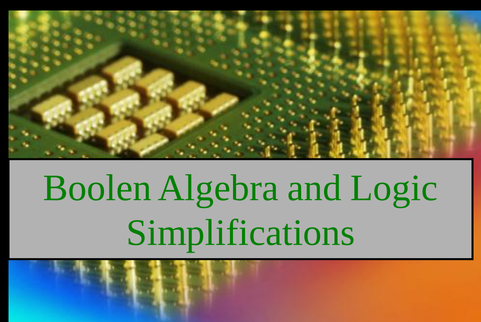
EEE251

Logic Design and Circuits

Assist.Prof. Yavuz BAYAM



Boolen Algebra and Logic Simplifications

October 23, 2012

Introduction

- The fundamental of digital electronic bases on hypothesis.
- Hypothesis can not be both true and false at the same time. It can be only true or only false.
- For example; ‘Water is frozen under 0^0C’

  ‘Sun revolves around the earth’

These ideas can be evaluated as The first idea is ‘true’ and the second one is ‘false’. Therefore, these ideas are accepted as hypothesis.

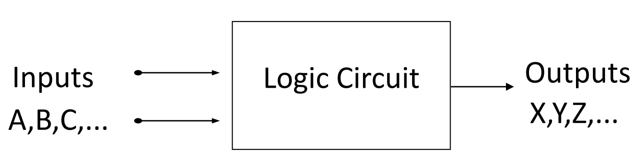
- ‘People who don not eat properly get sick’; this idea can not be evaluated as hypothesis because the only reason is not unhealthy feed to be sick (genetical, environmental conditions, etc.).
- The ideas which can not be identified as ‘true’ or ‘false’ are not not defined as hypothesis.

Boolean Algebra

- The hypothesis which can not be divided simpler hypothesis is **‘simple hypothesis’.**
- **‘complex hypothesis’** are obtained from simple hypothesis.
- The fundamental of Boolean Algebra bases on expressions of simple and complex hypothesis.
- Firstly, Boolean algebra was developed in 1854 by mathematician George Boole (1815-1864), then Peono, Whitehead, Bertrand Russell ve other mathematicians developed the Boolean Algebra used in digital electronic.
- Definitions and rules of Boolean algebra (postulates) were surveyed by E.V. Huntington in 1904.
- In 1938, the idea - ‘Boolean algebra can be used in switching systems’ was declared by Claude E. Shannon after Boolean algebra rules applied in electronic equipments.

Boolean Rules

- Logic circuits works in binary system and inputs / outputs get one of ‘0’ or ‘1’ values.
- Boolean rules are one of the methods used in the simplification of logic circuits. By simplifying logic equations with Boolean rules, the best circuit can be determined.
- In other words; **‘Boolean rules’** is also used for explaining the effetcs of inputs of digital circuits.
- In the equations written for operation of logic circuits, A, B, C, D, etc. characters are generally used as input values and X, Y, W, Z, etc. are generally used for outputs.



Important Boolean Rules

- Boolean rules is a symbolic system uses 'AND', 'OR' and 'NOT' basic logical operations
- Basic logical operations are binary sistems and thus they can be explained with two states are inverse each other:

  True – False, Yes – No, Open – Close, '1' – '0', etc.

- At the begining it was not pratctical system, but later it is extensively used and named as **'Boolean Algebra or Rules**'
- After integration of Boolean rules binary system, the fundamentals of digital electronic.
- As each system has its own rules, Boolean algebra has its own rules.

Basic Boolean Rules

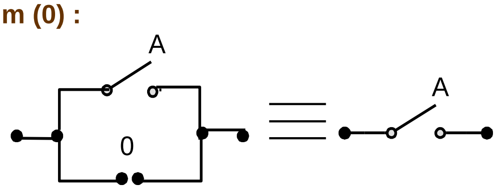
- Basic specifications of Boolean algebra: identity element, unit element, absorbing element, inverse element.

