CHAPTER 2

THE LOGIC OF COMPOUND STATEMENTS

Copyright © Cengage Learning. All rights reserved.

SECTION 2.4

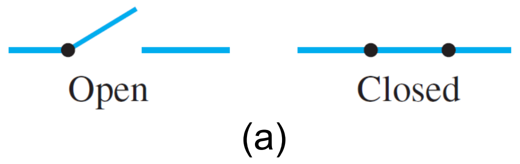
Application: Digital Logic Circuits

Copyright © Cengage Learning. All rights reserved.

Application: Digital Logic Circuits

The drawing in Figure 2.4.1(a) shows the appearance of the two positions of a simple switch. When the switch is closed, current can flow from one terminal to the other; when it is open, current cannot flow.

Imagine that such a switch is part of the circuit shown in figure 2.4.1(b). The light bulb turns on if, and only if, current flows through it. And this happens if, and only if, the switch is closed.



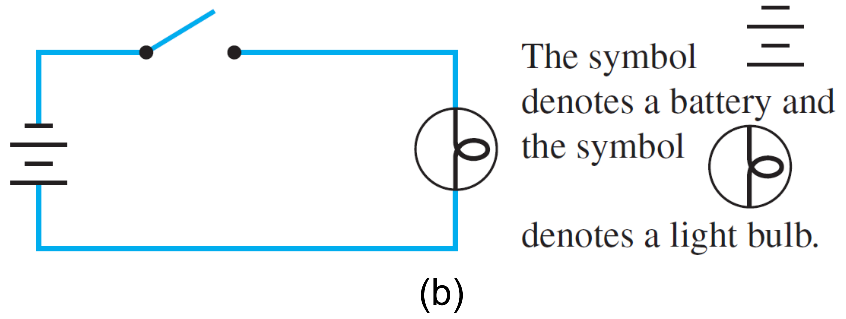
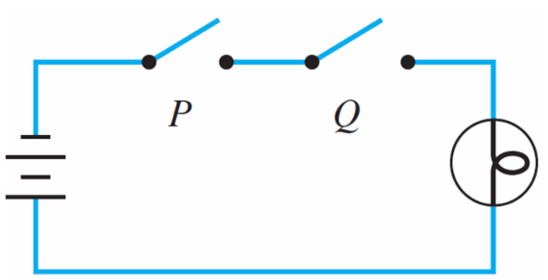


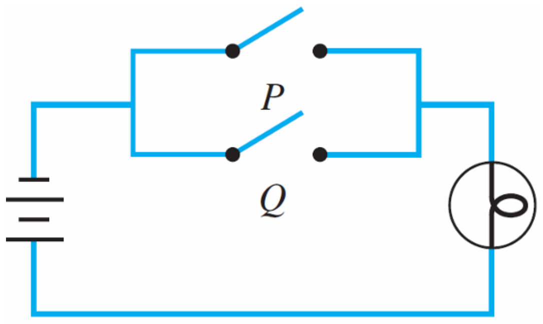
Figure 2.4.1

Application: Digital Logic Circuits

Now consider the more complicated circuits of Figures 2.4.2(a) and 2.4.2(b).



(a) Switches "in series"



(b) Switches "in parallel"

Figure 2.4.2

In the circuit of Figure 2.4.2(a) current flows and the light bulb turns on if, and only if, *both* switches *P* and *Q* are closed. The switches in this circuit are said to be **in series**.

Application: Digital Logic Circuits

In the circuit of Figure 2.4.2(b) current flows and the light bulb turns on if, and only if, *at least one* of the switches *P* or *Q* is closed. The switches in this circuit are said to be **in parallel**. All possible behaviors of these circuits are described by Table 2.4.1.

Switches		Light Bulb
P	***Q***	**State**
closed	closed	on
closed	open	off
open	closed	off
open	open	off

(a) Switches in Series

Switches		Light Bulb
P	***Q***	**State**
closed	closed	on
closed	open	on
open	closed	on
open	open	off

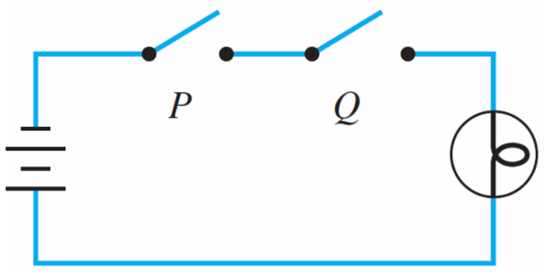
(b) Switches in Parallel

Table 2.4.1

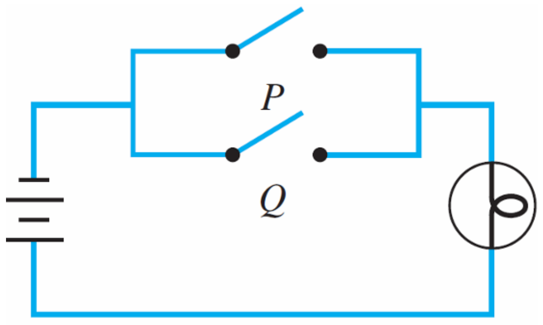
Application: Digital Logic Circuits

Observe that if the words *closed* and *on* are replaced by T and *open* and *off* are replaced by F, Table 2.4.1(a) becomes the truth table for *and* and Table 2.4.1(b) becomes the truth table for *or*.

Consequently, the switching circuit of Figure 2.4.2(a) is said to correspond to the logical expression $P \wedge Q$, and that of Figure 2.4.2(b) is said to correspond to $P \vee Q$.



(a) Switches "in series"



(b) Switches "in parallel"

Figure 2.4.2

Application: Digital Logic Circuits

More complicated circuits correspond to more complicated logical expressions. This correspondence has been used extensively in the design and study of circuits.

1940s-1950s: Switches replaced by transistors, leading to digital electronics (discrete signals instead of continuous ones). Components of digital electronics are **digital logic circuits.**

Electrical engineers continue to use the language of logic but they generally use the symbols 1 and 0 instead of T and F to denote these values. The symbols 0 and 1 are called **bits,** short for *b*inary dig*its*.

