Chapter 7

Programming Logic Gate Functions in PLCs

Chapter Outline

Technical Terms

combinational logic gates
sequential logic devices
Boolean expression
Boolean algebra
true state
logic high
false state
logic low
truth table
gate symbols
NOT gate
AND gate
OR gate
NAND gate
NOR gate
XOR gate
XNOR gate
Karnaugh map
Quine-McCluskey routine

Learning Objectives

After completing this chapter, you will be able to:

- Describe combinational and sequential logic gate circuits.
- Create PLC ladder logic programs for NOT, AND, OR, NAND, NOR, XOR, and XNOR logic gates.
- Create Boolean expressions and logic gate circuits from truth tables.
- Use the **Logic Converter** instrument in NI Multisim to create logic tables and Boolean expressions from logic gate circuits.
- Convert Boolean expressions to PLC ladder logic diagrams.
- Convert PLC ladder logic diagrams to logic gate circuits and Boolean expressions.

7.1 Introduction

The majority of PLC manufacturers use the ladder logic diagram programming language to program their programmable logic controllers (PLCs). Some manufacturers prefer using *logic gate circuits* or *Boolean expressions* to program their PLCs. Therefore, it is beneficial to know how to convert one type of PLC programming language to the other.

In this chapter, you will learn how to create logic gate circuits from ladder logic diagrams and vice versa. You will review the functions associated with the combinational logic gates. These gates are the NOT, AND, OR, NAND, NOR, XOR, and XNOR gates. You will learn how to create PLC ladder logic diagrams that emulate the functions of these gates.

7.2 Combinational and Sequential Logic Gate Circuits

Combinational logic gates: Logic devices in which the output of the device is dependent only on the present inputs to the device. There is no dependency on past inputs. Combinational logic gates do not require clock pulses to operate.

Sequential logic devices: Logic devices in which the output of the device is dependent on the present and past inputs to the device. Sequential logic devices require clock pulses to operate.

Combinational logic gates do not require clock pulses to operate. Their outputs depend only on their inputs. This means that the outputs of combinational logic gates are generated instantaneously. Generally, the combinational logic gates are simply called *logic gates*. Seven logic gates exist. Seven logic gates exist: NOT, AND, OR, NAND, NOR, XOR (exclusive OR), and XNOR (exclusive NOR). The gates in a circuit represent a simple Boolean expression. For example, two-input AND gates with inputs A and B and output Y graphically represent the expression $Y = A \cdot B$. **Figure 7-1** displays a logic gate circuit which shows the connection of logic gates for a Boolean expression.

Sequential logic devices have outputs that depend on their inputs as well as time. They require clock pulses. Therefore, an inherent delay time is always present for the sequential logic circuits. Flip-flop devices such as reset-set (RS), JK, delay (D), and toggle (T) are sequential logic devices. **Figure 7-2** displays a sequential logic circuit.

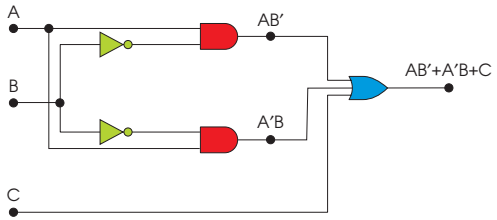


Figure 7-1. A three-input logic gate circuit.

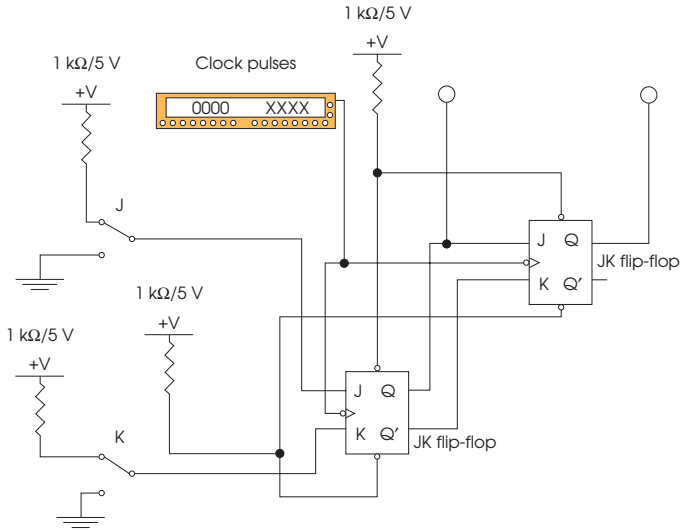


Figure 7-2. Sequential logic circuit. These circuits require clock pulses.

In this textbook, you will study only combinational logic gate circuits. However, you first need to review the concepts of Boolean expressions, truth tables, and gate symbols.

7.3 Boolean Expressions, Truth Tables, and Logic Gate Circuits

In basic algebra, you learned that every function has its own equation. Similarly in the field of digital electronics, every gate logic function has its own equation called a ***Boolean expression***. The nineteenth century British mathematician, George Boole, invented a type of algebra that uses only two conditions or states. The two states are *true* and *false*. This type of algebra using only two states is called ***Boolean algebra*** in honor of Boole.

In Boolean algebra, the ***true state*** is represented by the number one, called ***logic high*** or ***logic one***. The ***false state*** is represented by the number zero, called ***logic low*** or ***logic zero***. In the field of digital electronics, logic high is represented by the presence of a voltage potential.

Boolean expression: Names for equations in Boolean algebra.

Boolean algebra: Form of mathematics that uses two conditions or states: true and false.

True state: State in digital electronics that is represented with a number 1.

False state: State represented in digital electronics with a number zero.

Logic high: State in digital electronics that is represented with 5 volts. Also called *logic one.*

Logic low: State in digital electronics that is represented with zero volts. Also called *logic zero.*

Logic low is represented by the absence of a voltage potential. The logic high in a programmable logic controller is represented with five volts (+5 V), and the logic low is represented with zero volts (0 V).

By applying conventional algebra, you can plot a function's input and output points to create the characteristic of the function as a graph. This graph represents the function pictorially in an x–y coordinate system. The x-axis is for the input points and the y-axis is for the output points. In Boolean algebra, a table contains the digital input and output points. This table is called a ***truth table***. **Figure 7-3** displays a Boolean expression and its truth table. Note that a prime symbol (′) indicates the inverse value of the input. In Figure 7-3, the A with a prime represents the inverse of A.

Truth table: Table used to map Boolean expressions. Truth tables contain Boolean expression inputs and outputs.

Gate symbols: Symbols used to display logic gate devices.

There are schematic symbols for every combinational and sequential logic device. The schematic symbols for logic gates are called ***gate symbols***. Using logic gate symbols, one can create the logic gate circuits. **Figure 7-4** displays the logic gate circuit for Boolean expression in Figure 7-3.

In the following sections, you will learn what type of Boolean expressions, truth tables, and logic gate symbols are used for the seven logic gates introduced in Section 7.2.

Boolean expression: **Y = AB + A′C**

Truth Table

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

Figure 7-3. Boolean expression and its truth table.

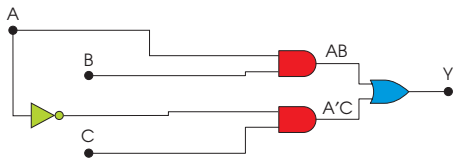


Figure 7-4. Logic circuit for Boolean expression in Figure 7-3.

7.4 NOT Gates or Inverters

NOT gate: Gate that generates a logic high output when all inputs are logic low.

The output of a *NOT gate* is the inverse of the input. The NOT gate is sometimes called an *inverter*. The function of a NOT gate is simulated by the electric circuit displayed in **Figure 7-5.** When the switch is closed, the electric bulb is short circuited, and it turns off. When the switch is open, electric current flows through the lightbulb, and the lightbulb turns on. Like the NOT gate, the output is on when the input is off and vice versa. The input is inverted to generate an output. **Figure 7-6** displays the NOT logic gate symbol, its Boolean expression, and its truth table.

Figure 7-7 displays that there are two different types of PLC ladder logic diagrams that perform the NOT function.

- In rung 0000, the XIO (examine if open) device is connected to the output. Therefore, the XIO device is normally closed and output zero is ON. When you press pushbutton #1 (I:0/0), the output zero (pilot light #1) is turned off. (Notice that address I:0/0 references the port 0 on module 0.)
- In rung 0001, pushbutton #2 (I:0/1) is connected to internal coil bit B3:0/0. (Notice that address I:0/1 references the port 1 on module 0.) In rung 0002, the internal contact bit B3:0/0 is inverted and connected to output one (pilot light #2). When normally open input I:0/1 is open, output one (O:0/1) is ON. Press input 0/1 to close it, then output one will turn OFF.

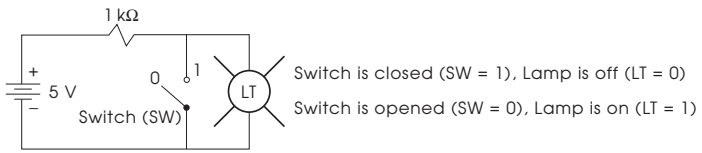
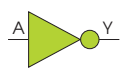


Figure 7-5. Electric circuit emulating the function of a NOT gate.

