

Figure 3.3 A P-type MOS transistor

0 volts, the transistor acts like an open circuit, breaking the circuit, and causing the bulb to not glow.

Figure 3.2c is a shorthand notation for describing the circuit of Figure 3.2b. Rather than always showing the power supply and the complete circuit, electrical engineers usually show only the terminals of the power supply. The fact that the power supply itself provides the completion of the completed circuit is well understood, and so is not usually shown.

The P-type transistor works exactly the opposite of the N-type transistor. Figure 3.3 shows the schematic representation of a P-type transistor. When the gate is supplied with 0 volts, the P-type transistor acts (more or less) like a piece of wire, closing the circuit. When the gate is supplied with 2.9 volts, the P-type transistor acts like an open circuit. Because the P-type and N-type transistors act in this complementary way, we refer to circuits that contain both P-type and N-type transistors as CMOS circuits, for *complementary metal oxide semiconductor*.

3.2 LOGIC GATES

One step up from the transistor is the logic gate. That is, we construct basic logic structures out of individual MOS transistors. In Chapter 2, we studied the behavior of the AND, the OR, and the NOT functions. In this chapter we construct transistor circuits that implement each of these functions. The corresponding circuits are called AND, OR, and NOT gates.

3.2.1 The NOT Gate (or, Inverter)

Figure 3.4 shows the simplest logic structure that exists in a computer. It is constructed from two MOS transistors, one P-type and one N-type. Figure 3.4a is the schematic representation of that circuit. Figure 3.4b shows the behavior of the circuit if the input is supplied with 0 volts. Note that the P-type transistor conducts and the N-type transistor does not conduct. The output is, therefore, connected to 2.9 volts. On the other hand, if the input is supplied with 2.9 volts, the P-type transistor does not

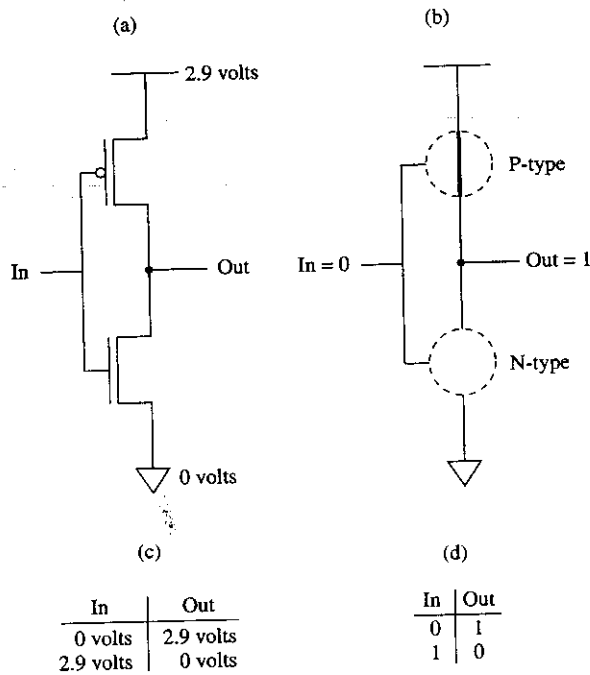


Figure 3.4 A CMOS inverter

conduct, but the N-type transistor does conduct. The output in this case is connected to ground (i.e., 0 volts). The complete behavior of the circuit can be described by means of a table, as shown in Figure 3.4c. If we replace 0 volts by the symbol 0 and 2.9 volts by the symbol 1, we have the truth table (Figure 3.4d) for the complement or NOT function, which we discussed in Chapter 2.

In other words, we have just shown how to construct an electronic circuit that implements the NOT logic function discussed in Chapter 2. We call this circuit a *NOT gate*, or an *inverter*.

3.2.2 OR and NOR Gates

First examine Figure 3.5. Figure 3.5a is a schematic containing two P-type and two N-type transistors.

Figure 3.5b shows the behavior of the circuit if *A* is supplied with 0 volts and *B* is supplied with 2.9 volts. In this case, the lower of the two P-type transistors produces an open circuit, and the output *C* is disconnected from the 2.9-volt power supply. However, the left-most N-type transistor acts like a piece of wire, connecting the output *C* to 0 volts.

Note that if both *A* and *B* are supplied with 0 volts, the two P-type transistors conduct, and the output *C* is connected to 2.9 volts. Note, further, that there is no ambiguity here, since both N-type transistors act as open circuits, and so *C* is disconnected from ground.

