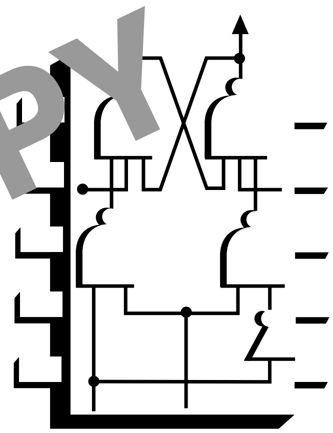
Digital Logic Systems

Table of Contents

DIGITAL LOGIC SYSTEMS

Lesson One

Digital Logic Fundamentals



TPC Training Systems

29101

Lesson

1 Digital Logic Fundamentals

TOPICS

Digital Logic
Boolean Algebra
Logical AND Function
Logical OR Function
Logical NOT Function
Digital Applications
Solid-State Switches
Positive and Negative Logic
NAND Logic
Combining Logic Circuits
TTL Logic
Integrated Circuit (IC) Logic Devices
7400 Series TTL Logic
CMOS Logic

OBJECTIVES

After studying this Lesson, you should be able to…

- Explain the difference between digital and analog circuits.
- Describe AND, NOT, and OR logic functions.
- Explain how solid-state switches can perform logic functions.
- Compare equivalent NAND and NOR gates using positive and negative logic.
- Discuss the importance of TTL and CMOS circuits.

KEY TECHNICAL TERMS

Digital circuit 1.01 a circuit whose inputs and outputs are represented by discrete values rather than continuous values

Binary circuit 1.02 a circuit having only two possible signal levels

Boolean algebra 1.04 a mathematical system in which a variable can have only one of two values

Truth table 1.08 a table that gives the outputs of a circuit for all possible combinations of inputs

NOT function 1.12 a function for which the output is an inversion of the input

Systems built on digital logic have become more and more common. As more complex digital electronic devices have become available at low cost, industries have used them to do jobs previously done by mechanical or electromechanical control devices. For this reason, familiarity with digital logic and digital systems has become increasingly important for today's technician.

In this Unit, you will read about logic circuits, from the simplest digital devices to complex systems. Lesson One introduces the basic logic devices and describes how they are used in solid-state logic circuitry.

Digital Logic

1.01 *Digital circuits* are electronic circuits whose inputs and outputs are discrete values that can be only one of two possible states, rather than continuous (or *analog*) values. In linear amplifiers, for example, the output voltage is equal to the input voltage times a gain factor. The output can have any value from zero to some specified maximum. The output is an analog of its input. That is, an analog output is proportional to some function of its input.

1.02 Theoretically, digital circuits may have any number of discrete inputs and outputs. However, the term *digital* is most commonly used to describe *binary* logic circuits. Binary circuits have two and only two possible signal levels. These levels (or *states*) are known by several names—for example, TRUE/FALSE, ON/OFF, HIGH/LOW, or 1/0. For practical reasons—particularly for TTL logic, discussed later in this Lesson—the LOW (OFF, FALSE) state is defined as 0 to +0.8 V and the HIGH (ON, TRUE) state is defined as +2 to 5 V. Any input level between +0.8 and 2 V is undefined and allows for component variation or voltage drift.

1.03 Automobile dashboard gauges provide good examples of the difference between analog and digital systems. For example, the fuel gauge is an analog meter that shows any level of fuel in the tank from empty to full. The pointer position is an analog of the fuel level. However, the alternator charge indicator on most cars is simply a light. The light has only two states, ON and OFF, and indicates only two conditions, charge and discharge. When the light is ON, the system is discharging. When the light is OFF, the system is charging. No intermediate levels of charge or discharge are shown. Thus, the alternator indicator acts as a digital monitoring system.

Boolean Algebra

1.04 All logic functions performed by digital circuits can be described by Boolean equations. That is, logic circuits perform Boolean algebra operations. *Boolean algebra* is a mathematical system in which a variable can have only one of two values. Logic functions can be implemented mechanically (with levers, hydraulic or pneumatic valves, or manual switches) or they may be statements of fact. The logic statements can even represent a chemical reaction or a statement of economics.

1.05 There are only three mathematical operations in Boolean algebra. They are the following:

- *AND* (logical multiplication)—written $A \cdot B$ or AB and pronounced "A and B."
- *OR* (logical addition)—written $A + B$ and pronounced "A or B."
- *NOT* (negation or complement)—written $\overline{A}$ and pronounced "not A."

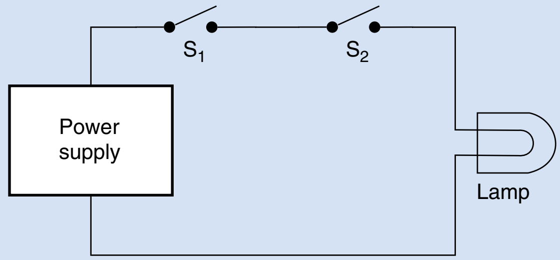
1.06 You can construct all logic systems, no matter how complex, by using just these three Boolean operations. As a result, all logic circuits (or gates) have a great deal in common, which is what makes mass production of logic circuits possible. The resulting price reductions have brought about a greater use of logic circuits in increasingly complex systems.

Logical AND Function

1.07 Figure 1-1 on the following page illustrates a binary two-element series circuit. This simple circuit is similar in operation to a two-input AND logic circuit (*AND gate*):

When S_1 is OFF and S_2 is OFF, the lamp is OFF.
When S_1 is OFF and S_2 is ON, the lamp is OFF.

Fig. 1-1. Two-element series circuit



When S_1 is ON and S_2 is OFF, the lamp is OFF.
When S_1 is ON and S_2 is ON, the lamp is ON.

