TABLE 1.5 $A + \overline{A}B = A + B$ Truth Table

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

Augustus DeMorgan, another nineteenth century English mathematician, worked out a logical transformation that is known as DeMorgan's law, which has great utility in simplifying and re-expressing Boolean equations. Simply put, DeMorgan's law states

$$A + B = \overline{\overline{A} \& \overline{B}} \quad \text{and} \quad A \& B = \overline{\overline{A} + \overline{B}}$$

These transformations are very useful, because they show the direct equivalence of AND and OR functions and how one can be readily converted to the other. XOR and XNOR functions can be represented by combining AND and OR gates. It can be observed from Table 1.3 that $A \oplus B = A\overline{B} + \overline{A}B$ and that $\overline{A \oplus B} = AB + \overline{A}\,\overline{B}$. Conversions between XOR/XNOR and AND/OR functions are helpful when manipulating and simplifying larger Boolean expressions, because simpler AND and OR functions are directly handled with DeMorgan's law, whereas XOR/XNOR functions are not.

1.3 THE KARNAUGH MAP

Generating Boolean equations to implement a desired logic function is a necessary step before a circuit can be implemented. Truth tables are a common means of describing logical relationships between Boolean inputs and outputs. Once a truth table has been created, it is not always easy to convert that truth table directly into a Boolean equation. This translation becomes more difficult as the number of variables in a function increases. A graphical means of translating a truth table into a logic equation was invented by Maurice Karnaugh in the early 1950s and today is called the *Karnaugh map*, or *K-map*. A K-map is a type of truth table drawn such that individual product terms can be picked out and summed with other product terms extracted from the map to yield an overall Boolean equation. The best way to explain how this process works is through an example. Consider the hypothetical logical relationship in Table 1.6.

TABLE 1.6 Function of Three Variables

A	B	C	Y
0	0	0	1
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

If the corresponding Boolean equation does not immediately become clear, the truth table can be converted into a K-map as shown in Fig. 1.3. The K-map has one box for every combination of inputs, and the desired output for a given combination is written into the corresponding box. Each axis of a K-map represents up to two variables, enabling a K-map to solve a function of up to four variables. Individual grid locations on each axis are labeled with a unique combination of the variables represented on that axis. The labeling pattern is important, because only one variable per axis is permitted to differ between adjacent boxes. Therefore, the pattern "00, 01, 10, 11" is not proper, but the pattern "11, 01, 00, 10" would work as well as the pattern shown.

K-maps are solved using the *sum of products* principle, which states that any relationship can be expressed by the logical OR of one or more AND terms. Product terms in a K-map are recognized by picking out groups of adjacent boxes that all have a state of 1. The simplest product term is a single box with a 1 in it, and that term is the product of all variables in the K-map with each variable either inverted or not inverted such that the result is 1. For example, a 1 is observed in the box that corresponds to A = 0, B = 1, and C = 1. The product term representation of that box would be $\overline{A}BC$. A brute force solution is to sum together as many product terms as there are boxes with a state of 1 (there are five in this example) and then simplify the resulting equation to obtain the final result. This approach can be taken without going to the trouble of drawing a K-map. The purpose of a K-map is to help in identifying minimized product terms so that lengthy simplification steps are unnecessary.

Minimized product terms are identified by grouping together as many adjacent boxes with a state of 1 as possible, subject to the rules of Boolean algebra. Keep in mind that, to generate a valid product term, all boxes in a group must have an identical relationship to all of the equation's input variables. This requirement translates into a rule that product term groups must be found in power-of-two quantities. For a three-variable K-map, product term groups can have only 1, 2, 4, or 8 boxes in them.

Going back to our example, a four-box product term is formed by grouping together the vertically stacked 1s on the left and right edges of the K-map. An interesting aspect of a K-map is that an edge wraps around to the other side, because the axis labeling pattern remains continuous. The validity of this wrapping concept is shown by the fact that all four boxes share a common relationship with the input variables: their product term is $\overline{B}$. The other variables, A and C, can be ruled out, because the boxes are 1 regardless of the state of A and C. Only variable B is a determining factor, and it must be 0 for the boxes to have a state of 1. Once a product term has been identified, it is marked by drawing a ring around it as shown in Fig. 1.4. Because the product term crosses the edges of the table, half-rings are shown in the appropriate locations.

There is still a box with a 1 in it that has not yet been accounted for. One approach could be to generate a product term for that single box, but this would not result in a fully simplified equation, because a larger group can be formed by associating the lone box with the adjacent box corresponding to A = 0, B = 0, and C = 1. K-map boxes can be part of multiple groups, and forming the largest groups possible results in a fully simplified equation. This second group of boxes is circled in Fig. 1.5 to complete the map. This product term shares a common relationship where A = 0, C = 1, and B

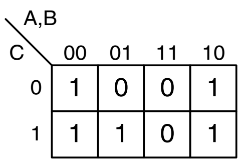

FIGURE 1.3 Karnaugh map for function of three variables.

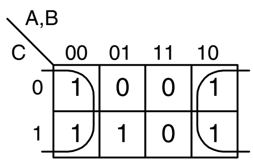

FIGURE 1.4 Partially completed Karnaugh map for a function of three variables.