Black Boxes and Gates

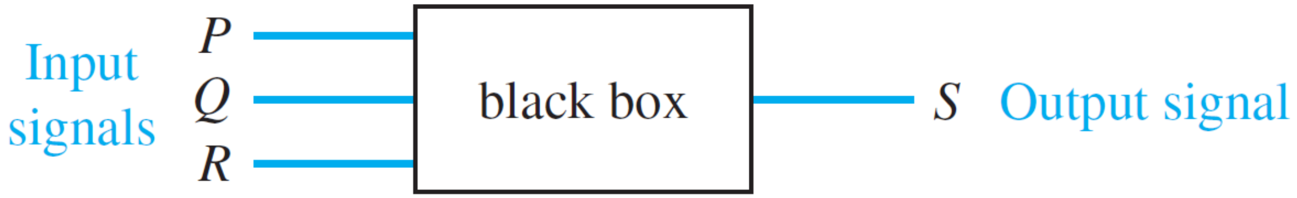
Black Boxes and Gates

Combinations of signal bits (1’s and 0’s) can be transformed into other combinations of signal bits (1’s and 0’s) by means of various circuits.

Because a variety of different technologies are used in circuit construction, computer engineers and digital system designers find it useful to think of certain basic circuits as black boxes.

Black Boxes and Gates

The inside of a black box contains the detailed implementation of the circuit and is often ignored while attention is focused on the relation between the **input** and the **output** signals.



The operation of a black box is completely specified by constructing an **input/output table** that lists all its possible input signals together with their corresponding output signals.

Black Boxes and Gates

For example, the black box picture has three input signals. Since each of these signals can take the value 1 or 0, there are eight possible combinations of input signals.

Black Boxes and Gates

One possible correspondence of input to output signals is as follows:

Input			Output
P	***Q***	***R***	***S***
1	1	1	1
1	1	0	0
1	0	1	0
1	0	0	1
0	1	1	0
0	1	0	1
0	0	1	1
0	0	0	0

An Input/Output Table

Black Boxes and Gates

An efficient method for designing more complicated circuits is to build them by connecting less complicated black box circuits. Three such circuits are known as NOT-, AND-, and OR-gates.

A **NOT-gate** (or **inverter**) is a circuit with one input signal and one output signal. If the input signal is 1, the output signal is 0.

Conversely, if the input signal is 0, then the output signal is 1. An **AND-gate** is a circuit with two input signals and one output signal. If both input signals are 1, then the output signal is 1.

Black Boxes and Gates

Otherwise, the output signal is 0. An **OR-gate** also has two input signals and one output signal. If both input signals are 0, then the output signal is 0. Otherwise, the output signal is 1.

The actions of NOT-, AND-, and OR-gates are summarized in Figure 2.4.3, where *P* and *Q* represent input signals and *R* represents the output signal.

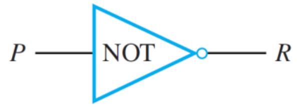
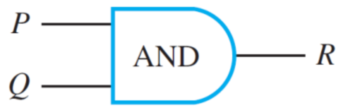
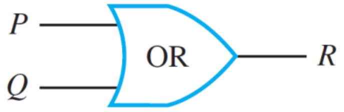
Type of Gate	Symbolic Representation	Action	
		Input	Output
		P	***R***
NOT		1	0
		0	1

Figure 2.4.3

Black Boxes and Gates

Type of Gate	Symbolic Representation	Action
AND		Input: P Q / Output: R 1 1 / 1 1 0 / 0 0 1 / 0 0 0 / 0
OR		Input: P Q / Output: R 1 1 / 1 1 0 / 1 0 1 / 1 0 0 / 0

AND

Input		Output
P	***Q***	***R***
1	1	1
1	0	0
0	1	0
0	0	0

OR

Input		Output
P	***Q***	***R***
1	1	1
1	0	1
0	1	1
0	0	0

Figure 2.4.3 (continued)

Black Boxes and Gates

It should be clear from Figure 2.4.3 that the actions of the NOT-, AND-, and OR-gates on signals correspond exactly to those of the logical connectives ~, $\wedge$, and $\vee$ on statements, if the symbol 1 is identified with T and the symbol 0 is identified with F.

Gates can be combined into circuits in a variety of ways. If the rules shown on the next page are obeyed, the result is a **combinational circuit**, one whose output at any time is determined entirely by its input at that time without regard to previous inputs.

Rules for a Combinational Circuit

Rules for a Combinational Circuit

Never combine two input wires. 2.4.1

A single input wire can be split partway and used as input for two separate gates. 2.4.2

An output wire can be used as input. 2.4.3

No output of a gate can eventually feed back into that gate. 2.4.4

Rule (2.4.4) is violated in more complex circuits, called **sequential circuits**, whose output at any given time depends both on the input at that time and also on previous inputs.

The Input/Output Table for a Circuit

The Input/Output Table for a Circuit

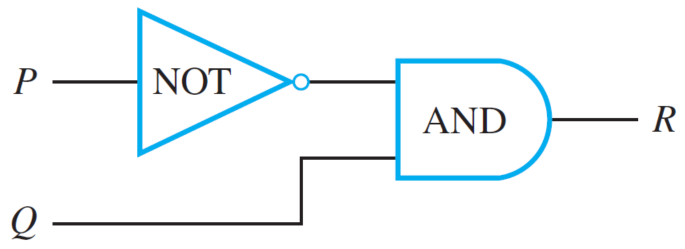
If you are given a set of input signals for a circuit, you can find its output by tracing through the circuit gate by gate.

Example 1 – *Determining Output for a Given Input*

Indicate the output of the circuits shown below for the given input signals.

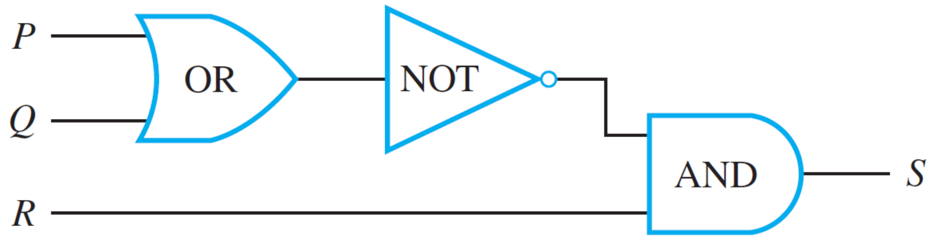
a.

Input signals: $P = 0$ and $Q = 1$



b.

