BOOLEAN ANALYSIS OF LOGIC CIRCUITS

Boolean algebra provides a concise way to express the operation of a logic circuit formed by a combination of logic gates so that the output can be determined for various combinations of input values.

Boolean Expression for a Logic Circuit

To derive the Boolean expression for a given logic circuit, begin at the leftmost inputs and work toward the final output, writing the expression for each gate. For the example circuit in Fig.(1), the Boolean expression is determined as follows:

1. *The expression for the left-most AND gate with inputs C and D is CD.*
2. *The output of the left-most AND gate is one of the inputs to the OR gate and B is the other input. Therefore, the expression for the OR gate is B + CD.*
3. *The output of the OR gate is one of the inputs to the right-most AND gate and A is the other input. Therefore, the expression for this AND gate is A(B + CD), which is the final output expression for the entire circuit.*

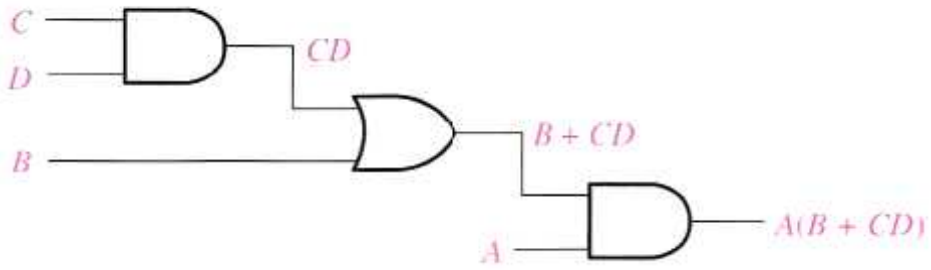

Fig1 A logic circuit showing the development of the Boolean expression for the output

Constructing a Truth Table for a Logic Circuit

Once the Boolean expression for a given logic circuit has been determined, a truth table that shows the output for all possible values of the input variables can be developed. The procedure requires that you evaluate the Boolean expression for all possible combinations of values for the input variables. In the case of the circuit in Fig.(2), there are four input variables (A, B, C, and D) and therefore sixteen ($2^4 = 16$) combinations of values are possible.

Evaluating the expression

To evaluate the expression ***(A(B+CD))****,first find the values of the variables that make the expression equal to 1, using the rules for Boolean addition and multiplication. In this case the expression equals 1 only if* ***A=1*** *and* ***B+CD=1****because*

A(B+CD)=1.1=1

Now determine when the ***B+CD*** *term equals* ***1****. The term* ***B+CD =1*** *if either* ***B=1****or* ***CD=1*** *or if the both* ***B*** *and* ***CD*** *equals* ***1****because*

B+CD = 1+0=1
B+CD = 0+1=1
B+CD = 1+1=1

*The term **CD =1** only if **C=1**and **D=1.***

*The summarize, the expression **A(B+CD)=1**when **A =1**and **B =1** regardless of the values of **C** and **D** or when **A=1**and **C = 1**and **D =1**regardless of the value of **B.** the expression **A(B+CD)=0** for all other value combinations of variables.*

Putting the Results in Truth Table format

The first step is to list the sixteen input variable combinations of 1s and 0s in a binary sequence as shown in Table 4-5. Next, place a 1 in the output column for each combination of input variables that was determined in the evaluation. Finally, place a 0 in the output column for all other combinations of input variables. These results are shown in the truth table in Table 1.

INPUTS				OUTPUT
A	B	C	D	A(B + CD)
0	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	0
0	1	1	1	0
1	0	0	0	0
1	0	0	1	0
1	0	1	0	0
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

SIMPLIFICATION USING BOOLEAN ALGEBRA

A simplified Boolean expression uses the fewest gates possible to implement a given expression.

Ex//

Using Boolean algebra techniques, simplify this expression:

$$AB + A(B + C) + B(B + C)$$

***Step 1**: Apply the distributive law to the second and third terms in the expression, as follows:*

$$AB + AB + AC + BB + BC$$

***Step 2**: Apply rule 7 **(BB = B)** to the fourth term*

$$AB+AB+AC+B+BC$$

***Step 3**: Apply rule 5 **(AB + AB = AB)** to the first two terms.*

$$AB + AC + B + BC$$

***Step 4**: Apply rule 10 **(B + BC = B)** to the last two terms.*

$$AB+AC+B$$

***Step 5**: Apply rule 10 **(AB + B = B)** to the first and third terms.*

$$B+AC$$

At this point the expression is simplified as much as possible.