Note that only the fourth set of conditions will turn the light ON—that is, the lamp is ON only if S_1 and S_2 are ON. If ON is defined as the *TRUE logic state*, or *logical 1*, the Boolean equation for this circuit is:

$$\text{ON} = S_1 \cdot S_2.$$

1.08 Another way to describe this logic function is with a truth table. A *truth table* gives the outputs of the circuit for all possible combinations of inputs. One form of truth table for the AND circuit of Fig. 1-1 is shown on the left in Fig. 1-2. If you think of all ON (or TRUE) conditions as logic 1, and all OFF (or FALSE) conditions as logic 0, the truth table will appear as on the right in Fig. 1-2. This is the way truth tables are usually shown. However, sometimes they are presented with TRUE/FALSE, HIGH/LOW, YES/NO, or other terms to designate the two possible logic states.

1.09 Each logic function has a standard symbol. Figure 1-3 shows the symbol for a two-input AND function. It also shows the Boolean equation for the function and the truth table for the AND function. This AND logic symbol represents any Boolean AND function, regardless of how it is implemented. That is, it does not necessarily represent an electronic circuit.

Fig. 1-2. Two-element truth tables

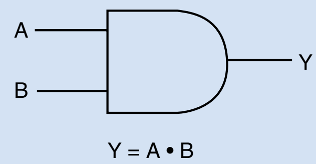
Table A

S_1	S_2	Lamp
OFF	OFF	OFF
OFF	ON	OFF
ON	OFF	OFF
ON	ON	ON

Table B

S_1	S_2	Lamp
0	0	0
0	1	0
1	0	0
1	1	1

Fig. 1-3. AND gate logic symbol and truth table



Logic symbol

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

Truth table

Logical OR Function

1.10 The two-element parallel circuit shown in Fig. 1-4 is similar in operation to a two-input OR logic circuit (*OR gate*). The main difference between this circuit and the two-input AND circuit of Fig. 1-1 is that the lamp in Fig. 1-4 is ON whenever S_1 is ON or S_2 is ON. The Boolean equation for this circuit is:

$$\text{ON} = S_1 + S_2.$$

The standard logic symbol and truth table for a two-input OR function are shown in Fig. 1-5.

1.11 Exclusive OR (XOR) is a special function—the output is TRUE if any one, but only one, of the inputs is TRUE. The Boolean equation for this function is:

$$\text{ON} = S_1 \oplus S_2.$$

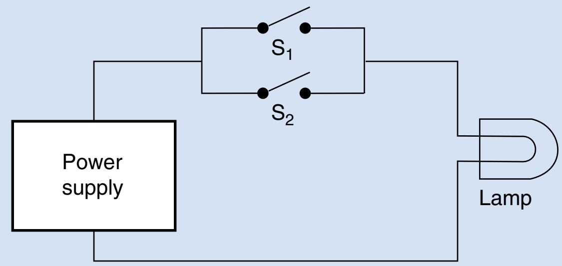
Logical NOT Function

1.12 The output of the logical *NOT function* is an inversion of its input. The NOT function is called a *complementing* function. For example, the complement of TRUE is FALSE, the complement of logical 1 is logical 0, and the complement of ON is OFF.

1.13 The circuit shown in Fig. 1-6 is similar in operation to the NOT function. The relay in this circuit is normally closed. Thus, if S_1 is closed, the relay is energized and S_2 is open (not closed). The Boolean equation for this example is:

$$S_2 = \overline{S_1}$$

Fig. 1-4. Two-element parallel circuit



The *overbar* (the line over S_1) is the convention used to indicate inversion.

1.14 The standard symbol and truth table for a NOT circuit are shown in Fig. 1-7. The NOT function is very important in implementing logic systems. NOT circuits are also called *inverters*. The small circle (*bubble*) at the inverter's output indicates a logic inversion.

Digital Applications

1.15 You will probably work with many digital control systems. A common two-state control system is a simple light switch. The switch is either ON or OFF. There is no in-between stage. If the switch is ON and power is available, the light goes on. If the switch is OFF or the power fails, the light goes off.

1.16 In industry, logic circuits are used to control and monitor various processes—for example, heating and lighting systems, chemical processes, and electronic alarm systems. They also are used to check battery charge, water temperature, and fuel level in automobiles. In addition, logic circuits control motors and hydraulic and mechanical systems.

Fig. 1-6. Relay acting as logical NOT function

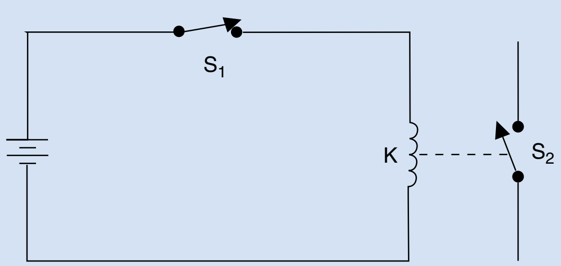
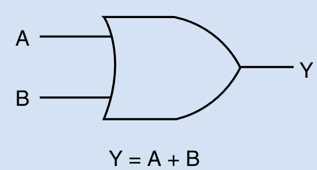


Fig. 1-5. OR gate logic symbol and truth table



Logic symbol

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

Truth table

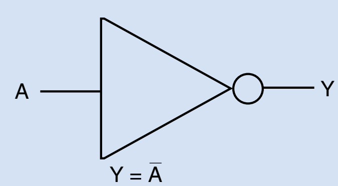
1.17 Logic systems operate on digital (binary) input signals and produce digital output signals. Common digital input sources can include position, level, pressure, and temperature switches, various manual switches, and outputs from other digital systems. Digital outputs are used to drive many kinds of equipment—for example, relays, indicator lamps, printers, stepper motors, and flow valves, to name just a few.

Application 1-1

A technician devised a circuit for closing windows automatically in case of rain. He used the simple,circuit shown at the top of the following page as a sensing device.