Input signals: $P = 1, \quad Q = 0, \quad R = 1$

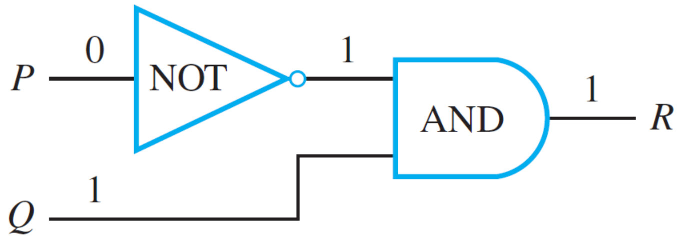


Example 1(a) – *Solution*

Move from left to right through the diagram, tracing the action of each gate on the input signals.

The NOT-gate changes *P* = *0* to a 1, so both inputs to the AND-gate are 1; hence the output *R* is 1.

This is illustrated by annotating the diagram as shown below.

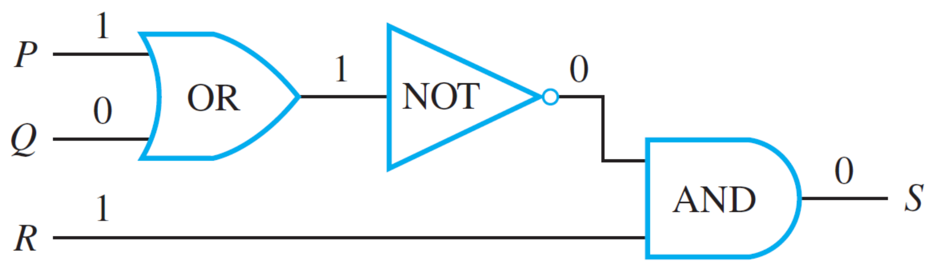


Example 1(b) – *Solution*

cont'd

The output of the OR-gate is 1 since one of the input signals, *P*, is 1. The NOT-gate changes this 1 into a 0, so the two inputs to the AND-gate are 0 and *R* = 1.

Hence the output *S* is 0. The trace is shown below.



The Boolean Expression Corresponding to a Circuit

The Boolean Expression Corresponding to a Circuit

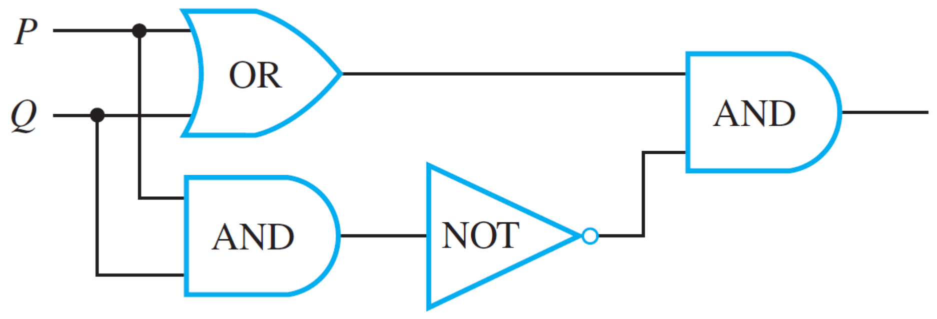
In logic, variables such as *p*, *q* and *r* represent statements, and a statement can have one of only two truth values: T(true) or F(false).

A statement form is an expression, such as $p \wedge (\sim q \vee r)$, composed of statement variables and logical connectives.

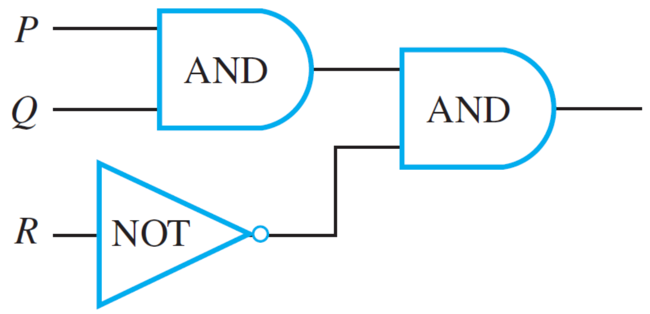
As noted earlier, one of the founders of symbolic logic was the English mathematician George Boole. In his honor, any variable, such as a statement variable or an input signal, that can take one of only two values is called a **Boolean variable.** An expression composed of Boolean variables and the connectives ~, $\wedge$, and $\vee$ is called a **Boolean expression.**

Example 3 – *Finding a Boolean Expression for a Circuit*

Find the Boolean expressions that correspond to the circuits shown below. A dot indicates a soldering of two wires; wires that cross without a dot are assumed not to touch.



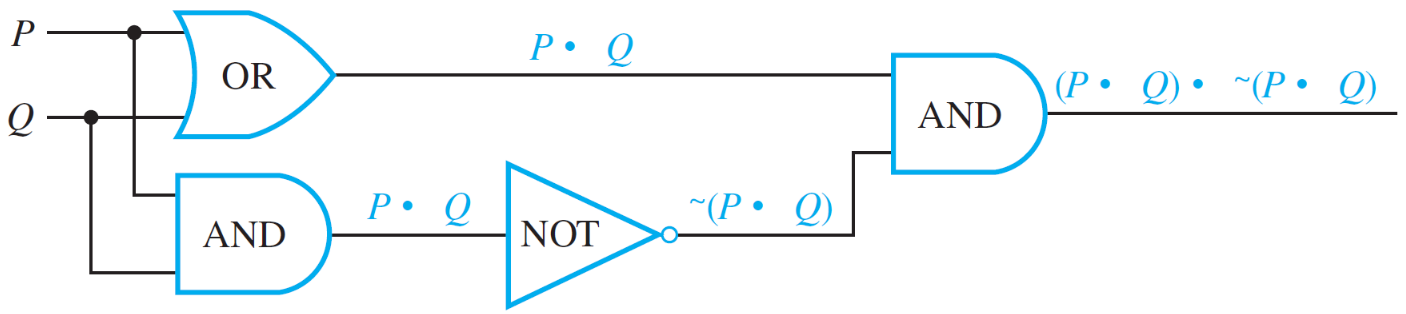
(a)



(b)

Example 3(a) – *Solution*

Trace through the circuit from left to right, indicating the output of each gate symbolically, as shown below.

