6. BOOLEAN LOGIC DESIGN

Topics:

- Boolean algebra
- Converting between Boolean algebra and logic gates and ladder logic
- Logic examples

Objectives:

- Be able to simplify designs with Boolean algebra

6.1 INTRODUCTION

The process of converting control objectives into a ladder logic program requires structured thought. Boolean algebra provides the tools needed to analyze and design these systems.

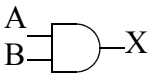
6.2 BOOLEAN ALGEBRA

Boolean algebra was developed in the 1800's by James Bool, an Irish mathematician. It was found to be extremely useful for designing digital circuits, and it is still heavily used by electrical engineers and computer scientists. The techniques can model a logical system with a single equation. The equation can then be simplified and/or manipulated into new forms. The same techniques developed for circuit designers adapt very well to ladder logic programming.

Boolean equations consist of variables and operations and look very similar to normal algebraic equations. The three basic operators are AND, OR and NOT; more complex operators include exclusive or (EOR), not and (NAND), not or (NOR). Small truth tables for these functions are shown in Figure 62. Each operator is shown in a simple equation with the variables A and B being used to calculate a value for X. Truth tables are a simple (but bulky) method for showing all of the possible combinations that will turn an output on or off.

Note: By convention a false state is also called off or 0 (zero). A true state is also called on or 1.

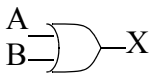
AND



$X = A \cdot B$

A	B	X
0	0	0
0	1	0
1	0	0
1	1	1

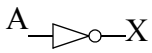
OR



$X = A + B$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	1

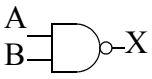
NOT



$X = \overline{A}$

A	X
0	1
1	0

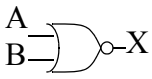
NAND



$X = \overline{A \cdot B}$

A	B	X
0	0	1
0	1	1
1	0	1
1	1	0

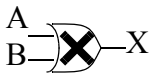
NOR



$X = \overline{A + B}$

A	B	X
0	0	1
0	1	0
1	0	0
1	1	0

EOR



$X = A \oplus B$

A	B	X
0	0	0
0	1	1
1	0	1
1	1	0

Note: The symbols used in these equations, such as + for OR are not universal standards and some authors will use different notations.

Note: The EOR function is available in gate form, but it is more often converted to its equivalent, as shown below.

$$X = A \oplus B = A \cdot \overline{B} + \overline{A} \cdot B$$

Figure 62 Boolean Operations with Truth Tables and Gates

In a Boolean equation the operators will be put in a more complex form as shown in Figure 63. The variable for these equations can only have a value of 0 for false, or 1 for true. The solution of the equation follows rules similar to normal algebra. Parts of the equation inside parenthesis are to be solved first. Operations are to be done in the sequence NOT, AND, OR. In the example the NOT function for C is done first, but the NOT over the first set of parentheses must wait until a single value is available. When there is a choice the AND operations are done before the OR operations. For the given

set of variable values the result of the calculation is false.

given

$$X = \overline{(A + B \cdot C)} + A \cdot (B + \overline{C})$$

assuming A=1, B=0, C=1

$$X = \overline{(1 + 0 \cdot 1)} + 1 \cdot (0 + \overline{1})$$

$$X = \overline{(1 + 0)} + 1 \cdot (0 + 0)$$

$$X = \overline{(1)} + 1 \cdot (0)$$

$$X = 0 + 0$$

$$X = 0$$

Figure 63 A Boolean Equation

The equations can be manipulated using the basic axioms of Boolean shown in Figure 64. A few of the axioms (associative, distributive, commutative) behave like normal algebra, but the other axioms have subtle differences that must not be ignored.

Idempotent

$A + A = A$ $\qquad$ $A \cdot A = A$

Associative

$(A + B) + C = A + (B + C)$ $\qquad$ $(A \cdot B) \cdot C = A \cdot (B \cdot C)$

Commutative

$A + B = B + A$ $\qquad$ $A \cdot B = B \cdot A$

Distributive

$A + (B \cdot C) = (A + B) \cdot (A + C)$ $\qquad$ $A \cdot (B + C) = (A \cdot B) + (A \cdot C)$

Identity

$A + 0 = A$ $\qquad$ $A + 1 = 1$

$A \cdot 0 = 0$ $\qquad$ $A \cdot 1 = A$

Complement

$A + \overline{A} = 1$ $\qquad$ $\overline{(\overline{A})} = A$

$A \cdot \overline{A} = 0$ $\qquad$ $\overline{1} = 0$

DeMorgan's

$\overline{(A + B)} = \overline{A} \cdot \overline{B}$ $\qquad$ $\overline{(A \cdot B)} = \overline{A} + \overline{B}$

Duality

interchange AND and OR operators, as well as all Universal, and Null sets. The resulting equation is equivalent to the original.

Figure 64 The Basic Axioms of Boolean Algebra

An example of equation manipulation is shown in Figure 65. The distributive axiom is applied to get equation (1). The idempotent axiom is used to get equation (2). Equation (3) is obtained by using the distributive axiom to move C outside the parentheses, but the identity axiom is used to deal with the lone C. The identity axiom is then used to simplify the contents of the parentheses to get equation (4). Finally the Identity axiom is used to get the final, simplified equation. Notice that using Boolean algebra has shown that 3 of the variables are entirely unneeded.

$$A = \overline{B} \cdot (C \cdot (\overline{D} + E + C) + \overline{F} \cdot C)$$

$$A = \overline{B} \cdot (\overline{D} \cdot C + E \cdot C + C \cdot C + \overline{F} \cdot C) \quad (1)$$

$$A = \overline{B} \cdot (\overline{D} \cdot C + E \cdot C + C + \overline{F} \cdot C) \quad (2)$$

$$A = \overline{B} \cdot C \cdot (\overline{D} + E + 1 + \overline{F}) \quad (3)$$

$$A = \overline{B} \cdot C \cdot (1) \quad (4)$$

$$A = \overline{B} \cdot C \quad (5)$$

Figure 65 Simplification of a Boolean Equation

Note: When simplifying Boolean algebra, OR operators have a lower priority, so they should be manipulated first. NOT operators have the highest priority, so they should be simplified last. Consider the example from before.

$$X = \overline{(A + B \cdot C)} + A \cdot (B + \overline{C})$$

$$X = \overline{(A) + (B \cdot C)} + A \cdot (B + \overline{C})$$ ← The higher priority operators are put in parentheses

$$X = \overline{(A)} \cdot \overline{(B \cdot C)} + A \cdot (B + \overline{C})$$ ← DeMorgan's theorem is applied

$$X = \overline{A} \cdot (\overline{B} + \overline{C}) + A \cdot (B + \overline{C})$$ ← DeMorgan's theorem is applied again

$$X = \overline{A} \cdot \overline{B} + \overline{A} \cdot \overline{C} + A \cdot B + A \cdot \overline{C}$$ ← The equation is expanded

$$X = \overline{A} \cdot \overline{B} + (\overline{A} \cdot \overline{C} + A \cdot \overline{C}) + A \cdot B$$ ← Terms with common terms are collected, here it is only NOT C

$$X = \overline{A} \cdot \overline{B} + \overline{C} \cdot (\overline{A} + A) + A \cdot B$$ ← The redundant term is eliminated

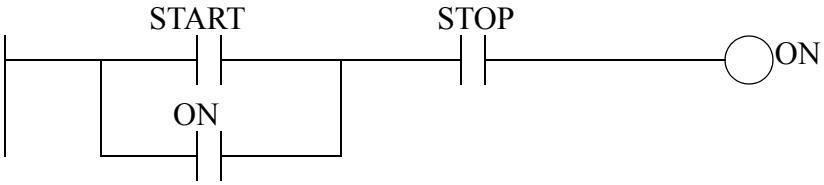
$$X = \overline{A} \cdot \overline{B} + \overline{C} + A \cdot B$$ ← A Boolean axiom is applied to simplify the equation further

6.3 LOGIC DESIGN

Design ideas can be converted to Boolean equations directly, or with other techniques discussed later. The Boolean equation form can then be simplified or rearranges, and then converted into ladder

logic, or a circuit.