The grid on the left rep-resents two metal forms mounted very close together, but not touching, on some kind of nonconductor. When the forms are dry, there is no connection to ground, and voltage is impressed on the input of the inverter through the resistor, R_1. This causes a LOW output from the inverter, which signals the rest of the circuit to leave the windows open. If rain lands on the forms and creates a bridge between them, current flows to ground,

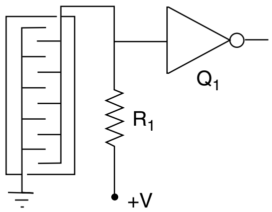
Fig. 1-7. NOT function logic symbol and truth table



Logic symbol

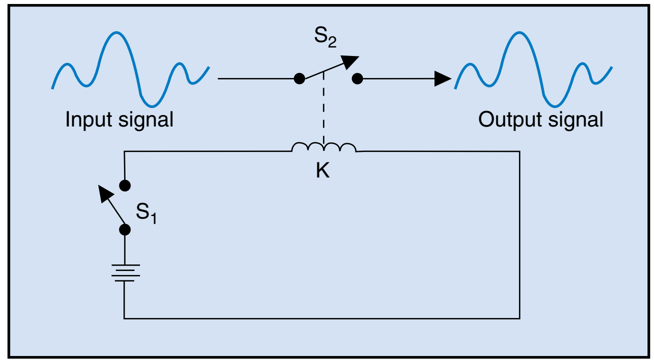
A	Y
1	0
0	1

Truth table



bringing the inverter side of the resistor to 0 V. This causes the inverter to switch and causes its output to go HIGH, signaling the rest of the circuit to energize and close the windows.

Fig. 1-8. Simple mechanical switch



Solid-State Switches

1.18 You have seen how Boolean algebra can represent different kinds of logic systems. These systems can be operated by electrical, mechanical, pneumatic, hydraulic, or other means. However, this Lesson is concerned only with how various logic functions are performed using *solid-state* switching circuits.

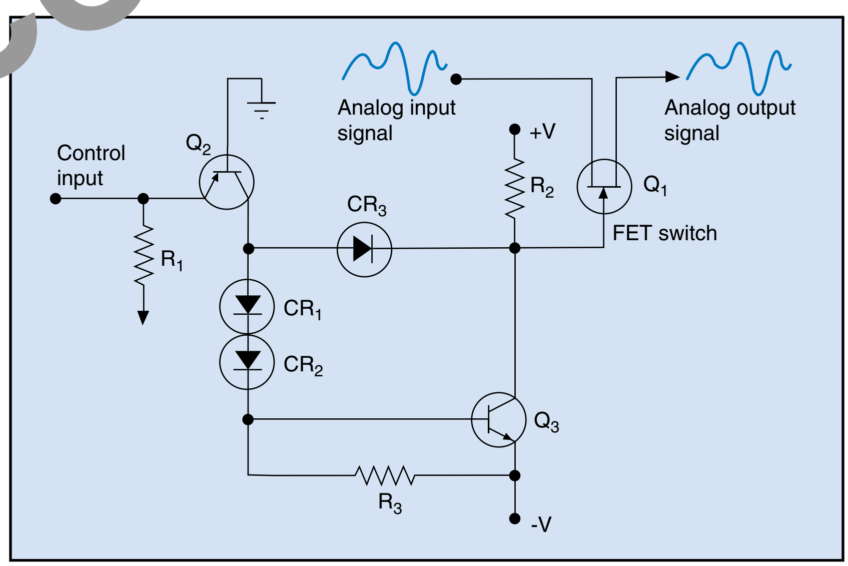
1.19 Solid-state switches can be either digital or analog devices. You may see solid-state switches in a system that does not use digital logic. One example is a mechanical switching device like the relay shown in Fig. 1-8. The relay passes whatever signal is at its input. The inputs and outputs in this case can be digital or analog signals. That is, the relay passes whatever signal is present when its contacts are closed, and it passes nothing when its contacts are open.

1.20 Solid-state analog switches, such as the one shown in Fig. 1-9, are similar to relays. When the control input is HIGH, the switch transistor (Q_1) is turned ON. When ON, Q_1 passes whatever signal is presented to it. Thus, the output is an analog of the input. When the control input is LOW, Q_1 has a high impedance and the input signal is not passed to the output.

1.21 Field-effect transistors (FETs) are usually used in solid-state analog switches. The reason is that when FETs are forward biased (turned ON), they pass current in both directions, much like mechanical switch contacts. However, a *logic switch* is not required to pass current in both directions or to reproduce the input signal at its output. A logic switch only has to tell its output which one of two possible states is at its input.

The Programmed Exercises on the next page will tell you how well you understand the material you have just read. Before starting the exercises, remove the Reveal Key from the back of your Book. Read the instructions printed on the Reveal Key. Follow these instructions as you work through the Programmed Exercises.

Fig. 1-9. Solid-state analog switch



1-1. The inputs and outputs of digital circuits are represented by ________ values.	1-1. DISCRETE or BINARY Ref: 1.02
1-2. Binary circuits have ________ possible inputs and outputs.	1-2. TWO Ref: 1.02
1-3. Logic functions can be described by ________ equations.	1-3. BOOLEAN Ref: 1.04
1-4. Name the three mathematical operations in Boolean algebra.	1-4. AND, OR, and NOT Ref: 1.05
1-5. A(n) ________ gives circuit outputs for all possible combinations of inputs.	1-5. TRUTH TABLE Ref: 1.08
1-6. In binary logic, the complement of logical 1 is logical ________.	1-6. 0 (ZERO) Ref: 1.12
1-7. A solid-state analog switch is similar to a(n) ________.	1-7. RELAY Ref: 1.20
1-8. Must a logic circuit output faithfully reproduce its input signal magnitude?	1-8. NO Ref: 1.21

Positive and Negative Logic

1.22 A basic AND circuit with its Boolean equation is shown in Fig. 1-10. This circuit uses two diodes and a resistor. If either input A or B is held LOW at 0 V, the associated diode will conduct and the output (Y) will be held near 0 V. Only if both inputs A and B are HIGH will the output be HIGH. If the logic is defined as:

TRUE = HIGH voltage state = 1
FALSE = LOW voltage state = 0.

Then the logic equation is:

$$Y = A \cdot B.$$

In other words, Y is TRUE if both A and B are TRUE. This statement expresses the AND logic function.

