

Digital Gates and Boolean Algebra

A digital circuit is a circuit in which only two logical values are present: 0 and 1. These two values might be represented by the presence or absence of an electrical voltage or a magnetic field or a pulse of light. Devices, called **gates**, are used to compute functions of these Boolean values. These gates form the basis on which all digital computers are built.

The following table lists some of the commonly used digital gates along with their graphical symbol and the mathematical symbol used with Boolean algebra.


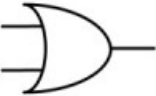

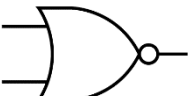

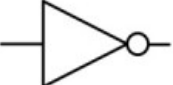
<u>Gate</u>	<u>Graphical Symbol</u>	<u>Boolean Symbol</u>	<u>Multiplication Table</u>												
AND		$A * B$ <i>or</i> AB	<table><tr><td><u>AND</u></td><td> </td><td>0</td><td>1</td></tr><tr><td>0</td><td> </td><td>0</td><td>0</td></tr><tr><td>1</td><td> </td><td>0</td><td>1</td></tr></table>	<u>AND</u>		0	1	0		0	0	1		0	1
<u>AND</u>		0	1												
0		0	0												
1		0	1												
OR		$A + B$	<table><tr><td><u>OR</u></td><td> </td><td>0</td><td>1</td></tr><tr><td>0</td><td> </td><td>0</td><td>1</td></tr><tr><td>1</td><td> </td><td>1</td><td>1</td></tr></table>	<u>OR</u>		0	1	0		0	1	1		1	1
<u>OR</u>		0	1												
0		0	1												
1		1	1												
XOR		$A \oplus B$	<table><tr><td><u>XOR</u></td><td> </td><td>0</td><td>1</td></tr><tr><td>0</td><td> </td><td>0</td><td>1</td></tr><tr><td>1</td><td> </td><td>1</td><td>0</td></tr></table>	<u>XOR</u>		0	1	0		0	1	1		1	0
<u>XOR</u>		0	1												
0		0	1												
1		1	0												
NOR		$\overline{A + B}$ <i>or</i> $!(A + B)$	<table><tr><td><u>NOR</u></td><td> </td><td>0</td><td>1</td></tr><tr><td>0</td><td> </td><td>1</td><td>0</td></tr><tr><td>1</td><td> </td><td>0</td><td>0</td></tr></table>	<u>NOR</u>		0	1	0		1	0	1		0	0
<u>NOR</u>		0	1												
0		1	0												
1		0	0												
NAND		$\overline{A * B}$ <i>or</i> $!(A * B)$	<table><tr><td><u>NAND</u></td><td> </td><td>0</td><td>1</td></tr><tr><td>0</td><td> </td><td>1</td><td>1</td></tr><tr><td>1</td><td> </td><td>1</td><td>0</td></tr></table>	<u>NAND</u>		0	1	0		1	1	1		1	0
<u>NAND</u>		0	1												
0		1	1												
1		1	0												
NOT		\bar{A} <i>or</i> $!A$	<table><tr><td><u>NOT</u></td><td> </td></tr><tr><td>0</td><td> 1</td></tr><tr><td>1</td><td> 0</td></tr></table>	<u>NOT</u>		0	1	1	0						
<u>NOT</u>															
0	1														
1	0														

Table: List of common digital gates

Introduction to Boolean Algebra

To define the digital circuits that can be built by combining gates, an algebra is needed in which variables and functions can take on only the values 0 and 1. Such an algebra was invented by an English mathematician, George Boole (1815-1864), and is called Boolean algebra. Using the symbols listed in the above table, we can enumerate several properties of Boolean algebra.