Aside: The logic for a seal-in circuit can be analyzed using a Boolean equation as shown below. Recall that the START is NO and the STOP is NC.



$$ON' = (START + ON) \cdot STOP$$

ON	STOP	START	ON'	
0	0	0	0	stop pushed, not active
0	0	1	0	stop pushed, not active
0	1	0	0	not active
0	1	1	1	start pushed, becomes active
1	0	0	0	stop pushed, not active
1	0	1	0	stop pushed, not active
1	1	0	1	active, start no longer pushed
1	1	1	1	becomes active and start pushed

If we can describe how a controller should work in words, we can often convert it directly to a Boolean equation, as shown in Figure 66. In the example a process description is given first. In actual applications this is obtained by talking to the designer of the mechanical part of the system. In many cases the system does not exist yet, making this a challenging task. The next step is to determine how the controller should work. In this case it is written out in a sentence first, and then converted to a Boolean expression. The Boolean expression may then be converted to a desired form. The first equation contains an EOR, which is not available in ladder logic, so the next line converts this to an equivalent expression (2) using ANDs, ORs and NOTs. The ladder logic developed is for the second equation. In the conversion the terms that are ANDed are in series. The terms that are ORed are in parallel branches, and terms that are NOTed use normally closed contacts. The last equation (3) is fully expanded and ladder logic for it is shown in Figure 67. This illustrates the same logical control function can be achieved with different, yet equivalent, ladder logic.

Process Description:

A heating oven with two bays can heat one ingot in each bay. When the heater is on it provides enough heat for two ingots. But, if only one ingot is present the oven may become too hot, so a fan is used to cool the oven when it passes a set temperature.

Control Description:

If the temperature is too high and there is an ingot in only one bay then turn on fan.

Define Inputs and Outputs:

B1 = bay 1 ingot present
B2 = bay 2 ingot present
F = fan
T = temperature overheat sensor

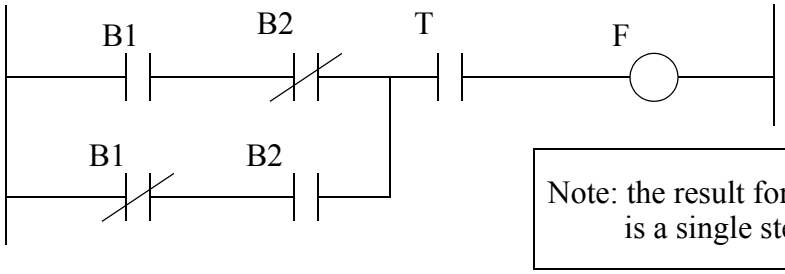
Boolean Equation:

$$F = T \cdot (B_1 \oplus B_2)$$

$$F = T \cdot (B_1 \cdot \overline{B_2} + \overline{B_1} \cdot B_2) \qquad (2)$$

$$F = B_1 \cdot \overline{B_2} \cdot T + \overline{B_1} \cdot B_2 \cdot T \qquad (3)$$

Ladder Logic for Equation (2):



Note: the result for conditional logic is a single step in the ladder

Warning: in spoken and written english OR and EOR are often not clearly defined. Consider the traffic directions "Go to main street then turn left or right." Does this *or* mean that you can drive either way, or that the person isn't sure which way to go? Consider the expression "The cars are red or blue.", Does this mean that the cars can be either red or blue, or all of the cars are red, or all of the cars are blue. A good literal way to describe this condition is "one or the other, but not both".

Figure 66 Boolean Algebra Based Design of Ladder Logic

Ladder Logic for Equation (3):

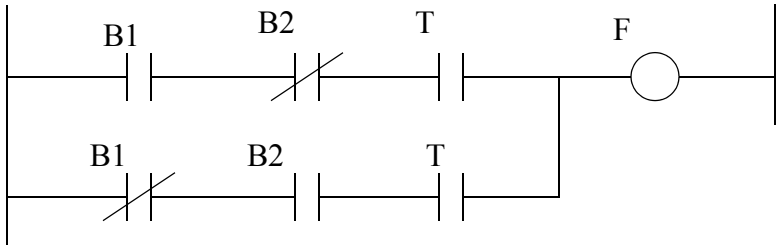
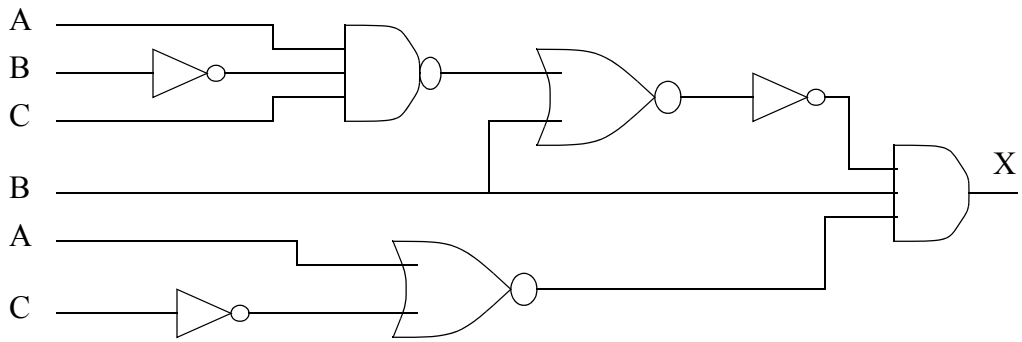


Figure 67 Alternate Ladder Logic

Boolean algebra is often used in the design of digital circuits. Consider the example in Figure 68. In this case we are presented with a circuit that is built with inverters, nand, nor and, and gates. This figure can be converted into a boolean equation by starting at the left hand side and working right. Gates on the left hand side are *solved* first, so they are put inside parentheses to indicate priority. Inverters are represented by putting a NOT operator on a variable in the equation. This circuit can't be directly converted to ladder logic because there are no equivalents to NAND and NOR gates. After the circuit is converted to a Boolean equation it is simplified, and then converted back into a (much simpler) circuit diagram and ladder logic.