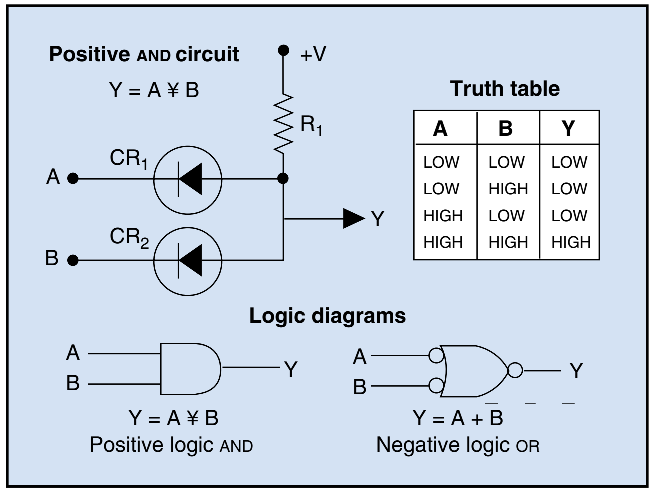
1.23 Note that the circuit in Fig. 1-10 corresponds to the AND function only if the logic is as defined—that is, *HIGH state TRUE*. HIGH state TRUE logic also is referred to as *HIGH TRUE logic, positive TRUE logic*, or simply *positive logic*. The active state is HIGH (logical 1) in positive logic.

1.24 Logic also can be defined as LOW state TRUE. That is, a signal near 0 V (logical 0) is defined as TRUE, and a signal near +5 V (logical 1) is defined as FALSE. This logic definition is called *LOW TRUE logic* or *negative logic*. It is important to understand that negative logic does not mean that the voltage signal necessarily goes negative. It simply means that the TRUE state has a lower value than the FALSE state. The active state is LOW (logical 0) in negative logic. Negative logic is used less frequently than positive logic.

1.25 The concept of positive and negative logic is the basis for alternate logic gates. In Fig. 1-10, the standard AND symbol is on the left and the alternate symbol is on the right. Alternate gates make it possible for one circuit to implement different logic functions, depending on how the logic is defined.

1.26 If the logic in Fig. 1-10 is defined as positive logic (TRUE = HIGH = 1), the circuit implements the AND function. That is, if both inputs A and B are HIGH, the output is HIGH (1), and any other set of inputs produces an output that is LOW (0). However, if the logic is defined as negative logic (TRUE = LOW = 0), then Y is LOW if either A or B is LOW. Thus, the same circuit that implements an AND function in positive logic performs an OR function in negative logic. In Boolean algebra, the two circuits are equivalent.

Fig. 1-10. Diode logic AND circuit



A	B	Y
LOW	LOW	LOW
LOW	HIGH	LOW
HIGH	LOW	LOW
HIGH	HIGH	HIGH

1.27 Logic circuits normally are defined in terms of positive logic. It is correct to call the circuit in Fig. 1-10 an AND circuit. Also, both symbols represent the AND circuit. It is important to remember, however, that the circuit will perform an OR function when used with negative TRUE (active LOW) logic inputs.

1.28 When you draw or interpret logic diagrams, remember that a bar over a term in a logic equation, as shown in Fig. 1-10, indicates that the function is inverted. Likewise, recall that a bubble on a logic symbol indicates that a signal is inverted. Each bubble is clearly attached to an input or output line and indicates that the signal on that line is inverted. A function may include one inverted signal or many. Remember that, for positive logic, the HIGH (active) signal is logical 1 and LOW is logical 0. For negative logic, the active signal is LOW (0).

NAND Logic

1.29 Although the circuit in Fig. 1-10 will produce the AND function, it is not satisfactory in large logic arrays where its output must drive additional logic circuits. Because there are no transistors in the circuit to provide gain, there is no way to reconstitute the signals if they become weak due to circuit loading.

Fig. 1-11. Three-input NAND gate

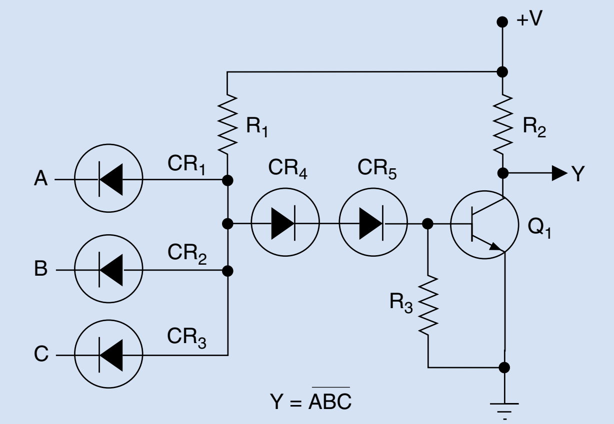
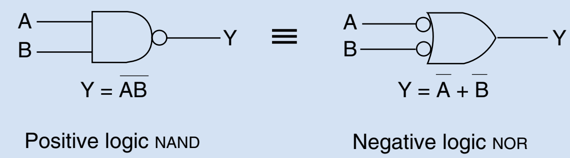


Fig. 1-13. NAND/NOR equality



A	B	Y
LOW	LOW	HIGH
LOW	HIGH	HIGH
HIGH	LOW	HIGH
HIGH	HIGH	LOW

Truth table

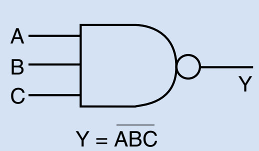
1.30 Figure 1-11 shows an improved logic gate—a *three-input* NAND (*NOT AND*) *gate*. If all three inputs are HIGH, the output is LOW. If one or more inputs are LOW, the output is HIGH. Therefore, in terms of positive logic, the logic equation is:

$$Y = \overline{A \cdot B \cdot C} = \overline{ABC}$$

For packaged logic circuits, the number of inputs is limited only by the number of connections that can be made to the package.

1.31 Figure 1-12 shows the logic symbol for the three-input NAND gate along with the corresponding truth table. The NAND circuit is especially important, because it is possible to construct any kind of logic system, from the simplest to the most complex, using only this kind of circuit.