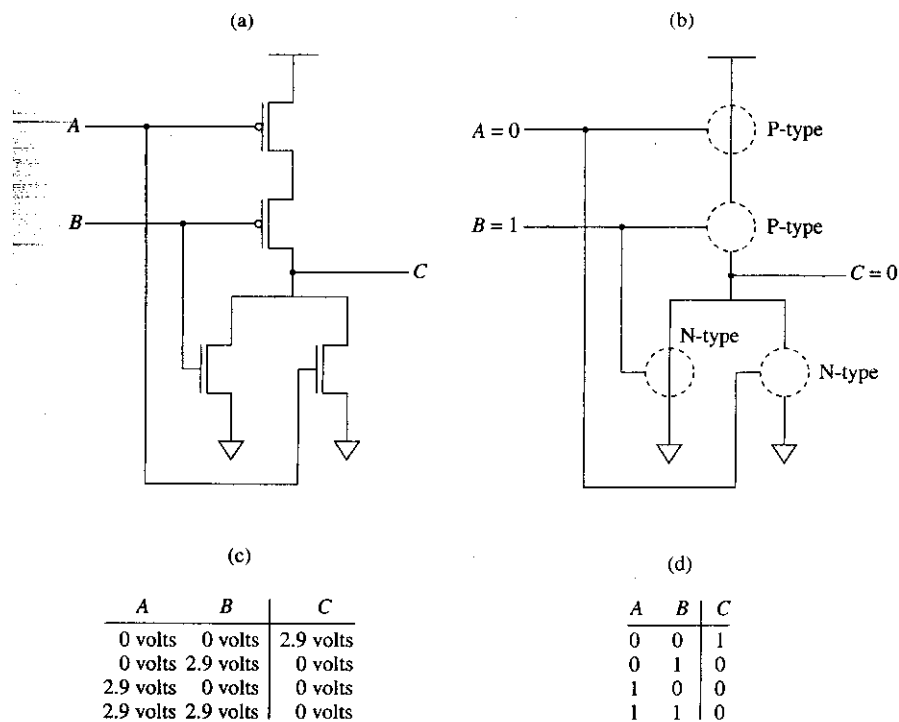


Figure 3.5 The NOR gate

If either A or B is supplied with 2.9 volts, the corresponding P-type transistor results in an open circuit. That is sufficient to break the connection from C to the 2.9-volt source. However, 2.9 volts supplied to the gate of one of the N-type transistors is sufficient to cause that transistor to conduct, resulting in C being connected to ground (i.e., 0 volts).

Figure 3.5c summarizes the complete behavior of the circuit of Figure 3.5a. It shows the behavior of the circuit for each of the four pairs of voltages with which A and B can be supplied. That is,

$$\begin{aligned}
 &A = 0 \text{ volts,} & B = 0 \text{ volts} \\
 &A = 0 \text{ volts,} & B = 2.9 \text{ volts} \\
 &A = 2.9 \text{ volts,} & B = 0 \text{ volts} \\
 &A = 2.9 \text{ volts,} & B = 2.9 \text{ volts}
 \end{aligned}$$

If we replace the voltages with their logical equivalents, we have the truth table of Figure 3.5d. Note that the output C is exactly the opposite of the logical OR function discussed in Chapter 2. In fact, it is the NOT-OR function, more typically abbreviated as NOR. We refer to the circuit that implements the NOR function as a NOR gate.

If we augment the circuit of Figure 3.5a by adding an inverter at the output, as shown in Figure 3.6a, we have at the output D the logical function OR. Figure 3.6a is the circuit for an OR gate. Figure 3.6b describes the behavior of this circuit if the