Boolean expression: **Y = A′**



NOT gate symbol

A	Y
0	1
1	0

Truth table

Figure 7-6. Boolean expression, gate symbol, and truth table for NOT logic gate.

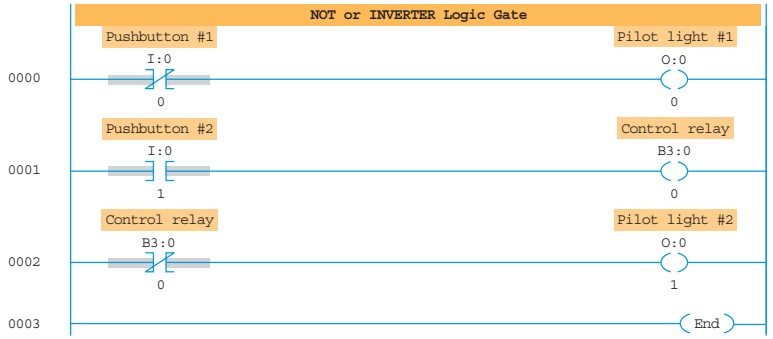


Figure 7-7. Two ways to program a NOT function in a PLC.

7.5 AND Gates

AND gate: Gate that only generates a logic high output when all inputs are logic high.

The function of an *AND gate* is simulated in the electric circuit displayed in **Figure 7-8.** Notice that the lamp will be on only when both switches are closed.

Figure 7-9 displays a two-input AND logic gate symbol, its Boolean expression, and its truth table. In the truth table, you can see that there is only one set of inputs that produces a logic high output.

Figure 7-10 displays a ladder logic diagram that performs the function of a two-input AND gate. When normally open inputs I:0/0 and I:0/1 are closed, output O:0/0 is energized.

7.6 OR Gates

OR gate: Gate that generates a logic high output in all states except when all inputs are logic low.

The function of an *OR gate* is simulated in the electric circuit displayed in **Figure 7-11.** Notice that the lamp will be ON when one *or* both of the switches are closed.

Figure 7-12 displays a two-input OR logic gate symbol, its Boolean expression, and its truth table. The truth table shows a logic high output for all combinations of inputs except where both A and B are low. When either input A, B, or both are on, the output is on.

Figure 7-13 displays a ladder logic diagram that performs the function of a two-input OR gate. When either normally open (NO) inputs I:0/0, I:0/1, or both are closed, output O:0/0 is energized.

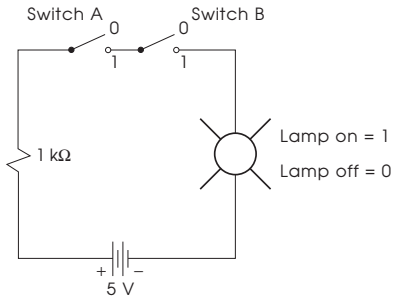
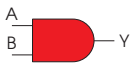


Figure 7-8. Electric circuit emulating an AND gate.

Boolean expression: **Y = A · B**



Two-input AND gate

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

Truth table

Figure 7-9. Boolean expression, gate symbol, and truth table for a two-input AND logic gate.

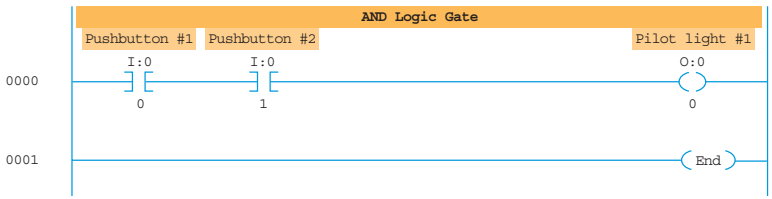


Figure 7-10. AND gate ladder logic diagram.

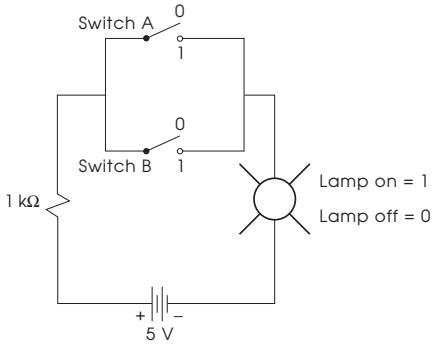
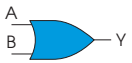


Figure 7-11. Electric circuit emulating an OR gate.

Boolean expression: **Y = A + B**



Two-input OR gate

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

Truth table

Figure 7-12. Boolean expression, gate symbol, and truth table for a two-input OR logic gate.

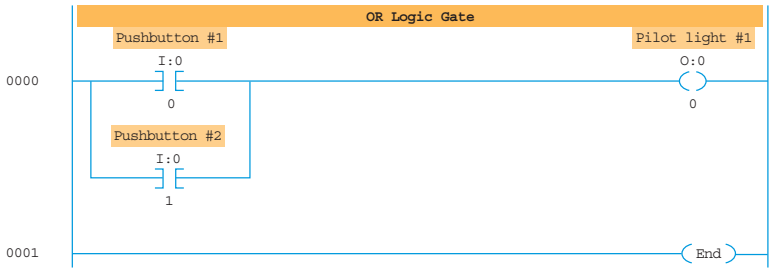


Figure 7-13. Ladder logic diagram for an OR gate.

7.7 NAND Gates

NAND gate: Gate that does not generate a logic high output when all inputs are logic low. An inverted AND.

The function of a *NAND gate* is simulated in the electric circuit displayed in **Figure 7-14.** Notice that the lamp will be off when both switches are closed. The NAND gate takes its name from NOT and AND. Its outputs are the inverse of the AND gate.

Figure 7-15 displays a two-input NAND logic gate symbol, its Boolean expression, and its truth table. Notice that the NAND gate can be built by connecting an AND gate in series with a NOT gate. Using the De-Morgan theorem, sometimes also called the Bubble method, you can convert a NAND gate to an OR gate with inverted inputs where $(A \cdot B)' = A' + B'$.

Figure 7-16 displays that there are two different types of ladder logic diagrams that perform the NAND function.

- Both normally closed inputs I:0/0 and I:0/1 must be energized (opened) to turn off the output O:0/0.

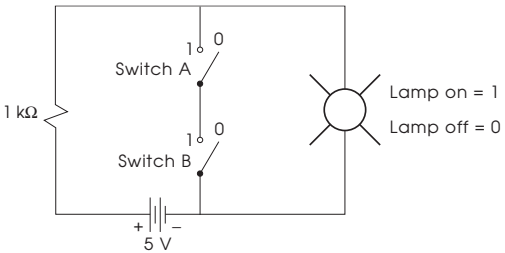
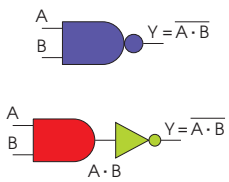


Figure 7-14. Electric circuit emulating a NAND gate.

Boolean expression: $Y = \overline{A \cdot B} = \overline{A} + \overline{B}$



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

Truth table

Figure 7-15. Boolean expression, gate symbol, and truth table for a NAND logic gate.

- When both normally open inputs I:0/2 and I:0/3 are energized (closed), the relay coil B3:0/0 is energized. Then the normally closed contact B3:0/0 is opened to turn off output O:0/1.

7.8 NOR Gates

NOR gate: Gate that generates a logic high output when all inputs are logic low. An inverted OR.

The function of a NOR logic gate is simulated in the electric circuit displayed in **Figure 7-17.** Notice that the lamp will be ON when both switches are open. The *NOR gate* takes its name from NOT and OR. Its outputs are the inverse of the OR gate.

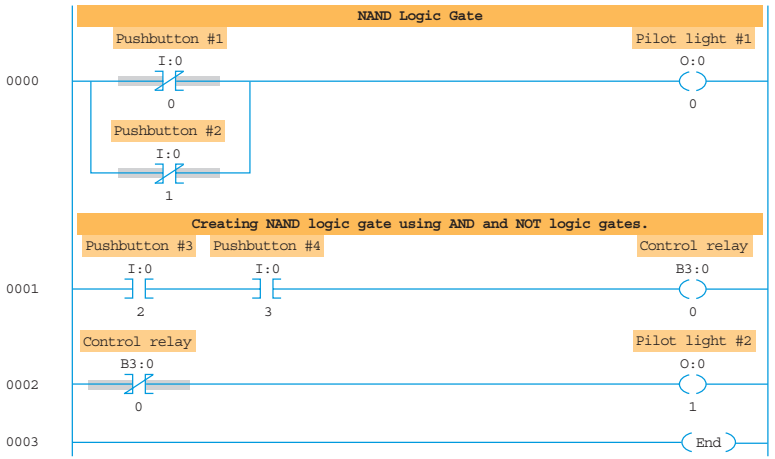


Figure 7-16. Ladder logic diagram for a NAND gate.

