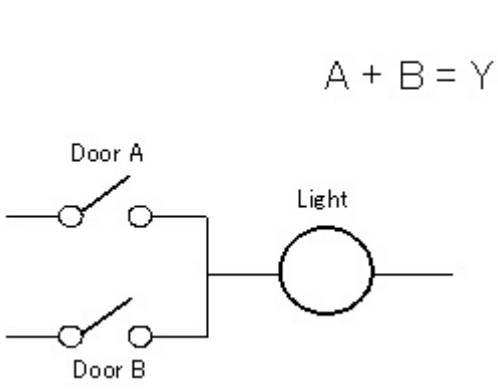
Electrical Tech Note — 542
Biosystems &Agricultural Engineering Department
Michigan State University

Boolean Math and Logic Gates

Cause and effect relationships in nature are frequently based upon simple yes or no decisions. Describing this in mathematical terms, a variable can only take on the values of zero (0) or one (1). This is the binary numbering system. There may be a series of conditions that must be met before an action will occur (AND operation). There also may be alternative sets of conditions that result in the same action (OR operation). In 1854 an English mathematician by the name of George Boole published a treatise on this subject. It was a form of mathematics that applied to the binary numbering system where the variables could only take on the values of zero and one. The basic mathematical operations are logical addition designated with a plus sign (+) and it represents an OR operation. The other basic mathematical operation is logical multiplication designated by a dot between the variables (•) and it represents an AND operation. Sometimes a dot between variables is not shown and is implied as in conventional mathematics. This mathematics became known as Boolean Algebra, and the rules of conventional algebra do not necessarily apply to this logical algebra based on binary numbering. Control systems be they mechanical or electrical, are binary in operation. Since the decisions that can be made to bring about an action can be very complex, this Boolean algebra was found to be useful to represent control systems mathematically and to determine the most simplistic set of decisions that were required to bring about an action. Modern day digital electronic devices function using AND and OR operations. Electronic logic gates constructed on micro chips form the basics of digital electronic devices. This Tech Note discusses the basic rules of Boolean algebra, electronic logic gates, and how they are related. It will be shown how to represent a Boolean expression with logic gates, and how to write a Boolean expression for a system of logic gates. This Tech Note will also discuss rules for simplification of Boolean expressions including the use of Karnaugh maps for this purpose.



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

Figure 542.1 *Two switches connected in parallel create what is known as a logic OR operation. The truth table for the OR operation is shown at the right.*

Electronic Logic Gates: The basic logic operations used in control systems are the AND where two or more devices are connected in series, and the OR where there are alternate parallel circuits. An example of an OR operation is the control circuit of Figure 542.1 where two alternative parallel paths operate a device. The example represents a warning light that is turned on if either of two doors to a cold storage are ajar. The two switches are connected in parallel to provide alternate paths. The light will be on (Y = 1) if either switch, A or B, is closed (either A = 1 "OR" B = 1). It is very useful in analyzing logic circuits to construct what is known as a "truth table". The truth table shows the conditions of the inputs (A and B) that will activate the output (Y = 1). When the output is activated (Y = 1) the Boolean expression is satisfied and the output is considered to be "true". The truth table for the OR operation is shown in Figure 542.1.

A truth table generally has the inputs to the system on the left and the output on the right separated from the inputs with a vertical line. All combinations of the inputs must be entered into the truth table. The number of input combinations is the number two raised to a power equal to the number of inputs. If there are two inputs the number of combinations is $2^2 = 4$. For three inputs the number of combinations is eight, and for four inputs the number of combinations is sixteen.

Note how the inputs are labeled in the truth table as compared to an actual system that is being analyzed. If the inputs originate at a binary "bus" such as a four line bus, note whether A represents the 1s line or the 8s line. For this Tech Note A represents the 1s line and D represents the 8s line. Be cautious of this when working with actual problems.

Rather than wiring an actual control circuit to activate the warning light of the example of Figure 542.1, all that is needed is a 0 or 1 signal from each cold storage door to know it's status. A one (1) signal means the door is ajar and the zero (0) signal means the door is closed. These signals can be supplied to an electronic "logic gate" that can perform the "OR" function. Simple logic gates that perform the OR and AND functions are shown in Figure 542.2. Standard symbols have been adopted for electronic OR and AND logic gates. The actual internal circuitry will vary depending upon the manufacturer. The logic gates shown in Figure 542.2 use two bi-polar transistors as switches. With both transistor switches open (A = 0 , and B = 0) the output is connected to ground through the resistor Ro. The output terminal is considered at the zero state (Y = 0). In the case of the OR gate, if either transistor switch closes, caused by a voltage applied to either input, current will flow from the voltage supply to ground resulting in an output of approximately +5 volts (Y = 1). In the case of the AND gate the two transistors are connected in series and both must be activated in order to complete the circuit. The transistors are connected in parallel for the OR gate.