Identities and Inverses

$$\begin{array}{llllll} A + 0 = A & A + 1 = 1 & A * 0 = 0 & A * 1 = A & A + A = A & A * A = A \\ A + \overline{A} = 1 & A * \overline{A} = 0 & \overline{\overline{A}} = A & & & \end{array}$$

Associative Properties

$$A + (B + C) = (A + B) + C \quad A * (B * C) = (A * B) * C$$

Commutative Properties

$$A + B = B + A \quad A * B = B * A$$

Distributive Properties

$$A * (B + C) = (A * B) + (A * C) \quad A + (B * C) = (A + B) * (A + C) \quad A + (A * B) = A$$

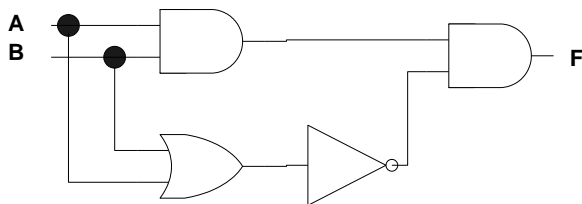
deMorgan's Properties

$$\overline{A + B} = \overline{A} * \overline{B} \quad \overline{A * B} = \overline{A} + \overline{B}$$

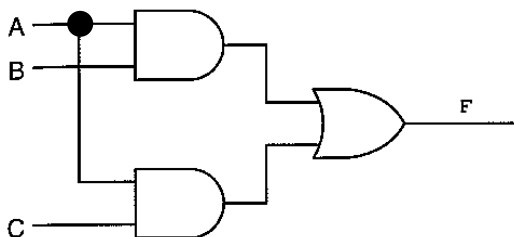
Circuit Simplification using Boolean algebra

Circuit designers often try to reduce the number of gates in their circuits in order to reduce component cost, printed board space, and power consumption. By converting a digital circuit to an algebraic expression, and then simplifying the expression using the algebraic properties listed above, the designer can then convert the simplified expression back into a simpler digital circuit that is equivalent to the original design.

Every digital circuit can be represented by a Boolean expression. For instance, the expressions for the following 2 circuits are listed to the right of each diagram.



$$F = (A * B) * (A + B)$$



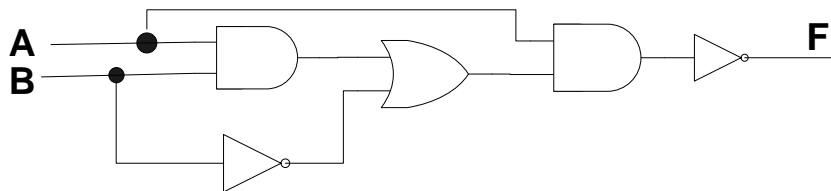
$$F = (A * B) + (A * C)$$

The reverse is also true – every Boolean expression is equivalent to a digital circuit. Can you draw digital circuits that are equivalent to the following Boolean expressions (without simplification)?

$$F = A * (B + C)$$

$$F = A * \overline{(B + (A * B))}$$

Circuit designers often use Boolean algebra to simplify a Boolean expression as much as possible in order to design an equivalent circuit that is simpler and more cost effective than their original design. As an example, we'll start with the digital circuit drawn below:



The Boolean expression equivalent to this circuit is $F = A * \overline{(A * B) + \overline{B}}$

Using Boolean algebra we can simplify the circuit:

$$\begin{aligned}
 F &= A * \overline{(A * B) + \overline{B}} \\
 F &= \overline{\overline{A} + ((A * B) + \overline{B})} && \text{(deMorgan's property)} \\
 F &= \overline{\overline{A} + (\overline{A * B}) * \overline{\overline{B}}} && \text{(deMorgan's property)} \\
 F &= \overline{\overline{A} + (\overline{A * B}) * B} && \text{(one of the rules about inverses)} \\
 F &= \overline{\overline{A} + ((\overline{A} + \overline{B}) * B)} && \text{(deMorgan's property)} \\
 F &= \overline{\overline{A} + ((\overline{A} * B) + (\overline{B} * B))} && \text{(distributive property)} \\
 F &= \overline{\overline{A} + ((\overline{A} * B) + 0)} && \text{(inverses)} \\
 F &= \overline{\overline{A} + (\overline{A} * B)} && \text{(0 is identity for + operator)} \\
 F &= \overline{\overline{A} * (1 + B)} && \text{(distributive property = "factoring")} \\
 F &= \overline{\overline{A} * 1} && \text{(one of the identity rules)} \\
 F &= \overline{\overline{A}} && \text{(1 is identity for * operator)}
 \end{aligned}$$

Thus, the original circuit is equivalent to the circuit: $A \longrightarrow \triangle \longrightarrow F$

Obviously, the new circuit is MUCH simpler than the original! It is interesting to note that the output, F, does not depend upon the value of B at all.