Identity Element in Sum (0) :

A + 0 = A

0 + 0 = 0

1 + 0 = 1

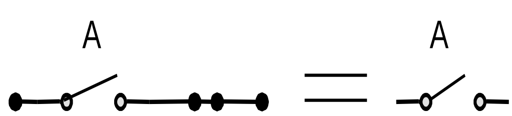


Identity Element in Multiplication (1) :

A . 1 = A

0 . 1 = 0

1 . 1 = 1



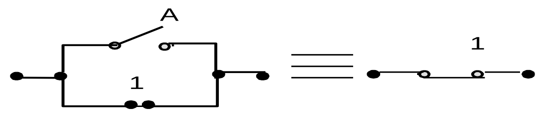
Basic Boolean Rules

Unit Element in Sum:

$A + 1 = 1$

$0 + 1 = 1$

$1 + 1 = 1$

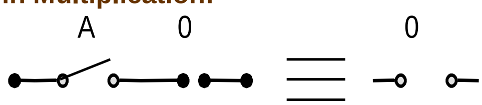


Absorbing Element in Multiplication:

$A \, . \, 0 = 0$

$0 \, . \, 0 = 0$

$1 \, . \, 0 = 0$



inverse Element:

If one variable is ‘0’, its inverse is ‘1’, and if the variable is ‘1’, its inverse is ‘0’. The inverse of one variable is shown with line or apostrophe:

$A = 0 \;\; => A' = 1$, $A = 1 \;\; => \;\; A' = 0$

The inverse of inverse of a variable is equal to itself: $(A'' = A)$.

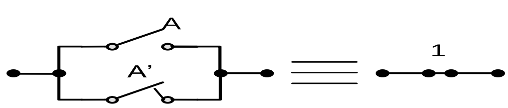
Basic Boolean Rules

1- The Operations of Summation and Multiplication :

$A + A' = 1$

$0 + 1 = 1$

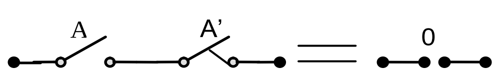
$1 + 0 = 1$



$A \, . \, A' = 0$

$0 \, . \, 1 = 0$

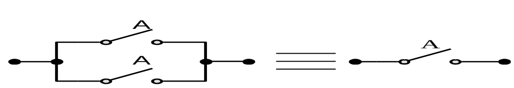
$1 \, . \, 0 = 0$



$A + A = A$

$0 + 0 = 0$

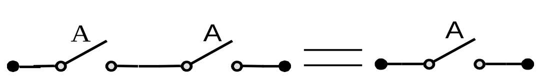
$1 + 1 = 1$



$A \, . \, A = A$

$0 \, . \, 0 = 0$

$1 \, . \, 1 = 1$



Boolean Addition

In Boolean algebra, a **variable** is a symbol used to represent an action, a condition, or data. A single variable can only have a value of 1 or 0.

The **complement** represents the inverse of a variable and is indicated with an overbar. Thus, the complement of A is $\overline{A}$.

A **literal** is a variable or its complement.

Addition is equivalent to the OR operation. The sum term is 1 if one or more if the literals are 1. The sum term is zero only if each literal is 0.

Example Determine the values of A, B, and C that make the sum term of the expression $\overline{A} + B + \overline{C} = 0$?

Solution Each literal must = 0; therefore $A = 1$, $B = 0$ and $C = 1$.

Boolean Multiplication

In Boolean algebra, multiplication is equivalent to the AND operation. The product of literals forms a product term. The product term will be 1 only if all of the literals are 1.

Example What are the values of the A, B and C if the product term of $A \cdot \overline{B} \cdot \overline{C} = 1$?

Solution Each literal must = 1; therefore $A = 1$, $B = 0$ and $C = 0$.

Basic Boolean Rules

2- Identity:

- This rule in Boolean Algebra is a different rule than other arithmatic operations

a) A + A = A (A+A+A+........+A = A) , b) A . A = A (A.A.A.A...........A = A)

3- Commutative Law:

- Comutative law in Boolean algebra is same with law used in arithmetic operations.

a) A + B = B + A   b) A . B = B . A

Commutative Laws

The **commutative laws** are applied to addition and multiplication. For addition, the commutative law states

In terms of the result, the order in which variables are ORed makes no difference.