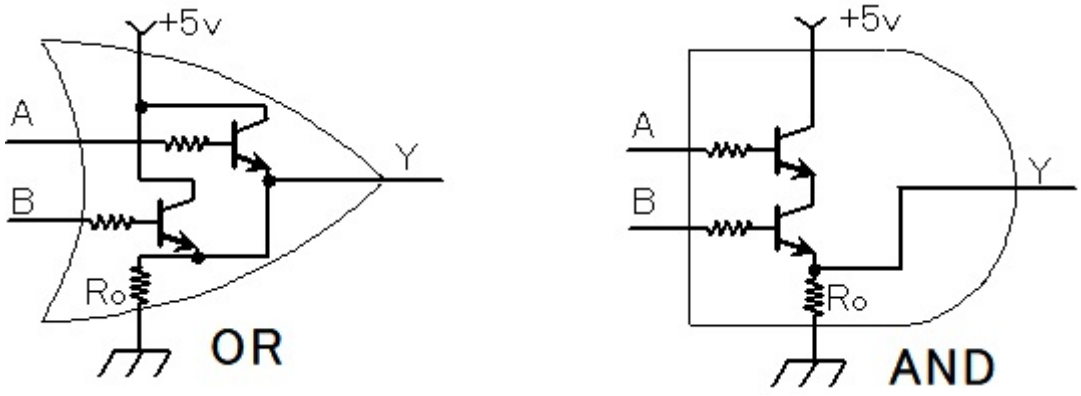
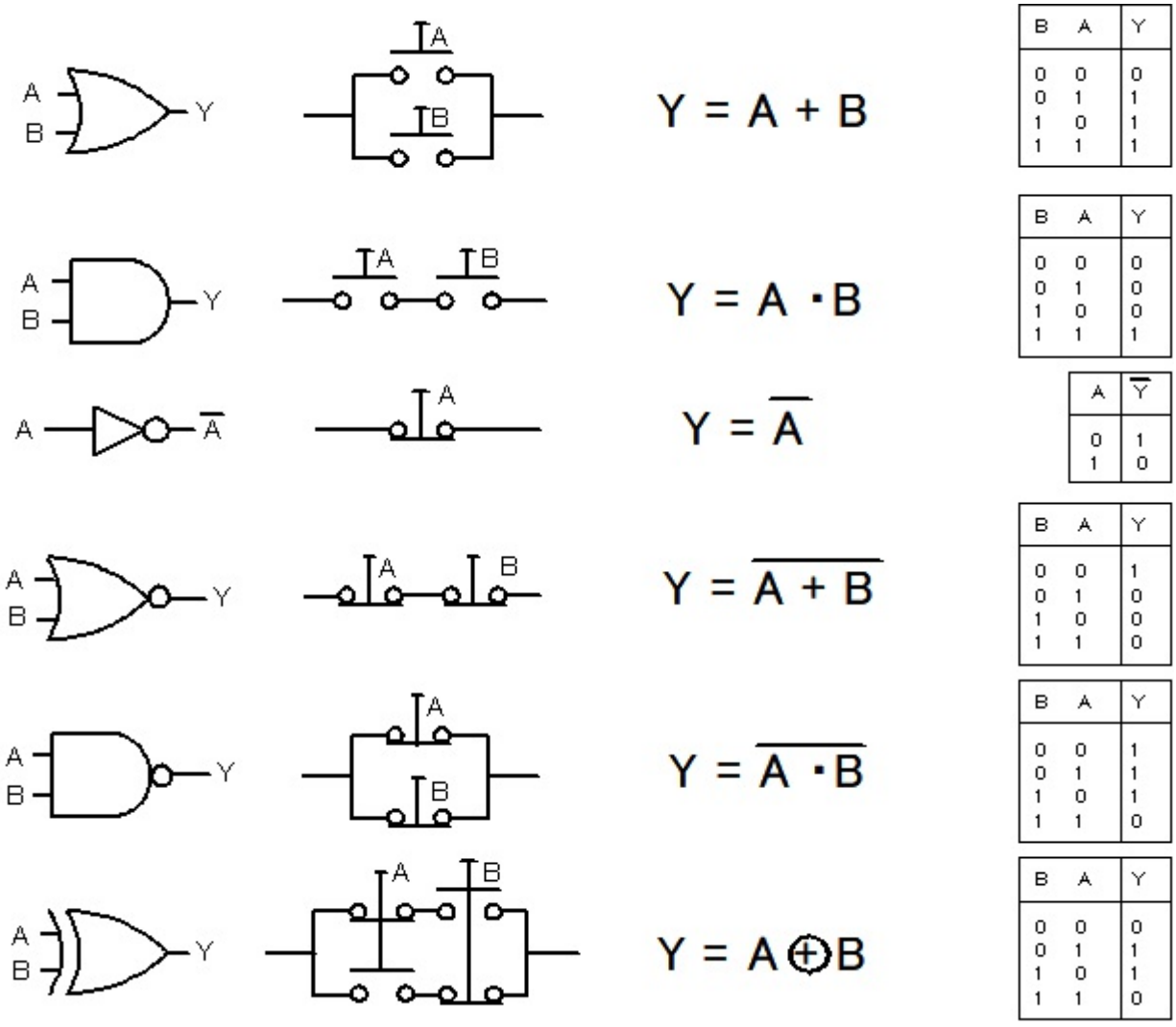


