

Figure 1-1 Block diagram of a digital computer.

computer organization

Computer organization is concerned with the way the hardware components operate and the way they are connected together to form the computer system. The various components are assumed to be in place and the task is to investigate the organizational structure to verify that the computer parts operate as intended.

computer design

Computer design is concerned with the hardware design of the computer. Once the computer specifications are formulated, it is the task of the designer to develop hardware for the system. Computer design is concerned with the determination of what hardware should be used and how the parts should be connected. This aspect of computer hardware is sometimes referred to as *computer implementation*.

computer architecture

Computer architecture is concerned with the structure and behavior of the computer as seen by the user. It includes the information formats, the instruc-

tion set, and techniques for addressing memory. The architectural design of a computer system is concerned with the specifications of the various functional modules, such as processors and memories, and structuring them together into a computer system.



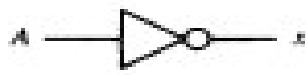
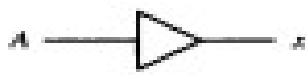




Name	Graphic symbol	Algebraic function	Truth table															
AND		$x = A \cdot B$ or $x = AB$	<table><tr><th>A</th><th>B</th><th>x</th></tr><tr><td>0</td><td>0</td><td>0</td></tr><tr><td>0</td><td>1</td><td>0</td></tr><tr><td>1</td><td>0</td><td>0</td></tr><tr><td>1</td><td>1</td><td>1</td></tr></table>	A	B	x	0	0	0	0	1	0	1	0	0	1	1	1
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
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									
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									
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
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
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
A	B	x																
0	0	0																
0	1	1																
1	0	1																
1	1	0																
Exclusive-NOR or equivalence		$x = (A \oplus B)'$ 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>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
A	B	x																
0	0	1																
0	1	0																
1	0	0																
1	1	1																

Figure 1-2 Digital logic gates.

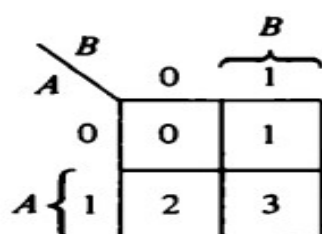
TABLE 1-1 Basic Identities of Boolean Algebra

(1) $x + 0 = x$	(2) $x \cdot 0 = 0$
(3) $x + 1 = 1$	(4) $x \cdot 1 = x$
(5) $x + x = x$	(6) $x \cdot x = x$
(7) $x + x' = 1$	(8) $x \cdot x' = 0$
(9) $x + y = y + x$	(10) $xy = yx$
(11) $x + (y + z) = (x + y) + z$	(12) $x(yz) = (xy)z$
(13) $x(y + z) = xy + xz$	(14) $x + yx = (x + y)(x + z)$
(15) $(x + y)' = x'y'$	(16) $(xy)' = x' + y'$
(17) $(x')' = x$	

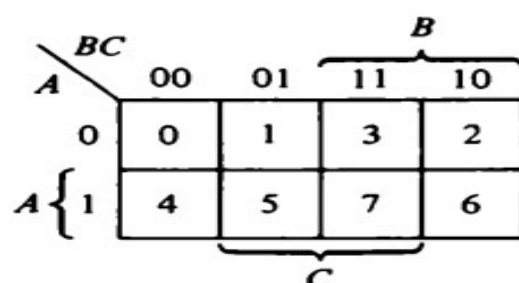
derived by means of DeMorgan's theorem. The general form of DeMorgan's theorem can be expressed as follows:

