synthesis and optimization. The designer needs to specify only the desired functional behavior, and the CAD software generates a cost-effective network that implements the required functionality.

A.5 PRACTICAL IMPLEMENTATION OF LOGIC GATES

Let us now turn our attention to the means by which logic variables can be represented and logic functions can be implemented in practice. The choice of a physical parameter to represent logic variables is obviously technology-dependent. In electronic circuits, either voltage or current levels can be used for this purpose.

To establish a correspondence between voltage levels and logic values or states, the concept of a *threshold* is used. Voltages above a given threshold are taken to represent one logic value, with voltages below that threshold representing the other. In practical situations, the voltage at any point in an electronic circuit undergoes small random variations for a variety of reasons. Because of this "noise," the logic state corresponding to a voltage level near the threshold cannot be reliably determined. To avoid such ambiguity, a "forbidden range" should be established, as shown in Figure A.10. In this case, voltages below $V_{0,max}$ represent the 0 value, and voltages above $V_{1,min}$ represent the 1 value. In subsequent discussion, we will often use the terms "low" and "high" to represent the voltage levels corresponding to logic values 0 and 1, respectively.

We will begin our discussion of electronic circuits that implement basic logic functions by considering simple circuits consisting of resistors and transistors that act as switches. Consider the circuits in Figure A.11. When switch S in Figure A.11*a* is closed, the output voltage V_{out} is equal to 0 (ground). When S is open, the output voltage V_{out} is equal to

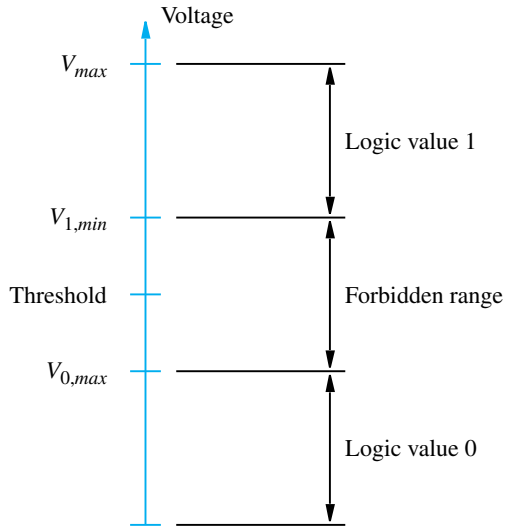


Figure A.10 Representation of logic values by voltage levels.

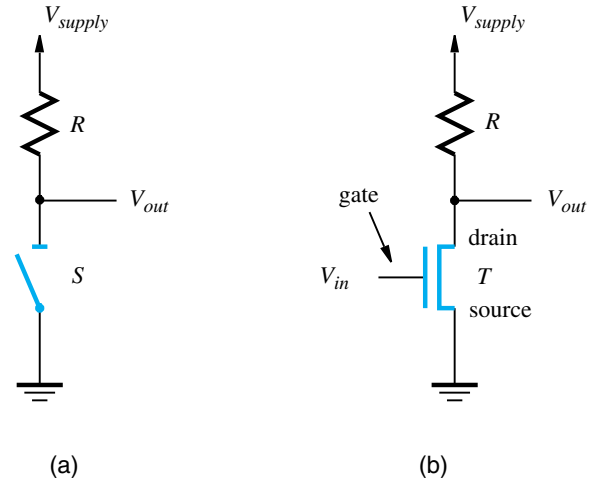


Figure A.11 An inverter circuit.

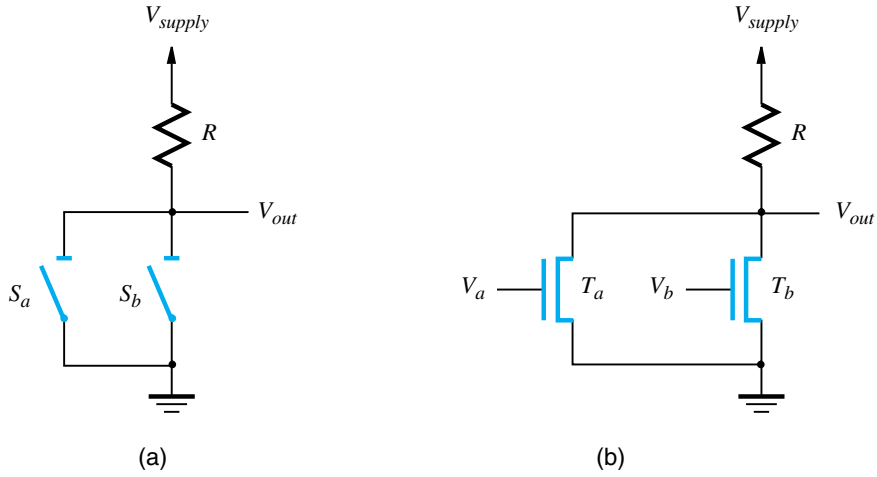


Figure A.12 A transistor circuit implementation of a NOR gate.

the supply voltage, V_{supply}. The same effect can be obtained in Figure A.11*b*, in which a transistor T is used to replace the switch S. When the input voltage applied to the gate of the transistor is 0 (that is, when $V_{in} = 0$), the transistor functions as an open switch, and $V_{out} = V_{supply}$. When V_{in} changes to V_{supply}, the transistor acts as a closed switch and the output voltage V_{out} is very close to 0. Thus, the circuit performs the function of a logic NOT gate.