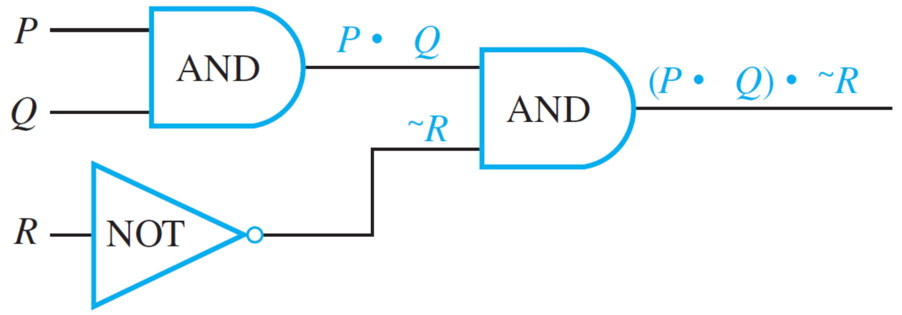


The final expression obtained, (*P* ∨ *Q*) ∧ ~(*P* ∧ *Q*), is the expression for exclusive or: *P* or *Q* but not both.

Example 3(b) – *Solution*

cont'd

The Boolean expression corresponding to the circuit is $(P \wedge Q) \wedge \sim R$, as shown below.



The Boolean Expression Corresponding to a Circuit

Observe that the output of the circuit shown in Example 3(b) is 1 for exactly one combination of inputs ($P = 1$, $Q = 1$, and $R = 0$) and is 0 for all other combinations of inputs.

The Boolean Expression Corresponding to a Circuit

For this reason, the circuit can be said to "recognize" one particular combination of inputs. The output column of the input/output table has a 1 in exactly one row and 0's in all other rows.

P	Q	R	$(P \wedge Q) \wedge \sim R$
1	1	1	0
1	1	0	1
1	0	1	0
1	0	0	0
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

Input/Output Table for a Recognizer

● **Definition**

A **recognizer** is a circuit that outputs a 1 for exactly one particular combination of input signals and outputs 0's for all other combinations.

The Circuit Corresponding to a Boolean Expression

Example 4 – *Constructing Circuits for Boolean Expressions*

Construct circuits for the following Boolean expressions.

a. $(\sim P \wedge Q) \vee \sim Q$ **b.** $((P \wedge Q) \wedge (R \wedge S)) \wedge T$

Solution:

a. Write the input variables in a column on the left side of the diagram. Then go from the right side of the diagram to the left, working from the outermost part of the expression to the innermost part.

Since the last operation executed when evaluating $(\sim P \wedge Q) \vee \sim Q$ is $\vee$, put an OR-gate at the extreme right of the diagram.

Example 4 – *Solution*

cont'd

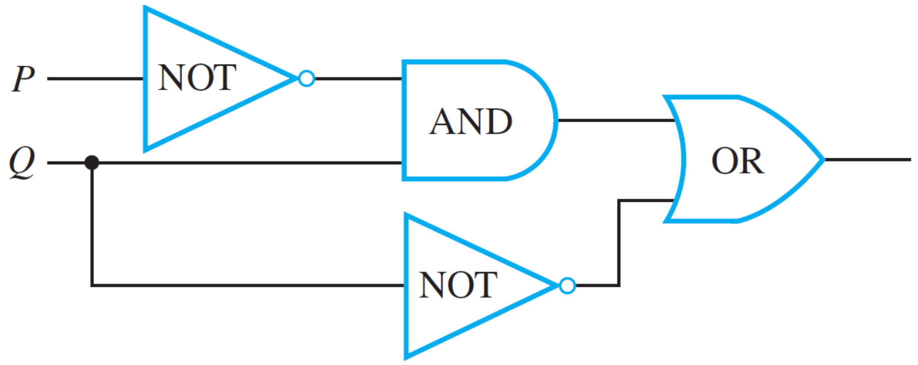
One input to this gate is $\sim P \wedge Q$, so draw an AND-gate to the left of the OR-gate and show its output coming into the OR-gate.

Since one input to the AND-gate is $\sim P$, draw a line from P to a NOT-gate and from there to the AND-gate. Since the other input to the AND-gate is Q, draw a line from Q directly to the AND-gate.

Example 4 – *Solution*

cont’d

The other input to the OR-gate is ~*Q*, so draw a line from *Q* to a NOT-gate and from the NOT-gate to the OR-gate. The circuit you obtain is shown below.



Example 4 – *Solution*

cont'd

b. To start constructing this circuit, put one AND-gate at the extreme right for the $\wedge$ between $((P \wedge Q) \wedge (R \wedge S))$ and T.

To the left of that put the AND-gate corresponding to the $\wedge$ between $P \wedge Q$ and $R \wedge S$.

To the left of that put the AND-gates corresponding to the $\wedge$'s between P and Q and between R and S.

Example 4 – *Solution*

cont'd

The circuit is shown in Figure 2.4.4.