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

For multiplication, the commutative law states

In terms of the result, the order in which variables are ANDed makes no difference.

$$\boxed{AB = BA}$$

Basic Boolean Rules

4- Associate Law:

- It is same with the rule used in Arithmetic operations.

a) (A + B) + C = A + (B + C) = A+B+C   b) (A . B) . C = A . (B . C) = A . B . C

5- Distributive Law:

- It is used in Boolean Algebra.

a) A . (B+C) = (A . B) + (A . C)   b) (A+B) . (A+C) = A+ (B . C)

6- Redundance Law:

- It is only used in Boolean Algebra.

a) A+ A . B = A   b) A . (A+B) = A

Associative Laws

The **associative laws** are also applied to addition and multiplication. For addition, the associative law states

When ORing more than two variables, the result is the same regardless of the grouping of the variables.

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

For multiplication, the associative law states

When ANDing more than two variables, the result is the same regardless of the grouping of the variables.

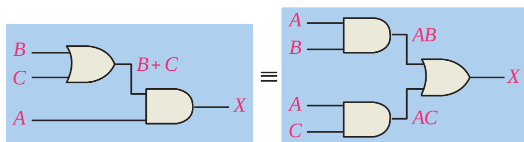
$$\boxed{A(BC) = (AB)C}$$

Distributive Law

The **distributive law** is the factoring law. A common variable can be factored from an expression just as in ordinary algebra. That is

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

The distributive law can be illustrated with equivalent circuits:



$A(B + C)$ $\quad$ $AB + AC$

Basic Boolean Rules

7- Minimization Law:

This is a kind of simplification rule.

a) A + A' . B = A + B

b) A .(A'+B) = A . B

8- De Morgan's Law:

It is used for siplifying of logic equations using 'NOR' and 'NAND' operations

a) $\overline{A.B}$ =A'+B'

b) $\overline{A+B}$ = A'.B'

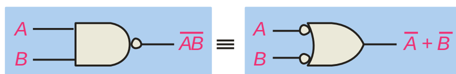
DeMorgan’s Theorem

DeMorgan’s 1st Theorem

The complement of a product of variables is equal to the sum of the complemented variables.

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

Applying DeMorgan’s first theorem to gates:



NAND Negative-OR

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

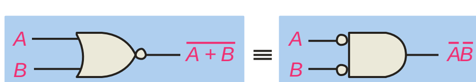
DeMorgan’s Theorem

DeMorgan’s 2nd Theorem

The complement of a sum of variables is equal to the product of the complemented variables.

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

Applying DeMorgan’s second theorem to gates:



NOR Negative-AND

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

DeMorgan's Theorem

Example Apply DeMorgan's theorem to remove the overbar covering both terms from the expression $X = \overline{\overline{C} + D}$.

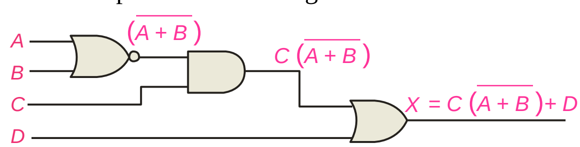
Solution To apply DeMorgan's theorem to the expression, you can break the overbar covering both terms and change the sign between the terms. This results in $X = \overline{\overline{C}} \cdot \overline{D}$. Deleting the double bar gives $X = C \cdot \overline{D}$.

Boolean Analysis of Logic Circuits

Combinational logic circuits can be analyzed by writing the expression for each gate and combining the expressions according to the rules for Boolean algebra.

Example Apply Boolean algebra to derive the expression for X.