The circuit is converted to a Boolean equation and simplified. The most nested terms in the equation are on the left hand side of the diagram.

$$X = \overline{\overline{\overline{(A \cdot \overline{B} \cdot C)} + B} \cdot B \cdot \overline{(A + \overline{C})}}$$

$$X = (\overline{A} + B + \overline{C} + B) \cdot B \cdot (\overline{A} \cdot C)$$

$$X = \overline{A} \cdot B \cdot \overline{A} \cdot C + B \cdot B \cdot \overline{A} \cdot C + \overline{C} \cdot B \cdot \overline{A} \cdot C + B \cdot B \cdot \overline{A} \cdot C$$

$$X = B \cdot \overline{A} \cdot C + B \cdot \overline{A} \cdot C + 0 + B \cdot \overline{A} \cdot C$$

$$X = B \cdot \overline{A} \cdot C$$

This simplified equation is converted back into a circuit and equivalent ladder logic.

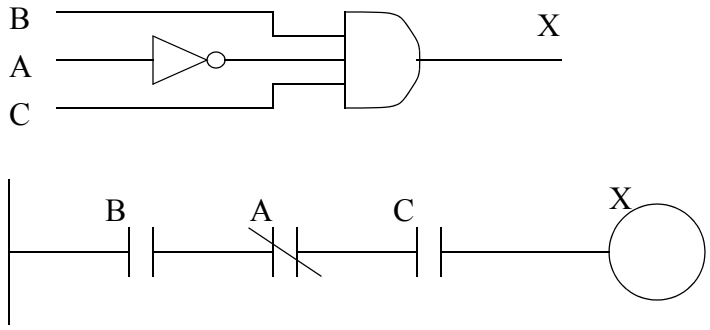


Figure 68 Reverse Engineering of a Digital Circuit

To summarize, we will obtain Boolean equations from a verbal description or existing circuit or ladder diagram. The equation can be manipulated using the axioms of Boolean algebra. after simplification the equation can be converted back into ladder logic or a circuit diagram. Ladder logic (and circuits) can behave the same even though they are in different forms. When simplifying Boolean equations that are to be implemented in ladder logic there are a few basic rules.

1. Eliminate NOTs that are for more than one variable. This normally includes replacing NAND and NOR functions with simpler ones using DeMorgan's theorem.
2. Eliminate complex functions such as EORs with their equivalent.

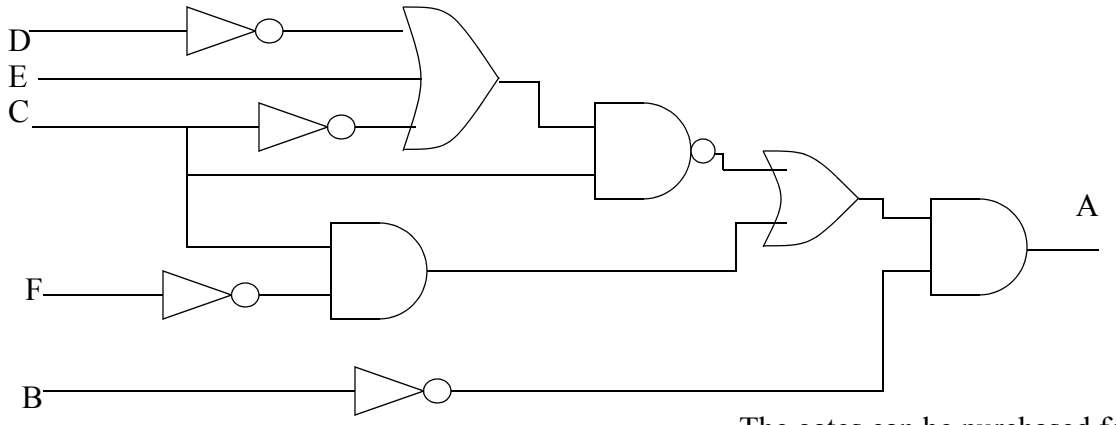
These principles are reinforced with another design that begins in Figure 69. Assume that the Boolean equation that describes the controller is already known. This equation can be converted into both a circuit diagram and ladder logic. The circuit diagram contains about two dollars worth of integrated circuits. If the design was mass produced the final cost for the entire controller would be under $50. The prototype of the controller would cost thousands of dollars. If implemented in ladder logic the cost for each controller would be approximately $500. Therefore a large number of circuit based controllers need to be produced before the break even occurs. This number is normally in the range of hundreds of units. There are some particular advantages of a PLC over digital circuits for the factory and some other applications.

- the PLC will be more rugged,
- the program can be changed easily
- less skill is needed to maintain the equipment

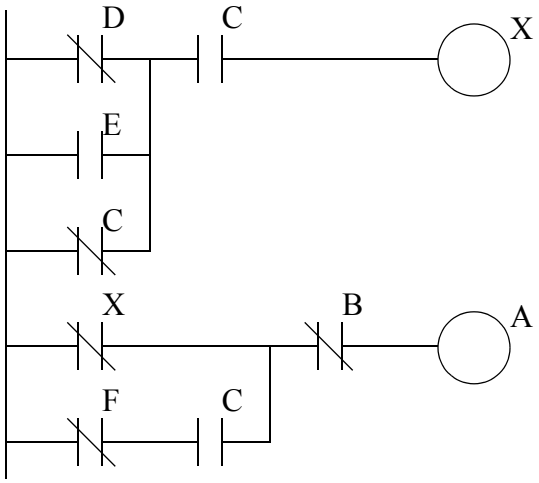
Given the controller equation;

$$A = \overline{B} \cdot (\overline{C \cdot (\overline{D} + E + \overline{C})} + \overline{F} \cdot C)$$

The circuit is given below, and equivalent ladder logic is shown.



The gates can be purchased for about $0.25 each in bulk. Inputs and outputs are typically 5V



An inexpensive PLC is worth at least a few hundred dollars

Consider the cost trade-off!

Figure 69 A Boolean Equation and Derived Circuit and Ladder Logic

The initial equation is not the simplest. It is possible to simplify the equation to the form seen in Figure 69. If you are a visual learner you may want to notice that some simplifications are obvious with ladder logic - consider the *C* on both branches of the ladder logic in Figure 70.