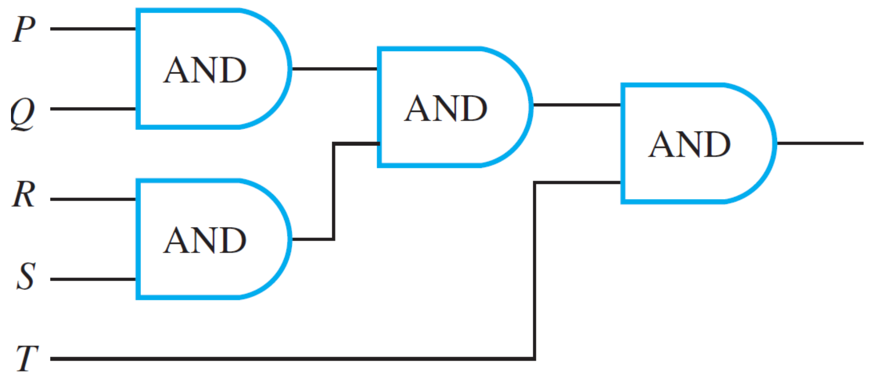
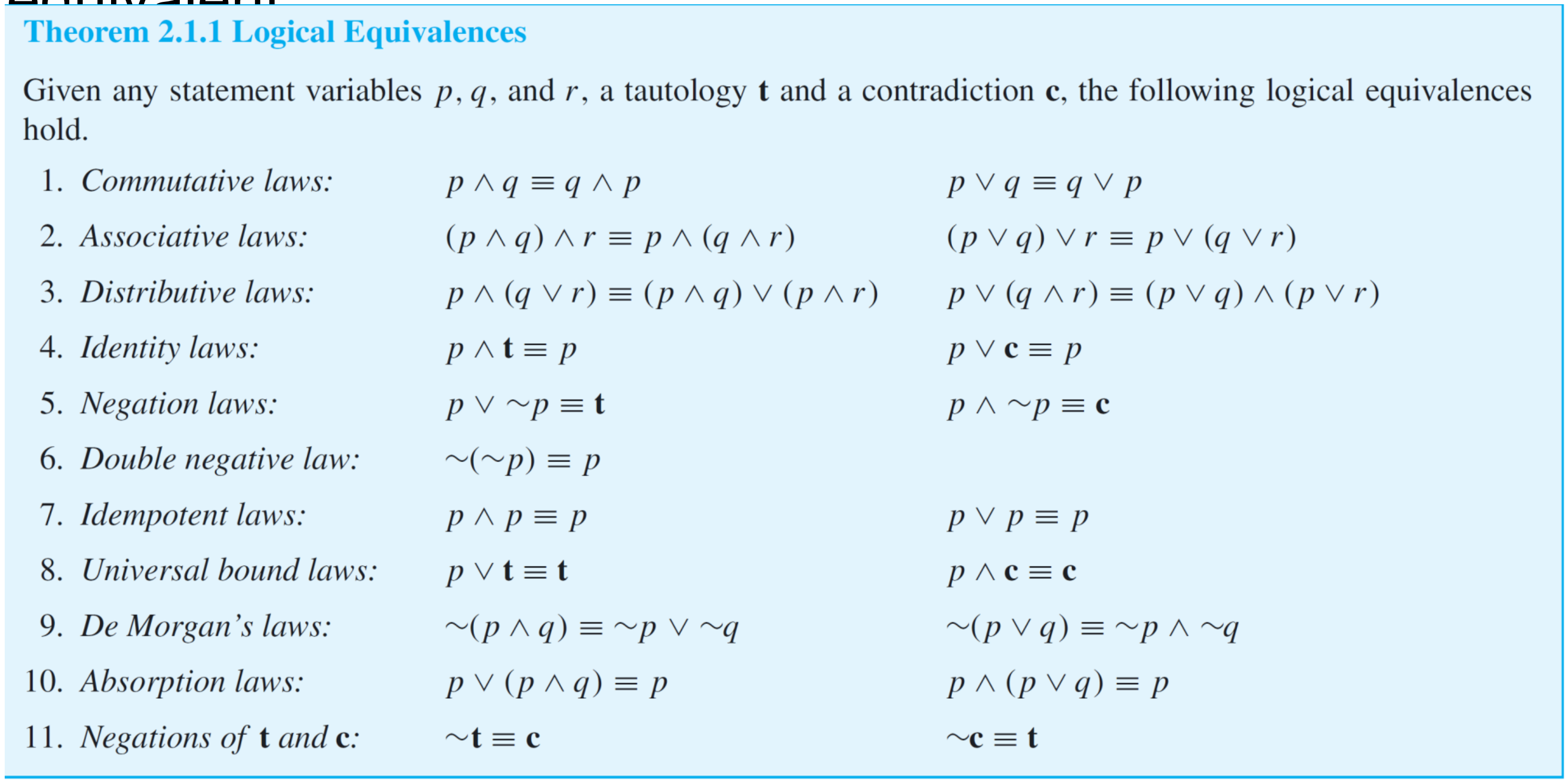


Figure 2.4.4

The Circuit Corresponding to a Boolean Expression

It follows from Theorem 2.1.1 that all the ways of adding parentheses to $P \wedge Q \wedge R \wedge S \wedge T$ are logically equivalent



Theorem 2.1.1 Logical Equivalences

Given any statement variables p, q, and r, a tautology $\mathbf{t}$ and a contradiction $\mathbf{c}$, the following logical equivalences hold.

1. *Commutative laws:*	$p \wedge q \equiv q \wedge p$	$p \vee q \equiv q \vee p$
2. *Associative laws:*	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	$(p \vee q) \vee r \equiv p \vee (q \vee r)$
3. *Distributive laws:*	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
4. *Identity laws:*	$p \wedge \mathbf{t} \equiv p$	$p \vee \mathbf{c} \equiv p$
5. *Negation laws:*	$p \vee {\sim}p \equiv \mathbf{t}$	$p \wedge {\sim}p \equiv \mathbf{c}$
6. *Double negative law:*	${\sim}({\sim}p) \equiv p$	
7. *Idempotent laws:*	$p \wedge p \equiv p$	$p \vee p \equiv p$
8. *Universal bound laws:*	$p \vee \mathbf{t} \equiv \mathbf{t}$	$p \wedge \mathbf{c} \equiv \mathbf{c}$
9. *De Morgan's laws:*	${\sim}(p \wedge q) \equiv {\sim}p \vee {\sim}q$	${\sim}(p \vee q) \equiv {\sim}p \wedge {\sim}q$
10. *Absorption laws:*	$p \vee (p \wedge q) \equiv p$	$p \wedge (p \vee q) \equiv p$
11. *Negations of* $\mathbf{t}$ *and* $\mathbf{c}$:	${\sim}\mathbf{t} \equiv \mathbf{c}$	${\sim}\mathbf{c} \equiv \mathbf{t}$

The Circuit Corresponding to a Boolean Expression

Thus, for example,

$((P \wedge Q) \wedge (R \wedge S)) \wedge T \equiv (P \wedge (Q \wedge R)) \wedge (S \wedge T)$.

It also follows that the circuit in Figure 2.4.5, which corresponds to $(P \wedge (Q \wedge R)) \wedge (S \wedge T)$, has the same input/output table as the circuit in Figure 2.4.4, which corresponds to $((P \wedge Q) \wedge (R \wedge S)) \wedge T$.

