By complementing both sides, we get the generalized DeMorgan's theorem:

$$[F(X_1, X_2, ..., X_n)]' = F^D(X_1', X_2', ..., X_n')$$

Amazing!

DUALITY IS GOOD FOR STUDENTS AND AUTHORS

You've seen that duality is the basis for the generalized DeMorgan's theorem. Going forward, it will halve the number of methods you must learn to manipulate and simplify logic functions. It also halved the amount of original material I had to write in these sections!

4.1.6 Standard Representations of Logic Functions

Before moving on to analysis and synthesis of combinational logic functions we'll introduce some necessary nomenclature and notation.

truth table

The most basic representation of a logic function is the *truth table*. Similar in philosophy to the perfect-induction proof method, this brute-force representation simply lists the output of the circuit for every possible input combination. Traditionally, the input combinations are arranged in rows in ascending binary counting order, and the corresponding output values are written in a column next to the rows. The general structure of a 3-variable truth table is shown below in Table 4-4.

The rows are numbered 0–7, corresponding to the binary input combinations, but this numbering is not an essential part of the truth table. The truth table for a particular 3-variable logic function is shown in Table 4-5. Each distinct pattern of eight 0s and 1s in the output column yields a different logic function; there are 2^8 such patterns. Thus, the logic function in Table 4-5 is one of 2^8 different logic functions of three variables.

The truth table for an n-variable logic function has 2^n rows. Obviously, truth tables are practical to write only for logic functions with a small number of variables, say, 10 for students and about 4–5 for everyone else.

Table 4-4 General truth table structure for a 3-variable logic function, F(X,Y,Z).

Row	X	Y	Z	F
0	0	0	0	F(0,0,0)
1	0	0	1	F(0,0,1)
2	0	1	0	F(0,1,0)
3	0	1	1	F(0,1,1)
4	1	0	0	F(1,0,0)
5	1	0	1	F(1,0,1)
6	1	1	0	F(1,1,0)
7	1	1	1	F(1,1,1)

Row	X	Y	Z	F
0	0	0	0	1
1	0	0	1	0
2	0	1	0	0
3	0	1	1	1
4	1	0	0	1
5	1	0	1	0
6	1	1	0	1
7	1	1	1	1

Table 4-5
Truth table for a particular 3-variable logic function, F(X, Y, Z).

The information contained in a truth table can also be conveyed algebraically. To do so, we first need some definitions:

- A *literal* is a variable or the complement of a variable. Examples: X, Y, X′, Y′. — *literal*
- A *product term* is a single literal or a logical product of two or more literals. Examples: Z′, $W \cdot X \cdot Y$, $X \cdot Y' \cdot Z$, $W' \cdot Y' \cdot Z$. — *product term*
- A *sum-of-products expression* is a logical sum of product terms. Example: $Z' + W \cdot X \cdot Y + X \cdot Y' \cdot Z + W' \cdot Y' \cdot Z$. — *sum-of-products expression*
- A *sum term* is a single literal or a logical sum of two or more literals. Examples: Z′, $W + X + Y$, $X + Y' + Z$, $W' + Y' + Z$. — *sum term*
- A *product-of-sums expression* is a logical product of sum terms. Example: $Z' \cdot (W + X + Y) \cdot (X + Y' + Z) \cdot (W' + Y' + Z)$. — *product-of-sums expression*
- A *normal term* is a product or sum term in which no variable appears more than once. A nonnormal term can always be simplified to a constant or a normal term using one of theorems T3, T3′, T5, or T5′. Examples of nonnormal terms: $W \cdot X \cdot X \cdot Y'$, $W + W + X' + Y$, $X \cdot X' \cdot Y$. Examples of normal terms: $W \cdot X \cdot Y'$, $W + X' + Y$. — *normal term*
- An n-variable *minterm* is a normal product term with n literals. There are 2^n such product terms. Some examples of 4-variable minterms: $W' \cdot X' \cdot Y' \cdot Z'$, $W \cdot X \cdot Y' \cdot Z$, $W' \cdot X' \cdot Y \cdot Z'$. — *minterm*
- An n-variable *maxterm* is a normal sum term with n literals. There are 2^n such sum terms. Examples of 4-variable maxterms: $W' + X' + Y' + Z'$, $W + X' + Y' + Z$, $W' + X' + Y + Z'$. — *maxterm*

There is a close correspondence between the truth table and minterms and maxterms. A minterm can be defined as a product term that is 1 in exactly one row of the truth table. Similarly, a maxterm can be defined as a sum term that is 0 in exactly one row of the truth table. Table 4-6 shows this correspondence for a 3-variable truth table.

Table 4-6 Minterms and maxterms for a 3-variable logic function, F(X, Y, Z).