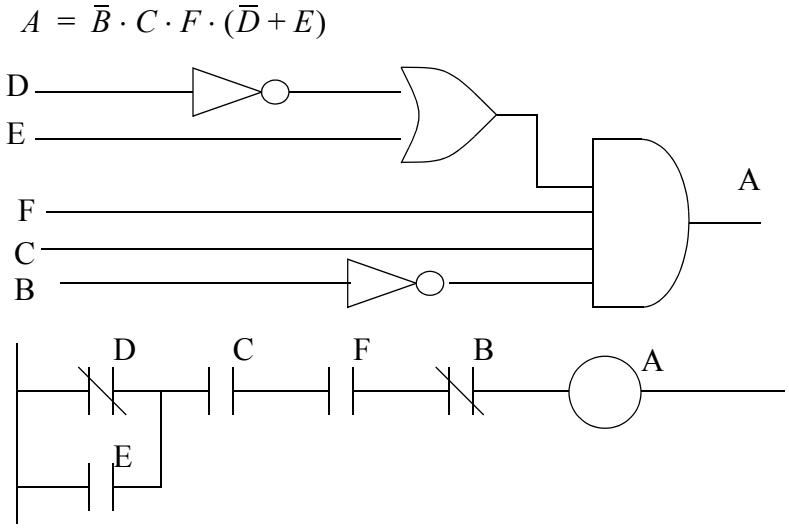


Figure 70 The Simplified Form of the Example

The equation can also be manipulated to other forms that are more routine but less efficient as shown in Figure 71. The equation shown is in disjunctive normal form - in simpler words this is ANDed terms ORed together. This is also an example of a canonical form - in simpler terms this means a standard form. This form is more important for digital logic, but it can also make some PLC programming issues easier. For example, when an equation is simplified, it may not look like the original design intention, and therefore becomes harder to rework without starting from the beginning.

$$A = (\overline{B} \cdot C \cdot \overline{D}) + (\overline{B} \cdot C \cdot E) + (\overline{B} \cdot C \cdot \overline{F})$$

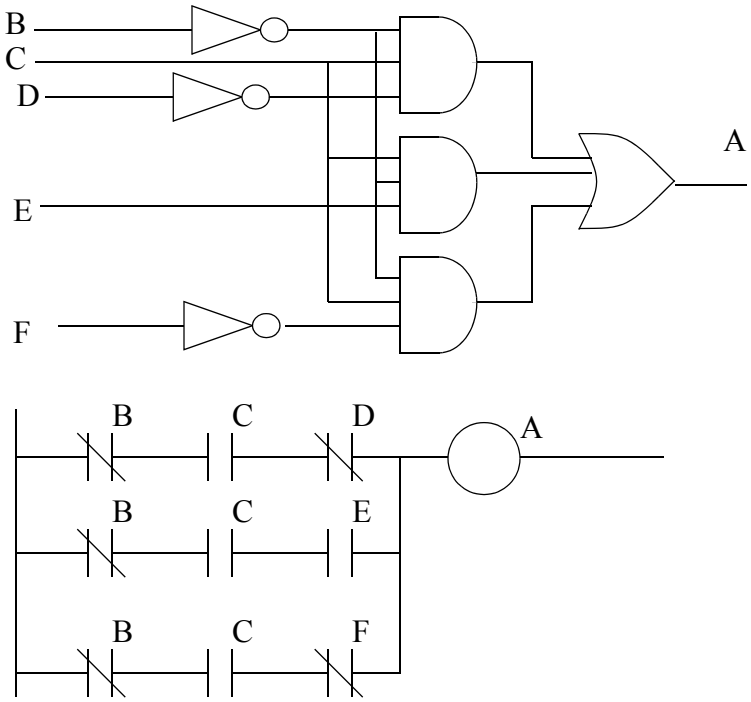


Figure 71 A Canonical Logic Form

6.3.1 Boolean Algebra Techniques

There are some common Boolean algebra techniques that are used when simplifying equations. Recognizing these forms are important to simplifying Boolean Algebra with ease. These are itemized, with proofs in Figure 72.

$A + C\overline{A} = A + C$ proof:

$A + C\overline{A}$
$(A + C)(A + \overline{A})$
$(A + C)(1)$
$A + C$

$AB + A = A$ proof:

$AB + A$
$AB + A1$
$A(B + 1)$
$A(1)$
A

$\overline{A + B + C} = \overline{A}\overline{B}\overline{C}$ proof:

$\overline{A + B + C}$
$\overline{(A + B) + C}$
$\overline{(A + B)}\overline{C}$
$(\overline{A}\overline{B})\overline{C}$
$\overline{A}\overline{B}\overline{C}$

Figure 72 Common Boolean Algebra Techniques

6.4 COMMON LOGIC FORMS

Knowing a simple set of logic forms will support a designer when categorizing control problems. The following forms are provided to be used directly, or provide ideas when designing.

6.4.1 Complex Gate Forms

In total there are 16 different possible types of 2-input logic gates. The simplest are AND and OR, the other gates we will refer to as *complex* to differentiate. The three popular complex gates that have been discussed before are NAND, NOR and EOR. All of these can be reduced to simpler forms with only ANDs and ORs that are suitable for ladder logic, as shown in Figure 73.

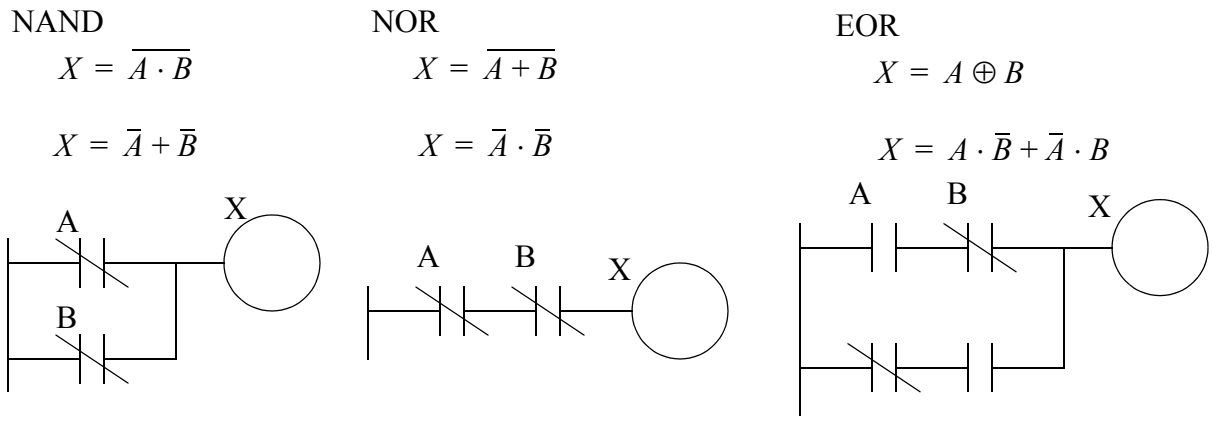
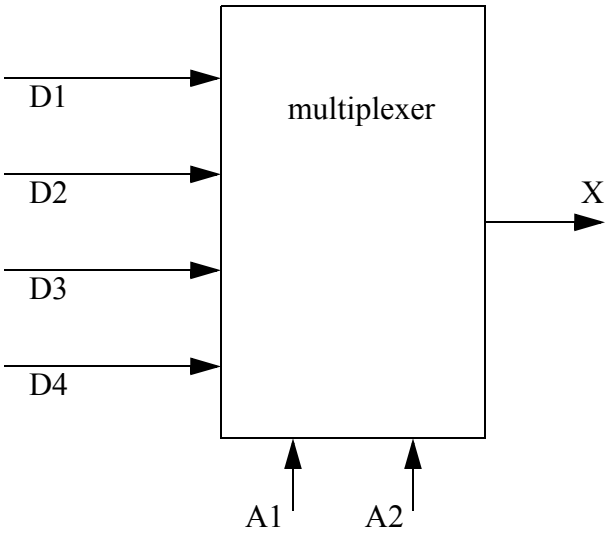


Figure 73 Conversion of Complex Logic Functions

6.4.2 Multiplexers

Multiplexers allow multiple devices to be connected to a single device. These are very popular for telephone systems. A telephone *switch* is used to determine which telephone will be connected to a limited number of lines to other telephone switches. This allows telephone calls to be made to somebody far away without a dedicated wire to the other telephone. In older telephone switch boards, operators physically connected wires by plugging them in. In modern computerized telephone switches the same thing is done, but to digital voice signals.