Occasionally, we use a truth table to prove two Boolean expressions (and, thus, two digital circuits) are equivalent. We'll use this technique to show that $A + (B * C) = (A + B) * (A + C)$.

<u>A</u>	<u>B</u>	<u>C</u>	<u>B * C</u>	<u>A + (B * C)</u>	<u>(A + B)</u>	<u>(A + C)</u>	<u>(A + B) * (A + C)</u>
0	0	0	0	0	0	0	0
0	0	1	0	0	0	1	0
0	1	0	0	0	1	0	0
0	1	1	1	1	1	1	1
1	0	0	0	1	1	1	1
1	0	1	0	1	1	1	1
1	1	0	0	1	1	1	1
1	1	1	1	1	1	1	1

Since the A, B, & C columns include every possible combination of 1's and 0's for these 3 variables and since the circled columns are identical, the two Boolean expressions are equivalent.

Designing Digital Circuits using Truth Tables

There is another way truth tables prove to be useful. Each truth table is equivalent to a Boolean expression and digital circuit. Thus, we can design a circuit by specifying the needed output for every combination of input values. We'll design a circuit that implements the following truth table:

Input variables			Output (equals whatever we desire)
<u>A</u>	<u>B</u>	<u>C</u>	<u>F</u>
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

To find the equivalent Boolean circuit, circle every row in the truth table for which the F value equals 1.

Input variables			Output
<u>A</u>	<u>B</u>	<u>C</u>	<u>F</u>
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	0

The desired Boolean expression will be in the "Sum Of Products" (SOP) format. That is, it will contain 1 or more terms connected by the OR operator (+). A term is defined to be one or more variables (or the NOT of a variable) connected by the AND operator. It is customary to use concatenation instead of the "*" character to represent the AND operator.

Thus, an expression in SOP format might look something like: $F = ABC + \overline{A}BC + A\overline{B}C + \overline{A}\overline{B}C + ABC$

For the truth table in the example above, the SOP expression will have 3 terms, one for each circled row. If the value of an input variable in a circled row is 0, the corresponding variable in the expression will have the 'NOT' operator added. Thus, the 1st circled row (4th row in the table) corresponds to the term, $\overline{A}BC$. The entire Boolean expression is $F = \overline{A}BC + A\overline{B}C + ABC$. Can you draw a circuit that is equivalent to this expression? What does this circuit do?

(In a later section, we'll learn about a generalized way to draw a circuit given a SOP expression.)

Exercise

What Boolean expression is equivalent to the following truth table?

Input variables			Output
\overline{A}	\overline{B}	\overline{C}	\overline{F}
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Karnaugh Maps

Boolean expressions in SOP format occur so often in circuit design that a special simplification procedure has been developed that uses a geometric technique called a Karnaugh map. Karnaugh maps may be used to simplify SOP expressions that contain 2, 3, or 4 variables. We'll illustrate the technique using 4 variables.

Suppose we are given the expression, $F = \overline{A}\overline{B}\overline{C}D + \overline{A}B\overline{C}D + A\overline{B}\overline{C}D + ABCD + \overline{A}\overline{B}C\overline{D} + \overline{A}BC\overline{D} + A\overline{B}C\overline{D} + A\overline{B}C\overline{D}$. Although it is certainly possible to simplify this expression using the rules and properties of Boolean algebra, it could prove a long and arduous task. Instead we'll draw the following box with 4 rows and 4 columns and labels.

	00	01	11	10
00				
01				
11				
10				

Box in 3rd row, 4th column has row label '11' and column label '10'. These labels correspond to the term: $ABC\overline{D}$

The row labels represent the possible values for the A and B variables. A '0' corresponds to a variable that has the NOT operator appended. For instance, the fourth row has the label '10' which corresponds to the pair $A\overline{B}$. The column labels represent the possible values for the C and D variables. The square in the 3rd row of the 4th column has a row label of '11' and a column label of '10'. Thus, this little square corresponds to the term $ABC\overline{D}$.

