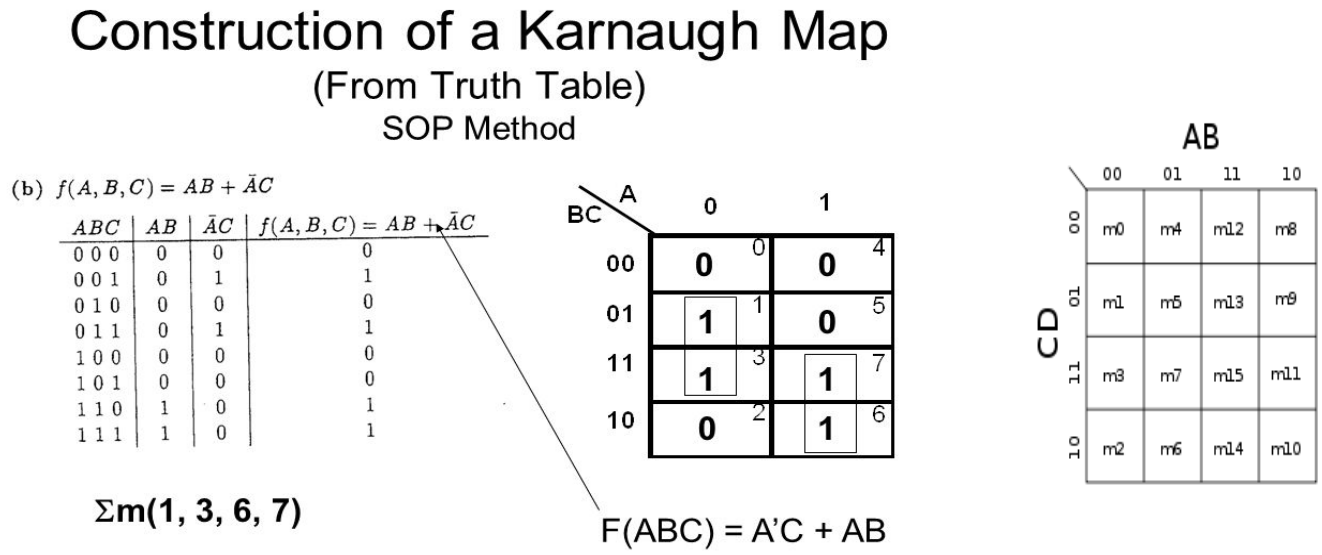
Sum of products equation:
Locate each output 1 in the truth table and write down the fundamental product. This kind of representation of a truth table is also known as canonical sum form.



Logic Circuit:
After writing the sum-of-products equation, the corresponding logic circuit by drawing an AND-OR network.

Truth table to Karnaugh map:

A Karnaugh map is a visual display of the fundamental products needed for a sum-of-products solution. For instance, here is how to convert below Table into its Karnaugh mapY = F(A, B, C) = L_m(1,3,6,7). First, draw the blank map of Fig. The vertical column is labelled AB,A'B, AB' and A'B'. With this order, only one variable changes from complemented to uncomplemented form (or vice versa) as you move downward. Minterms in the equation gets mapped into corresponding positions in the map.



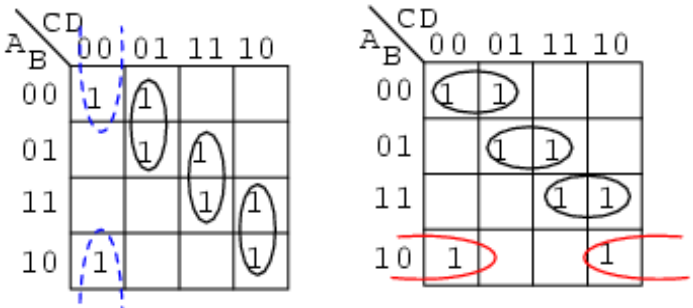
Four Variable maps:

Many digital computers and systems process 4-bit numbers. For instance, some digital chips will work with nibbles like 0000, 0001, 0010, and so on. For this reason, logic circuits are often designed to handle four input variables (or their complements).