Fig. 1-12. Three-input NAND logic



Logic symbol

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

Truth table

1.32 If the NAND circuit is used with negative logic, then if one or more of the inputs is LOW (0 V), the output is HIGH. That is:

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

This is an example of a *NOR* (*NOT OR*) function. Figure 1-13 shows the equality between the two gate symbols for a two-input NAND circuit and also shows the equality in truth table form.

1.33 You can construct a NOT function from a NAND gate by tying all of its inputs except one to a TRUE (active) logic level voltage. Then the circuit, with its one remaining input, has the same output as the inverting logic switch.

Combining Logic Circuits

1.34 Recall that there are only three mathematical operations in Boolean algebra—AND, OR, and NOT. Because all three of these mathematical operations can be implemented by NAND logic gates, all logic functions can be obtained from combinations of this one kind of gate.

1.35 The following is an example of how NAND gates can be combined to provide more complex logic functions. Imagine a simple machine that has a shuttle moving back and forth between two end points. Suppose that the system is to produce a warning signal if the shuttle passes either end point. Also suppose that the machine has the following digital outputs on the next page:

- S_1, the left-end limit switch, is HIGH if the switch is tripped
- S_2, the right-end limit switch, is HIGH if the switch is tripped
- R, a direction-sensing line, is HIGH when the shuttle is moving to the right.

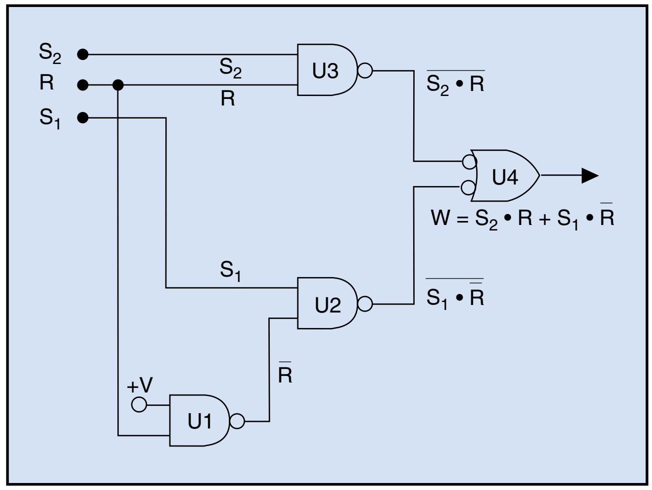
1.36 These three outputs are fed to a logic circuit that produces a warning signal (W) whenever S_2 is tripped and the shuttle is moving right or whenever S_1 is tripped and the shuttle is moving left. The logic equation for this set of conditions is:

$$W = (S_2 \cdot R) + (S_1 \cdot \overline{R})$$

Note that the letter R indicates motion to the right. Therefore $\overline{R}$, the complement of R, indicates motion to the left.

1.37 The circuit equivalent to this logic equation can be constructed with four NAND gates, as shown in Fig. 1-14 at right above. Inverter U1 is a two-input NAND gate with one input tied HIGH. The other inverter input, R, is active HIGH (1), but the inverter output is active LOW. The inverter output is therefore labeled $\overline{R}$.

Fig. 1-14. Combining logic circuits



1.38 U2 also is a NAND gate. Its output is LOW if both inputs are HIGH. That is, if S_1 is HIGH AND $\overline{R}$ is HIGH, the output of U2 is LOW. Therefore, its output is labeled $\overline{S_1 \cdot \overline{R}}$.

1.39 U3 is a NAND gate with S_2 and R as its inputs. Its output is $\overline{S_2 \cdot R}$. That is, its output is LOW if S_2 is HIGH AND R is HIGH.

1.40 U4 also is a NAND gate, but its inputs are active LOW. The NAND gate functions as a negative logic NOR gate. The output of U4 (W) is HIGH if either S_2 and R is LOW or S_1 and $\overline{R}$ is LOW. Notice that this logic circuit combines NOT, AND, and OR functions using only NAND gates.

1.41 U4 is drawn as a negative-input OR gate, the alternate NAND symbol. Many people believe that this makes the logic easier to follow, because it represents more closely the function actually performed by U4 in this circuit. On the other hand, others are opposed to showing bubbles on inputs or using double inversions. U4 also may be shown simply as a NAND gate. Both ways of drawing the circuit are correct, and you will probably run across both in your work.

TTL Logic

1.42 Although all logic functions can be built using only NAND gates, it is not always the most convenient or efficient method. Figure 1-15 at left shows a simplified TTL (*transistor-transistor logic*) positive-input NOR gate. You can use this gate when you need an OR or NOR function and negative TRUE inputs are not

Fig. 1-15. TTL NOR logic gate

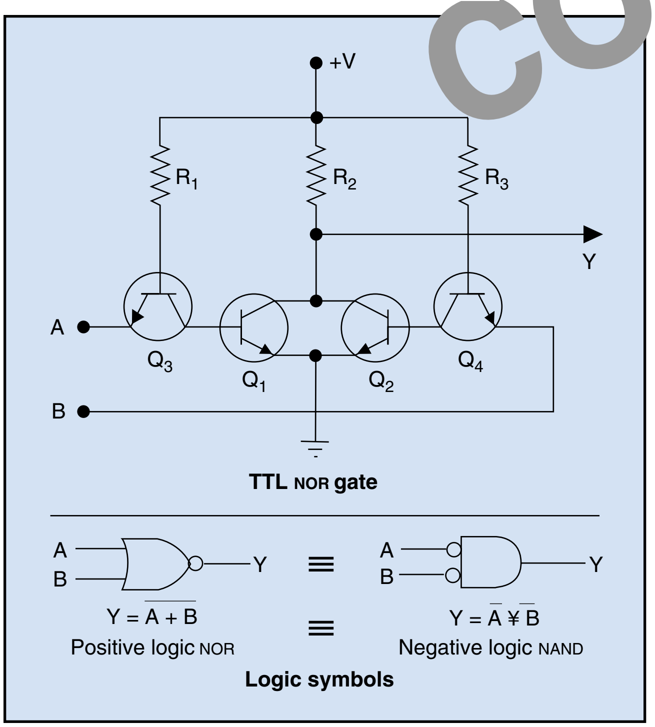
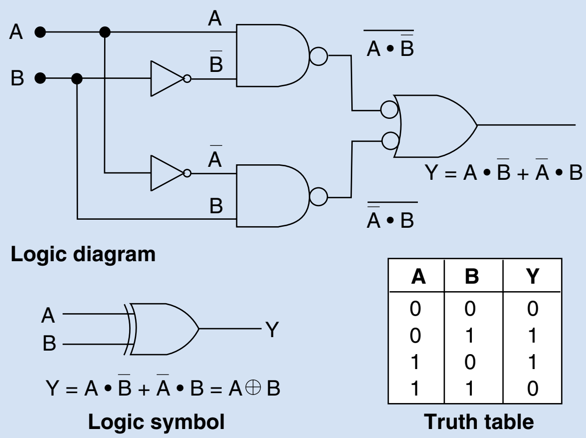