$$(x_1 + x_2 + x_3 + \cdots + x_n)' = x_1' x_2' x_3' \cdots x_n'$$

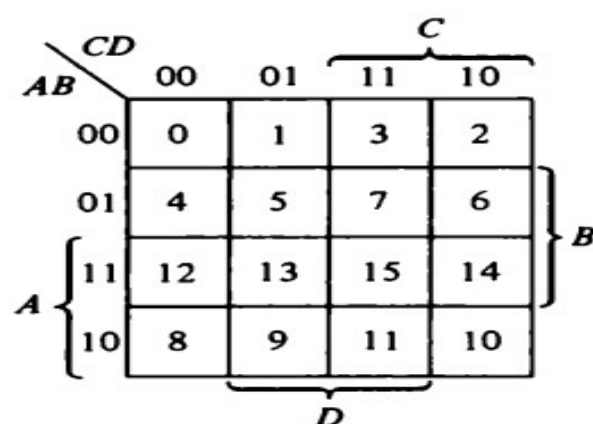
$$(x_1 x_2 x_3 \cdots x_n)' = x_1' + x_2' + x_3' + \cdots + x_n'$$



(a) Two-variable map



(b) Three-variable map



(c) Four-variable map

Figure 1-7 Maps for two-, three-, and four-variable functions.

Combinational Circuit:

SECTION 1-5 Combinational Circuits

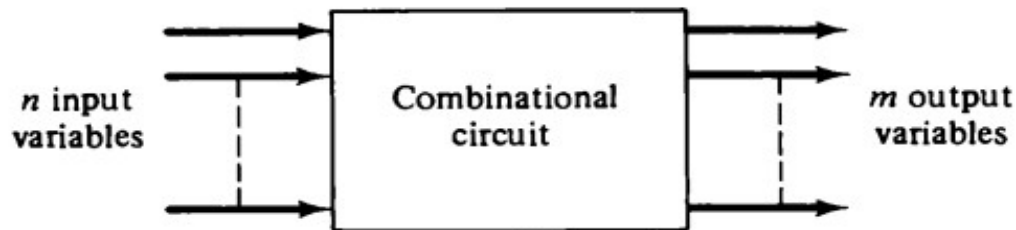


Figure 1-15 Block diagram of a combinational circuit.

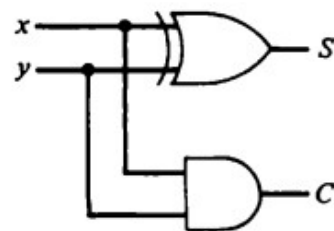
The design of combinational circuits starts from the verbal outline of the problem and ends in a logic circuit diagram. The procedure involves the following steps:

1. The problem is stated.
2. The input and output variables are assigned letter symbols.
3. The truth table that defines the relationship between inputs and outputs is derived.
4. The simplified Boolean functions for each output are obtained.
5. The logic diagram is drawn.

Half-Adder:

x	y	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

(a) Truth table



(b) Logic diagram

Figure 1-16 Half-adder.

(for carry) to the two output variables. The truth table for the half-adder is shown in Fig. 1-16(a). The C output is 0 unless both inputs are 1. The S output represents the least significant bit of the sum. The Boolean functions for the two outputs can be obtained directly from the truth table:

$$S = x'y + xy' = x \oplus y$$

$$C = xy$$

Full-Adder:

TABLE 1-2 Truth Table for Full-Adder

Inputs			Outputs	
x	y	z	C	S
0	0	0	0	0
0	0	1	0	1
0	1	0	0	1
0	1	1	1	0
1	0	0	0	1
1	0	1	1	0
1	1	0	1	0
1	1	1	1	1

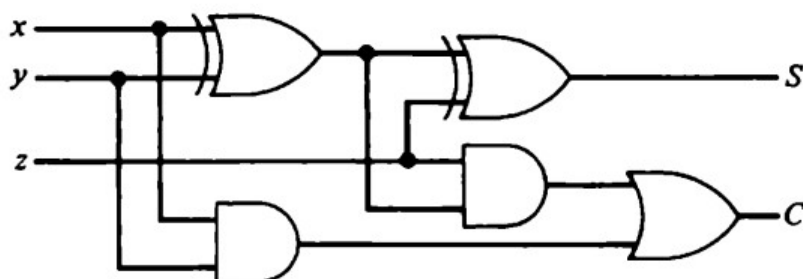
the exclusive-OR relation of the variables (see the discussion at the end of Sec. 1-2). The squares with 1's for the C output may be combined in a variety of ways. One possible expression for C is