Row	X	Y	Z	F	Minterm	Maxterm
0	0	0	0	F(0,0,0)	$X' \cdot Y' \cdot Z'$	$X + Y + Z$
1	0	0	1	F(0,0,1)	$X' \cdot Y' \cdot Z$	$X + Y + Z'$
2	0	1	0	F(0,1,0)	$X' \cdot Y \cdot Z'$	$X + Y' + Z$
3	0	1	1	F(0,1,1)	$X' \cdot Y \cdot Z$	$X + Y' + Z'$
4	1	0	0	F(1,0,0)	$X \cdot Y' \cdot Z'$	$X' + Y + Z$
5	1	0	1	F(1,0,1)	$X \cdot Y' \cdot Z$	$X' + Y + Z'$
6	1	1	0	F(1,1,0)	$X \cdot Y \cdot Z'$	$X' + Y' + Z$
7	1	1	1	F(1,1,1)	$X \cdot Y \cdot Z$	$X' + Y' + Z'$

minterm number
minterm i

An *n*-variable minterm can be represented by an *n*-bit integer, the *minterm number*. We'll use the name *minterm i* to denote the minterm corresponding to row *i* of the truth table. In minterm *i*, a particular variable appears complemented if the corresponding bit in the binary representation of *i* is 0; otherwise, it is uncomplemented. For example, row 5 has binary representation 101 and the corresponding minterm is $X \cdot Y' \cdot Z$. As you might expect, the correspondence for maxterms is just the opposite: in *maxterm i*, a variable is complemented if the corresponding bit in the binary representation of *i* is 1. Thus, maxterm 5 (101) is $X' + Y + Z'$. Note that all of this makes sense only if we know the number of variables in the truth table, three in the examples.

maxterm i

Based on the correspondence between the truth table and minterms, we can easily create an algebraic representation of a logic function from its truth table. The *canonical sum* of a logic function is a sum of the minterms corresponding to truth-table rows (input combinations) for which the function produces a 1 output. For example, the canonical sum for the logic function in Table 4-5 on page 197 is

canonical sum

$$\begin{aligned} F &= \Sigma_{X,Y,Z}(0,3,4,6,7) \\ &= X' \cdot Y' \cdot Z' + X' \cdot Y \cdot Z + X \cdot Y' \cdot Z' + X \cdot Y \cdot Z' + X \cdot Y \cdot Z \end{aligned}$$

minterm list
on-set

Here, the notation $\Sigma_{X,Y,Z}(0,3,4,6,7)$ is a *minterm list* and means "the sum of minterms 0, 3, 4, 6, and 7 with variables X, Y, and Z." The minterm list is also known as the *on-set* for the logic function. You can visualize that each minterm "turns on" the output for exactly one input combination. Any logic function can be written as a canonical sum.

canonical product

The *canonical product* of a logic function is a product of the maxterms corresponding to input combinations for which the function produces a 0 output. For example, the canonical product for the logic function in Table 4-5 is

$$\begin{aligned} F &= \Pi_{X,Y,Z}(1,2,5) \\ &= (X + Y + Z') \cdot (X + Y' + Z) \cdot (X' + Y + Z') \end{aligned}$$

Here, the notation $\Pi_{X,Y,Z}(1,2,5)$ is a *maxterm list* and means "the product of maxterms 1, 2, and 5 with variables X, Y, and Z." The maxterm list is also known as the *off-set* for the logic function. You can visualize that each maxterm "turns off" the output for exactly one input combination. Any logic function can be written as a canonical product.

maxterm list

off-set

It's easy to convert between a minterm list and a maxterm list. For a function of n variables, the possible minterm and maxterm numbers are in the set $\{0, 1, \ldots, 2^n - 1\}$; a minterm or maxterm list contains a subset of these numbers. To switch between list types, take the set complement, for example,

$$\begin{aligned}
\Sigma_{A,B,C}(0,1,2,3) &= \Pi_{A,B,C}(4,5,6,7) \\
\Sigma_{X,Y}(1) &= \Pi_{X,Y}(0,2,3) \\
\Sigma_{W,X,Y,Z}(0,1,2,3,5,7,11,13) &= \Pi_{W,X,Y,Z}(4,6,8,9,10,12,14,15)
\end{aligned}$$

We have now learned five possible representations for a combinational logic function:

1. A truth table.
2. An algebraic sum of minterms, the canonical sum.
3. A minterm list using the Σ notation.
4. An algebraic product of maxterms, the canonical product.
5. A maxterm list using the Π notation.

Each one of these representations specifies exactly the same information; given any one of them, we can derive the other four using a simple mechanical process.

4.2 Combinational-Circuit Analysis

We analyze a combinational logic circuit by obtaining a formal description of its logic function. Once we have a description of the logic function, a number of other operations are possible:

- We can determine the behavior of the logic circuit for various input combinations.
- We can manipulate an algebraic description to suggest different circuit structures for the logic function.
- We can transform an algebraic description into a standard form corresponding to an available circuit structure. For example, a sum-of-products expression corresponds directly to the circuit structure used in PLAs (programmable logic arrays), and a truth table corresponds to the lookup memory used in most FPGAs (field programmable gate arrays).
- We can use an algebraic description of the circuit's functional behavior in the analysis of a larger system that includes the circuit.