

## 1-2 Logic Gates

Binary information is represented in digital computers by physical quantities called *signals*. Electrical signals such as voltages exist throughout the computer in either one of two recognizable states. The two states represent a binary variable that can be equal to 1 or 0. For example, a particular digital computer may employ a signal of 3 volts to represent binary 1 and 0.5 volt to represent binary 0. The input terminals of digital circuits accept binary signals of 3 and 0.5 volts and the circuits respond at the output terminals with signals of 3 and 0.5 volts to represent binary input and output corresponding to 1 and 0, respectively.

*gates*

Binary logic deals with binary variables and with operations that assume a logical meaning. It is used to describe, in algebraic or tabular form, the manipulation and processing of binary information. The manipulation of binary information is done by logic circuits called *gates*. Gates are blocks of hardware that produce signals of binary 1 or 0 when input logic requirements are satisfied. A variety of logic gates are commonly used in digital computer systems. Each gate has a distinct graphic symbol and its operation can be described by means of an algebraic expression. The input–output relationship of the binary variables for each gate can be represented in tabular form by a *truth table*. The basic logic gates are AND and inclusive OR with multiple inputs and NOT with a single input. Each gate with more than one input is sensitive to either logic 0 or logic 1 input at any one of its inputs, generating the output according to its function. For example, a multi-input AND gate is sensitive to logic 0 on any one of its inputs, irrespective of any values at other inputs.

The names, graphic symbols, algebraic functions, and truth tables of eight logic gates are listed in Fig. 1-2, with applicable sensitivity input values. Each gate has one or two binary input variables designated by  $A$  and  $B$  and one binary output variable designated by  $x$ . The AND gate produces the AND logic function: that is, the output is 1 if input  $A$  and input  $B$  are both equal to 1; otherwise, the output is 0. These conditions are also specified in the truth table for the AND gate. The table shows that output  $x$  is 1 only when both input  $A$  and input  $B$  are 1. The algebraic operation symbol of the AND function is the same as the multiplication symbol of ordinary arithmetic. We can either use a dot between the variables or concatenate the variables without an operation symbol between them. **AND gates may have more than two inputs, and by definition, the output is 1 if and only if all inputs are 1.**

OR

The OR gate produces the inclusive-OR function; that is, the output is 1 if input  $A$  or input  $B$  or both inputs are 1; otherwise, the output is 0. The algebraic symbol of the OR function is  $+$ , similar to arithmetic addition. OR gates may have more than two inputs, and by definition, the output is 1 if any input is 1.

*inverter*

The inverter circuit inverts the logic sense of a binary signal. It produces the NOT, or complement, function. The algebraic symbol used for the logic complement is **either a prime or a bar over the variable symbol**. In this book we use a prime for the logic complement of a binary variable, while a bar over the letter is reserved for designating a complement microoperation as defined in Chap. 4.

The small circle in the output of the graphic symbol of an inverter designates a logic complement. A triangle symbol by itself designates a buffer circuit. A

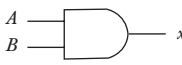
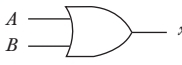
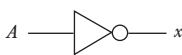
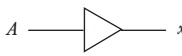
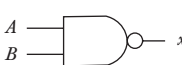
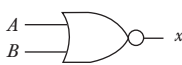
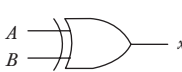
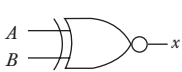
Name	Graphic symbol	Algebraic function	Truth table	Input sensitivity															
AND		$x = A \cdot B$ or $x = AB$	<table><tr><th>A</th><th>B</th><th>x</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	x	0	0	0	0	1	0	1	0	0	1	1	1	0
A	B	x																	
0	0	0																	
0	1	0																	
1	0	0																	
1	1	1																	
OR		$x = A + B$	<table><tr><th>A</th><th>B</th><th>x</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	x	0	0	0	0	1	1	1	0	1	1	1	1	1
A	B	x																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	1																	
Inverter		$x = A'$	<table><tr><th>A</th><th>x</th></tr><tr><td>0</td><td>1</td></tr><tr><td>1</td><td>0</td></tr></table>	A	x	0	1	1	0	Not Applicable									
A	x																		
0	1																		
1	0																		
Buffer		$x = A$	<table><tr><th>A</th><th>x</th></tr><tr><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td></tr></table>	A	x	0	0	1	1	Not Applicable									
A	x																		
0	0																		
1	1																		
NAND		$x = (AB)'$	<table><tr><th>A</th><th>B</th><th>x</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	x	0	0	1	0	1	1	1	0	1	1	1	0	0
A	B	x																	
0	0	1																	
0	1	1																	
1	0	1																	
1	1	0																	
NOR		$x = (A + B)'$	<table><tr><th>A</th><th>B</th><th>x</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	x	0	0	1	0	1	0	1	0	0	1	1	0	1
A	B	x																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	0																	
Exclusive-OR (XOR)		$x = A \oplus B$ or $x = A'B + AB'$	<table><tr><th>A</th><th>B</th><th>x</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>1</td></tr><tr><td>1</td><td>0</td><td>1</td></tr><tr><td>1</td><td>1</td><td>0</td></tr></table>	A	B	x	0	0	0	0	1	1	1	0	1	1	1	0	Not Applicable
A	B	x																	
0	0	0																	
0	1	1																	
1	0	1																	
1	1	0																	
Exclusive-NOR or equivalence		<del><math>x = (A \oplus B)'</math></del> <del>or</del> <del><math>x = A'B + AB'</math></del>	<table><tr><th>A</th><th>B</th><th>x</th></tr><tr><td>0</td><td>0</td><td>1</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	x	0	0	1	0	1	0	1	0	0	1	1	1	Not Applicable
A	B	x																	
0	0	1																	
0	1	0																	
1	0	0																	
1	1	1																	

Figure 1-2 Digital logic gates with applicable input sensitivity values.