We can now discuss the implementation of more complex logic functions. Figure A.12 shows a circuit realization for a NOR gate. In this case, V_{out} in Figure A.12*a* is high only when both switches S_a and S_b are open. Similarly, V_{out} in Figure A.12*b* is high only when

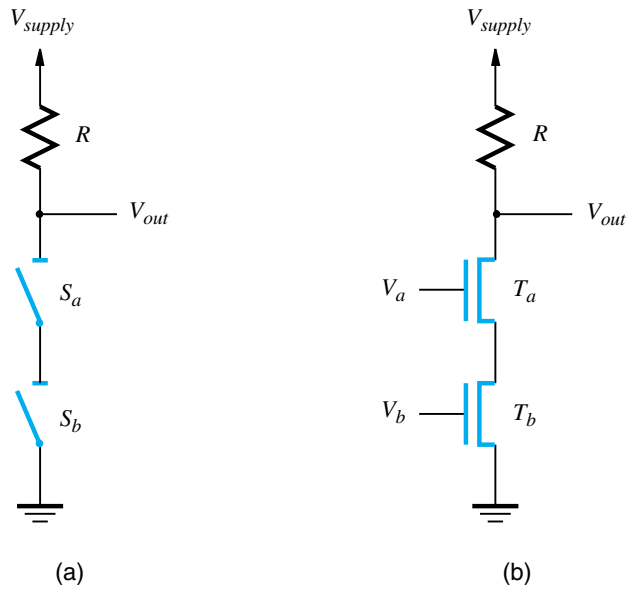


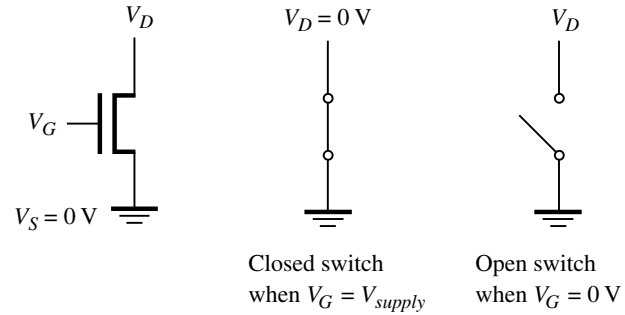
Figure A.13 A transistor circuit implementation of a NAND gate.

both inputs V_a and V_b are low. Thus, the circuit is equivalent to a NOR gate in which V_a and V_b correspond to two logic variables x_1 and x_2. We can easily verify that a NAND gate can be obtained by connecting the transistors in series as shown in Figure A.13. The logic functions AND and OR can be implemented using NAND and NOR gates, respectively, followed by the inverter of Figure A.11.

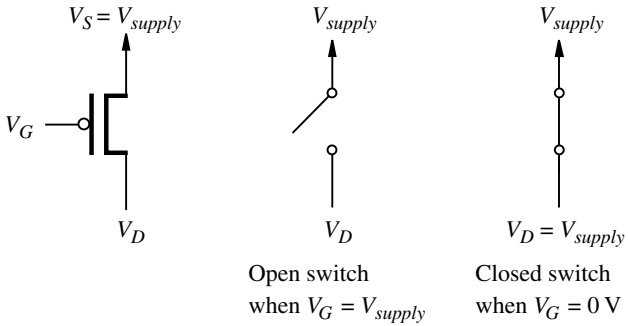
Note that NAND and NOR gates are simpler in their circuit implementations than AND and OR gates. Hence, it is not surprising to find that practical realizations of logic functions use NAND and NOR gates extensively. Many of the examples given in this book show circuits consisting of AND, OR, and NOT gates for ease of understanding. In practice, logic circuits contain all five types of gates.