In Figure 74 a multiplexer is shown that will take one of four inputs bits D1, D2, D3 or D4 and make it the output X, depending upon the values of the address bits, A1 and A2.



A1	A2	X
0	0	X=D1
0	1	X=D2
1	0	X=D3
1	1	X=D4

Figure 74 A Multiplexer

Ladder logic form the multiplexer can be seen in Figure 75.

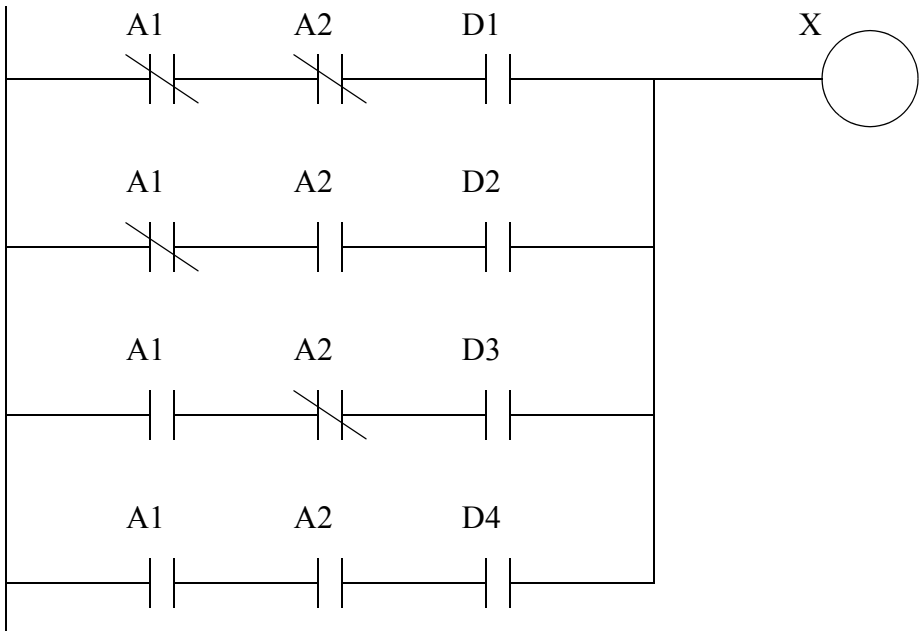


Figure 75 A Multiplexer in Ladder Logic

6.5 SIMPLE DESIGN CASES

The following cases are presented to illustrate various combinatorial logic problems, and possible solutions. It is recommended that you try to satisfy the description before looking at the solution.

6.5.1 Basic Logic Functions

Problem: Develop a program that will cause output D to go true when switch A and switch B are closed or when switch C is closed.

Solution:

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

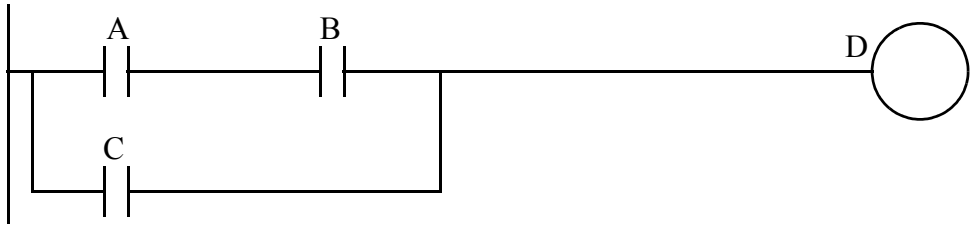


Figure 76 Sample Solution for Logic Case Study A

Problem: Develop a program that will cause output D to be on when push button A is on, or either B or C are on.

Solution:

$$D = A + (B \oplus C)$$

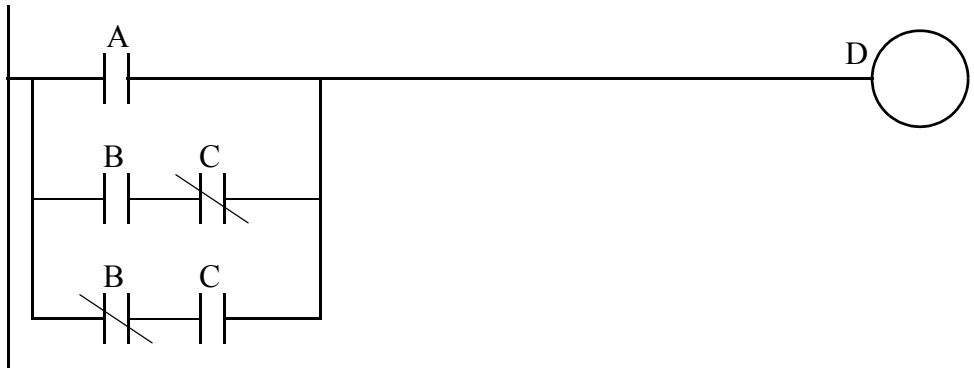


Figure 77 Sample Solution for Logic Case Study B

6.5.2 Car Safety System

Problem: Develop Ladder Logic for a car door/seat belt safety system. When the car door is open, and the seatbelt is not done up, the ignition power must not be applied. If all is safe then the key will start the engine.

Solution:

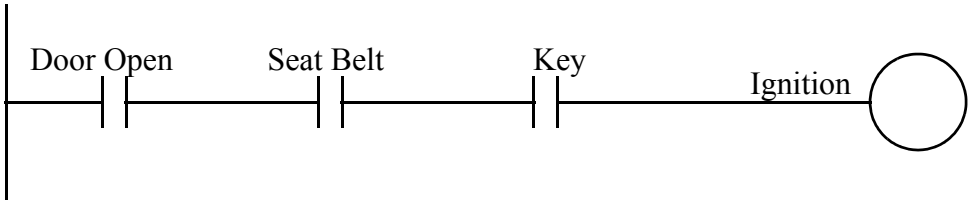


Figure 78 Solution to Car Safety System Case

6.5.3 Motor Forward/Reverse

Problem: Design a motor controller that has a forward and a reverse button. The motor forward and reverse outputs will only be on when one of the buttons is pushed. When both buttons are pushed the motor will not work.