Solution Write the expression for each gate:



Applying DeMorgan's theorem and the distribution law:

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

Rules of Boolean Algebra

1. $A + 0 = A$
2. $A + 1 = 1$
3. $A \cdot 0 = 0$
4. $A \cdot 1 = 1$
5. $A + A = A$
6. $A + \overline{A} = 1$
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. $(A + B)(A + C) = A + BC$

Simplification of Logic equations using Boolean Rules

- Complex logic equations or expressions can be simplified using mathematical rules in Boolean
- Electronic circuits created from simplified logic expressions can be cheaper and simpler.
- ‘ ' ’ punctuation is used to represent the inverse of variables in logic operations, ‘--’ punctuation is used to represent the inverse of conjunct expressions.

Simplification of Logic equations using Boolean Rules

Example: Prove that the equation A+ A . B = A is true

$$A+A.B = A\,\underbrace{(1+B)}_{1} = A.1 = A \qquad (1+B = 1)$$

Example: Prove that the equation A . (A+B) = A is true

$$A.(A+B) = \underbrace{A.A}_{A} + A.B = A + A.B = A.\underbrace{(1+B)}_{1} = A.1 = A$$

Example: Prove that the equation (A+B) . (A+C) = A + (B.C) is true

$$(A+B) . (A+C) = \underbrace{A . A}_{A} + A . C + B . A + B . C = \underline{\underline{A}} + \underline{\underline{A}}.C + \underline{\underline{A}}.B + B.C$$

$$= A\,\underbrace{(1+C+B)}_{1} + B.C$$

$$= A.1 + B.C = A + B.C$$

Simplification of Logic equations using Boolean Rules

Example: Prove that the equation A + A' . B = A + B is true

$$A + A'.B = \overline{\overline{A + A'.B}} = \overline{A'.\overline{(A'.B)}} = \overline{A'.(\underbrace{A''}_{A}+B')}$$

$$= \overline{A'.(A+B')} = \overline{\underbrace{A'.A}_{0} + A'.B'} = \overline{A'.B'} = \overline{\overline{A+B}} = A+B$$

Example: Prove that the equation A .(A'+B) = A . B is true

$$A.(A'+B) = \underbrace{A.A'}_{0} + A.B = 0 + A.B = A.B$$

Simplification of Logic equations using Boolean Rules

Example: Prove that the equation A+B+C = A’.B’.C’ is true

$$\overline{A+\underbrace{B+C}} = \overline{A+X} = A'.X' = A'.(\overline{B+C}) = A'.(B'.C') = A'.B'.C'$$

(B+C=X) olarak varsayalım.

Example: Simplify the expression F= A’.B+A+A.B

$$A'.B + A + A.B = A.\underbrace{(1+B)}_{1} + A'.B = \overline{\overline{A + A'.B}} = \overline{A'.(\overline{A'.B})} = \overline{A'.(\underbrace{A''}_{A}+B')}$$

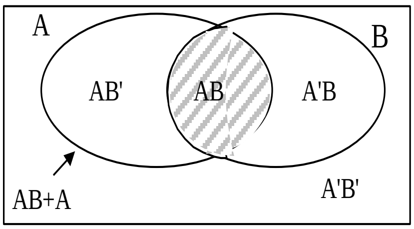
$$= \overline{A'.\underbrace{A''}_{A} + A'.B'} = \overline{\underbrace{A'.A}_{0} + A'.B'}$$

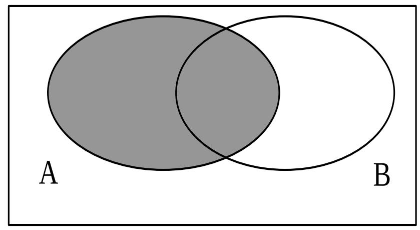
$$= \overline{A'.B'} = \overline{\overline{A+B}} = A+B$$

VENN Diagram

- Venn diagram is a kind of a method to show the relations between Boolean variables using shapes
- In other words; Venn Diagram visually show all possible logical relations between a finite collection of sets.
- In this method, a circle is used to represent each variable. All points in the circle shows thw whole circle. All points out of the circle are represented as ‘reverse of variable’.
- For example; If A=1, A'=0, so all points in A are 1, all out points represents 0.
- The intersection of two sets A and B is swhon as ($A \cap B$) and its union is shown as ($A \cup B$).
- The out of set A is shown as A'

