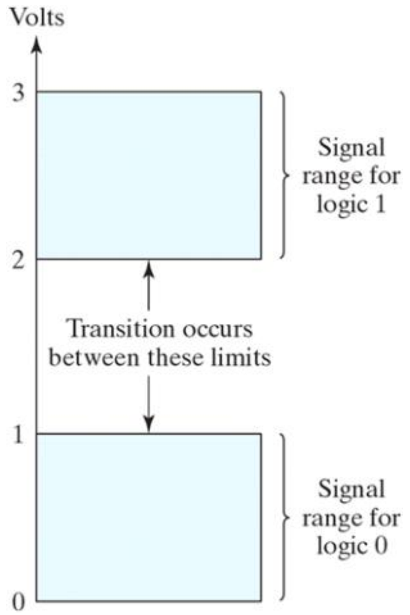
Binary Logic and Gates:

September 11, 2018 1:27 PM

We study Boolean algebra as foundation for designing and analyzing digital systems!

- Binary variables take on one of two values.



Basic logical operators

September 11, 2018 2:24 PM

- Basic logical operators are the logic functions AND, OR and NOT. Logical operators operate on binary values and binary variables.

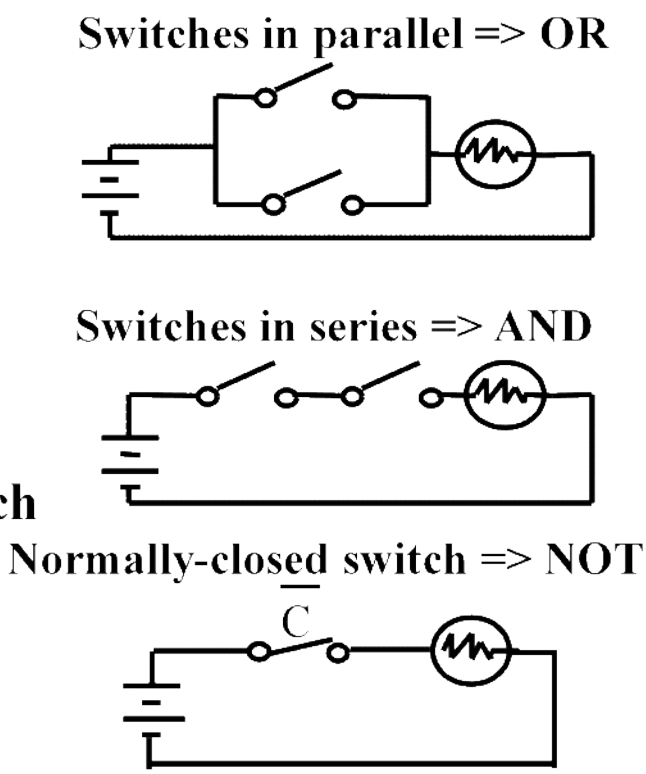
Operator Definitions

September 11, 2018 1:54 PM

Operations are defined on the values "0" and "1" for each operator:

AND	OR	NOT
$0 \cdot 0 = 0$	$0 + 0 = 0$	$\overline{0} = 1$
$0 \cdot 1 = 0$	$0 + 1 = 1$	$\overline{1} = 0$
$1 \cdot 0 = 0$	$1 + 0 = 1$	
$1 \cdot 1 = 1$	$1 + 1 = 1$	

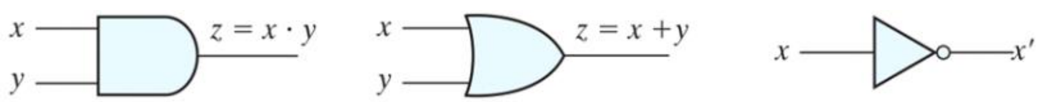
- **Using Switches**
  - For inputs:
    - logic 1 is <u>switch closed</u>
    - logic 0 is <u>switch open</u>
  - For outputs:
    - logic 1 is <u>light on</u>
    - logic 0 is <u>light off</u>.
  - NOT uses a switch such that:
    - logic 1 is <u>switch open</u>
    - logic 0 is <u>switch closed</u>



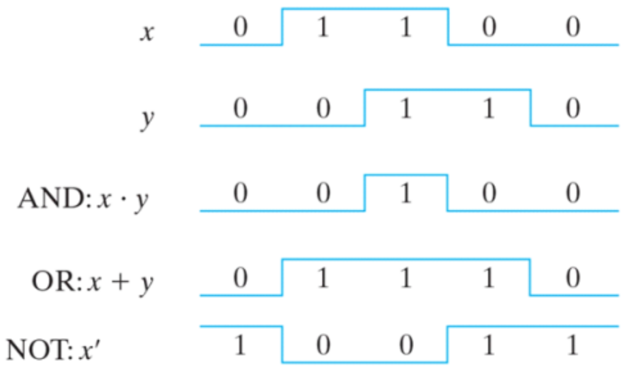
Logic gates

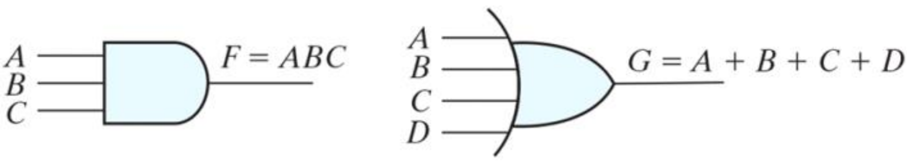
September 11, 2018 2:26 PM

- <u>Logic gates</u> implement logic functions.