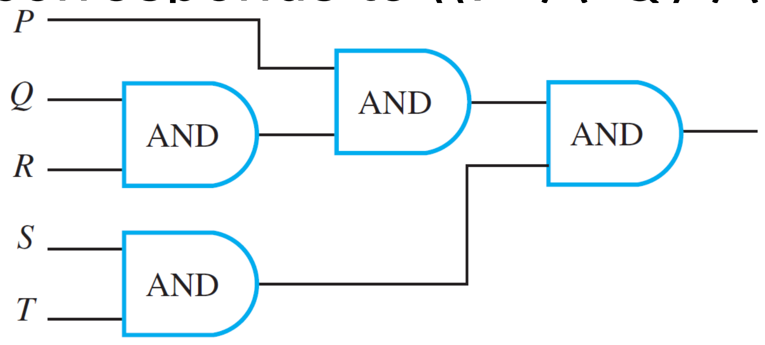


Figure 2.4.5

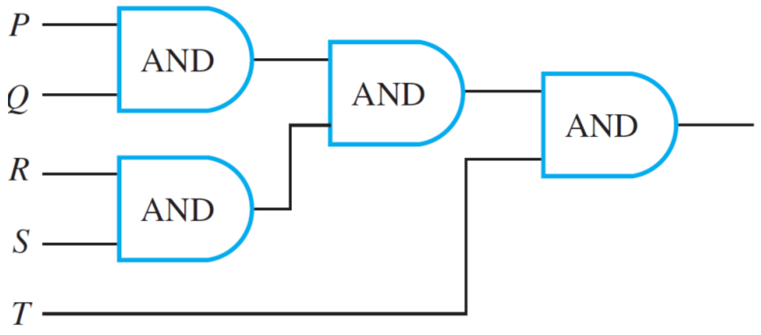


Figure 2.4.4

The Circuit Corresponding to a Boolean Expression

Each of the circuits in Figures 2.4.4 and 2.4.5 is, therefore, an implementation of the expression $P \wedge Q \wedge R \wedge S \wedge T$. Such a circuit is called a **multiple-input AND-gate** and is represented by the diagram shown in Figure 2.4.6.

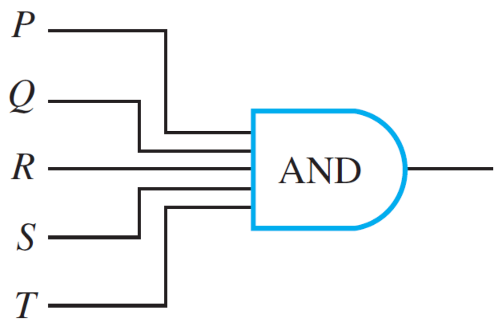


Figure 2.4.6

Multiple-input OR-gates are constructed similarly.

Finding a Circuit That Corresponds to a Given Input/Output Table

Example 5 – *Designing a Circuit for a Given Input/Output Table*

Design a circuit for the following input/output table:

Input			**Output**
P	***Q***	***R***	***S***
1	1	1	1
1	1	0	0
1	0	1	1
1	0	0	1
0	1	1	0
0	1	0	0
0	0	1	0
0	0	0	0

Example 5 – *Solution*

First construct a Boolean expression with this table as its truth table. To do this, identify each row for which the output is 1—in this case, the first, third, and fourth rows.

For each such row, construct an *and* expression that produces a 1 (or true) for the exact combination of input values for that row and a 0 (or false) for all other combinations of input values.

For example, the expression for the first row is $P \wedge Q \wedge R$ because $P \wedge Q \wedge R$ is 1 if $P = 1$ and $Q = 1$ and $R = 1$, and it is 0 for all other values of P, Q, and R.

Example 5 – *Solution*

cont'd

The expression for the third row is $P \wedge {\sim}Q \wedge R$ because $P \wedge {\sim}Q \wedge R$ is 1 if $P = 1$ and $Q = 0$ and $R = 1$, and it is 0 for all other values of P, Q, and R. Similarly, the expression for the fourth row is $P \wedge {\sim}Q \wedge {\sim}R$.

Now any Boolean expression with the given table as its truth table has the value 1 in case $P \wedge Q \wedge R = 1$, or in case $P \wedge {\sim}Q \wedge R = 1$, or in case $P \wedge {\sim}Q \wedge {\sim}R = 1$, and in no other cases.

It follows that a Boolean expression with the given truth table is

$$(P \wedge Q \wedge R) \vee (P \wedge {\sim}Q \wedge R) \vee (P \wedge {\sim}Q \wedge {\sim}R).$$ 2.4.5

Example 5 – *Solution*

cont'd

The circuit corresponding to this expression has the diagram shown in Figure 2.4.7.

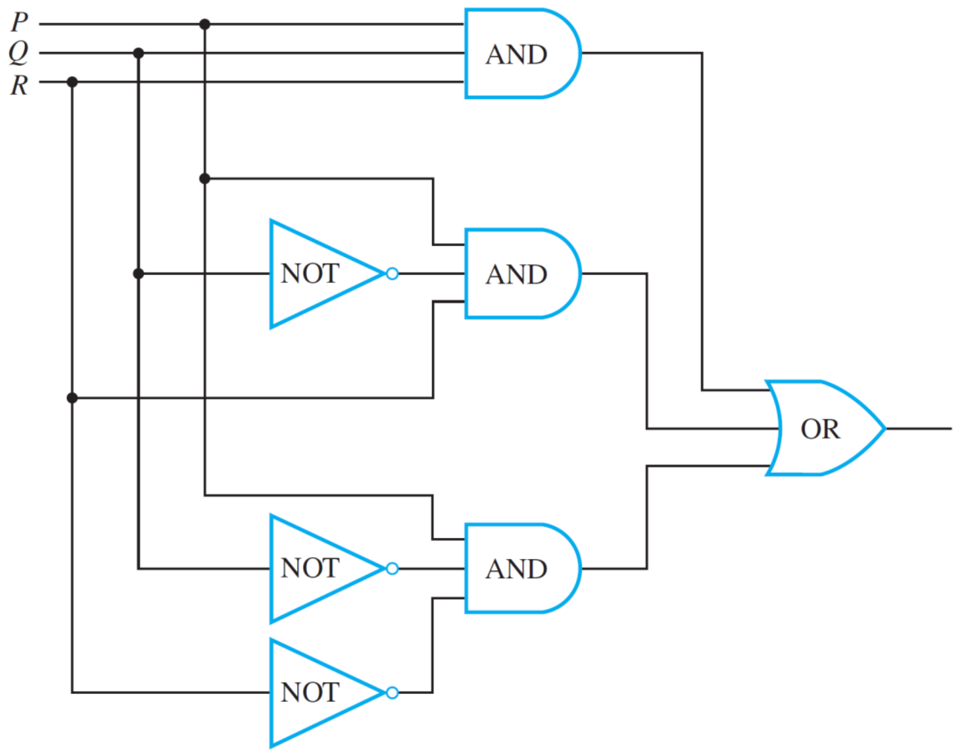


Figure 2.4.7

Example 5 – *Solution*

cont'd

Observe that expression

$$(P \wedge Q \wedge R) \vee (P \wedge {\sim}Q \wedge R) \vee (P \wedge {\sim}Q \wedge {\sim}R). \quad 2.4.5$$