Figure 542.2 *The OR and AND operations can be performed by electronic logic gates consisting basically of two transistors either connected in parallel for the OR operation or in series for the AND operation.*

Logic Operations: In addition to the OR operation and the AND operation, there is one additional operation that is necessary and that is to change a zero to a one or a one into a zero. This is called the NOT operation and the electronic device that performs this operation is called an inverter. The symbol for an inverter (NOT) is a triangle with a small circle at the tip as shown in Figure 542.3. In a Boolean expression the NOT function is usually indicated with a line over the variable ($\overline{A}$). The value of $\overline{A}$ is the complement of A. If A = 1 then $\overline{A}$ =0. If A = 0 then $\overline{A}$ =1. Sometimes a small circle is placed at the input to a digital gate or device. This small circle represents an inverter.



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

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

A	$\overline{Y}$
0	1
1	0

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

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

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

Figure 542.3 *The symbol for a two input logic gate is shown on the left followed by an equivalent control circuit with momentary push buttons. The A and B input is the actual act of pressing on the push button. The Boolean expression is shown in the third column followed by the truth table for the gate on the right.*

The OR, AND, NOT are the three basic logic operations. There are other useful logic gates, but they can all be constructed from these three basic logic operations. Figure 542.3 shows the OR gate, AND gate, and the Inverter (NOT) along with an equivalent control circuit and the Boolean expression. The logic gates shown in Figure 542.3 are only two input gates, but gates with more inputs are common.

A very useful modification to the AND and OR gate is the NOT function connected to the output. The gates formed by this modification are the NOT AND shortened to NAND and the NOT OR shortened to NOR. The symbol for these gates has a small circle connected to the output. These gates are also shown in Figure 542.3. Finally there is sometimes a need for an OR gate that gives an output (Y = 1) when the input A or B is one (1) but not both. This means Y = 1 for A = 1 or B = 1, but the output is zero (Y = 0) for both A = 1 and B = 1. This gate is called an Exclusive OR and is shortened to XOR. This gate is also shown in Figure 542.3.

Boolean Laws, Rules, and Theorems: Boolean math provides a means of expressing a control system strategy in a form where it can be analyzed and manipulated. One important use of Boolean math is to express a logic process in a mathematical format and then see if it can be reduced to a more simplistic set of decisions. It is not uncommon for a logic process to have redundant components that can be eliminated. Basic laws, rules, and theorems that are useful for Boolean expression analysis are listed in Table 542.1. It is important to be able to apply these laws, rules, and theorems. It is useful to be able to write a Boolean expression from a set of logic gates and to draw a diagram of logic gates from a Boolean expression.

Table 542.1 Boolean math laws, rules, and theorems.

Commutative Law::	$A + B = B + A$ $A \bullet B = B \bullet A$
Associative Law:	$A + (B + C) = (A + B) + C$ $A \bullet (B \bullet C) = (A \bullet B) \bullet C$
Distributive Law:	$A (B + C) = AB + AC$ $(A + B) (C + D) = AC + AD + BC + BD$
Absorption Law:	$A + A \bullet B = A(1 + B) = A$ $A \bullet (A + B) = A$ $(A + B) \bullet (A + C) = A + B \bullet C$ $A \bullet B + B \bullet C + \overline{A} \bullet C = A \bullet B + \overline{A} \bullet C$
Rules:	$A \bullet 0 = 0$ $A \bullet 1 = A$ $A + 0 = A$ $A + 1 = 1$ $A \bullet A = A$ $A + A = A$ $A \bullet \overline{A} = 0$ $A + \overline{A} = 1$ $A + \overline{A}B = A + B$ $\overline{A} + AB = \overline{A} + B$ $\overline{\overline{A}} = A$ $\overline{A} B + C (A + \overline{B}) = \overline{A} B + C$
DeMorgan's theorem:	$\overline{A+B} = \overline{A} \bullet \overline{B}$ $\overline{A \bullet B} = \overline{A} + \overline{B}$ $\overline{A+B+C} = \overline{A} \bullet \overline{B} \bullet \overline{C}$ $\overline{A \bullet B \bullet C} = \overline{A} + \overline{B} + \overline{C}$

The **commutative law** is shown in Table 542.1 for a two input logic gate but the same principle applies to logic gates with more inputs. All this means is that it does not matter if the inputs are connected to the gate as A and B or as B and A. The gate works with either configuration. The implications of the **associative law** are not necessarily obvious. Note in Table 542.1 there are three inputs and in one case they are grouped as A alone and B, C together, and in the other case A, B are grouped with C alone. What this is saying is that if there are three inputs, but you only have two input gates available, you can accomplish the desired result by feeding two inputs (B + C) into one gate and the output of that gate fed into a second gate along with input A. This is illustrated in Figure 542.4 for the Boolean expression A + (B + C). This is the same as feeding A, B, C into a three input gate.