$$C = xy + (x'y + xy')z$$

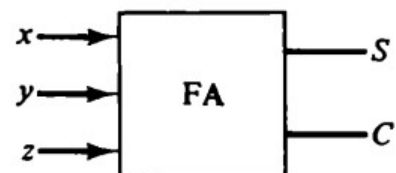
Realizing that $x'y + xy' = x \oplus y$ and including the expression for output S , we obtain the two Boolean expressions for the full-adder:

$$S = x \oplus y \oplus z$$

$$C = xy + (x \oplus y)z$$



(a) Logic diagram

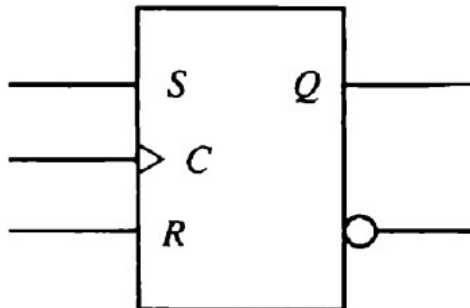


(b) Block diagram

Figure 1-18 Full-adder circuit.

SR flip-flops:

SECTION 1-6 Flips-Flops



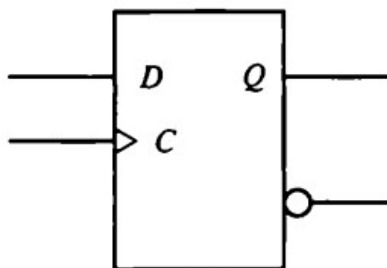
(a) Graphic symbol

S	R	$Q(t + 1)$	
0	0	$Q(t)$	No change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	?	Indeterminate

(b) Characteristic table

Figure 1-19 SR flip-flop.

D flip-flops:



(a) Graphic symbol

D	$Q(t + 1)$	
0	0	Clear to 0
1	1	Set to 1

(b) Characteristic table

Figure 1-20 D flip-flop.

is determined from the D input. The relationship can be expressed by characteristic equation:

$$Q(t + 1) = D$$

JK flip-flops & T flip-flops:

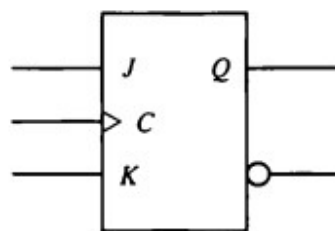
JK Flip-Flop

A *JK* flip-flop is a refinement of the *SR* flip-flop in that the indeterminate condition of the *SR* type is defined in the *JK* type. Inputs *J* and *K* behave like inputs *S* and *R* to set and clear the flip-flop, respectively. When inputs *J* and *K* are both equal to 1, a clock transition switches the outputs of the flip-flop to their complement state.

The graphic symbol and characteristic table of the *JK* flip-flop are shown in Fig. 1-21. The *J* input is equivalent to the *S* (set) input of the *SR* flip-flop, and the *K* input is equivalent to the *R* (clear) input. Instead of the indeterminate condition, the *JK* flip-flop has a complement condition $Q(t + 1) = Q'(t)$ when both *J* and *K* are equal to 1.

T Flip-Flop

Another type of flip-flop found in textbooks is the *T* (toggle) flip-flop. This flip-flop, shown in Fig. 1-22, is obtained from a *JK* type when inputs *J* and *K* are connected to provide a single input designated by *T*. The *T* flip-flop

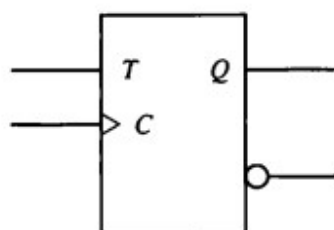


(a) Graphic symbol

J	K	$Q(t+1)$	
0	0	$Q(t)$	No change
0	1	0	Clear to 0
1	0	1	Set to 1
1	1	$Q'(t)$	Complement

(b) Characteristic table

Figure 1-21 JK flip-flop.