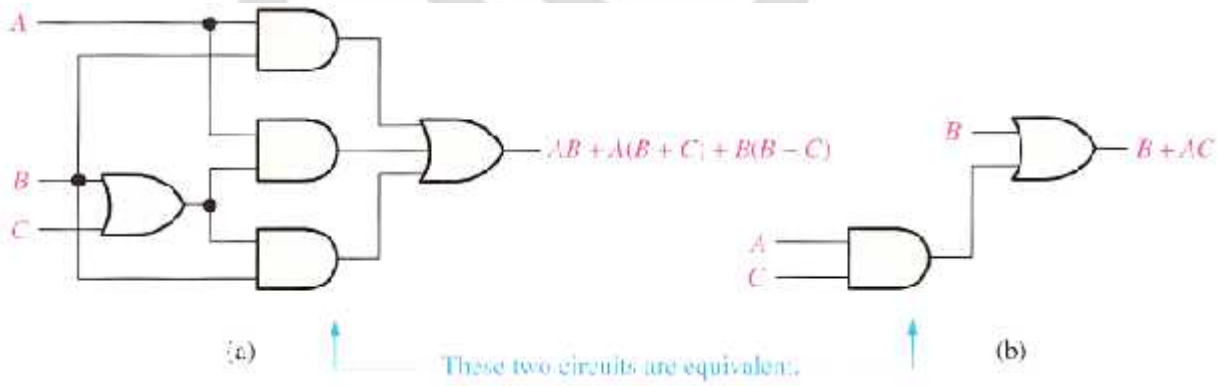

Fig2

EX// simplify the following Boolean expression:

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

Note that brackets and parentheses mean the same thing: the term inside is multiplied (ANDed) with the term outside.

Step1: *Apply the distributive law to the terms within the brackets.*

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

Step2: *apply rule 8 (**BB = 0**) to the second term within the parentheses*

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

Step3: *apply rule3 **(A.0.D =0)***

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

Step4: *apply rule1 (drop the 0)*

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

Step5: *apply the distributive law*

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

Step6: *apply rule 7(**CC=C**)*

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

Step7: *Factor out* $\bar{B}C$

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

Step8: *apply rule 6* $(A+\bar{A}=1)$

$$\bar{B}C$$

STANDARD FORMS OF BOOLEAN EXPRESSIONS

All Boolean expressions, regardless of their form, can be converted into either of two standard forms: the sum-of-products form or the product-ofsums form. Standardization makes the evaluation, simplification, and implementation of Boolean expressions much more systematic and easier.

A binary variable may appear either in its normal form (x) or in its complement form ($\bar{x}$). Now consider two binary variables x and y combined with an ***AND*** *operation. Since each variable may appear in either form, there are four possible combinations: $\bar{x}\,\bar{y}$, $\bar{x}\,y$, $x\,\bar{y}$,and $x\,y$. Each of these four AND terms is called a minterm, or a standard product. In a similar manner,* ***n*** *variables can be combined to form 2^n minterms. The 2^n different minterms may be determined by a method similar to the one shown in Table* ***2*** *for three variables.*

Each minterm is obtained from an AND term of the n variables, with each variable being primed if the corresponding bit of the binary number is a 0 and unprimed if a 1. A symbol for each minterm is also shown in the table and is of the form m_j, where the subscript ***j*** *denotes the decimal equivalent of the binary number of the minterm designated.*

In a similar fashion, n variables forming an OR term, with each variable being primed or unprimed, provide 2^n possible combinations, called maxterms, or standard sums. The eight maxterms for three variables, together with their symbolic designations, are listed in Table 2 . Any 2^n maxterms for ***n*** *variables may be determined similarly. It is important to note that (1) each maxterm is obtained from an OR term of the n variables, with each variable being unprimed if the corresponding bit is a 0 and primed if a 1, and (2) each* ***maxterm is the complement of its corresponding minterm and vice versa.***

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.

For example, the function f1 in Table 2 is determined by expressing the combinations 001, 100, and 111 as $x\,\bar{y}\,\bar{z}$, $x\,\bar{y}\,\bar{z}$, and xyz, respectively. Since each one of these minterms results in ***f1 = 1****, we have*