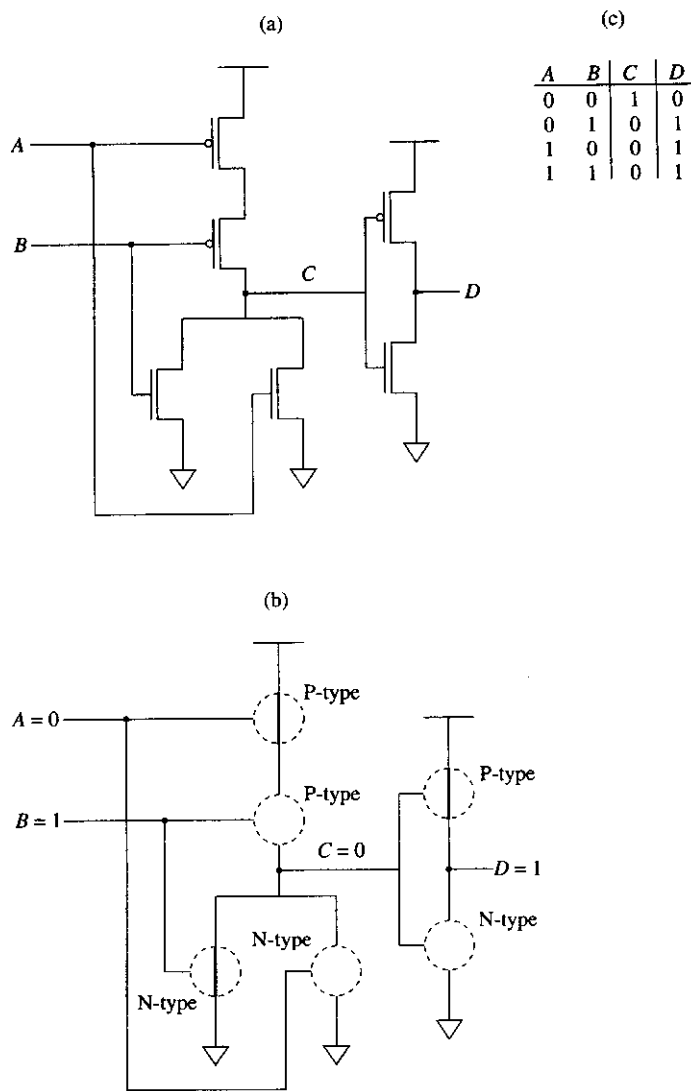


Figure 3.6 The OR gate

input variable A is set to 0 and the input variable B is set to 1. Figure 3.6c shows the circuit's truth table.

3.2.3 AND and NAND Gates

Next, we will examine Figure 3.7. Note that if either A or B is supplied with 0 volts, there is a direct connection from C to the 2.9-volt power supply. The fact that C is at 2.9 volts means the N-type transistor whose gate is connected to C provides a path

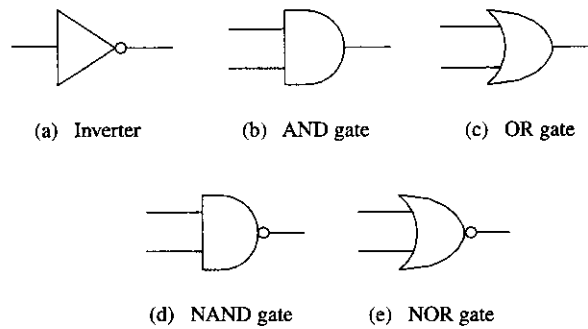


Figure 3.8 Basic logic gates

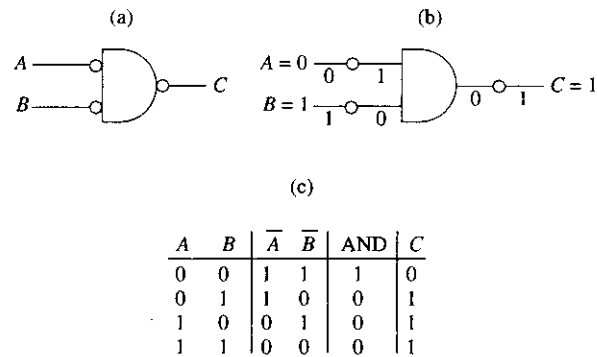


Figure 3.9 DeMorgan's law

3.2.4 DeMorgan's Law

Note (see Figure 3.9a) that one can complement an input before applying it to a gate. Consider the effect on the two-input AND gate if we apply the complements of A and B as inputs to the gate, and also complement the output of the AND gate. The bubbles at the inputs to the AND gate designate that the inputs A and B are complemented before they are used as inputs to the AND gate.

Figure 3.9b shows the behavior of this structure for the input combination $A = 0$, $B = 1$. For ease of representation, we have moved the “bubbles” away from the inputs and the output of the AND gate. That way, we can more easily see what happens to each value as it passes through a bubble.

Figure 3.9c summarizes by means of a truth table the behavior of the logic circuit of Figure 3.9a for all four combinations of input values. Note that the NOT of A is represented as \bar{A} .

(a)			
A	B	C	OUT
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

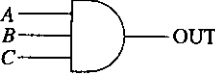
(b)	
A B C	

Figure 3.10 A three-input AND gate

We can describe the behavior of this circuit algebraically:

$$\overline{\overline{A \text{ AND } B}} = A \text{ OR } B$$

This equivalence is known as DeMorgan's law. Is there a similar result if one inverts both inputs to an OR gate, and then inverts the output?

3.2.5 Larger Gates

Before we leave the topic of logic gates, we should note that the notion of AND, OR, NAND, and NOR gates extends to larger numbers of inputs. One could build a three-input AND gate or a four-input OR gate, for example. An n -input AND gate has an output value of 1 only if ALL the input variables have values of 1. If any of the n inputs has a value of 0, the output of the n -input AND gate is 0. An n -input OR gate has an output value of 1 if ANY of the input variables has a value of 1. That is, an n -input OR gate has an output value of 0 only if ALL n -input variables have values of 0.

Figure 3.10 illustrates a three-input AND gate. Figure 3.10a shows its truth table. Figure 3.10b shows the symbol for a three-input AND gate.

Can you draw a transistor-level circuit for a three-input AND gate? How about a four-input OR gate?

3.3 COMBINATIONAL LOGIC STRUCTURES

Now that we understand the workings of the basic logic gates, the next step is to build some of the logic structures that are important components of the microarchitecture of a computer.

There are fundamentally two kinds of logic structures, those that include the storage of information and those that do not. In Sections 3.4 and 3.5, we will deal with structures that store information. In this section, we will deal with those that do not. These structures are sometimes referred to as *decision elements*. Usually, they