Fig. 1-16. Exclusive OR gate



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

Truth table

available. If input A is HIGH, Q_1 conducts. If input B is HIGH, Q_2 conducts. Because Q_1 and Q_2 share collector resistor R_2, Y is LOW if either A or B is HIGH.

1.43 Remember that this circuit is called a NOR only because it is customary to name the circuits according to their positive logic function. Figure 1-15 also shows that there are two correct logic symbols for the NOR circuit, just as there are for the NAND circuit.

1.44 As with other logic gates, the *exclusive OR* circuit can be made with NAND gates, as shown in Fig. 1-16. The logic symbol and truth table for an exclusive OR gate also are shown in Fig. 1-16. Note the operational symbol in the equation: Y = A ⊕ B. Recall that exclusive OR means that the gate's output is TRUE if one and only one of its inputs is TRUE. In this way it differs from the standard OR gate, whose out put is TRUE if either one or both of its inputs are TRUE.

Integrated Circuit (IC) Logic Devices

1.45 Only a few kinds of logic circuits are needed to perform many different functions, so logic circuits can be easily packaged in modules. Standard modules are mass-produced, greatly reducing their cost. They are simply plugged in or connected to perform various logic functions.

1.46 Early logic modules were made by assembling discrete transistors, diodes, and resistors into compact plastic packages. In 1959, the first integrated circuit (IC) logic became available commercially. Integrated circuits are devices in which the components are integrated into a single silicon chip, rather than being assembled from discrete components. The development of IC logic resulted in lower cost, lower power dissipation, higher speed, and much higher packaging density.

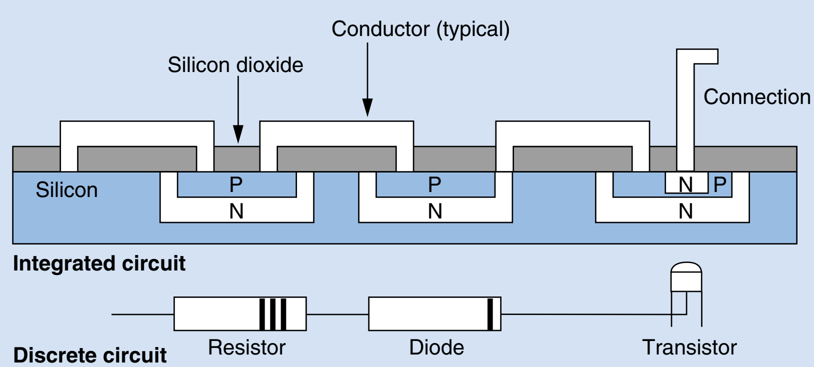
1.47 Figure 1-17 shows the construction of an IC along with its equivalent discrete circuit:

- A resistor is formed in the layers of the IC by the small section of *p*-type silicon between two conductors.
- A diode is formed in the layers of the IC by the *pn* junction.
- An *npn* transistor is formed in the layers of the IC by the pair of *pn* junctions.

This kind of integrated circuit, about one-quarter of an inch square, can contain well over a quarter of a million elements. ICs are packaged in various ways, the most common being the flat dual in-line package (*DIP*) of plastic or ceramic with leads for socket mounting, through-hole soldering, or the more modern surface mounting technology (*SMT*) on circuit cards.

1.48 The first manufactured IC logic circuits generally copied existing discrete circuit designs. As methods and techniques improved, IC logic evolved into transistor-transistor logic, commonly known and

Fig. 1-17. IC construction



written as TTL or T^2L. High-volume users can obtain application-specific integrated circuits (*ASICs*) with their designs built in by the IC manufacturer.

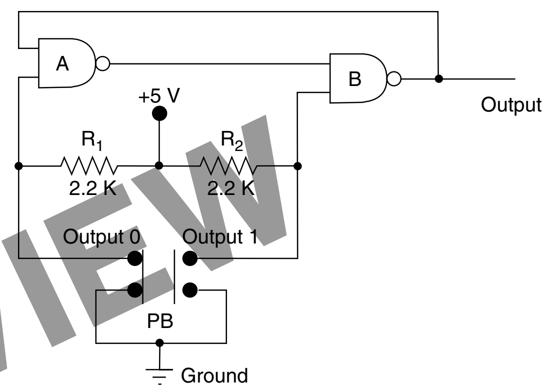
7400 Series TTL Logic

1.49 Figure 1-18 shows a typical TTL two-input NAND gate, a 7400 series quad, which is one of the 7400 series TTL ICs. The logic diagram and pin connections for the package also are shown. It is possible to construct the logic circuit in Fig. 1-14, on page 14, by connecting the four NAND gates contained in this single IC package. The input is to a multiple-emitter transistor, rather than to separate transistors or diodes. If either input A or B is LOW, Q_1 conducts, cutting off Q_2. These conditions cause Q_3 to conduct and Q_4 to be cut off, so the output is HIGH. If both A and B are HIGH, Y is LOW. The output is from a two-transistor (Q_3 and Q_4) circuit called a *totem-pole* or *active pull-up* circuit. Its main advantage is that it provides fast output transitions.