Solution:

$$F = BF \cdot \overline{BR}$$

$$R = \overline{BF} \cdot BR$$

where,

F = motor forward
R = motor reverse
BF = forward button
BR = reverse button

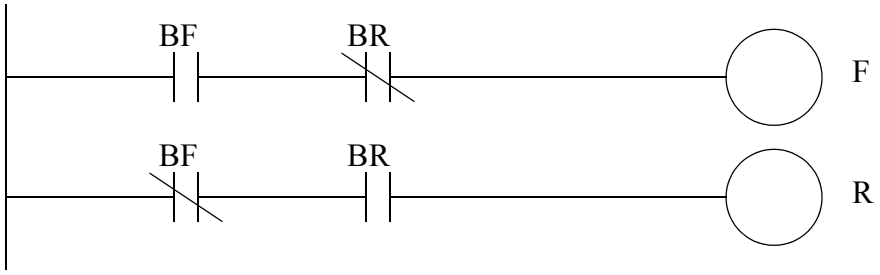


Figure 79 Motor Forward, Reverse Case Study

6.5.4 A Burglar Alarm

Consider the design of a burglar alarm for a house. When activated an alarm and lights will be activated to encourage the unwanted guest to leave. This alarm be activated if an unauthorized intruder is detected by window sensor and a motion detector. The window sensor is effectively a loop of wire that is a piece of thin metal foil that encircles the window. If the window is broken, the foil breaks breaking the conductor. This behaves like a normally closed switch. The motion sensor is designed so that when a person is detected the output will go on. As with any alarm an activate/deactivate switch is also needed. The basic operation of the alarm system, and the inputs and outputs of the controller are

itemized in Figure 80.

The inputs and outputs are chosen to be;

A = Alarm and lights switch (1 = on)
W = Window/Door sensor (1 = OK)
M = Motion Sensor (0 = OK)
S = Alarm Active switch (1 = on)

The basic operation of the alarm can be described with rules.

1. If alarm is on, check sensors.
2. If window/door sensor is broken (turns off), sound alarm and turn on lights

Note: As the engineer, it is your responsibility to define these items before starting the work. If you do not do this first you are guaranteed to produce a poor design. It is important to develop a good list of inputs and outputs, and give them simple names so that they are easy to refer to. Most companies will use wire numbering schemes on their diagrams.

Figure 80 Controller Requirements List for Alarm

The next step is to define the controller equation. In this case the controller has 3 different inputs, and a single output, so a truth table is a reasonable approach to formalizing the system. A Boolean equation can then be written using the truth table in Figure 81. Of the eight possible combinations of alarm inputs, only three lead to alarm conditions.

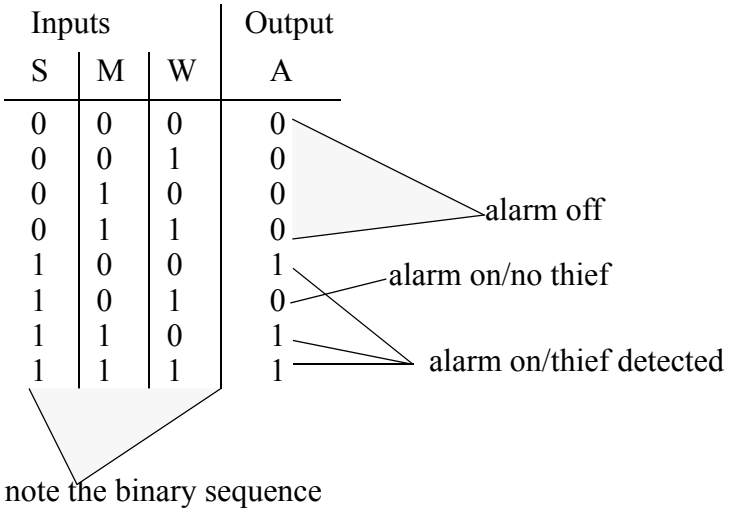


Figure 81 Truth Table for the Alarm

The Boolean equation in Figure 82 is written by examining the truth table in Figure 81. There are three possible alarm conditions that can be represented by the conditions of all three inputs. For example take the last line in the truth table where when all three inputs are on the alarm should be one. This leads to the last term in the equation. The other two terms are developed the same way. After the equation has been written, it is simplified.

$$A = (S \cdot \overline{M} \cdot \overline{W}) + (S \cdot M \cdot \overline{W}) + (S \cdot M \cdot W)$$

$$\therefore A = S \cdot (\overline{M} \cdot \overline{W} + M \cdot \overline{W} + M \cdot W)$$

$$\therefore A = S \cdot ((\overline{M} \cdot \overline{W} + M \cdot \overline{W}) + (M \cdot \overline{W} + M \cdot W))$$

$$\therefore A = (S \cdot \overline{W}) + (S \cdot M) = S \cdot (\overline{W} + M)$$

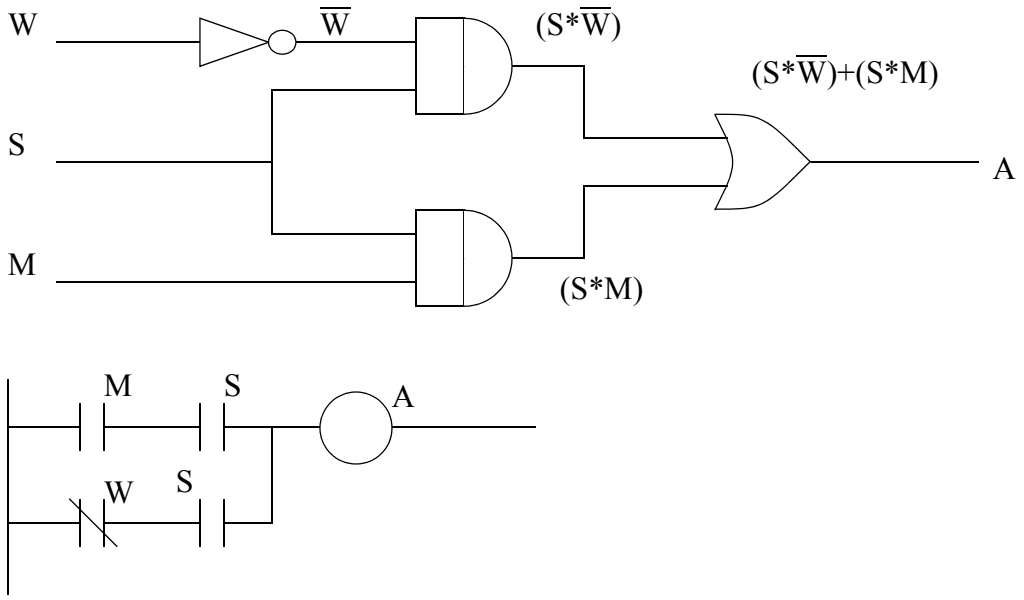


Figure 82 A Boolean Equation and Implementation for the Alarm

The equation and circuits shown in Figure can also be further simplified, as shown in Figure 83.