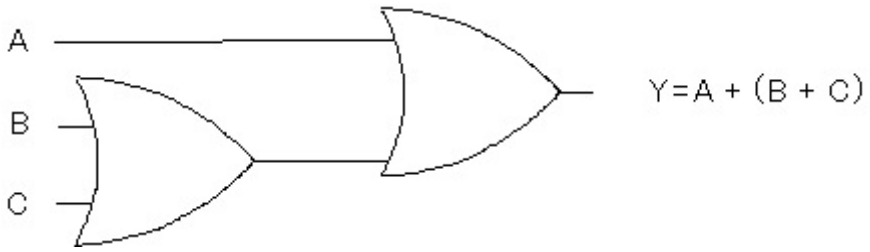


Figure 542.4 The associative law makes it clear that inputs can be combined in different ways to accomplish an objective such as this case where there are three inputs but only two input gates are available.

The **distributive law** demonstrates that the rules for multiplying and factoring an algebraic expression apply to a Boolean expression except that the square of a variable is simply the variable itself. Table 542.1 gives two examples of the distributive law. Look carefully at the second example (A + B)(C + D) = AC + AD + BC + BD. Figure 542.5 shows this Boolean expression represented with logic gates. In one case two OR gates feed into an AND gate. In the other case four AND gates feed into an OR gate. The left side of this expression is shown as the product of two sums. The right side of the expression is shown as the sum of four products. Summing is not a legal move in Boolean math, but that is how it is referenced. Remember that plus (+) means an OR operation. The real significance of the format of the previous expression is that with Boolean math it is sometimes necessary to put an expression into the **product of sums** format (POS) and sometimes it is necessary to put the expression in the **sum of products** format (SOP). Later in this Tech Note a technique of expression reduction called Karnaugh mapping will be explained. In order to enter the expression into the Karnaugh map the expression must first be put into the sum of products format.

There is a digital chip made to act as a sum of products device. The device is called an AND-OR-INVERT (AOI) gate. The chip has four AND gates feeding into an OR gate with an inverter at the output. Two of the AND gates have two inputs and two have three inputs. If an input on one of the AND gates is not needed that input is tied to a one state.

Study the absorption law and rules in Table 542.1 carefully as they are very useful in reducing Boolean expressions into the most simplistic form. If you have any doubts about the validity of any of the expressions, diagram them with logic gates and analyze the results. You will find in every case the rules are true. If the variables are separated with a dot (•) they are inputs to an AND gate. If the variables are separated with a plus sign (+) they are inputs to a OR gate.

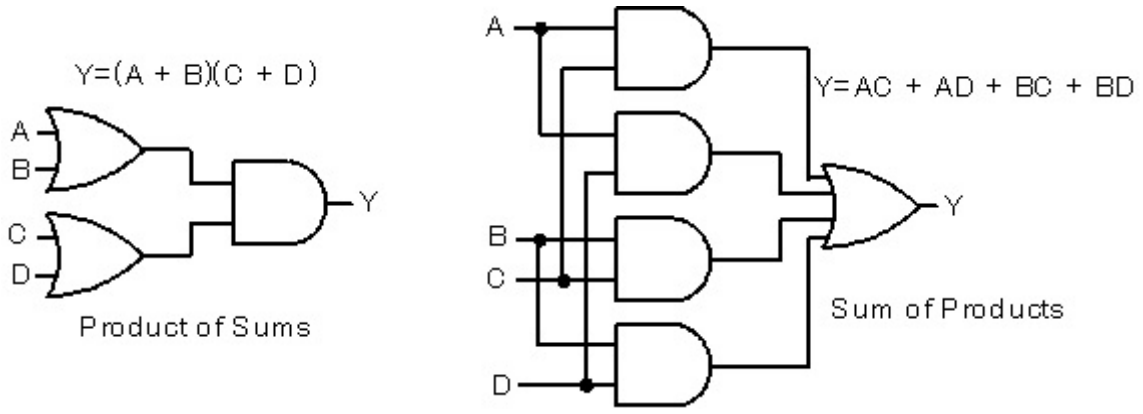
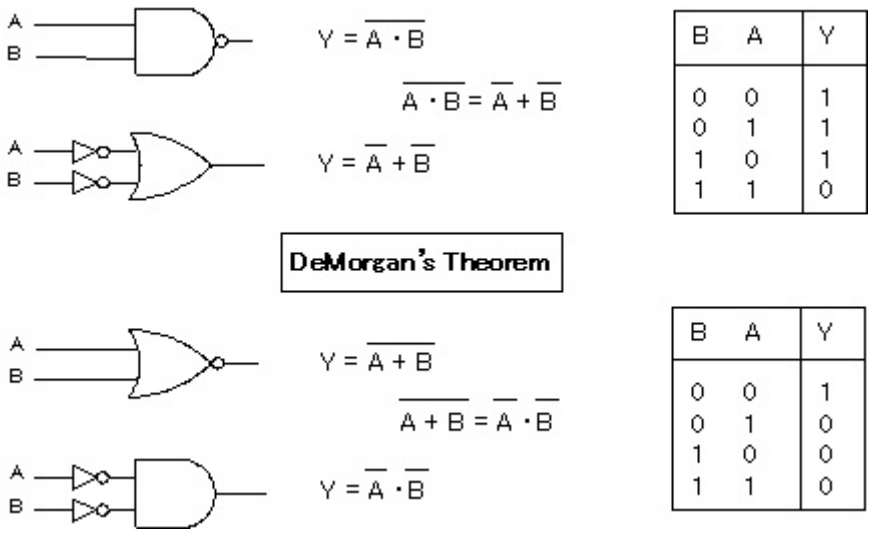