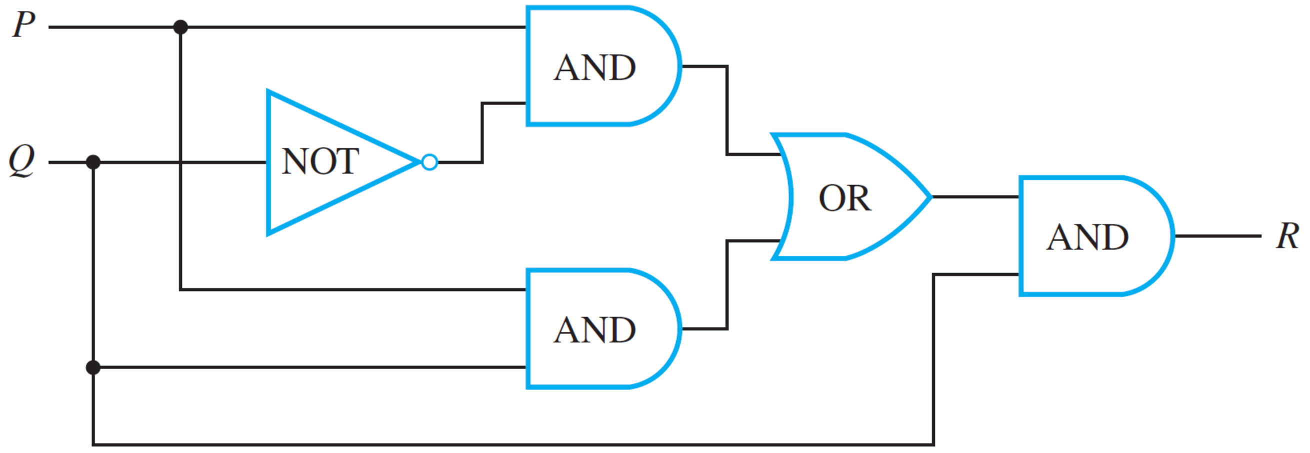
is a disjunction of terms that are themselves conjunctions in which one of *P* or ~*P*, one of *Q* or ~*Q*, and one of *R* or ~*R* all appear.

Such expressions are said to be in **disjunctive normal form** or **sum-of-products form**.

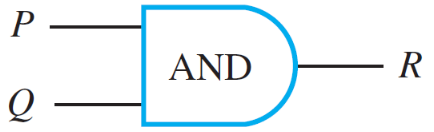
Simplifying Combinational Circuits

Simplifying Combinational Circuits

Consider the two combinational circuits shown in Figure 2.4.8.



(a)



(b)

Figure 2.4.8

Simplifying Combinational Circuits

If you trace through circuit (a), you will find that its input/output table is

Input		Output
P	*Q*	*R*
1	1	1
1	0	0
0	1	0
0	0	0

which is the same as the input/output table for circuit (b). Thus these two circuits do the same job in the sense that they transform the same combinations of input signals into the same output signals.

Simplifying Combinational Circuits

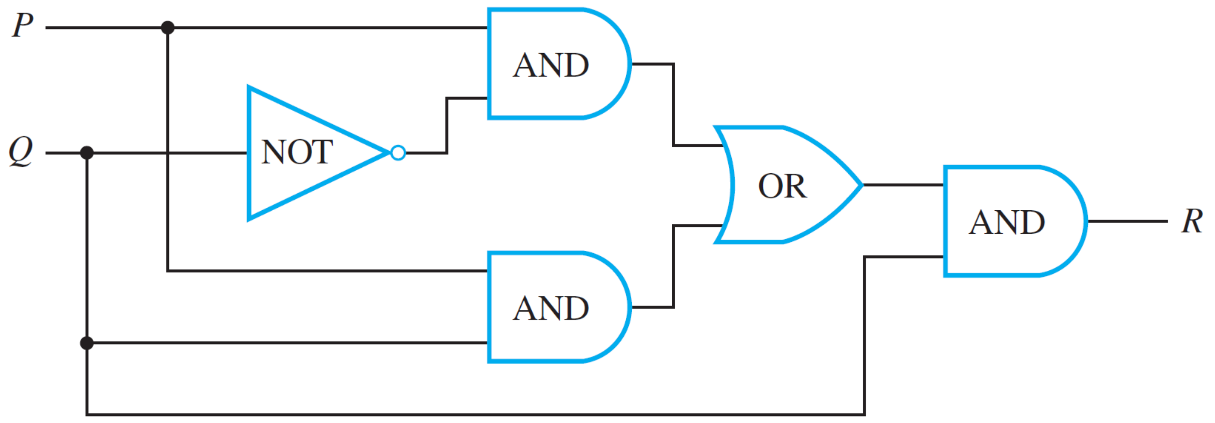
Yet circuit (b) is simpler than circuit (a) in that it contains many fewer logic gates. Thus, as part of an integrated circuit, it would take less space and require less power.

• Definition

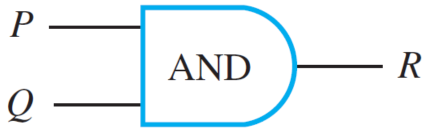
Two digital logic circuits are **equivalent** if, and only if, their input/output tables are identical.

Example 6 – *Showing That Two Circuits Are Equivalent*

Find the Boolean expressions for each circuit in Figure 2.4.8. Use Theorem 2.1.1 to show that these expressions are logically equivalent when regarded as statement forms.