Pairs, Quads and Octets:

The map contains a pair of ls that are horizontally adjacent (next to each other). The first l represents the product A'B'C'D'; the second l stands for the product
A'B'C'D.

$$Out = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}B\bar{C}D + \bar{A}BCD + ABCD + ABC\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D}$$



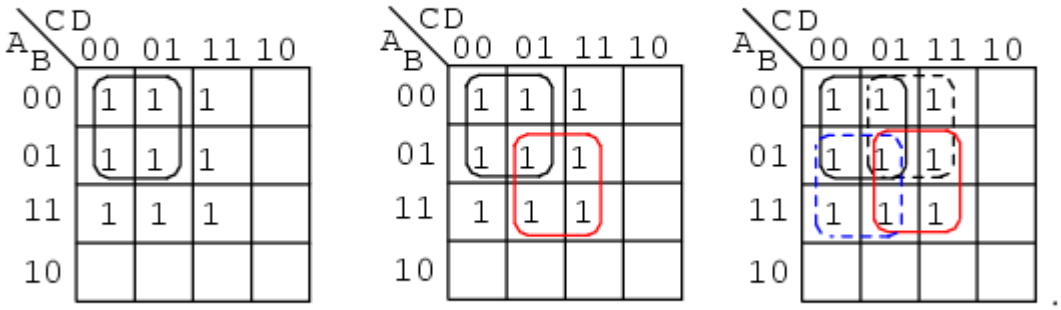
$$Out = B\bar{C}\bar{D} + \bar{A}\bar{C}D + BCD + AC\bar{D}$$

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

The quad:

A quad is a group of four ls that are horizontally or vertically adjacent. The ls may be end-to-end, or in the form of a square, as in Fig. When a quad is seen, always encircle it because it leads to a simpler product. In fact, a quad eliminates two variables and their complements.

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



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

The octet:

Besides pairs and quads, there is one more group to adjacent 1 s to look for: the octet. This is a group of eight 1 s like those of Fig. An octet like this eliminates three variables and their

complements.

ab \ cd	c'd' 00	c'd 01	cd 11	cd' 10
a'b' 00	0	1	1	1
a'b 01	0	1	1	0
ab 11	0	1	1	0
ab' 10	0	1	1	0

Dont Care conditions:

In some digital systems, certain input conditions never occur during normal operation; therefore, the corresponding output never appears. Since the output never appears, it is indicated by an X in the truth table. Figure shows a K-map with dont care conditions.

K-Map:

a:

DC \ BA	00	01	11	10
00	1	0	1	1
01	0	1	1	1
11	X	X	X	X
10	1	1	X	X

$F_a(D,C,B,A) = D + B + CA + C'A'$

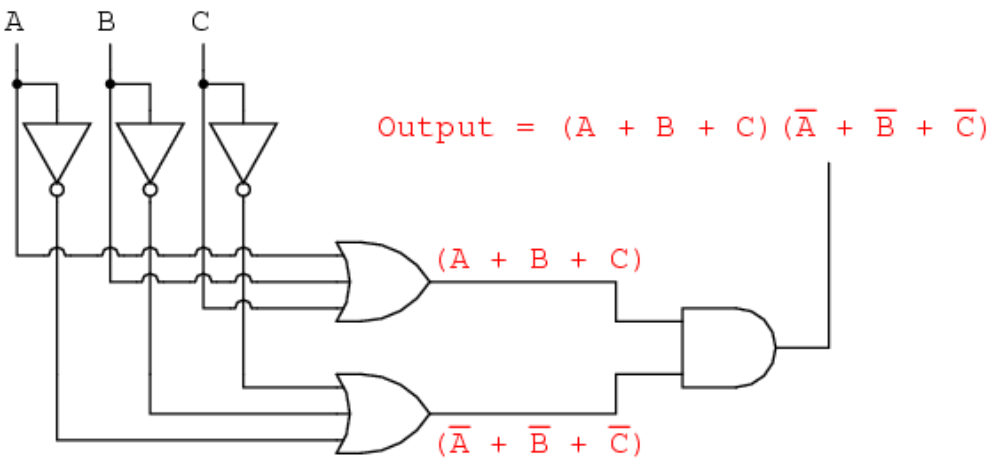
Product of Sum Methods:

Given a truth table, identify the fundamental sums needed for a logic design. Then by ANDing these sums, we get the product-of-sums equation corresponding to the truth table. But there are some differences between the two approaches. With the sum-of-products method, the fundamental product produces an output 1 for the corresponding input condition. But with the product-of-sums method, the fundamental sum produces an output O for the corresponding input condition. The fundamental sum are also called Max terms.

Variables			Min terms	Max terms
A	B	C	m_i	M_i
0	0	0	A' B' C' = m 0	A + B + C = M 0
0	0	1	A' B' C = m 1	A + B + C' = M 1
0	1	0	A' B C' = m 2	A + B' + C = M 2
0	1	1	A' B C = m 3	A + B' + C' = M 3
1	0	0	A B' C' = m 4	A' + B + C = M 4
1	0	1	A B' C = m 5	A' + B + C' = M 5
1	1	0	A B C' = m 6	A' + B' + C = M 6
1	1	1	A B C = m 7	A' + B' + C' = M 7

Logic circuit:

The logic circuit is got by drawing an OR-AND network. An example of Product of Sum logic circuit is shown in figure below.



Conversion between SOP and POS:

In SOP, each one at output gives one AND term which is finally ORed. In POS, each zero gives one OR term which is finally ANDed. Thus SOP and POS occupy complementary locations in a truth table and one representation can be obtained from the other by

(i) Identifying complementary locations,

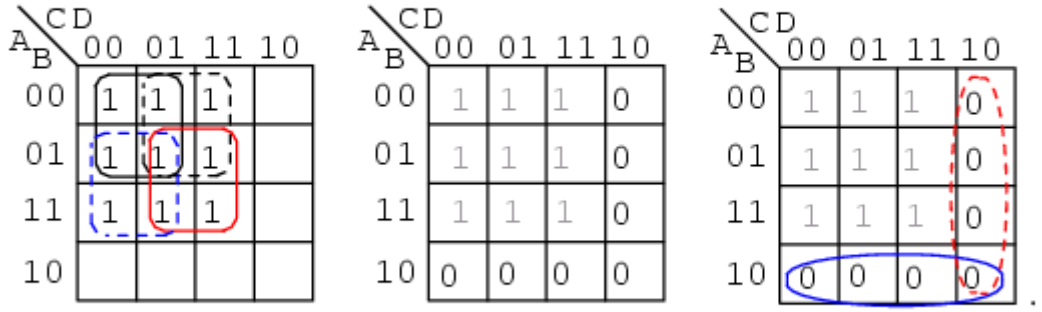
(ii) Changing minterm to maxterm or reverse, and finally

(iii) Changing summation by product or reverse.

Product-of-Sums simplification:

Simplification of POS is possible using K-map. One can use a similar technique as followed in SOP representation but by forming largest group of zeros and then replacing each group by a sum term. The variable going in the formation of sum term is inverted if it remains constant with a value 1 in the group and it is not inverted if that value is 0. Finally, all the sum terms are ANDed to get simplest POS form.

$$\text{Out} = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}CD + \bar{A}B\bar{C}\bar{D} + \bar{A}B\bar{C}D + \bar{A}BCD + AB\bar{C}\bar{D} + AB\bar{C}D + ABCD$$



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

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

Duality:

Given a logic circuit, we can find its dual circuit as follows: Change each AND gate to an OR gate, change each OR gate to an AND gate, and complement all input-output signals. An equivalent statement of duality is this: Change each NAND gate to a NOR gate, change each NOR gate to a NAND gate, and complement all input-output signals.