Figure 542.5 *According to the distributive law the two Boolean expressions are equivalent where in the left hand case the gates are in the product of sums format and in the right hand case they are in the sum of products format.*

DeMorgan's Theorem: The NAND gate is simply an AND gate with an inverter attached to the output. If inputs to the gate are A, B, the output is A dot B with an inverter symbol above both variables ($\overline{A \bullet B}$). This is shown in Figure 542.3 and Figure 542.6. When working with a Boolean expression it is awkward to have two or more variables combined under one inverter symbol. Usually it is desirable to be able to separate the variables so they can be manipulated individually. When variables are combined under one inverter symbol, they **cannot** be simply separated. For example $\overline{A \bullet B} \neq \overline{A} \bullet \overline{B}$. It was discovered that if the inputs A, B were first inverted then used as inputs to an OR gate the output would be NOT A OR NOT B, ($\overline{A} + \overline{B}$). This is illustrated in Figure 542.6. What is significant about this discovery is that the truth table for $\overline{A} + \overline{B}$ is equivalent to the truth table for $\overline{A \bullet B}$. This also works for more than two variables as shown in Table 542.1. It was also discovered that $\overline{A+B} = \overline{A} \bullet \overline{B}$. This became known as DeMorgan's theorem and it is a significant tool when working with Boolean expressions.



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

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

Figure 542.6 *DeMorgan's theorem shows how an OR gate can substitute for a NAND gate, and how an AND gate can substitute for a NOR gate.*

Here is an example of DeMorgan's theorem where the variable A is inverted before it is fed into a NAND gate. The variable B is not inverted. As shown in Figure 542.7, the output of the gate converts to A + $\overline{B}$. The purpose of applying DeMorgan's theorem was to convert the Boolean expression to a form where each variable can be manipulated individually.

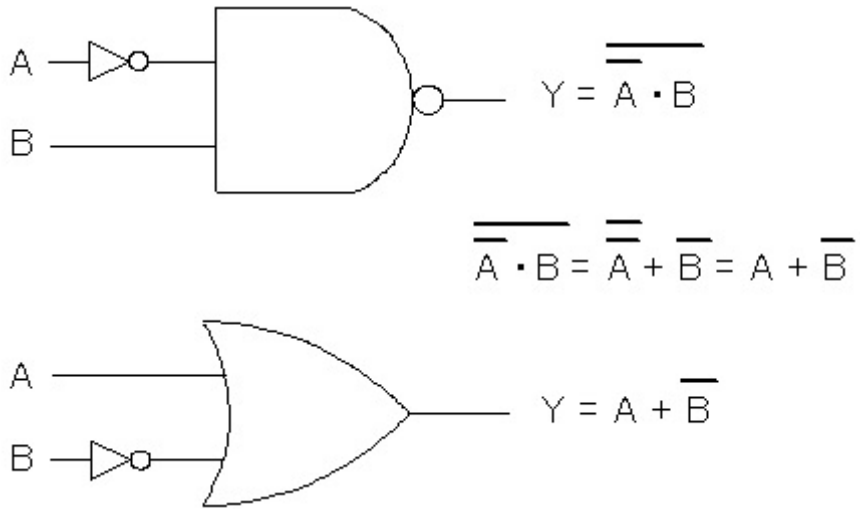
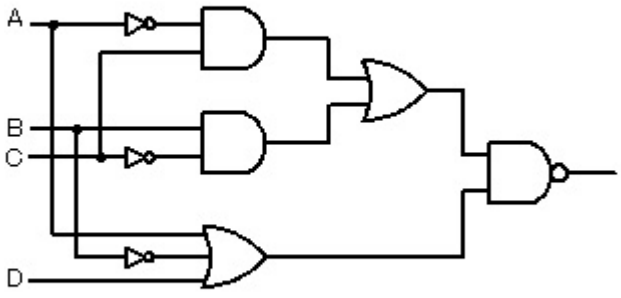