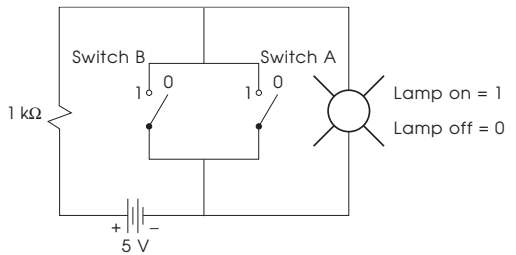
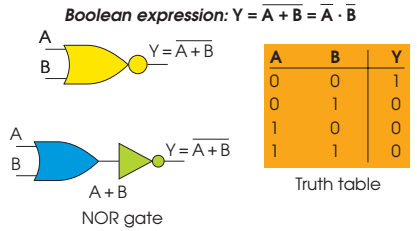


Figure 7-17. Electric circuit emulating a NOR gate.

Figure 7-18 displays a two-input NOR logic gate symbol, its Boolean expression, and its truth table. Notice the NOR gate can be built by connecting an OR gate in series with a NOT gate. Using the De-Morgan theorem, you can convert a NOR gate to an AND gate with inverted inputs where $(A + B)' = A' \cdot B'$.

Figure 7-19 displays that there are two different types of ladder logic diagrams that perform the NOR gate function.

- Both normally closed inputs I:0/0 and I:0/1 must be de-energized (remain closed) to turn on the output O:0/0.



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

Truth table

Figure 7-18. Boolean expression, gate symbol, and truth table for a NOR logic gate.

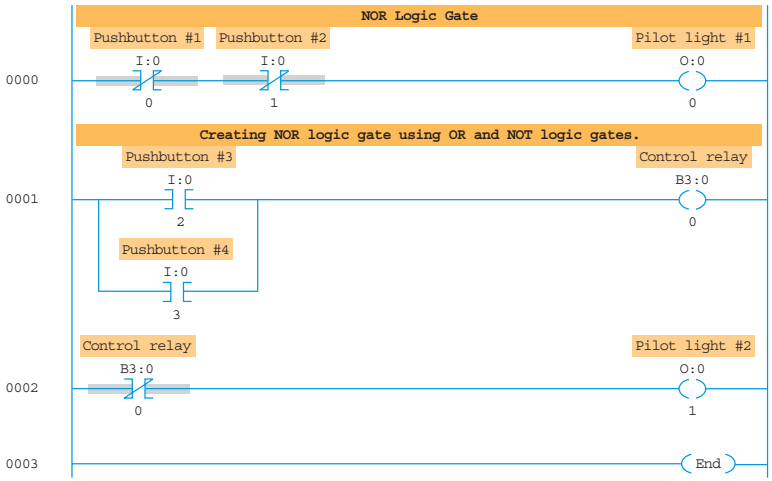


Figure 7-19. Ladder logic diagram for a NOR gate.

- When both normally open inputs I:0/2 and I:0/3 are de-energized, the relay coil B3:0/0 is de-energized. Then the normally closed contact B3:0/0 remains closed to turn on output O:0/1.

7.9 XOR (Exclusive OR) Gates

XOR gate: Gate that generates a logic high output when one input is logic high and the other input is logic low or vice versa.

The function of an *XOR (exclusive OR)* gate is simulated in the electric circuit displayed in **Figure 7-20.** Notice that the lamp will be on if one switch is open while the other switch is closed.

Figure 7-21 displays an XOR logic gate symbol, its Boolean expression, and its truth table. Looking at the truth table, you can see that either inputs A or B (but not both) must be high to produce a high output. One input must be ON and the other one OFF in order to have the output ON.

Figure 7-22 displays a ladder logic diagram that performs the function of an XOR gate. When I:0/0 is on, I:0/1 must be off and vice versa in order to turn on output O:0/0. When either Pushbutton #1 or Pushbutton #2 is pressed, the output is ON. When both pushbuttons are pressed, output is OFF.

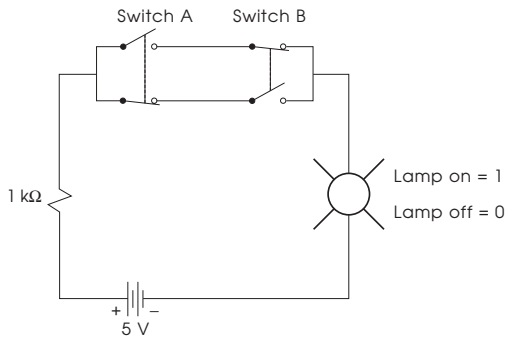
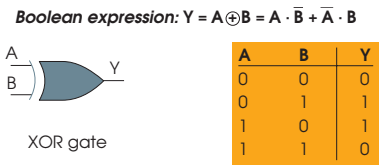


Figure 7-20. Electric circuit emulating an XOR gate.



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

Figure 7-21. Boolean expression, gate symbol, and truth table for an XOR logic gate.

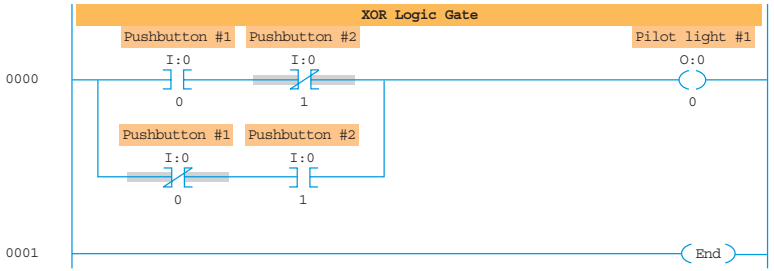


Figure 7-22. Ladder logic diagram for an XOR gate.

7.10 XNOR (Exclusive NOR) Gates

The function of an *XNOR (exclusive NOR)* gate is simulated in the electric circuit displayed in **Figure 7-23.** Notice that the lamp will be on when either both switches are open or closed. The lamp will not be on if only switch A is activated, or if only switch B is activated.

XNOR gate: Gate that generates a logic high output when either both inputs are logic high or both inputs are logic low.

Figure 7-24 displays an XNOR logic gate symbol, its Boolean expression, and its truth table. Notice the XNOR gate can be built by connecting an XOR gate in series with the NOT gate. When either or both pushbuttons are pressed or not pressed, the output is ON.

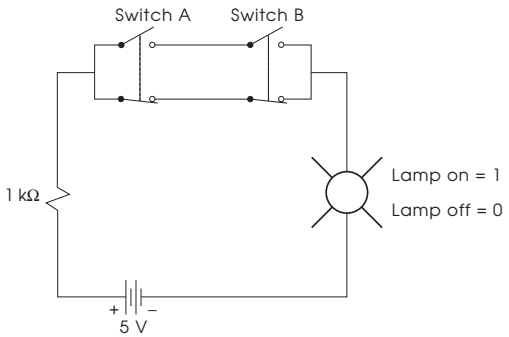


Figure 7-23. Electric circuit emulating an XNOR gate.

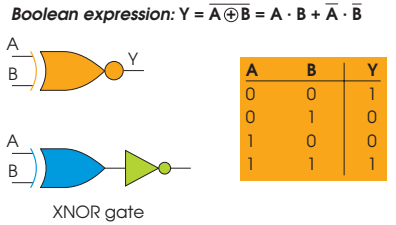


Figure 7-24. Boolean expression, gate symbol, and truth table for an XNOR logic gate.

Figure 7-25 displays that there are two different types of ladder logic diagrams that perform the XNOR function.

- Both inputs I:0/0 and I:0/1 must be on or off to turn on the output O:0/0.
- When I:0/2 is on and input I:0/3 is off or vice versa (i.e., XOR gate), the relay coil B3:0/0 is energized. Then the normally closed contact B3:0/0 opens to turn off output O:0/1.

7.11 Simplifying Boolean Expressions

In previous sections, the use of Boolean expressions, truth tables, and logic gate circuits were studied. You saw how logic gates were converted to PLC ladder logic diagrams. In this and the next two sections, you will study how to create PLC ladder logic diagrams from truth tables.

To convert a truth table to a PLC ladder logic diagram, you must first find its *simplified Boolean expression*. The next step is to use the gate logic to PLC ladder diagram conversion routine to create the PLC ladder logic diagram. Three methods are used to simplify Boolean expressions.

- Karnaugh maps.
- Quine-McCluskey routine.
- Electronic simulation software.

Karnaugh Maps

Karnaugh map: A tool that can be used to simplify Boolean expressions. This is an older, difficult method for Boolean expression simplification.

Karnaugh maps (K-Map) are graphical representations of truth tables. They use columns and rows to represent each term in a truth table. For an *n*-variable input truth table, there are 2^n boxes in a Karnaugh map. For example, for a two-input truth table, four boxes (2^2) are needed. A K-Map has a box for every line in the truth table.