VENN Diagram



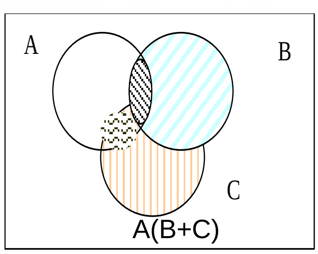


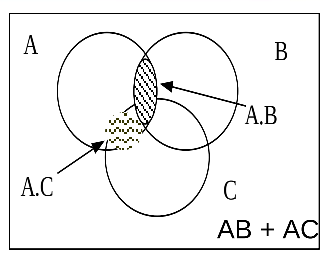
Sets in VENN Diagram

AB +A = A eequation with VENN Diagram.

- The figure shows the equation A+AB
- When the figure is investigated, figure shows that the area of the equation AB+A is same with the area of set A.

VENN Diagram





Showing distribute law with VENN Diagram.

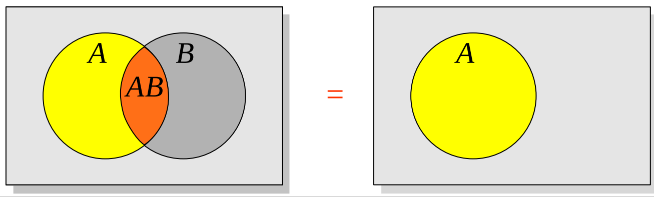
- The figure shows that the equation A.(B+C) = (A.B) + (A.C) (distribute law) with Venn diagram.
- In this diagram, there are 3 sets
- 8 different areas can be defined for a diagram with 3 variables.
- In this example, the union points of A, B, C variables and the area of 'AB+AC' expression.

Rules of Boolean Algebra

Rules of Boolean algebra can be illustrated with *Venn* diagrams. The variable A is shown as an area.

The rule $A + AB = A$ can be illustrated easily with a diagram. Add an overlapping area to represent the variable B.

The overlap region between A and B represents AB.



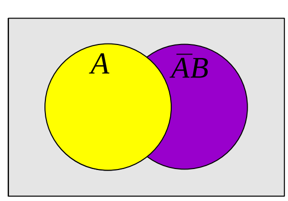
The diagram visually shows that $A + AB = A$. Other rules can be illustrated with the diagrams as well.

Rules of Boolean Algebra

Example Illustrate the rule $A + \bar{A}B = A + B$ with a Venn diagram.

Solution

This time, $\bar{A}$ is represented by the blue area and B again by the red circle. The intersection represents $\bar{A}B$. Notice that $A + \bar{A}B = A + B$



Rules of Boolean Algebra

Rule 12, which states that $(A + B)(A + C) = A + BC$, can be proven by applying earlier rules as follows:

$$\begin{aligned}(A + B)(A + C) &= AA + AC + AB + BC \\ &= A + AC + AB + BC \\ &= A(1 + C + B) + BC \\ &= A \cdot 1 + BC \\ &= A + BC\end{aligned}$$

This rule is a little more complicated, but it can also be shown with a Venn diagram, as given on the following slide…

Three areas represent the variables A, B, and C.

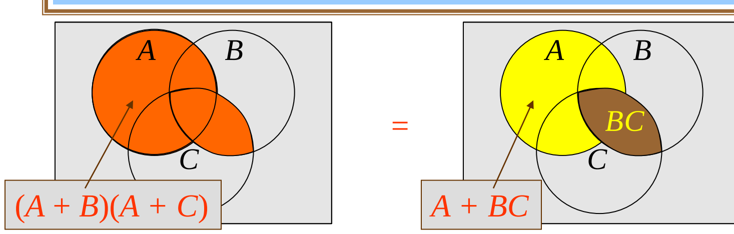
The area representing $A + B$ is shown in yellow.

The area representing $A + C$ is shown in red.