1.50 Of all logic families, the 74 series TTL is by far the most common for circuits with a small (less than 12) or medium (12 to 99) number of gates. The main components in these devices are bipolar digital ICs. The 74 series TTL provides a wide variety of logic functions. It traditionally operates from a single +5-V power supply, but standards for lower voltages (mainly 3.3 V) have been developed. Most IC logic manufacturers offer some version of this family.

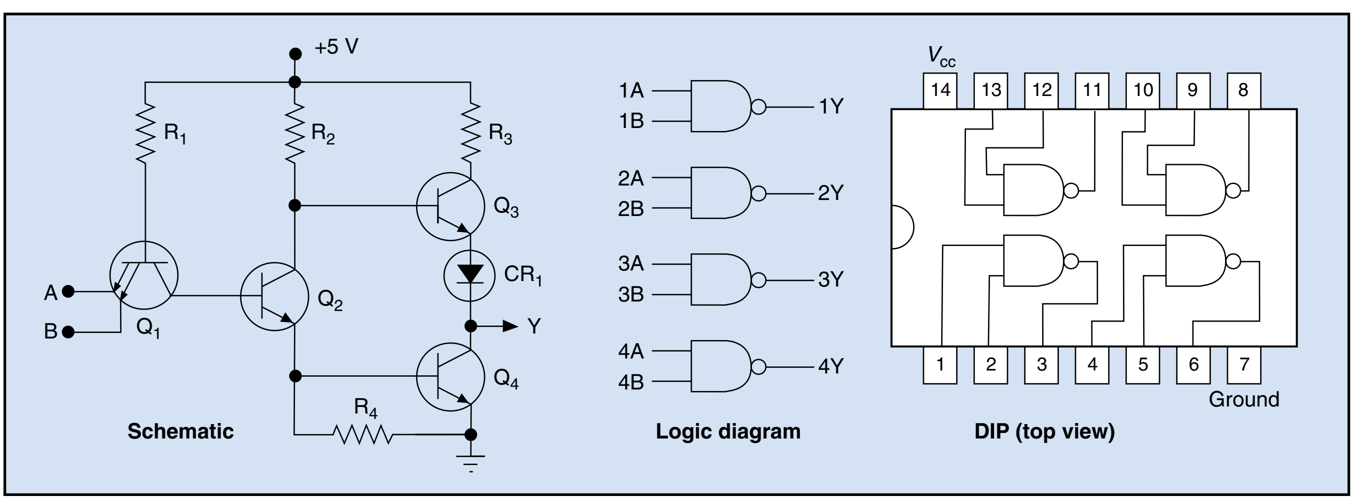
Application 1-2

A technician had to design a circuit that gave a logic HIGH when a pushbutton switch was pressed and a logic LOW when not pressed. This circuit was part of a device that had to switch rapidly without any contact bounce. She decided on the circuit shown below, using two NAND gates in a 7400 IC chip.



When the switch is not pressed, one input of NAND gate A is connected to ground, ensuring that A's output is HIGH. A's output is one input to NAND B. The other input to B is HIGH from +5 V through the

Fig. 1-18. 7400 quad two-input NAND gate



resistor. With both inputs to B HIGH, its output is LOW. Pressing the switch causes one input to A to go HIGH from +5 V through its resistor, and the input to B goes LOW (to ground) through the closed switch. This makes B's output go LOW. Feedback ensures that both of A's inputs change for the correct output.

1.51 Many logic circuits are easier to design at an operating voltage other than +5 V. However, due to the longstanding popularity and widespread use of the 74 series TTL family, many circuits have been designed to run on +5 V and to have logic levels compatible with 74 family logic. Equipment often is specified as having TTL input/output levels. This means that the circuit's logic levels are the same as those of the 74 family. TTL logic levels are defined as follows:

HIGH = logical 1 = 2 to 5 V
LOW = logical 0 = 0 to 0.8 V.

1.52 Several series or subfamilies of the 74 family of TTL logic have been developed over the years in order to meet specific needs. One series (the 74H series) is designed for the purpose of reducing time delays in processing signals caused by stray capacitance and stored charges within the transistors. These delays can be reduced by decreasing circuit resistances. However, the improved performance comes at the cost of higher power dissipation.

1.53 Other systems require low power dissipation, so a low-power family called the 74L series was developed. However, the low-power operation was achieved at the expense of speed.

1.54 The next development in TTL logic (called *Schottky* TTL) operates at twice the speed of the 74H series with about the same power dissipation. This family, known as 74S, uses high-speed Schottky diodes to clamp the transistor collectors to prevent saturation. This circuit is shown in Fig. 1-19. Three other series also are available:

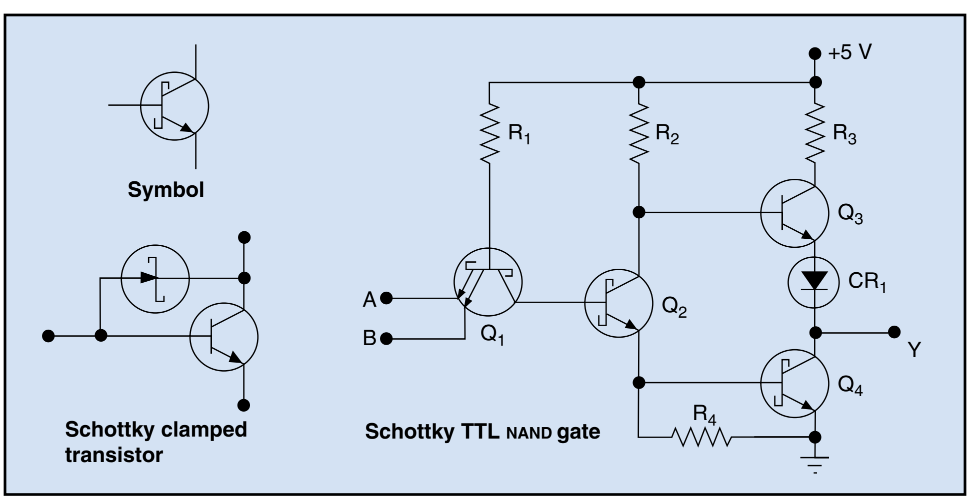
- low-power Schottky (74LS)
- advanced Schottky (74AS)
- advanced low-power Schottky (74ALS).