Binary numbers are placed above each column and to the left of each row. **Figure 7-26** shows how binary numbers related to the input values are placed in a two-input, three-input, and four-input K-Map.

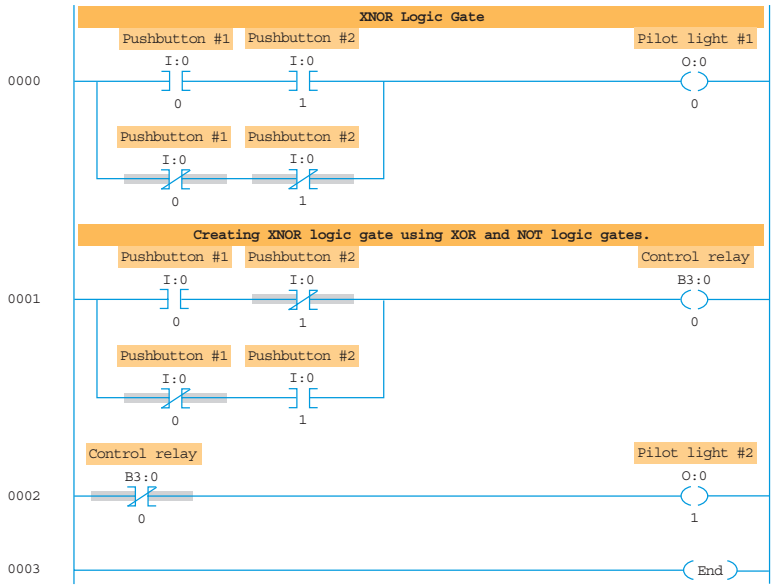


Figure 7-25. Ladder logic diagram for an XNOR gate.

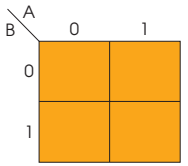
Notice that the input values are placed so that the values for adjacent columns and rows change only a single bit. For example, for a three-input K-Map, binary numbers related to inputs A and B are placed above the columns in the order 00, 01, 11, and 10. Binary numbers related to input C are placed to the left of the rows in the order 0 and 1. This ordering follows the Gray code system explained in Chapter 3.

To use the K-Map, the expression must be in a sum of products (SOP) form, such as AB′ + BC. This means that the Boolean expression consists of groups that are created from ANDed inputs. Then, the groups are summed (ORed) to create the entire Boolean expression. Use the following steps and refer to **Figure 7-27** to simplify the Boolean expressions using K-Maps.

1. Select an appropriate K-Map that has the correct number of input boxes, such as two-input and three-input. As stated, for an *n*-variable input truth table, there will be 2^n boxes. Therefore, for a two-variable (A and B) input table, there will be 2^2 boxes, or 4 boxes.

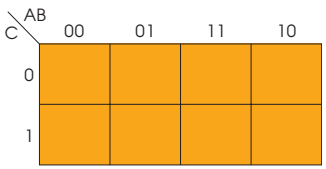
Two-Input Karnaugh Map

A	B	Y
0	0	
0	1	
1	0	
1	1	



Three-Input Karnaugh Map

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



Four-Input Karnaugh Map

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

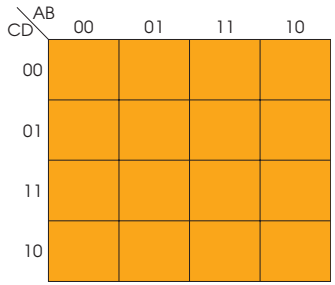


Figure 7-26. Two-input, three-input, and four-input K-Maps.

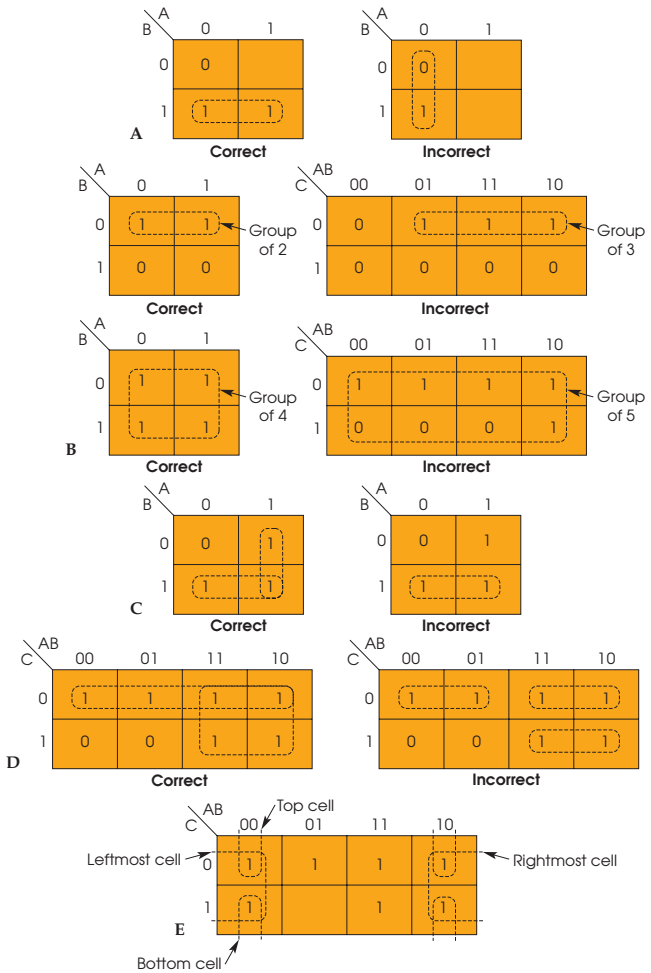
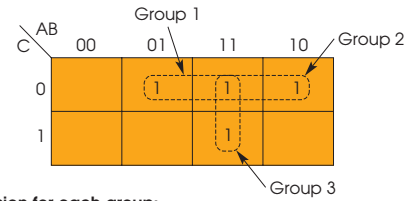


Figure 7-27. Simplifying Boolean expressions using K-Maps. A—Grouping pairs of binary 1s in adjacent cells. B—Grouping even number of 1s in adjacent cells. C—Grouping 1s in the adjacent cells. D—Groups must be as large as possible. E—Groups can wrap around the K-Map.

2. Plot only the terms in which Y = 1.
3. Follow the rules below for grouping the 1s in the K-Map that lead to simplifying the expression.
   - Adjacent groups with binary number 1 in them must be combined in groups of 1, 2, 4, 8, 16, and so on. See Figure 7-27A.
   - Each group must contain an even number of binary 1s. See Figure 7-27B.
   - Every 1 in adjacent cells must be included in a group. See Figure 7-27C.
   - The same 1 can be used in two or more overlapping groups. See Figure 7-27C.
   - Each group should be as large as possible. See Figure 7-27D.
   - The map can be considered closed, so that the end boxes are grouped adjacently (top and bottom, or left and right). Figure 7-27E shows how groups wrap around the K-Map.
4. Write the Boolean expressions for each group, and then simplify the expression by retaining only the common variables. See **Figure 7-28.**
5. Then, sum the common variables from each group to create the simplified sum of product (SOP) Boolean expression. See Figure 7-28.

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



Boolean expression for each group:
Group 1 = A′(B C′) + A(B C′)
Common variables = BC′

Group 2 = (A B C′) + (A B′ C′)
Common variables = AC′

Group 3 = (A B C′) + (A B C)
Common variables = AB

Simplified SOP Boolean expression:
Y = BC′ + AC′ + AB

Figure 7-28. Simplified sum of product (SOP) Boolean Expression.

Example 7-1

The example displayed in **Figure 7-29** illustrates how to use a Karnaugh map to find the simplified SOP Boolean expression. Examine the steps used to simplify.

1. There are three inputs: A, B, and C. Therefore, select a three-input Karnaugh map. Note that the map will have 2^3 (8) boxes.
2. Plot only the terms in which Y = 1.
3. Group the adjacent logic highs (1s). Remember, each group should be as large as possible. For this Karnaugh map, there is only one group.
4. Then, write the expression for the group and then simplify the expression by retaining the common variable(s). The simplified SOP Boolean expression is Y = C.

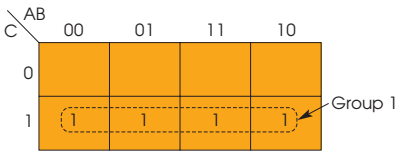
Quine-McCluskey Routine

For more than five inputs, the Karnaugh map method becomes very difficult. Therefore, for more than five input variables, the *Quine-McCluskey routine* is a better method for simplifying Boolean expressions. The Quine-McCluskey routine is a complicated method that uses the Boolean algebraic simplification rules to find the simplified Boolean expression. We will not study the theory of the Quine-McCluskey method in this textbook, but you should know that this method exists and might be used in an advanced course.

Quine-McCluskey routine: Tool used as an advanced Boolean expression simplification routine.