(a) Two-input AND gate (b) Two-input OR gate (c) NOT gate or inverter





(a) Three-input AND gate (b) Four-input OR gate

Boolean Algebra

September 11, 2018 1:39 PM

Boolean Algebra: a useful mathematical system for specifying and transforming logic functions.

Postulates and Theorems of Boolean Algebra.

Postulate 2	(a)	$x + 0 = x$	(b)	$x \cdot 1 = x$
Postulate 5	(a)	$x + x' = 1$	(b)	$x \cdot x' = 0$
Theorem 1	(a)	$x + x = x$	(b)	$x \cdot x = x$
Theorem 2	(a)	$x + 1 = 1$	(b)	$x \cdot 0 = 0$
Theorem 3, involution		$(x')' = x$		
Postulate 3, commutative	(a)	$x + y = y + x$	(b)	$xy = yx$
Theorem 4, associative	(a)	$x + (y + z) = (x + y) + z$	(b)	$x(yz) = (xy)z$
Postulate 4, distributive	(a)	$x(y + z) = xy + xz$	(b)	$x + yz = (x + y)(x + z)$
Theorem 5, DeMorgan	(a)	$(x + y)' = x'y'$	(b)	$(xy)' = x' + y'$
Theorem 6, absorption	(a)	$x + xy = x$	(b)	$x(x + y) = x$

Boolean Functions

September 11, 2018 1:39 PM

Any Boolean function can be represented by an expression (consisting of binary variables, the constants 0 and 1).

For example, consider the function F_1 represented by the expression:

$F_1 = x + y'.z$

Function F_1 has three variables x, y, and z, where these variables and/or their complements may appear in a Boolean function.

- A **literal** is a Boolean variable or its complement in a Boolean function.

- A **term** is the expression formed by literals and operations at one level.

For example, F_1 has three literals x, y' and z and two terms

Why the Number of literals and terms are important in a Boolean expression?

The number of lierals in a term shows the number of inputs for a specific gate. Also, the number of terms shows the number og gates needed to implement a Boolean expression using gates.

Example: Find the number of literals and terms in the following expression:

$F = x.y + x'.z + y.z$

Truth Tables

September 11, 2018 1:58 PM

Consider the function F represented by the expression:

Now we want to see the value of this function for all different combinations of variables:

Truth table: a tabular listing of the values of a function for all possible combinations of values on its arguments

- In general, the truth table has rows, where n is the number of variables used in the function expression.
- It gives all different combinations of binary values for each variable.
- A Boolean expression can also be implemented by logic gates

Truth Tables - Cont'd

September 11, 2018 1:58 PM

- The truth table representation of a Boolean function is unique, while the Boolean expression is not unique.

  For example consider the following Boolean functions.

  $F_2 = x'.y'.z + x'.y.z + x.y'$

  $F_3 = x'.z + x.y'$

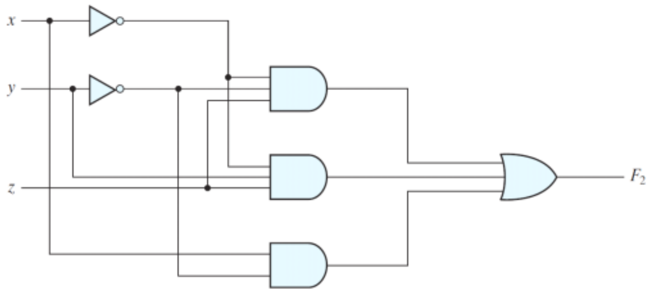
  The truth table for these functions are:

x	y	z	F_2	F_3
0	0	0		
0	0	1		
0	1	0		
0	1	1		
1	0	0		
1	0	1		
1	1	0		
1	1	1		

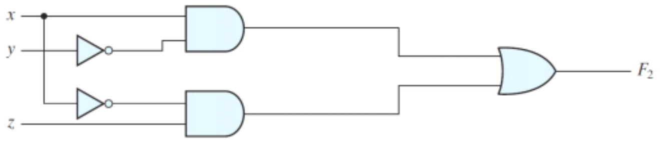
Function Simplification

September 11, 2018 2:06 PM

Lets look at the implementation of functions F_1 and F_2 using logic gates.



$F_2 = x'y'z + x'yz + xy'$



$F_2 = xy' + x'z$

- To implement a Boolean function each term requires a gate and each variable within the term designates an input to the gate.
- Reducing the number of terms and/or literals, the implementation of the expression will be simplified.