CMOS Logic

1.55 The most common logic family using unipolar digital ICs as its main components is CMOS, which stands for complementary metal-oxide semiconductor. CMOS logic is most prominent in circuits using large numbers of gates, notably in computers.

1.56 In CMOS, a logical 0 is defined as 0 to 1.5 V, and a logical 1 is 3.5 to 5 V. Input voltage may range from 3 to 18 V, but usually is 5 V if CMOS logic is used in a circuit with TTL logic.

Fig. 1-19. Schottky clamped logic



1-9. In positive logic, HIGH voltage is logical ________ and low voltage is logical ________.	1-9. 1 (ONE); 0 (ZERO) Ref: 1.28
1-10. The term NAND means the same as ________.	1-10. NOT AND Ref: 1.30
1-11. By defining the inpouts to a NAND circuit as active LOW, you produce a circuit that is equivalent to the ________ fuction.	1-11. NOR Ref: 1.32
1-12. You can obtain a not function by tying all but one input of a(n) ________ gate toi an active logic level voltage.	1-12. NAND Ref: 1.33
1-13. Can all logic functions, no matter how complex, be obtained from combinations of NAND gates?	1-13. YES Ref: 1.34
1-14. The equation $Y = A \oplus B$ expresses a(n) ________ gate.	1-14. EXCLUSIVE OR Ref: 1.44, Fig. 1-16
1-15. The inputs of a typical two-input TTL NAND gate are to transistor with multiple ________ rather than to multiple transistors.	1-15. EMITTERS Ref: 1.49
1-16. Schottky clamped logic results in circuits with greater ________.	1-16. SPEED Ref: 1.54

Answer the following questions by marking an "X" in the box next to the best answer.

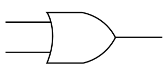
1-1. Which of the following are the three basic Boolean operations?

- ☐ a. AND, NAND, NOR
- ☐ b. AND, OR, NOT
- ☐ c. NAND, NOR, NOT
- ☐ d. OR, NOR, NOT

1-2. The equation Y = AB expresses a(n) _______ function.

- ☐ a. AND
- ☐ b. NAND
- ☐ c. NOR
- ☐ d. OR

1-3. Which logic function does this symbol represent?



- ☐ a. AND
- ☐ b. NAND
- ☐ c. NOR
- ☐ d. OR

1-4. A typical solid-state analog switch is similar to a(n)

- ☐ a. logic switch
- ☐ b. NOT circuit
- ☐ c. OR gate
- ☐ d. relay

1-5. Solid-state analog switches usually use _______, because they pass current in both directions when turned ON.

- ☐ a. DTLs
- ☐ b. FETs
- ☐ c. NOT gates
- ☐ d. OR gates

1-6. Negative logic means that the

- ☐ a. active state is LOW
- ☐ b. output is always 0
- ☐ c. value of the FALSE state is lower than the value of the TRUE state
- ☐ d. voltage signal is negative

1-7. A bubble on the output of a logic symbol means that the output signal is

- ☐ a. always low
- ☐ b. a negative voltage
- ☐ c. disconnected
- ☐ d. inverted

1-8. All logic functions can be obtained from combinations of _______ gates.

- ☐ a. AND
- ☐ b. NAND
- ☐ c. NOT
- ☐ d. OR

1-9. Which of the following signal values for logical 0 and logical 1 are TTL-compatible?

- ☐ a. -400 mV and +400 mV
- ☐ b. -5 V and +5 V
- ☐ c. -3.2 V and +1.3 V
- ☐ d. 0 V and +5 V

1-10. The logic family that is used most often in computers is

- ☐ a. CMOS
- ☐ b. ECL
- ☐ c. 74ALS series TTL
- ☐ d. 74H series TTL

SUMMARY

Digital logic circuits now perform many functions previously performed by mechanical or electromechanical means. These circuits are used in logic systems in which the variables have only one of two values. The only operations are AND, OR (XOR), and NOT (Boolean algebra), which produce one of two possible outputs. The AND function is expressed as Y = A · B or AB, the OR function is expressed as Y = A + B (XOR: Y = A ⊕ B), and the NOT function is expressed as Y = $\overline{A}$.

All three Boolean operations can be performed with various combinations of a single kind of circuit—the NAND (NOT AND) gate—although other logic circuits are also used. In particular, the NOR (NOT OR) gate can be used to construct other logic circuits.

Solid-state switches operate much like relays. Their main components usually are FETs, which pass current in both directions. A logic switch, on the other hand, does not need to pass current in both directions or to reproduce the input signal at its output.

Logic circuits normally are defined in terms of positive logic, in which HIGH (active) voltage is logical 1 and LOW voltage is logical 0. In negative logic, The active voltage is LOW (logical 0). A bubble on an input or output line represents an inverted signal.

The development of digital ICs resulted in logic circuits with higher performance and lower cost. Several families of IC logic are in common use today, each with its own advantages. The TTL logic families are very common, and many other logic families are designed to be TTL-compatible. The CMOS logic family is most common in computers.

Answers to Self-Check Quiz

1-1. b. AND, OR, NOT. Ref: 1.05

1-2. a. AND. Ref: 1.09, Fig. 1-3

1-3. d. OR. Ref: 1.10, Fig. 1-5

1-4. d. Relay. Ref: 1.20

1-5. b. FETs. Ref: 1.21

1-6. a. Active state is LOW. Ref: 1.24

1-7. d. Inverted. Ref: 1.28

1-8. b. NAND. Ref: 1.34

1-9. d. 0 V and +5 V. Ref: 1.51

1-10. a. CMOS. Ref: 1.55