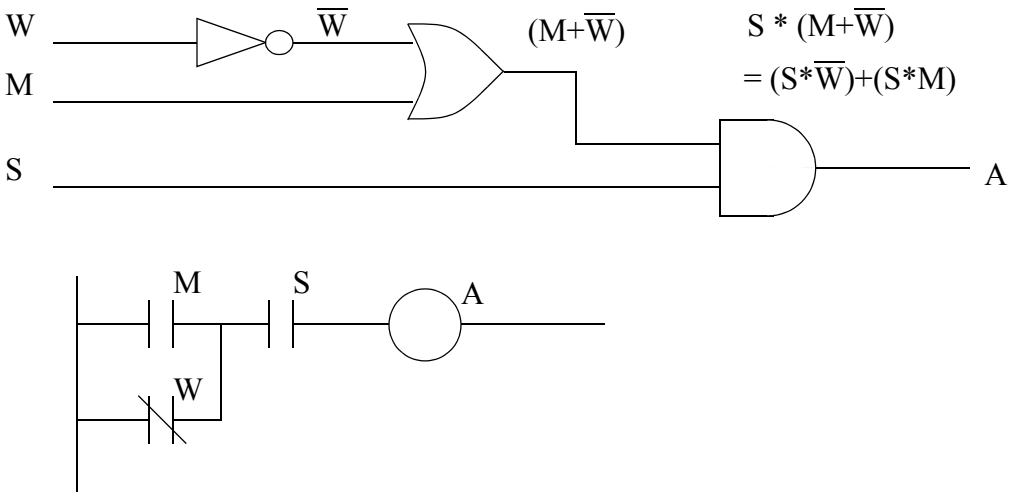


Figure 83 The Simplest Circuit and Ladder Diagram

Aside: The alarm could also be implemented in programming languages. The program below is for a Basic Stamp II chip. (www.parallaxinc.com)

```
w = 1; s = 2; m = 3; a = 4
input m; input w; input s
output a
loop:
if (in2 = 1) and (in1 = 0 or in3 = 1) then on
low a; goto loop ‘alarm off
on:
high a; goto loop ‘alarm on
```

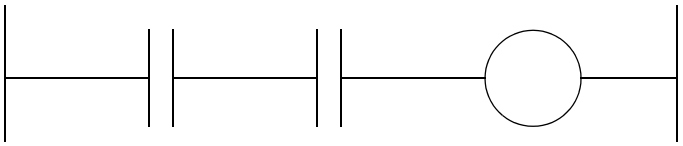
Figure 84 Alarm Implementation Using A High Level Programming Language

6.6 SUMMARY

- Logic can be represented with Boolean equations.
- Boolean equations can be converted to (and from) ladder logic or digital circuits.
- Boolean equations can be simplified.
- Different controllers can behave the same way.
- Common logic forms exist and can be used to understand logic.
- Truth tables can represent all of the possible state of a system.

6.7 PRACTICE PROBLEMS

1. Is the ladder logic in the figure below for an AND or an OR gate?



2. Draw a ladder diagram that will cause output D to go true when switch A and switch B are closed or when switch C is closed.

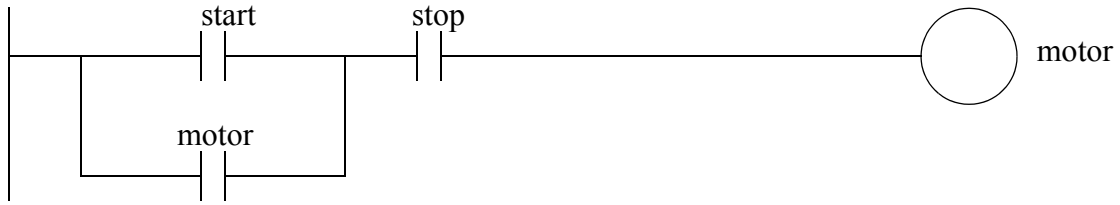
3. Draw a ladder diagram that will cause output D to be on when push button A is on, or either B or C are on.

4. Design ladder logic for a car that considers the variables below to control the motor *M*. Also add a second output that uses any outputs not used for motor control.

   - doors opened/closed (D)
   - keys in ignition (K)
   - motor running (M)
   - transmission in park (P)
   - ignition start (I)

5. a) Explain why a stop button must be normally closed and a start button must be normally open.

b) Consider a case where an input to a PLC is a normally closed stop button. The contact used in the ladder logic is normally open, as shown below. Why are they both not the same? (i.e., NC or NO)



6. Make a simple ladder logic program that will turn on the outputs with the binary patterns when the corresponding buttons are pushed.

OUTPUTS								INPUTS
H	G	F	E	D	C	B	A	
1	1	0	1	0	1	0	1	Input X on
1	0	1	0	0	0	0	1	Input Y on
1	0	0	1	0	1	1	1	Input Z on

7. Convert the following Boolean equation to the simplest possible ladder logic.

$$X = A \cdot \overline{(\overline{A} + \overline{A} \cdot B)}$$

8. Simplify the following boolean equations.

a) $A(B + AB)$

b) $\overline{A(\overline{B + AB})}$

c) $\overline{A}(B + AB)$

d) $\overline{\overline{A}(\overline{B + AB})}$

9. Simplify the following Boolean equations,

a) $\overline{(A + B)} \cdot \overline{(A + \overline{B})}$

b) $ABCD + \overline{A}BCD + ABC\overline{D} + AB\overline{C}\overline{D}$

10. Simplify the Boolean expression below.

$$((A \cdot \overline{B}) + \overline{(\overline{B} + A)}) \cdot C + (\overline{B} \cdot C + B \cdot C)$$

11. Given the Boolean expression a) draw a digital circuit and b) a ladder diagram (do not simplify), c) simplify the expression.

$$X = A \cdot \overline{B} \cdot C + \overline{(C + B)}$$

12. Simplify the following Boolean equation and write corresponding ladder logic.

$$Y = \overline{\overline{(AB\overline{C}D + AB\overline{C}\overline{D} + \overline{A}BCD + \overline{A}\overline{B}CD)} + D}$$

13. For the following Boolean equation,

$$X = A + B(A + C\overline{B} + D\overline{A}C) + ABCD$$

a) Write out the logic for the unsimplified equation.
b) Simplify the equation.
c) Write out the ladder logic for the simplified equation.

14. a) Write a Boolean equation for the following truth table. (Hint: do this by writing an expression for each

line with a true output, and then ORing them together.)

A	B	C	D	Result
0	0	0	0	1
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	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	1

b) Write the results in a) in a Boolean equation.
c) Simplify the Boolean equation in b)

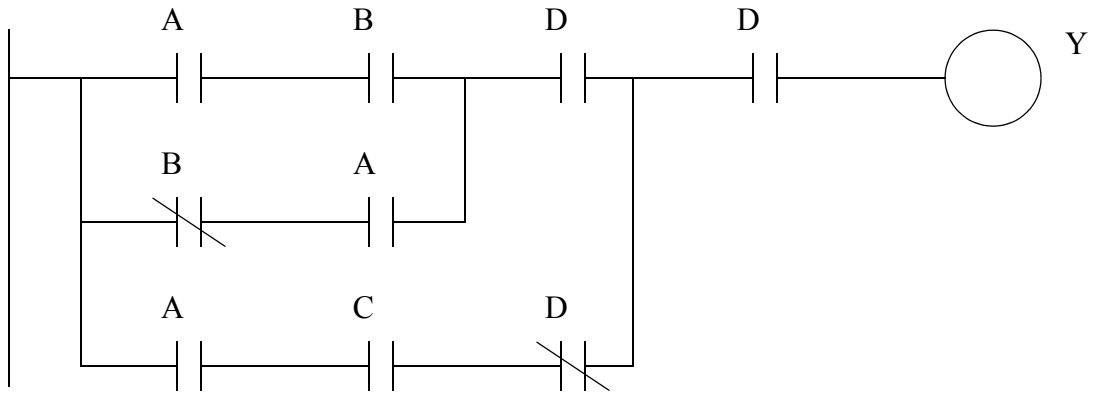
15. Simplify the following Boolean equation, and create the simplest ladder logic.