Example 7-1

A	B	C	Y
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



Boolean expression for each group:
Group 1 = A′ B′Ⓒ + A′ BⒸ + A BⒸ + A B′Ⓒ
Common variable(s) = C

Simplified SOP Boolean expression:
Y = C

Figure 7-29. Using a Karnaugh map to find a simplified Boolean expression for Example 7-1.

Electronic Circuit Simulation

Using electronic circuit simulation software to find the simplified Boolean expression is the easiest method. This type of software allows you to enter the input and output data and solves and simplifies the expression for you.

An example of this type of software is NI Multisim. In Example 7-2, you will learn how to use NI Multisim to find the simplified Boolean expression of a truth table. If you have access to this software, work through the following example.

Example 7-2

Open the NI Multisim program. From the **Instruments** toolbar, click the **Logic Converter** icon. Then, click a space in the work area to place the converter. Double-click the **Logic Converter** image to open the **Logic Converter** dialog box. **Figure 7-30** displays the **Logic Converter** instrument and dialog box. Click the inputs A, B, C, and D since your example truth table has those four inputs. Next, click the output column and type the output bits for the truth table displayed in **Figure 7-31** (column Y). Click the **Simplify** button to find the simplified Boolean expression:

A′CD + B′CD + BC′D + ABCD′

The answer appears in the rectangle at the bottom of Figure 7-30. The simplified Boolean expression consists of the sum of four product

Truth Table:

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

Figure 7-30. Truth table for Example 7-2.

expressions. Therefore, four rungs must be connected in parallel. Each rung has series input devices on it. **Figure 7-32** displays the PLC ladder logic diagram created from the simplified Boolean expression.

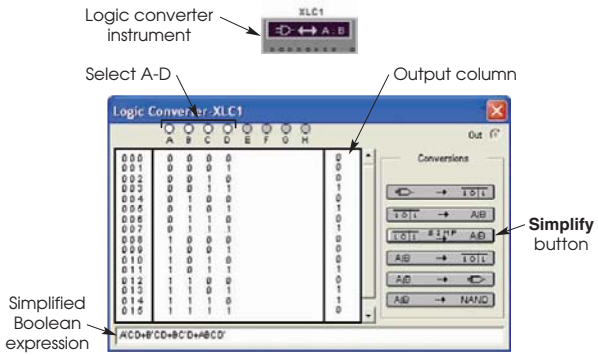


Figure 7-31. Using the **Logic Converter** instrument to find the simplified Boolean expression for Example 7-2.

Example 7-2

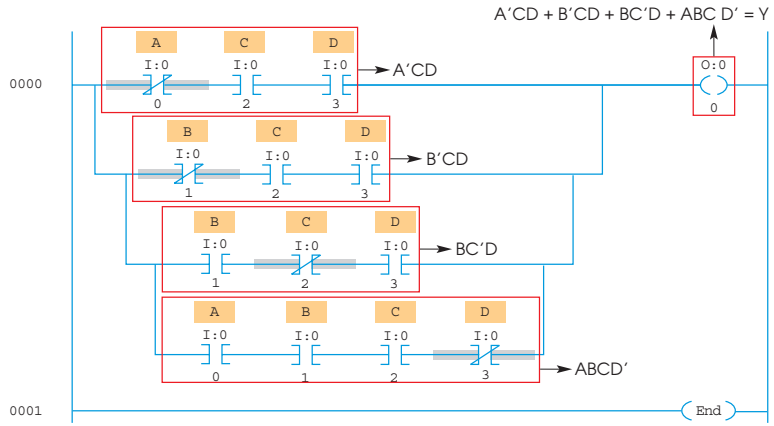


Figure 7-32. PLC ladder logic diagram for Example 7-2.

7.12 Creating PLC Ladder Logic Diagrams from Logic Gate Circuits

In previous sections, you learned how to create PLC ladder logic diagrams for the logic gates. To create the ladder logic diagram from a logic gate circuit, you must convert each gate to its equivalent ladder logic diagram. Note that these simple conversions were discussed in sections 7.4 through 7.10. In this section, three examples are used to illustrate how to create PLC ladder logic diagrams for logic gate circuits. Study the procedures used in the following examples.

Example 7-3

Create the PLC ladder logic diagram for the logic gate circuit displayed in **Figure 7-33.**

Examine Figure 7-33. The pilot light red (PLTR) output section has three inputs: PBR, PBG, and SW. Pushbutton red (PBR) and pushbutton green (PBG) are inputs to an XOR logic gate. The output of the XOR logic gate and the inverted switch (SW) are inputs to a two-input AND logic gate. These inputs generate the pilot light red (PLTR) output.

The two-input AND logic gate output is also fed into a two-input NAND logic gate. The temperature switch (TSW) is another input to the NAND logic gate. The output generated from the NAND logic gate is labeled pilot light white (PLTW).

Using the transformations described in Section 7.7 (on NAND gates) and Section 7.9 (on XOR gates), you can generate a PLC ladder logic diagram. **Figure 7-34** displays the PLC ladder logic diagram for Example 7-3.

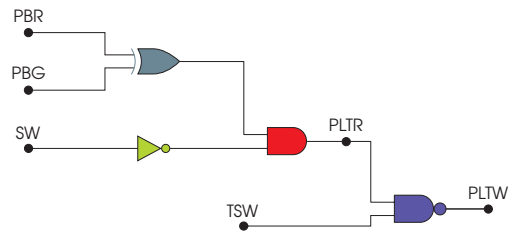


Figure 7-33. Logic gate circuit for Example 7-3.

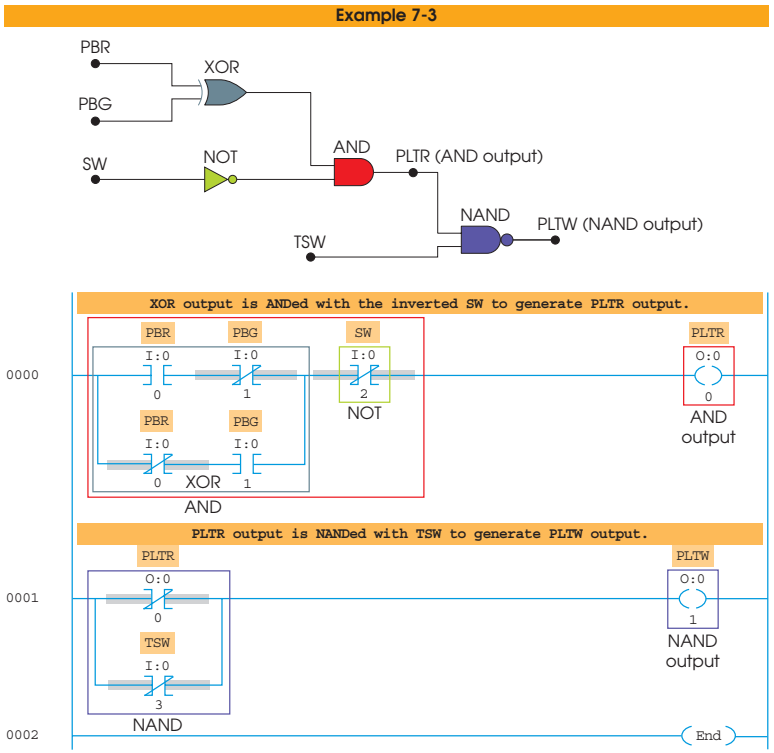


Figure 7-34. PLC ladder logic diagram for Example 7-3.

Example 7-4

Create the PLC ladder logic diagram for the logic gate circuit displayed in **Figure 7-35.**

Pushbutton red (PBR) and pushbutton green (PBG) are inputs to a two-input AND gate. The output of the AND logic gate and the switch (SW) are inputs to a two-input OR logic gate. The output of the OR logic gate and an inverted temperature switch (TSW) are inputs to a second AND logic gate. This AND logic gate generates the output for pilot light red (PLTR).

Using the transformations from Section 7.5 (AND gates) and Section 7.6 (OR gates) you can create the PLC ladder logic diagram. **Figure 7-36** displays the PLC ladder logic diagram for Example 7-4.

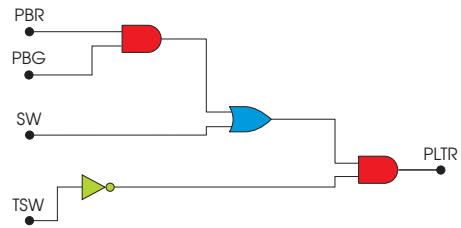


Figure 7-35. Logic gate circuit for Example 7-4.