The overlap of red and yellow is shown in orange.

The overlapping area between B and C represents BC.

ORing with A gives the same area as before.



Truth table

- Truth tables show input values with their all combinations and output values according to functions.
- When a truth table is created, a states are appeared according to number of inputs
- If there are ‘n’ input values, there are 2^n different states.
- For example; for a statement with 2 variable there are $2^2 = 4$ different cases, for a statement with 3 variable there are $2^3 = 8$ different cases are obtained.
- **For example:** If the inputs are A and B in a system, what is the output values in the operation of A+B of the system?

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

Truth table

For example: The output of a system with A and B input values is shown with f=A.B equaion. Let’s show the input and output values.

A	B	A . B
0	0	0
0	1	0
1	0	0
1	1	1

Truth table

Example: Lets show the values of A and B variables and the results of all operations according to these inputs in the following truth table.

A	B	A'	B'	A+B	A.B	A+A'	A.A'	B+B'	B.B'	A+B'	A'+B
0	0	1	1	0	0	1	0	1	0	1	1
0	1	1	0	1	0	1	0	1	0	0	1
1	0	0	1	1	0	1	0	1	0	1	0
1	1	0	0	1	1	1	0	1	0	1	1

Truth table

Example : Lets prove the truth table of $\overline{A+B}$ = A' **.** B' De Morgan theorem.

If the values of columns represent both side of the equation are same values, the truth of equation is proved using truth table.

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

Truth table

Example: Let’s prove the equation $\overline{(A . B)}$ = A'+B' with truth table.

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

Example: Prove that F = A + A . B = A

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

Truth table

Example : Let’s prove that F = A . (B+C) = (A . B) + (A . C)

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

Truth table

Example : Let's prove that F = A + A . B + A'. C + C' . D = A + C + D

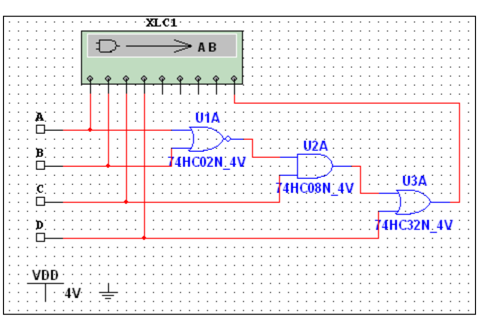
A	A'	B	B'	C	C'	D	D'	AB	A'C	C'D	A+AB+A'C+C'D	A+C+D
0	1	0	1	0	1	0	1	0	0	0	0	0
0	1	0	1	0	1	1	0	0	0	1	1	1
0	1	0	1	1	0	0	1	0	1	0	1	1
0	1	0	1	1	0	1	0	0	1	0	1	1
0	1	1	0	0	1	0	1	0	0	0	0	0
0	1	1	0	0	1	1	0	0	0	1	1	1
0	1	1	0	1	0	0	1	0	1	0	1	1
0	1	1	0	1	0	1	0	0	1	0	1	1
1	0	0	1	0	1	0	1	0	0	0	1	1
1	0	0	1	0	1	1	0	0	0	1	1	1
1	0	0	1	1	0	0	1	0	0	0	1	1
1	0	0	1	1	0	1	0	0	0	0	1	1
1	0	1	0	0	1	0	1	1	0	0	1	1
1	0	1	0	0	1	1	0	1	0	0	1	1
1	0	1	0	1	0	0	1	1	0	0	1	1
1	0	1	0	1	0	1	0	1	0	0	1	1

If the last two column are same, the equation is true.

Boolean Analysis of Logic Circuits

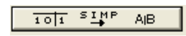
Example Use Multisim to generate the truth table for the circuit in the previous example.