$$Y = \overline{\overline{C}\left(\overline{A} + \left(\overline{\overline{A} + (\overline{B}\,\overline{C}(A + \overline{B\overline{C}}))}\right)\right)}$$

16. Simplify the following boolean equation with Boolean algebra and write the corresponding ladder logic.

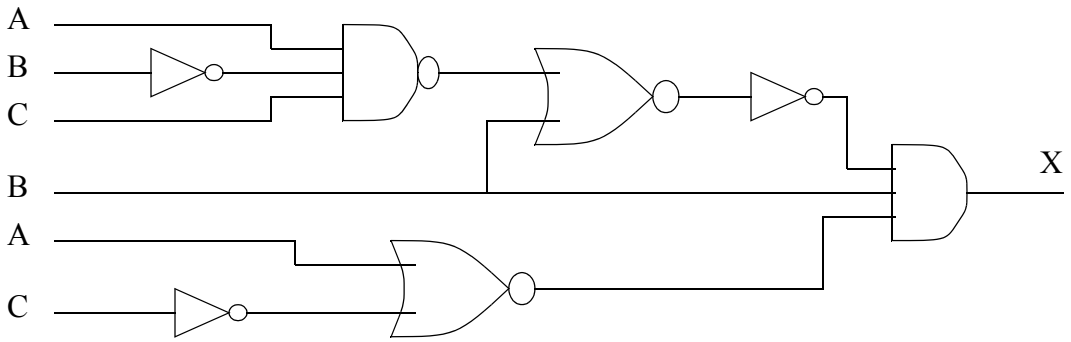
$$X = \overline{(A + B \cdot \overline{A}) + \overline{(C + D + E\overline{C})}}$$

17. Convert the following ladder logic to a Boolean equation. Then simplify it, and convert it back to simpler ladder logic.



18. a) Develop the Boolean expression for the circuit below.
b) Simplify the Boolean expression.

c) Draw a simpler circuit for the equation in b).



19. Given a system that is described with the following equation,

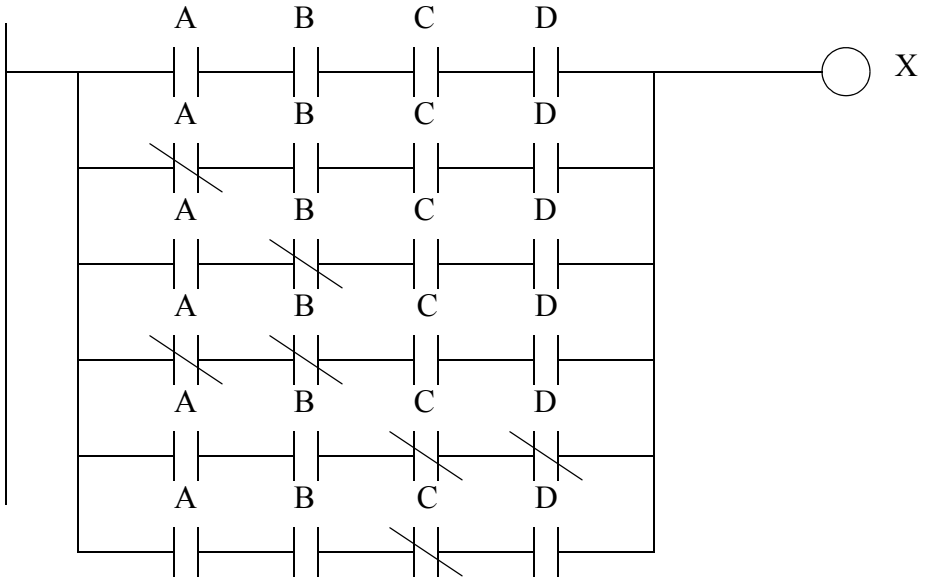
$$X = A + (B \cdot (\overline{A} + C) + C) + A \cdot B \cdot (\overline{D} + \overline{E})$$

a) Simplify the equation using Boolean Algebra.
b) Implement the original and then the simplified equation with a digital circuit.
c) Implement the original and then the simplified equation in ladder logic.

20. Simplify the following and implement the original and simplified equations with gates and ladder logic.

$$A + (\overline{B} + \overline{C} + \overline{D}) \cdot (B + \overline{C}) + A \cdot B \cdot (\overline{C} + \overline{D})$$

21. Convert the following ladder logic to a Boolean equation. Simplify the equation and convert it back to ladder logic.



22. Use Boolean equations to develop simplified ladder logic for the following truth table where A, B, C and D

are inputs, and X and Y are outputs.

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

6.8 ASSIGNMENT PROBLEMS

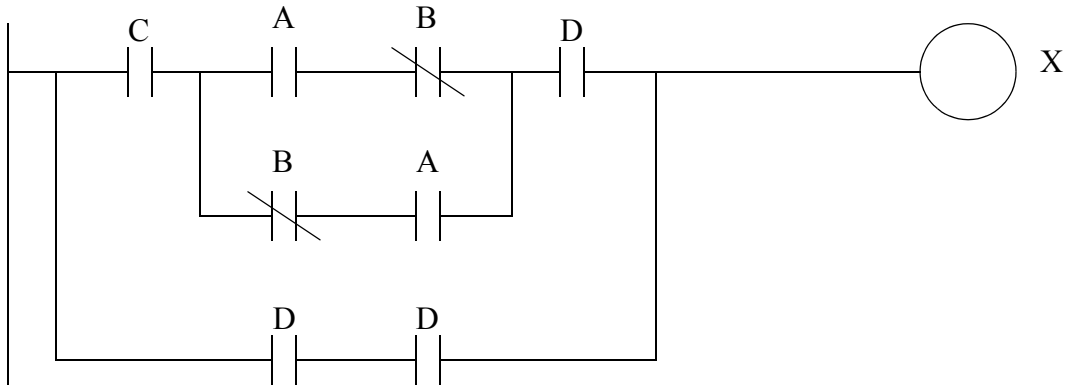
1. Simplify the following Boolean equation and implement it in ladder logic.

$$X = A + BA + B\overline{C} + \overline{D + C}$$

2. Simplify the following Boolean equation and write a ladder logic program to implement it.

$$X = (A\overline{B}C + \overline{A}BC + A\overline{B}\overline{C} + AB\overline{C} + ABC)$$

3. Convert the following ladder logic to a Boolean equation. Simplify the equation using Boolean algebra, and then convert the simplified equation back to ladder logic.



4. Convert the truth table below to a Boolean equation, and then simplify it. The output is X and the inputs are A, B, C and D.

A	B	C	D	X
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	0
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	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

5. Simplify the following Boolean equation. Convert both the unsimplified and simplified equations to ladder logic.

$$X = \overline{(AB\overline{C})}(A + BC)$$

6. Convert the following ladder logic to a Boolean equation. Simplify the equation and convert it back to ladder logic.