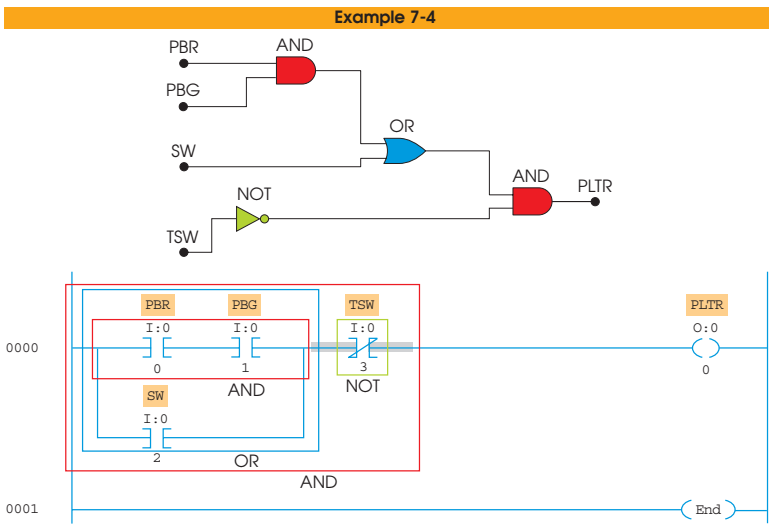


Figure 7-36. PLC ladder logic diagram for Example 7-4.

Example 7-5

Create the PLC ladder logic diagram for the logic gate circuit displayed in **Figure 7-37.**

Inputs A and B are fed into an XNOR logic gate. Inputs D and E are fed into a NOR logic gate. Outputs of the XNOR and NOR logic gates plus input C are fed into a three-input OR logic gate. The three-input OR logic gate generates output Y. **Figure 7-38** displays the PLC ladder logic diagram for Example 7-5.

7.13 Creating PLC Ladder Logic Diagrams from Boolean Expressions

Some manufacturers use Boolean expressions to program their PLCs. In this section, you will learn how to use Boolean expressions to create PLC ladder logic diagrams.

Work through the following three examples that illustrate how to create ladder logic diagrams for Boolean expressions.

Example 7-6

Create the PLC ladder logic diagram for the following Boolean expression.

$$Y = A' + B + CD + EB$$

To create the diagram, each rung or each portion of a rung is created by replacing the Boolean letter with the inputs that match. **Figure 7-39** summarizes the Boolean expressions and ladder diagrams for the logic gates covered in Sections 7-4 through 7-10.

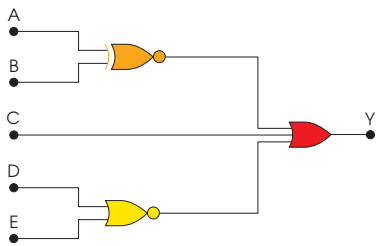


Figure 7-37. Logic gate circuit for Example 7-5.

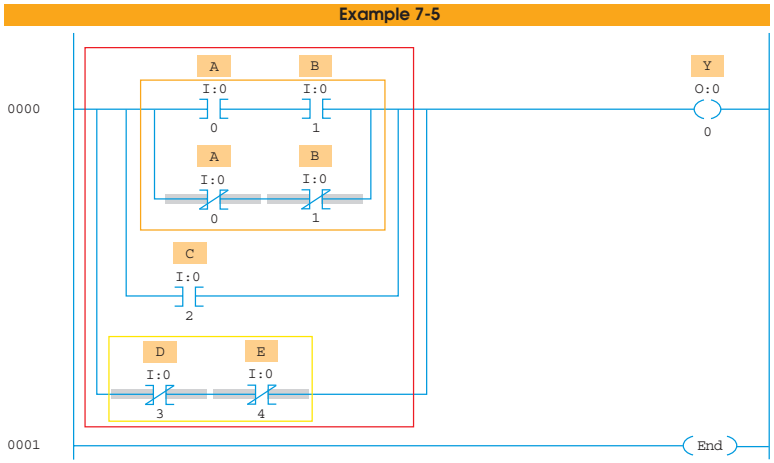


Figure 7-38. PLC ladder logic diagram for Example 7-5.

Figure 7-40 displays the PLC ladder logic diagram. Notice that inverted A, B, CD, and EB inputs are in parallel (OR). Inputs C and D are in series (AND). Inputs E and B are also in series (AND).

Example 7-7

Create the PLC ladder logic diagram for the following Boolean expression.

$$Y = (AB)' + AC + BC$$

Figure 7-41 displays the PLC ladder logic diagram. Notice that inverted A and B inputs are in series to generate output at the control relay. Then, the inverted output of AB is in parallel with AC and BC.

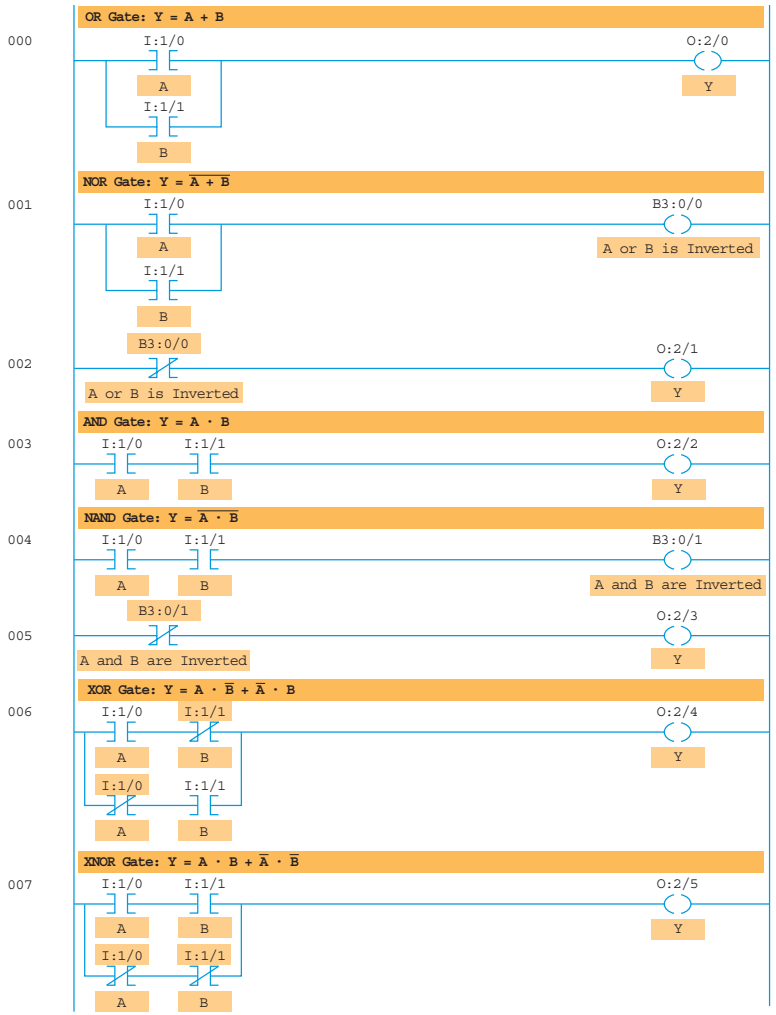


Figure 7-39. Boolean expression and ladder diagram summary.

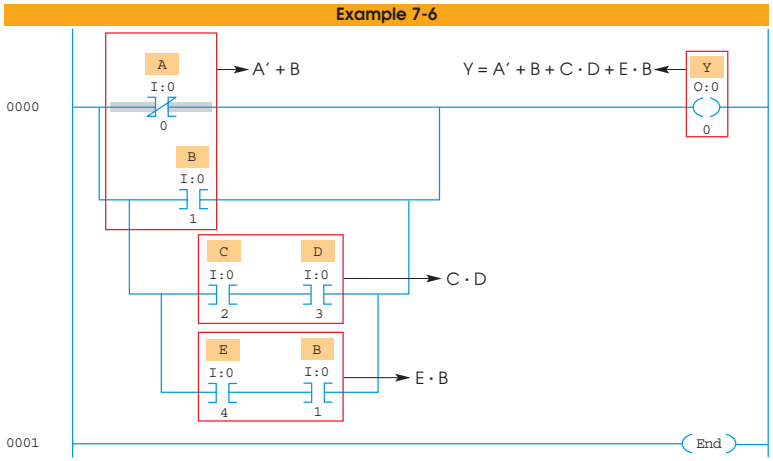


Figure 7-40. PLC ladder logic diagram for Example 7-6.

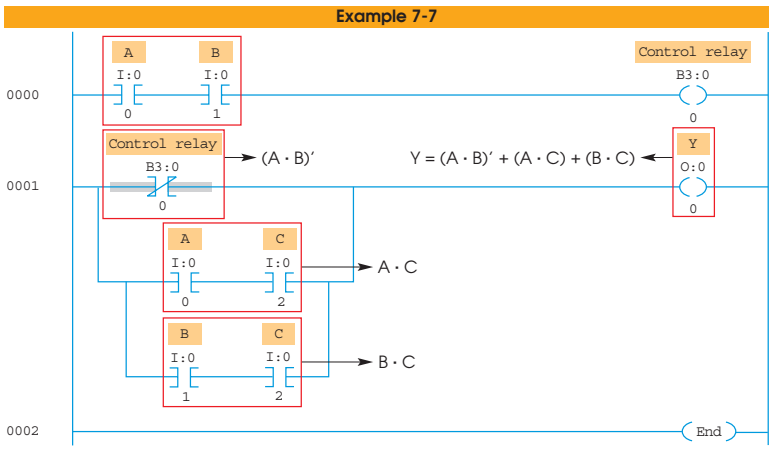


Figure 7-41. PLC ladder logic diagram for Example 7-7.