Figure 542.7 *Example of the application of DeMorgan's theorem where the expression is converted to a form where each variable can be manipulated individually.*

Another example of the application of DeMorgan's theorem is shown in Figure 542.8 where the last gate in the system of logic gates is a NAND gate that inverts the output. The top line of the Boolean expression represents the gates as shown in the diagram. With repeated applications of DeMorgan's theorem the expression is converted to the sum of products format. This is a necessary process before an expression can be entered into a Karnaugh map to determine if it can be further simplified.



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

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

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

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

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

$$Y = A\overline{B} + AC + \overline{B}\,\overline{\overline{C}} + C\overline{C}^{\,0} + \overline{\overline{A}}\,B\,\overline{D}$$

$$Y = A\overline{B} + AC + \overline{B}\,\overline{\overline{C}} + \overline{\overline{A}}\,B\,\overline{D}$$

Figure 542.8 *A Boolean expression is written for a system of logic gates then by applying DeMorgan's theorem it is converted tp a sum of products format.*

Universal Logic Gates: Logic gates are versatile tools when working with digital electronic circuits. All of the basic gates can be constructed using the AND, OR, and inverter (NOT). Two gates can be used to construct all other logic gate functions and those are the NAND and NOR. As a result they are often called the "universal" gates. Using these two gates to create other gates may not be very efficient, but in a pinch when the desired gate is not available, they can be used to get the job accomplished. Figure 542.9 shows two ways each gate can be converted to an inverter. The simple method is to tie all of the inputs together so there is one input and one output as shown in Figure 542.9. The other way to make an inverter out of a NAND, NOR gate is also shown in Figure 542.9. In the case of the NAND gate, use one input and tie the other inputs to the one state (+5v). In the case of the NOR, use one input and tie the others to the zero state (ground).

Sometimes a gate has more inputs than are needed. The unused inputs usually just can't be ignored. Usually they must be connected either to the one state or the zero state. This is also true with other types of digital electronic devices. Figure 542.9 also shows how unused inputs can be taken out of the circuit. The examples shown are for three input gates where only two inputs are needed. In the case of the AND and NAND, tie the unused inputs to the one state (+5v). In the case of the OR and NOR, tie the unused inputs to the zero state (ground).

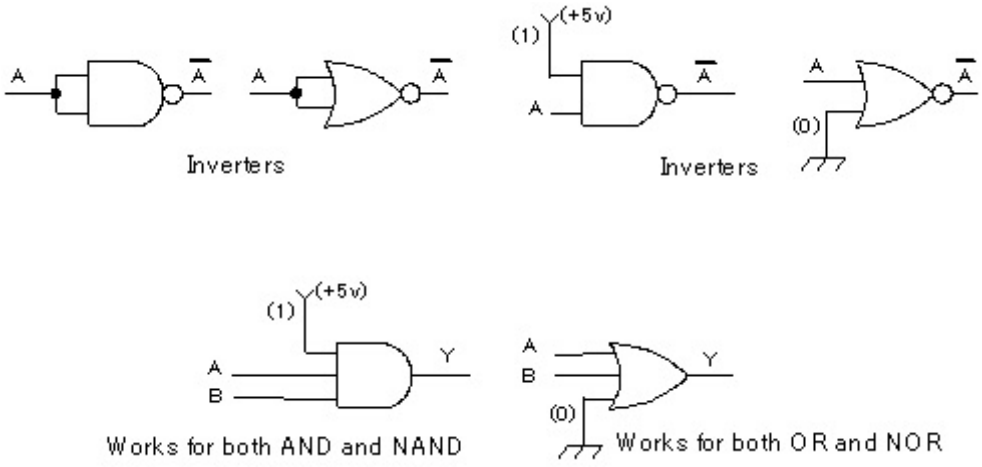


Figure 542.9 *Two methods are shown to use the NAND and NOR gate as an inverter. Also shown is the method of dealing with extra gate inputs that are not needed.*

Boolean Expression Reduction: Reducing a Boolean expression to a simple form is sometimes difficult. The question always remains if the expression can be reduced further. Here is an example of expression reduction using the laws and rules of Table 542.1. As a starting point this expression is in the sum of products format.