Important note: the sequence of the labels must be 00, 01, 11, & 10. This sequence is known as a "gray code"; each label differs from its two neighbors by exactly one digit.. It is incorrect to label the rows and/or columns using the numerical sequence, 00, 01, 10, & 11.

Now that we've drawn the box with 4 rows and columns and have explained the correspondence between the labels and SOP terms, we need to insert some values in the table. For each term in the SOP expression, insert a 1 in the corresponding box. For the expression, $F = \bar{A}\bar{B}\bar{C}D + \bar{A}BCD + A\bar{B}\bar{C}D + ABCD + \bar{A}\bar{B}C\bar{D} + \bar{A}BC\bar{D} + AB\bar{C}\bar{D} + ABC\bar{D}$, the box should now appear as shown below.

	00	01	11	10
00				1
01		1	1	1
11		1	1	1
10				1

← Example: This box has row label '01' and column label '10'. These labels correspond to the term: $\bar{A}BC\bar{D}$

The result is a Karnaugh map. We're now ready for the simplification procedure. The idea is to enclose every '1' in the boxes within ellipses that obey the rules below. Since each ellipse will correspond to a term in the simplified expression, we want to accomplish the task with as *few* ellipses as possible. Also, it is beneficial to make the ellipses as *large* as possible. So why not just draw 1 large ellipse that encloses all of the 1's? Because, doing so will usually violate one or more of the rules!

Rules

1. An ellipse may NOT enclose a square that is blank.
2. The number of 1's enclosed in an ellipse MUST be a power of 2 (i.e. 1, 2, 4, 8, or 16).
3. It is OK for a 1 to be enclosed in more than one ellipse.
4. If more than 1 square is enclosed in an ellipse, they must be horizontally or vertically adjacent.
5. The left-most column is considered to be adjacent to the right-most column. Similarly, the top row is adjacent to the bottom row.

Goals

1. Use a minimum of ellipses to enclose *every* '1'.
2. Make the ellipses as big as possible.

Following the rules above, we can construct the two ellipses shown below.

	00	01	11	10
00				1
01		1	1	1
11		1	1	1
10				1

Both ellipses only enclose squares that contain a '1'; and the number of 1's is a power of 2 (in these cases, 4). You should convince yourself that at least 2 ellipses are required and these two are the largest such ellipses that enclose all of the 1's. Since there are 2 ellipses, the simplified SOP expression will have 2 terms.

For each ellipse, observe which label values are consistently the same for every enclosed square. For instance, the middle circle encloses 4 squares. The row labels are 01 and 11; and we note that the second digit (which you should recall corresponds to the B variable) is 1 both times. Similarly, the second digit in the 2 column labels is always 1 for all 4 squares. The middle circle corresponds to the term: BD .

The long ellipse in the right-most column encloses 4 squares that occupy all 4 rows – none of the digits in the row labels are consistently the same for every enclosed square. However, the column label is always '10' for every square. Thus, the term that corresponds to this ellipse is $C\bar{D}$.

The simplified expression is thus, $F = BD + C\bar{D}$. I think you will agree that this is MUCH better than the expression we started with: $F = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + \bar{A}B\bar{C}\bar{D} + \bar{A}BC\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}CD + \bar{A}B\bar{C}D + \bar{A}BCD$. Just to emphasize the point, these two expressions are functionally equivalent. Which expression would you prefer to convert into a digital circuit?

Example

Recall the third and fifth rule for drawing ellipses:

- It is OK for a 1 to be enclosed in more than one ellipse.
- The left-most column is considered to be adjacent to the right-most column. Similarly, the top row is adjacent to the bottom row.

These will prove beneficial in the following example.

	00	01	11	10
00		1	1	
01				
11		1		
10		1	1	

Since the top row and bottom rows are considered adjacent (rule 5), one ellipse wraps around the top edge to the bottom edge to enclose four 1's. Although the square at '1101' could be enclosed with an ellipse of size 1, it is beneficial to extend the ellipse to include the square immediately below it, so that the ellipse size is now, 2 (using rule 3). The simplified Boolean expression is $F = \bar{B}D + A\bar{C}\bar{D}$.