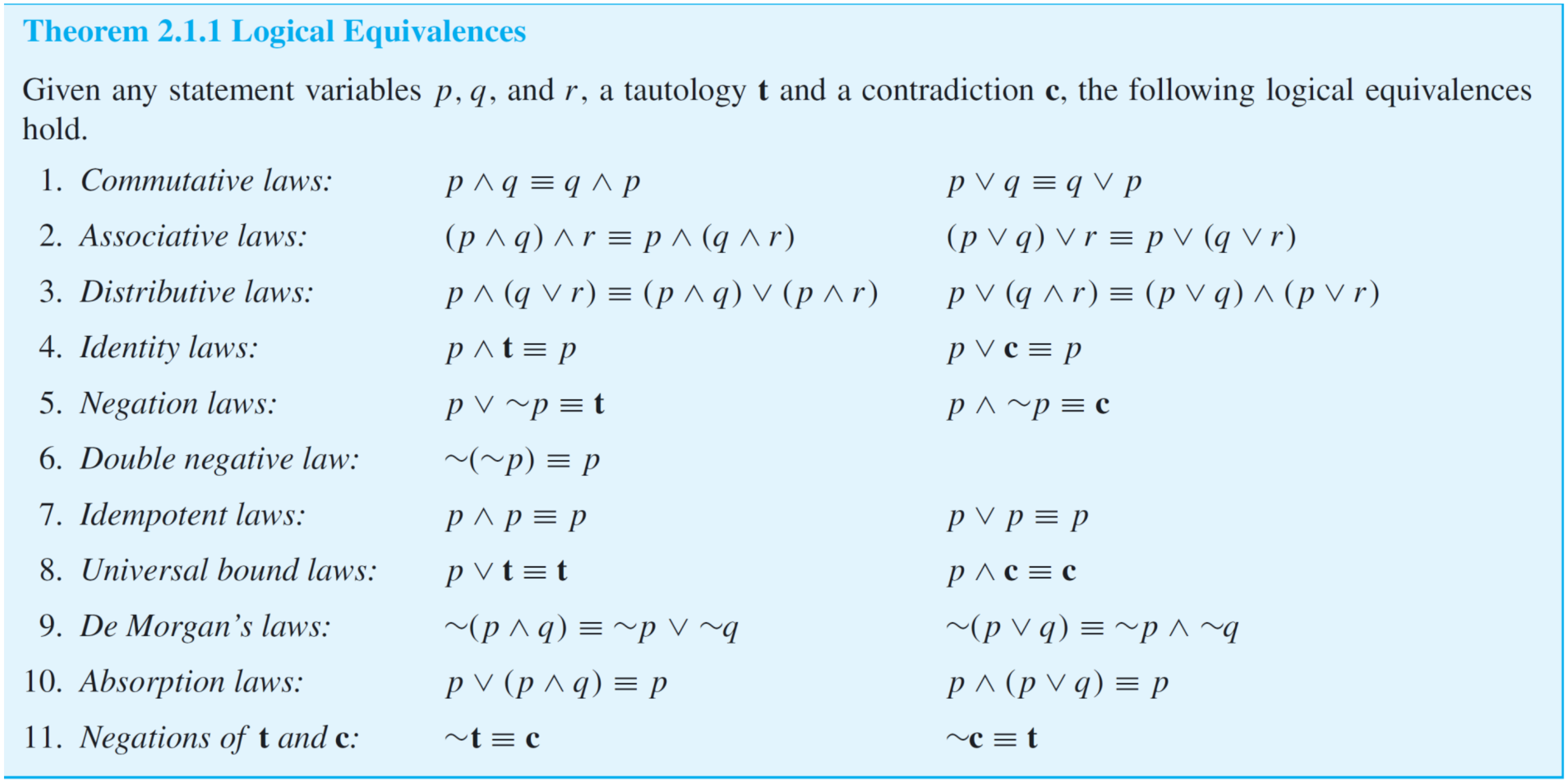
(a)



(b)

Figure 2.4.8

Example 6 – *Showing That Two Circuits Are Equivalent* cont'd



Theorem 2.1.1 Logical Equivalences

Given any statement variables p, q, and r, a tautology $\mathbf{t}$ and a contradiction $\mathbf{c}$, the following logical equivalences hold.

1. *Commutative laws:*	$p \wedge q \equiv q \wedge p$	$p \vee q \equiv q \vee p$
2. *Associative laws:*	$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$	$(p \vee q) \vee r \equiv p \vee (q \vee r)$
3. *Distributive laws:*	$p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
4. *Identity laws:*	$p \wedge \mathbf{t} \equiv p$	$p \vee \mathbf{c} \equiv p$
5. *Negation laws:*	$p \vee \sim p \equiv \mathbf{t}$	$p \wedge \sim p \equiv \mathbf{c}$
6. *Double negative law:*	$\sim(\sim p) \equiv p$	
7. *Idempotent laws:*	$p \wedge p \equiv p$	$p \vee p \equiv p$
8. *Universal bound laws:*	$p \vee \mathbf{t} \equiv \mathbf{t}$	$p \wedge \mathbf{c} \equiv \mathbf{c}$
9. *De Morgan's laws:*	$\sim(p \wedge q) \equiv \sim p \vee \sim q$	$\sim(p \vee q) \equiv \sim p \wedge \sim q$
10. *Absorption laws:*	$p \vee (p \wedge q) \equiv p$	$p \wedge (p \vee q) \equiv p$
11. *Negations of* **t** *and* **c**:	$\sim\mathbf{t} \equiv \mathbf{c}$	$\sim\mathbf{c} \equiv \mathbf{t}$

Example 6 – *Solution*

The Boolean expressions that correspond to circuits (a) and (b) are $((P \wedge \sim Q) \vee (P \wedge Q)) \wedge Q$ and $P \wedge Q$, respectively.

By Theorem 2.1.1,

$$
\begin{aligned}
&((P \wedge \sim Q) \vee (P \wedge Q)) \wedge Q \\
&\qquad \equiv (P \wedge (\sim Q \vee Q)) \wedge Q && \text{by the distributive law} \\
&\qquad \equiv (P \wedge (Q \vee \sim Q)) \wedge Q && \text{by the commutative law for } \vee
\end{aligned}
$$

Example 6 – *Solution*

cont'd

$$\equiv (P \wedge \mathbf{t}) \wedge Q \qquad \text{by the negation law}$$

$$\equiv P \wedge Q \qquad \text{by the identity law.}$$

It follows that the truth tables for $((P \wedge {\sim}Q) \vee (P \wedge Q)) \wedge Q$ and $P \wedge Q$ are the same.

Hence the input/output tables for the circuits corresponding to these expressions are also the same, and so the circuits are equivalent.