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

$$(\overline{A}\,\overline{B} + A\,B) = 1 \quad \text{(from Table 542.1)}$$

$$Y = \overline{C} + A\,\overline{B}\,\overline{C} + \overline{A}\,B = (1 + A\,\overline{B})\,\overline{C} + \overline{A}\,B$$

$$(1 + A\,\overline{B}) = 1 \quad \text{(from Table 542.1)}$$

$Y = \overline{C} + \overline{A}\,B$ The original equation reduces to "(Not C) or (Not A) and B.

In this case it seems clear that this is probably the most simple from of the expression. In other cases it is not necessarily clear. The technique of Karnaugh mapping helps to visualize when the most simple Boolean expression has been achieved.

Karnaugh Mapping: This is a technique that can be used to eliminate redundant steps in a Boolean expression and reduce it to as simple a form as practical. Generally if there are only two variables this technique is not necessary, but it can be used. It is often used when there are three or four variables. If there are more than four variables using this technique can become cumbersome. Figure 542.10 shows a Karnaugh map for three variables and four variables. The Karnaugh map is a rectangular grid with as many spaces as there are combinations of variables in the truth table. The number of spaces is 2 raised to the power equal to the number of variables. For two variables $2^2 = 4$, three variables, $2^3 = 8$, four variables $2^4 = 16$ spaces. The labeling of the rows and columns in the Karnaugh map must be done such that only one variable changes from one row or column to the adjacent row or column. For example, if $\overline{B}$ changes to B, then A or $\overline{A}$ must remain unchanged. Once the Karnaugh map has been constructed, the Boolean expression must be converted to the sum of products format. The following is a three variable Boolean expression in the sum of products format and a four variable expression in the sum of products format. If these expressions were to be converted to gates they would be AND gates feeding into an OR gate. The objective is to see if it is possible to simplify these expressions.

$$Y = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + \overline{A}\,B\,\overline{C}$$

$$Y = \overline{A}\,\overline{B}\,\overline{C} + A\,\overline{C}\,\overline{D} + A\,\overline{B} + A\,B\,C\,\overline{D} + \overline{A}\,\overline{B}\,C$$

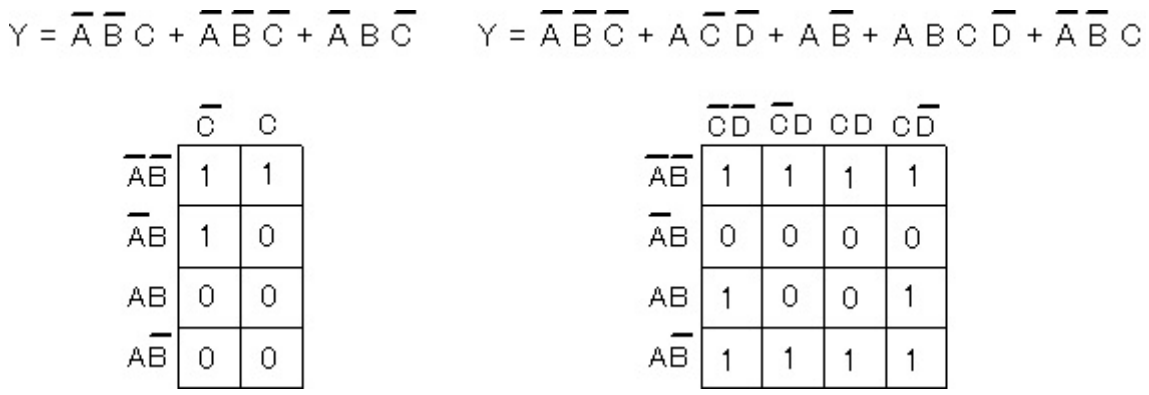


Figure 542.10 *Construct a Karnaugh map for the number of variables in the Boolean expression so that from one row or column to the adjacent row or column only one variable changes from X to $\overline{X}$ or vice versa.*

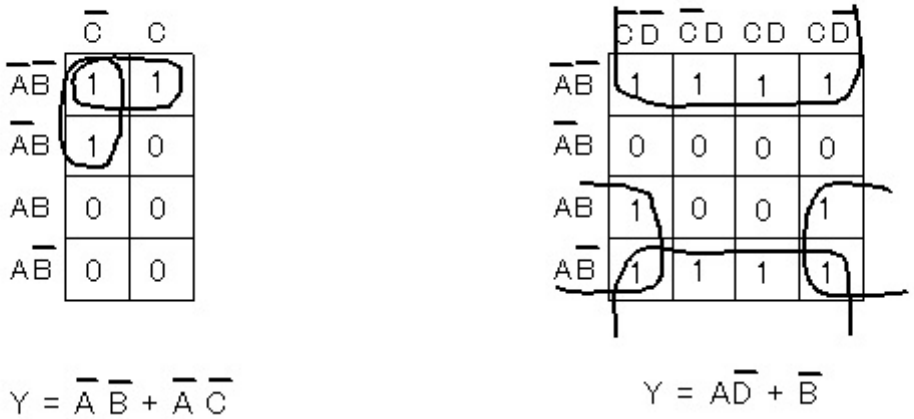
The next step in the process is to enter a one (1) in each square of the Karnaugh map for each set of products that are satisfied. In Figure 542.10 the Karnaugh maps have been filled in for the two previous expressions. If a variable is missing from a set of products in an expression than the value of the missing variable does not matter. The statement is satisfied when the variable is either zero or one. To make sure the map is filled in completely it is sometimes useful to construct a truth table for the expression then transfer the values to the Karnaugh map. The truth table for each of the previous expressions is shown in Table 542.2.