Function Simplification - Cont'd

September 13, 2018 1:20 PM

Example: $F = x + x'.y$

Example: G= $x.y + x'.z + y.z$

Complement of a Function

September 11, 2018 3:18 PM

Complement of a function can be obtained by using DeMorgan's theorem in general case as follows:

$$(A + B + C + \cdots E)' = A'.B'.C' \ldots E'$$

$$(A.B.C \ldots E)' = A' + B' + C' + \cdots E'$$

Example: $F = x.(y'.z' + y.z)$

Canonical and Standard Forms

September 13, 2018 1:34 PM

Minterm: A *minterm* is a product term in a Boolean function in which every variable is present is either in normal or in complemented form. It is also called standard product.

- n variables can be combined to form 2^n minterms.
- A symbol for each minterm is of the form m_j, where the subscript j denotes the decimal equivalent of the binary number of the minterm designated.

Maxterm: A *maxterm* is a sum term in a Boolean function in which every variable is present is either in normal or in complemented form. It is also called standard sum.

- n variables can be combined to form 2^n maxterms.
- A symbol for each maxterm is of the form M_j, where the subscript j denotes the decimal equivalent of the binary number of the maxterm designated.

Minterms and Maxterms for Three Binary Variables

			Minterms		Maxterms	
x	**y**	**z**	**Term**	**Designation**	**Term**	**Designation**
0	0	0	$x'y'z'$	m_0	$x + y + z$	M_0
0	0	1	$x'y'z$	m_1	$x + y + z'$	M_1
0	1	0	$x'yz'$	m_2	$x + y' + z$	M_2
0	1	1	$x'yz$	m_3	$x + y' + z'$	M_3
1	0	0	$xy'z'$	m_4	$x' + y + z$	M_4
1	0	1	$xy'z$	m_5	$x' + y + z'$	M_5
1	1	0	xyz'	m_6	$x' + y' + z$	M_6
1	1	1	xyz	m_7	$x' + y' + z'$	M_7

Canonical and Standard Forms - Cont'd

September 13, 2018 3:05 PM

- A Boolean function can be expressed algebraically from a given truth table by forming a minterm for each combination of the variables that produces a 1 in the function and then taking the OR of all those terms.

Example: Function f_1 and f_2 can be defined from the following truth table using minterms as follows:

Functions of Three Variables

x	y	z	Function f_1	Function f_2
0	0	0	0	0
0	0	1	1	0
0	1	0	0	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

- A Boolean function can be expressed algebraically from a given truth table by forming a maxterm for each combination of the variables that produces a 0 in the function and then taking the AND of all those terms.

Example: Function f_1 and f_2 can be defined from the above truth table using maxterms as follows:

Canonical form: Boolean functions expressed as a sum of minterms or product of maxterms are said to be in canonical form.

Canonical and Standard Forms - Cont'd

September 13, 2018 3:34 PM

Example: Express function $F = A + B'C$ as a sum of minterms:

1- Method 1: Using truth table

2- Using Boolean Algebra

Example: Express function $F = xy + x'z$ as a sum of minterms:

1- Method 1: Using truth table

1- Using Boolean Algebra

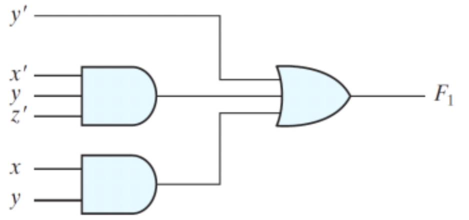
Sum of Products (SOP) and product of sums (POS)

September 13, 2018 4:23 PM

There are two standard form of representations for Boolean expression.

1- The **sum of products (SOP)** is a Boolean expression in which different product terms of inputs (ANDing) are being summed together (ORing). The function F_1 in the below is in the form of SOP:

$F_1 = y' + xy + x'yz'$

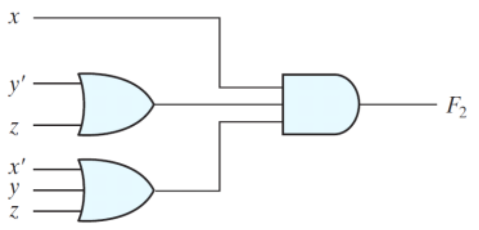


Types of Sum Of Product (SOP) Forms:

- Canonical SOP Form
- Non-Canonical SOP Form
- Minimal SOP Form

2- The **product of sums (POS)** is a Boolean expression in which products (ANDing) of different sum terms of inputs (ORing) are taken. The function F_2 in the below is in the form of POS.

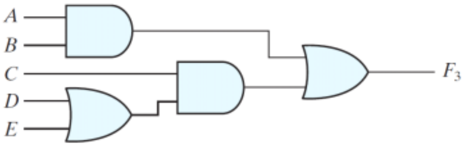
$F_2 = x(y' + z)(x' + y + z')$



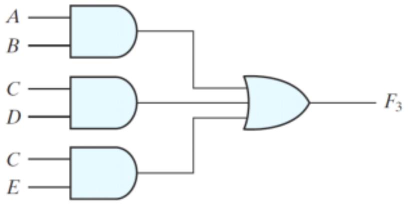
Types of Product of Sums (POS) Forms:

- Canonical POS Form
- Non-Canonical POS Form
- Minimal POS Form

Non-Standard form of a Boolean expression:



(a) $AB + C(D + E)$



(b) $AB + CD + CE$