(a) Graphic symbol

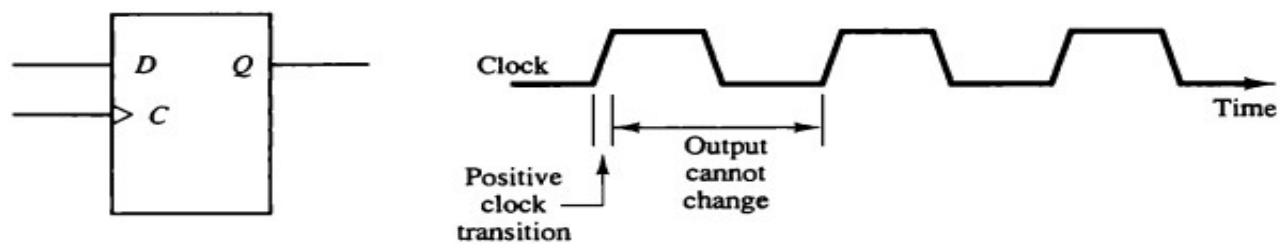
T	$Q(t+1)$	
0	$Q(t)$	No change
1	$Q'(t)$	Complement

(b) Characteristic table

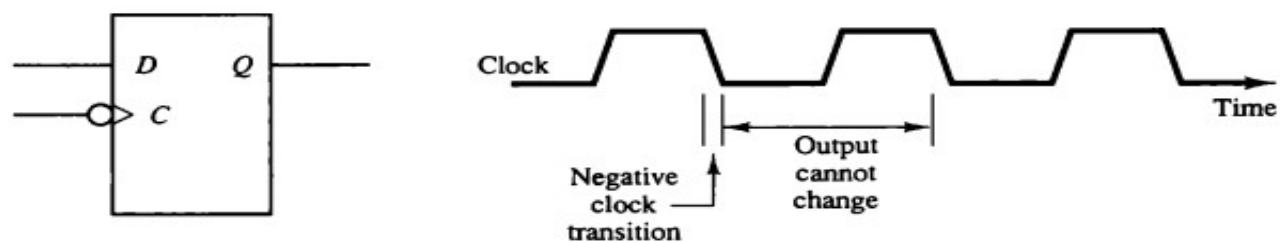
Figure 1-22 T flip-flop.

therefore has only two conditions. When $T = 0$ ($J = K = 0$) a clock transition does not change the state of the flip-flop. When $T = 1$ ($J = K = 1$) a clock transition complements the state of the flip-flop. These conditions can be expressed by a characteristic equation:

$$Q(t+1) = Q(t) \oplus T$$



(a) Positive-edge-triggered *D* flip-flop.



(b) Negative-edge-triggered *D* flip-flop.

Figure 1-23 Edge-triggered flip-flop.

TABLE 1-3 Excitation Table for Four Flip-Flops

SR flip-flop				<i>D</i> flip-flop		
$Q(t)$	$Q(t + 1)$	<i>S</i>	<i>R</i>	$Q(t)$	$Q(t + 1)$	<i>D</i>
0	0	0	×	0	0	0
0	1	1	0	0	1	1
1	0	0	1	1	0	0
1	1	×	0	1	1	1

<i>JK</i> flip-flop				<i>T</i> flip-flop		
$Q(t)$	$Q(t + 1)$	<i>J</i>	<i>K</i>	$Q(t)$	$Q(t + 1)$	<i>T</i>
0	0	0	×	0	0	0
0	1	1	×	0	1	1
1	0	×	1	1	0	1
1	1	×	0	1	1	0

Figure 1-24 Block diagram of a clocked synchronous sequential circuit.