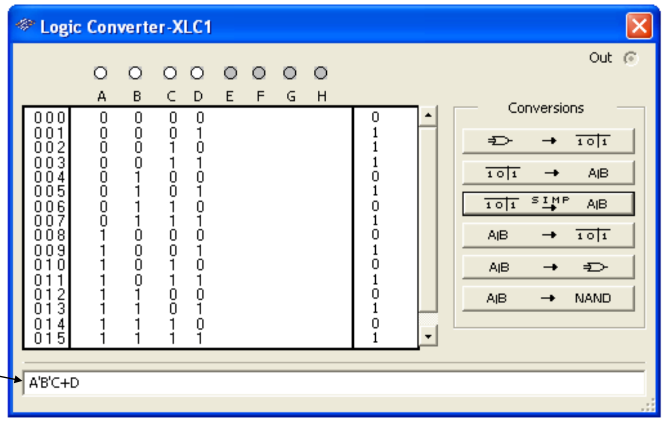
Solution Set up the circuit using the Logic Converter as shown. (Note that the logic converter has no "real-world" counterpart.)



Double-click the Logic Converter top open it. Then click on the conversion bar on the right side to see the truth table for the circuit (see next slide).

Boolean Analysis of Logic Circuits

The simplified logic expression can be viewed by clicking 



Simplified expression

Producing Summation of Minterms and Multiplication of Maxterms

- The simplified function can be expressed with minterms and maxterms for some spesific applications.
- For those expandations, each minterm or maxterm expressions has to have all variables.
- If there is not a certain variable in the combination, (x + x') is added and operated in AND with the combination.

Example : Let’s express f=A+B'C with summation of minterms.

In the first combination there is only A. Therefore B and C variables must be added to expression. Firstly, ‘B’ variabşe is added like;

A=A(B+B')=AB+AB'

is obtained. Then ‘C’ variable is added like;

A = AB(C+C') + AB'(C+C') = ABC + ABC' + AB'C + AB'C'

an equation has all variables is obtained.

Producing Summation of Minterms and Multiplication of Maxterms

- For the second combination B'C, A variable is added;
- B'C = B'C(A+A') = B'CA + B'CA' is obtained.
- Then both combinations are collected;

- f = ABC+ABC'+AB'C+AB'C'+AB'C+A'B'C

İs obtained. AB'C combination was used two times, so the function is smplified using x + x = x Boolean rule;

- f = A+B'C = A'B'C+ AB'C'+ AB'C+ ABC'+ ABC = $m_1+m_4+m_5+m_6+m_7$

- In case of ∑ symbol used;
- F(A,B,C) = ∑ (1,4,5,6,7) expressed.

- To express a simplified function with maxterm combinations, each above steps is made.

Producing Summation of Minterms and Multiplication of Maxterms

- **Example:** Let's express F = (A'+B).(A+C).(B+C) with maxterm.

One variable is missed in each combination;

- A'+B = A' + B + CC' = (A'+B+C) . (A'+B+C')
- A+C = A + C + BB' = (A+B+C) . (A+B'+C)
- B+C = B + C + AA' = (A+B+C) . (A'+B+C) are obtained.

These equations are collected (if there are two same terms, that is written only once);

- F = (A+B+C) . (A+B'+C) . (A'+B+C) . (A'+B+C')
- = $M_0.M_2.M_4.M_5$ function is produced.

- '∏' is used to show maxterm expression;

- F(A,B,C) = ∏ (0,2,4,5)

Producing Summation of Minterms and Multiplication of Maxterms

- When ‘Minterm’ and ‘Maxterm’ are investigated for one function, you can see that minterm and maxterm are complements of each other.

- Because, while '1‘s are taken to produce minterm expression, for maxterm '0' values are taken.
- For example; $f_{(A,B,C)} = \sum(1,4,5,6,7)$
  $= m_1 + m_4 + m_5 + m_6 + m_7$

The complement of this function is;

$$f'_{(A,B,C)} = \sum(0,2,3) = m_0+m_2+m_3$$

- Equal value of f’ is found using De Morgan rule,
- $f' = m_0+m_2+m_3 \Rightarrow f = m_0+m_2+m_3 = m_0'.m_2'.m_3'$
- $= M_0.M_2.M_3$
- $= \prod(0,2,3)$ is obtained. Therefore $m_i' = M_i$
- In the same way $M_i' = m_i$ can be seen from truth table.