Example 7-8

Create the PLC ladder logic diagram for the following Boolean expression.

$$Y = (A + B) \cdot (C + D)$$

Figure 7-42 displays the PLC ladder logic diagram. Notice that A is in parallel with B and C is in parallel with D. Then, (A + B) and (C + D) are in series.

7.14 Creating Logic Gate Circuits from PLC Ladder Logic Diagrams

In Sections 7.11, 7.12, and 7.13, you learned how to create PLC ladder logic diagrams from truth tables, logic gate circuits, and Boolean expressions. In this section, you will see the reverse process. You will use examples to study how to convert PLC ladder logic diagrams to logic gate circuits. The first step in this process is to find the Boolean expression that represents the ladder logic diagram. You can then draw the logic gate circuit using the Boolean expression similar. You can also use the logic converter instrument in the NI Multisim program to find truth tables and Boolean expressions from the logic gate circuits. Three examples are used to illustrate how to convert PLC ladder logic diagrams to logic gate circuits.

Example 7-9

Create the logic gate circuit for the PLC ladder logic diagram displayed in **Figure 7-43.** First, turn the PLC ladder logic diagram shown in Figure 7-43 into a Boolean expression as shown in the ladder

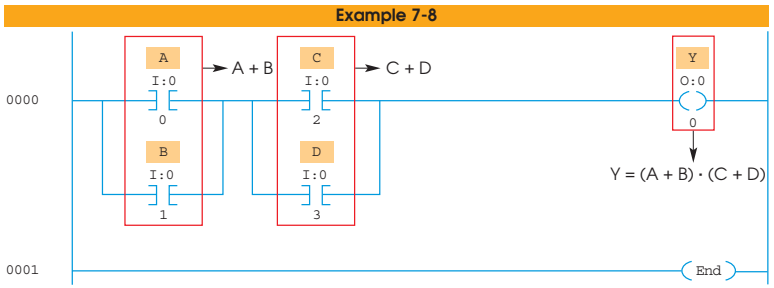


Figure 7-42. PLC ladder logic diagram for Example 7-8.

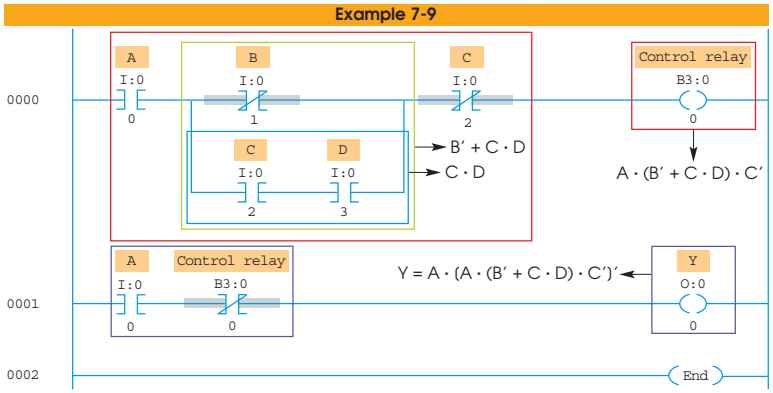


Figure 7-43. PLC ladder logic diagram for Example 7-9.

diagram. Notice that in rung 0000, inverted input B is in parallel (ORed) with serial (ANDed) inputs C and D. This is represented by the following expression:

$$B' + C \cdot D$$

This combination (B′ + C · D) is in serial (ANDed) with input A and inverted input C′ in rung 0000. Therefore, the control relay output for this rung is the following:

$$A \cdot (B' + C \cdot D) \cdot C'$$

In rung 0001, input A is in serial (ANDed) with the inverted output of the control relay from rung 0000 to generate the Boolean expression for output Y:

$$Y = A \cdot [A \cdot (B' + C \cdot D) \cdot C']'$$

Next, create the logic gate circuit using the Boolean expression from Figure 7-43. **Figure 7-44** displays the logic gate circuit for this example. Notice that ANDed inputs C and D and inverted input B are ORed for the following Boolean expression:

$$B' + C{\cdot}D$$

Then, this is ANDed with input A and inverted input C for the following Boolean expression:

$$A \cdot (B' + C \cdot D) \cdot C'$$

Finally, this output is inverted and ANDed with input A for the output of Y:

$$Y = A \cdot [A \cdot (B' + C{\cdot}D) \cdot C']'$$

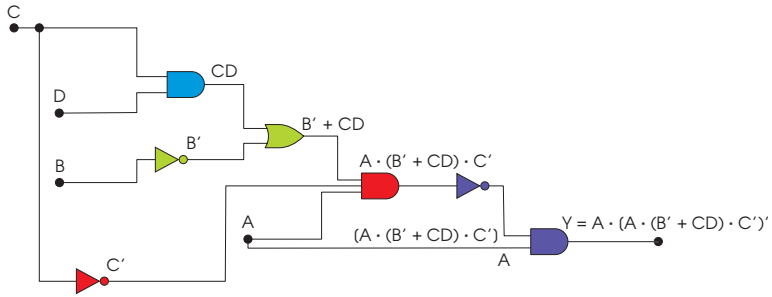


Figure 7-44. Logic gate circuit for Example 7-9.

Example 7-10

Create the logic gate circuit for the PLC ladder logic diagram displayed in **Figure 7-45.** First, turn the PLC ladder logic diagram shown in Figure 7-45 into a Boolean expression as shown in the ladder diagram. Notice that in rung 0000, inputs C and D in the bottom parallel branches create an XOR logic gate (CD′ + C′D). This is in parallel (ORed) with input B, forming the following Boolean expression:

$$B + (C \cdot D' + C' \cdot D)$$

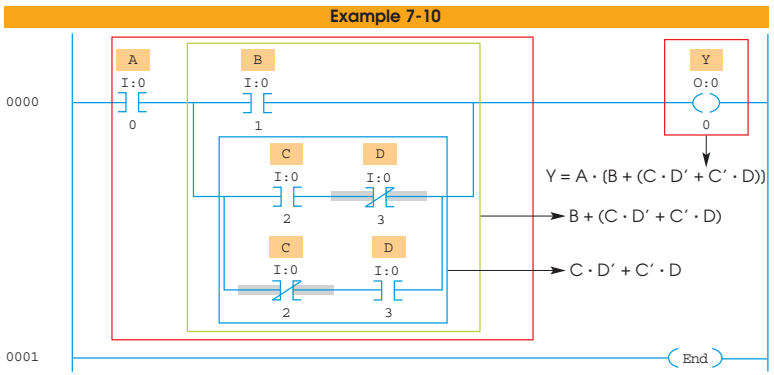


Figure 7-45. PLC ladder logic diagram for Example 7-10.

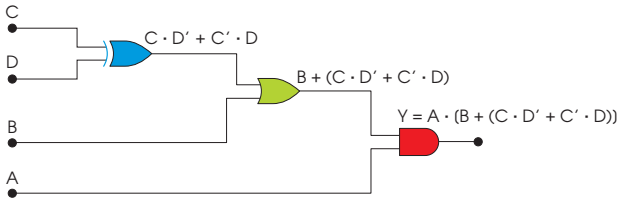


Figure 7-46. Logic gate circuit for Example 7-10.

Finally, this combination is in serial (ANDed) with input A to generate output Y:

$$Y = A \cdot [B + (C \cdot D' + C' \cdot D)]$$

Next, create the logic gate circuit using the Boolean expression from Figure 7-45. **Figure 7-46** displays the logic gate circuit for this example. Notice that inputs C and D are XORed and then ORed with input B. This combination is then ANDed with input A for the output of Y:

$$Y = A \cdot [B + (C \cdot D' + C' \cdot D)]$$

Example 7-11

Create the logic gate circuit for the PLC ladder logic diagram displayed in **Figure 7-47.** First, turn the PLC ladder logic diagram shown in Figure 7-47 into a Boolean expression as shown in the ladder diagram. Notice that in rung 0000, parallel (ORed) inputs A, B, and inverted C are in parallel (ORed) with the output from the control relay which is in series (ANDed) with inverted input D, forming the following Boolean expression:

$$(A + B + C') + [D' \cdot (A + B + C' + D')]$$

Finally, inverted input E′ in rung 0001 is in serial (ANDed) with the output of the control relay. This is equal to the output of Y:

$$Y = E' \{(A + B + C') + [D' \cdot (A + B + C' + D')]\}$$

Next, create the logic gate circuit using the Boolean expression from Figure 7-47. **Figure 7-48** displays the logic gate circuit for this example.