Table 542.2 Truth tables for the Boolean expressions, $Y = \overline{A}\,\overline{B}\,C + \overline{A}\,B\,\overline{C} + A\,\overline{B}\,\overline{C}$, and $Y = \overline{A}\,\overline{B}\,\overline{C} + A\,\overline{C}\,\overline{D} + A\,\overline{B} + A\,B\,C\,\overline{D} + \overline{A}\,\overline{B}\,C$

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

Before proceeding to the next step it is important to understand the Karnauth map. Even though the map is flat on the page with edges, the actual map has no edges. The map is continuous. The top edge and the bottom edge are the same and there is no edge. The top row is adjacent to the bottom row. The left edge and the right edge are the same and there is no edge. The left column is adjacent to the right column. There are no corners. The corners are the same point. It sounds like something out of a weird dream, but it's not all that bad. Now that it is understood that the map is continuous with no edges, circle groups of ones (1s) that make up blocks of 2, 4, or 8. If a one is isolated by itself it is circled as a block on only one. Make sure all of the 1s are included in some block of either 1, 2, 4, or 8. The previous Karnaugh map examples are shown in Figure 542.11 with the groups of 1s circled. Note that for the four variable example there is a group of four and a group of eight.

$Y = \overline{A}\,\overline{B}\,C + \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C}$ $\quad$ $Y = \overline{A}\,\overline{B}\,\overline{C} + A\,\overline{C}\,\overline{D} + A\,\overline{B} + A\,B\,C\,\overline{D} + \overline{A}\,\overline{B}\,C$



$Y = \overline{A}\,\overline{B} + \overline{A}\,\overline{C}$ $\quad$ $Y = A\overline{D} + \overline{B}$

Figure 542.11 *Remember that the Karnaugh map is continuous and does not have edges. Circle the 1s on the map in the largest groups possible of 1, 2, 4, or 8.*

Now examine each circled group in the Karnaugh map and write down the variable or variables in the group that remain unchanged. First look at the 4 by 2 Karnaugh map. In one circle the variables that do not change are $\overline{A}\ \overline{B}$ and in the other circle the variables that do not change are $\overline{A}\ \overline{C}$. The simplified form of the expression is $Y = \overline{A}\,\overline{B}\,C + \overline{A}\,\overline{B}\,\overline{C} + \overline{A}\,B\,\overline{C} = \overline{A}\ \overline{B} + \overline{A}\ \overline{C}$. This expression can be represented with two 2-input AND gates feeding into one 2-input OR gate. In the case of the 4 by 4 Karnaugh map, $A\,\overline{D}$ remain unchanged for the circle of four and only $\overline{B}$ remains unchanged for the circle of eight. The simplified form of the expression is $Y = \overline{A}\,\overline{B}\,\overline{C} + A\,\overline{C}\,\overline{D} + A\,\overline{B} + A\,B\,C\,\overline{D} + \overline{A}\,\overline{B}\,C = \overline{B} + A\,\overline{D}$. This expression can be represented with one 2-input AND gate and one 2-input OR gate.

Here are the steps summarized for reducing an expression to the most simple form possible using the technique of the Karnaugh map. The same result can be accomplished by using the Boolean laws, rules, and theorems.

1. Put the Boolean expression in the sum of products form.
2. Fill in the truth table for the Boolean expression.
3. Construct a Karnaugh map and put a 1 in each square that satisfies the Boolean sum of products expression.
4. Circle the 1s in the map in the largest groups possible of 1, 2, 4, or 8 and be sure all of the 1s are included in some group.
5. For each group circled in the Karnaugh map identify the variable or variables in that group that do not change.
6. Write the simplified form of the Boolean expression consisting of a sum of products where each set of products are the variables of each group that do not change.

Conclusion: Engineers and technicians that work with digital electronics or control systems need to have a working knowledge of electronic gates and Boolean math. For a typical application it may be necessary to bring about an action when a digital bus reaches a specific number or numbers. This means it is necessary to be able to convert from the binary to the decimal numbering system as well as know how to use logic gates to bring about the desired action at the appropriate point. It is important to be able to write a Boolean expression from a set of logic gates and to be able to represent a Boolean expression with logic gates. Logic gates are commonly used components in digital electronic systems and instrumentation.