Producing Summation of Minterms and Multiplication of Maxterms

- For converting of minterm and maxterm expressions to each other;
  - ‘$\sum$ and $\prod$ sybols are changed and the numbers are also converted
  - $F(A,B,C) = \prod(0,2,4,5)$
  - changing the Maxterm expression to Minterm
  - $f(A,B,C) = \sum(1,3,6,7)$

SOP and POS forms

Boolean expressions can be written in the **sum-of-products** form (**SOP**) or in the **product-of-sums** form (**POS**). These forms can simplify the implementation of combinational logic, particularly with PLDs. In both forms, an overbar cannot extend over more than one variable.

An expression is in SOP form when two or more product terms are summed as in the following examples:

$\overline{A}\,\overline{B}\,\overline{C} + A\,B$ $\quad$ $A\,B\,\overline{C} + \overline{C}\,\overline{D}$ $\quad$ $C\,D + \overline{E}$

An expression is in POS form when two or more sum terms are multiplied as in the following examples:

$(A + B)(\overline{A} + C)$ $\quad$ $(A + B + \overline{C})(B + D)$ $\quad$ $(\overline{A} + B)C$

SOP Standard form

In **SOP standard form**, every variable in the domain must appear in each term. This form is useful for constructing truth tables or for implementing logic in PLDs.

You can expand a nonstandard term to standard form by multiplying the term by a term consisting of the sum of the missing variable and its complement.

Example

Convert $X = \overline{A}\,\overline{B} + A\,B\,C$ to standard form.

Solution

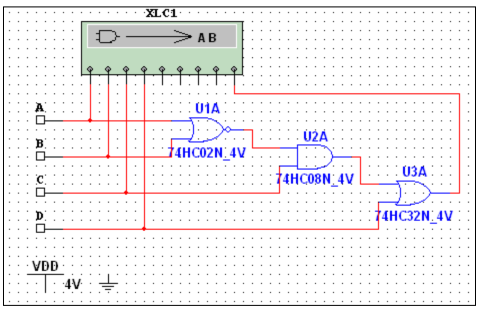
The first term does not include the variable C. Therefore, multiply it by the $(C + \overline{C})$, which = 1:

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

SOP Standard form

The Logic Converter in Multisim can convert a circuit into standard SOP form.

Example Use Multisim to view the logic for the circuit in standard SOP form.

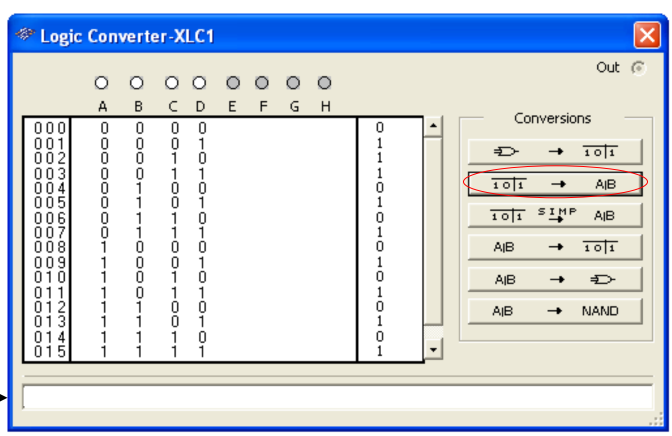


Solution

Click the truth table to logic button on the Logic Converter.

See next slide…

SOP Standard form



SOP Standard form

POS Standard form

In **POS standard form**, every variable in the domain must appear in each sum term of the expression.

You can expand a nonstandard POS expression to standard form by adding the product of the missing variable and its complement and applying rule 12, which states that $(A + B)(A + C) = A + BC$.

Example Convert $X = (\overline{A} + \overline{B})(A + B + C)$ to standard form.

Solution The first sum term does not include the variable C. Therefore, add $C\overline{C}$ and expand the result by rule 12.

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

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