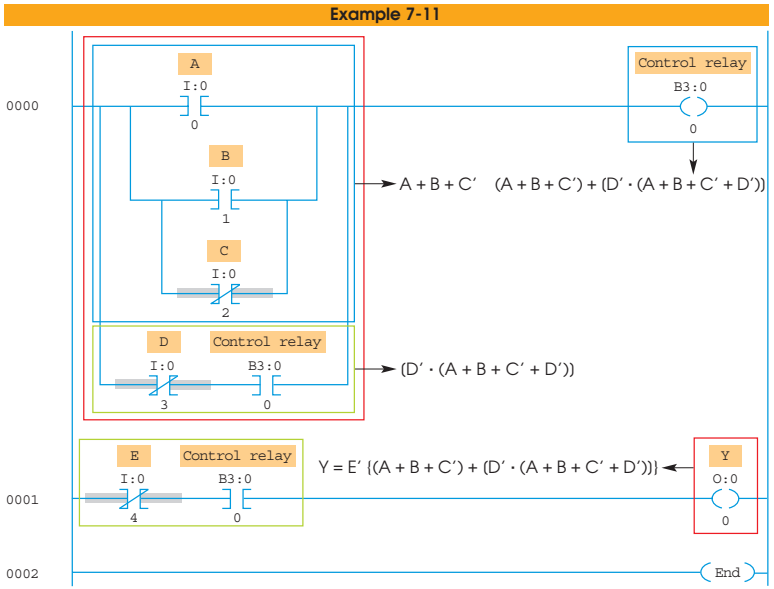


Figure 7-47. PLC ladder logic diagram for Example 7-11.

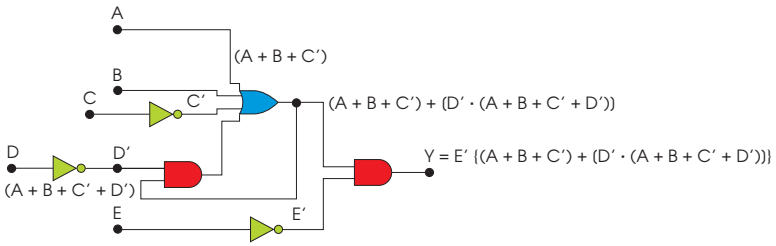


Figure 7-48. Logic gate circuit for Example 7-11.

Summary

- Combinational logic gates do not require clock pulses to operate.
- Combinational logic gates are called *logic gates.* There are seven logic gates: NOT, AND, OR, NAND, NOR, XOR (exclusive OR), and XNOR (exclusive NOR).
- Sequential logic devices require clock pulses and have outputs that depend on their inputs as well as time.
- Flip-flop devices such as reset-set (RS), JK, delay (D), and toggle (T) are sequential logic devices.
- Every gate logic function has its own equation called a Boolean expression.
- A type of algebra using only two states is called Boolean algebra in honor of Boole.
- In Boolean algebra, the true state is represented by the number one, called logic high or logic one. The false state is represented by the number zero, called logic low or logic zero.
- In Boolean algebra, a table, called a truth table, contains the digital input and output points.
- A prime symbol (′) indicates the inverse value of the input.
- The output of a NOT gate is the inverse of the input. The NOT gate is sometimes called an inverter.
- Karnaugh maps (K-Maps) are graphical representations of truth tables that use columns and rows to represent each term in a truth table.

Review Questions

Complete each of the following sentences with the correct word(s).

1. When the input to a NOT gate is a logic high, the output is a logic _____.
2. You can create a NOT gate ladder logic diagram using either one rung or _____ rungs.
3. In an one-rung NOT gate ladder logic diagram, the input instruction must be normally _____.
4. In a two-rung NOT gate ladder logic diagram, the input instruction must be normally _____.
5. In a two-rung NOT gate ladder logic diagram, you must use a(n) _____ contact.
6. In an AND gate ladder logic diagram, when both inputs are closed, the output is _____.
7. In an AND gate ladder logic diagram, when both inputs are open the output is _____.

8. In an AND gate ladder logic diagram, when one input is closed and the other one is open, the output is _____.
9. An AND gate ladder logic diagram requires _____ rung(s).
10. In a NAND gate ladder logic diagram, when both inputs are open, the output is _____.
11. In a NAND gate ladder logic diagram, when both inputs are closed, the output is _____.
12. In a NAND gate ladder logic diagram, when one input is closed and the other one is open, the output is _____.
13. You can create a NAND gate logic diagram using either one rung or _____.
14. In an one-rung NAND gate ladder logic diagram, the inputs must be in _____.
15. In an OR gate ladder logic diagram, when both inputs are closed, the output is _____.
16. In an OR gate ladder logic diagram, when both inputs are open, the output is _____.
17. In an OR gate ladder logic diagram, when one input is closed and the other one is open, the output is _____.
18. An OR gate ladder logic diagram requires _____ rung(s).
19. The input for an OR gate ladder logic diagram must be normally _____.
20. In a NOR gate ladder logic diagram, when both inputs are open, the output is _____.
21. In a NOR gate ladder logic diagram, when both inputs are closed, the output is _____.
22. In a NOR gate ladder logic diagram, when one input is closed and the other one is open, the output is _____.
23. In a one-rung NOR gate ladder logic diagram, both inputs must be normally _____.
24. In a two-rung NOR gate ladder logic diagram, inputs are connected in _____.
25. In an XOR gate ladder logic diagram, when both inputs are open, the output is _____.
26. In an XOR gate ladder logic diagram, when both inputs are closed, the output is _____.
27. In an XOR gate ladder logic diagram, when one input is closed and the other one is open, the output is _____.
28. The inputs in an XOR gate ladder logic diagram are connected in _____ and _____.
29. The inputs in an XOR gate ladder logic diagram are normally _____ and _____.

30. In an XNOR gate ladder logic diagram, when both inputs are open, the output is _____.
31. In an XNOR gate ladder logic diagram, when both inputs are closed, the output is _____.
32. In an XNOR gate ladder logic diagram, when one input is closed and the other one is open, the output is _____.
33. In a one-rung XNOR gate ladder logic diagram, the normally closed inputs are connected in _____.

Specify if the following statements are true or false.

34. Both inputs to an AND gate must be high to produce a high output.
35. Both inputs to a NAND gate must be high to produce a high output.
36. Inverting a NAND gate will result in creating an AND gate.
37. Only one input to an OR gate must be high to produce a high output.
38. Both inputs to a NOR gate must be high to produce a high output.
39. Inverting an OR gate will result in creating a NOR gate.
40. Both inputs to an XOR gate must be high to produce a high output.
41. Both inputs to an XNOR gate must be high to produce a high output.
42. All three inputs to a three-input AND gate must be high to produce a high output.

Create a PLC ladder diagram for the truth tables in the following problems.

43.

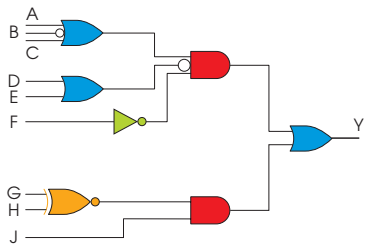
A	B	C	I	Y
0	0	0	I	0
0	0	1	I	0
0	1	0	I	0
0	1	1	I	1
1	0	0	I	0
1	0	1	I	1
1	1	0	I	0
1	1	1	I	1

44.

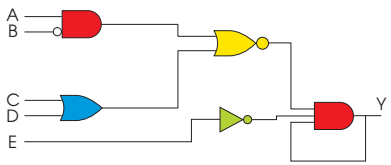
A	B	C		Y
0	0	0		0
0	0	1		0
0	1	0		0
0	1	1		1
1	0	0		0
1	0	1		1
1	1	0		1
1	1	1		0

Create a PLC ladder diagram for the relay logic diagrams in the following problems.

45.

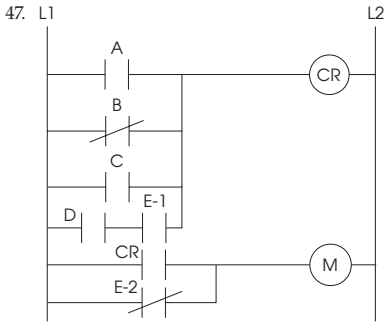


46.

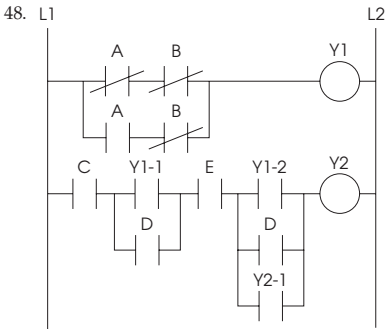


Create a relay logic diagram for the ladder diagrams in the following problems.

47.



48.



Create a PLC ladder diagram for the Boolean expressions in the following problems.

49. Y = (A′ + B) + (A′ + B + C′)′
50. Y = A′B′C + AB′ + A′BC′
51. Y = B′(A + C) + C(A′ + B) + AC
52. Y = (A′B + AB′) × (A′B′ + AB) + ABC