$$f1 = \bar{x}\,\bar{y}\,z + x\,\bar{y}\,\bar{z} + xyz = m1 + m4 + m7$$

Table 2, Minterms and Maxterms for Three Binary Variables

X	Y	Z	Minterms		Maxterms	
			Term	Designation	Term	Designation
0	0	0	$\overline{X}\,\overline{Y}\,\overline{Z}$	m_0	$X+Y+Z$	M_0
0	0	1	$\overline{X}\,\overline{Y}\,Z$	m_1	$X+Y+\overline{Z}$	M_1
0	1	0	$\overline{X}\,Y\,\overline{Z}$	m_2	$X+\overline{Y}+Z$	M_2
0	1	1	$\overline{X}\,Y\,Z$	m_3	$X+\overline{Y}+\overline{Z}$	M_3
1	0	0	$X\,\overline{Y}\,\overline{Z}$	m_4	$\overline{X}+Y+Z$	M_4
1	0	1	$X\,\overline{Y}\,Z$	m_5	$\overline{X}+Y+\overline{Z}$	M_5
1	1	0	$X\,Y\,\overline{Z}$	m_6	$\overline{X}+\overline{Y}+Z$	M_6
1	1	1	$X\,Y\,Z$	m_7	$\overline{X}+\overline{Y}+\overline{Z}$	M_7

Sum of Minterms

The minterms whose sum defines the Boolean function are those which give the 1's of the function in a truth table.

EX// *Express the Boolean function* $F = A + \overline{B}C$ *as a sum of minterms. The function has three variables: A, B, and C.*

The first term **A** *is missing two variables; therefore,*

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

This function is still missing one variable, so

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

$= ABC + AB\overline{C} + A\overline{B}C + A\overline{B}\,\overline{C}$

The second term B_C is missing one variable; hence,

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

Combining all terms, we have

$F = A + \overline{B}C$

$= ABC + AB\overline{C} + A\overline{B}C + A\overline{B}\,\overline{C} + \overline{A}\overline{B}C$

But AB_C appears twice, and according to theorem 1 (x + x = x), it is possible to remove one of those occurrences. Rearranging the minterms in ascending order, we finally obtain

$F = \overline{A}\overline{B}C + A\overline{B}C + AB\overline{C} + ABC$

$= m1 + m4 + m5 + m6 + m7$

When a Boolean function is in its sum-of-minterms form, it is sometimes convenient to express the function in the following brief notation:

$F(A, B, C) = \sum(1, 4, 5, 6, 7)$

The summation symbol ∑ stands for the ORing of terms

An alternative procedure for deriving the minterms of a Boolean function is to obtain the truth table of the function directly from the algebraic expression and then read the minterms from the truth table.

A	B	C	F
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Product of Maxterms

Each of the 2^{2n} functions of n binary variables can be also expressed as a product of maxterms.

To express a Boolean function as a product of maxterms, it must first be brought into a form of OR terms. This may be done by using the distributive law, $x + yz = (x + y)(x + z)$. *Then any missing variable* x *in each OR term is ORed with* $x\,\bar{x}$ *The procedure is clarified in the following example.*

***EX**//Express the Boolean function* $F = xy + x\bar{z}$ *as a product of maxterms.*

First, convert the function into OR terms by using the distributive law:

$$F = xy + \bar{x}z = (xy + \bar{x})(xy + z)$$
$$= (x + \bar{x})(y + \bar{x})(x + z)(y + z)$$
$$= (\bar{x} + y)(x + z)(y + z)$$

The function has three variables: x, y, and z. Each OR term is missing one variable; therefore,

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

Combining all the terms and removing those which appear more than once, we finally obtain

$$F = (x + y + z)(x + \bar{y} + z)(\bar{x} + y + z)(\bar{x} + y + \bar{z})$$
$$= M0 \quad M2 \quad M4 \quad M5$$

A convenient way to express this function is as follows:

$$F(x, y, z) = \Pi(0, 2, 4, 5)$$

*The product symbol, **Π**, denotes the ANDing of maxterms; the numbers are the indices of the maxterms of the function*

Conversion between Canonical Forms

The complement of a function expressed as the sum of minterms equals the sum of minterms missing from the original function. This is because the original function is expressed by those minterms which make the function equal to 1, whereas its complement is a 1 for those minterms for which the function is a 0. As an example, consider the function

$$F(A, B, C) = \sum(1, 4, 5, 6, 7)$$

This function has a complement that can be expressed as

$$\overline{F}(A, B, C) = \sum(0, 2, 3) = m_0 + m_2 + m_3$$

Now, if we take the complement of $\overline{F}$ by DeMorgan's theorem, we obtain F in a different form:

$$F = \overline{(m_0 + m_2 + m_3)} = \overline{m_0}\,.\,\overline{m_2}\,.\,\overline{m_3} = M_0M_2M_3 = \Pi(0, 2, 3)$$

The last conversion follows from the definition of minterms and maxterms as shown in Table 2 . From the table, it is clear that the following relation holds:

$$\overline{m_j} = M_j$$

*That is, the **maxterm with subscript j is a complement of the minterm with the same subscript j and vice versa.***