A.5.1 CMOS CIRCUITS

Figures A.11 through A.13 illustrate the general structure of circuits implemented using *NMOS technology*. The name derives from the fact that the transistors used to realize the logic functions are of NMOS type. Two types of *metal-oxide semiconductor* (MOS) transistors are available for use as switches. An n-channel transistor is said to be of NMOS-type, and it behaves as a closed switch when its gate input is raised to the positive power supply voltage, V_{supply}, as indicated in Figure A.14*a*. The opposite behavior is achieved with a p-channel transistor, which is said to be of PMOS type. It acts as an open switch when the gate voltage, V_G, is equal to V_{supply}, and as a closed switch when $V_G = 0$, as indicated in Figure A.14*b*. Note that the graphical symbol for a PMOS transistor has a bubble on the gate input to indicate that its behavior is complementary to that of an NMOS transistor. Note



(a) NMOS transistor



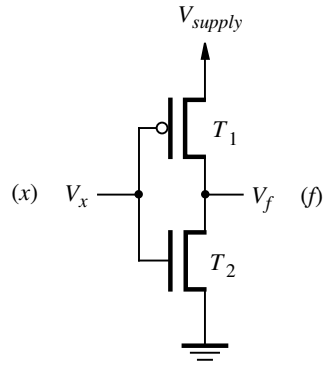
(b) PMOS transistor

Figure A.14 NMOS and PMOS transistors in logic circuits.

also that the names source and drain are associated with the opposite terminals of PMOS transistors in comparison with NMOS transistors. The source of an NMOS transistor is connected to ground, while the source of a PMOS transistor is connected to V_{supply}. This naming convention is due to the nature of the current that flows through these transistors.

A drawback of the circuits in Figures A.11 through A.13 is their power consumption. In the state in which the switches are closed to provide a path between ground and the pull-up resistor R, there is current flowing from V_{supply} to ground. In the opposite state, in which switches are open, there is no path to ground and there is no current flowing. (In MOS transistors no current flows through the gate terminal.) Thus, depending on the states of its gates, there may be significant power consumption in a logic circuit.

An effective solution to the power consumption problem lies in using both NMOS and PMOS transistors to implement circuits that do not dissipate power when in a steady state. This approach leads to the CMOS (complementary metal-oxide semiconductor) technology. The basic idea of CMOS circuits is illustrated by the inverter circuit in Figure A.15. When $V_x = V_{supply}$, which corresponds to the input x having the logic value 1, transistor T_1 is



(a) Circuit

x	V_x	T_1	T_2	V_f	f
0	low	on	off	high	1
1	high	off	on	low	0

(b) Truth table and transistor states

Figure A.15 CMOS realization of a NOT gate.

turned off and T_2 is turned on. Thus, T_2 pulls the output voltage V_f down to 0. When V_x changes to 0, transistor T_1 turns on and T_2 turns off. Thus, T_1 pulls the output voltage V_f up to V_{supply}. Therefore, the logic values of x and f are complements of each other, and the circuit implements a NOT gate.

A key feature of this circuit is that transistors T_1 and T_2 operate in a complementary fashion; when one is on, the other is off. Hence, there is always a closed path from the output point f to either V_{supply} or ground. But, there is no closed path between V_{supply} and ground at any time except during a very short transition period when the transistors are changing their states. This means that the circuit does not dissipate appreciable power when it is in a steady state. It dissipates power only when it is switching from one logic state to another. Therefore, power dissipation in this circuit is dependent on the rate at which state changes take place.

We can now extend the CMOS concept to circuits that have n inputs, as shown in Figure A.16. NMOS transistors are used to implement the pull-down network, such that a closed path is established between the output point f and ground when a desired function $F(x_1, \ldots, x_n)$ is equal to 0. The pull-up network is built with PMOS transistors, such that a closed path is established between the output f and V_{supply} when $F(x_1, \ldots, x_n)$ is equal to 1. The pull-up and pull-down networks are functional complements of each other, so that in steady state there exists a closed path only between the output f and either V_{supply} or ground, but not both.

The pull-down network is implemented in the same way as shown in Figures A.11 through A.13. Figure A.17 gives the implementation of a NAND gate, and Figure A.18 gives a NOR gate. Figure A.19 shows how an AND gate is realized by inverting the output of a NAND gate.

In addition to low power dissipation, CMOS circuits have the advantage that MOS transistors can be implemented in very small sizes and thus occupy a very small area on an integrated circuit chip. This results in two significant benefits. First, it is possible to fabricate chips containing billions of transistors, which has led to the realization of modern

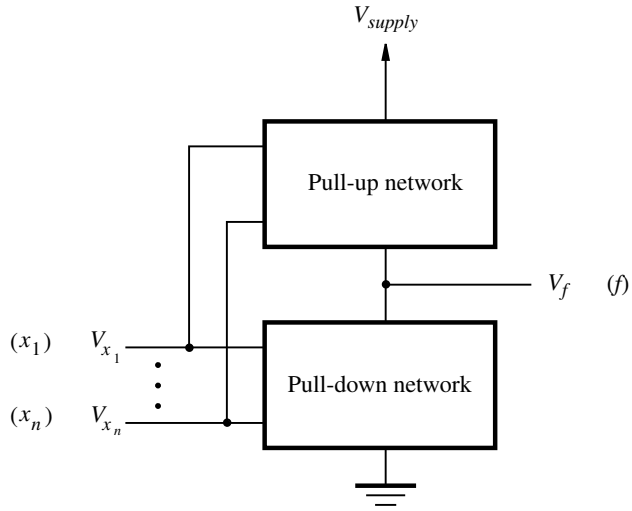
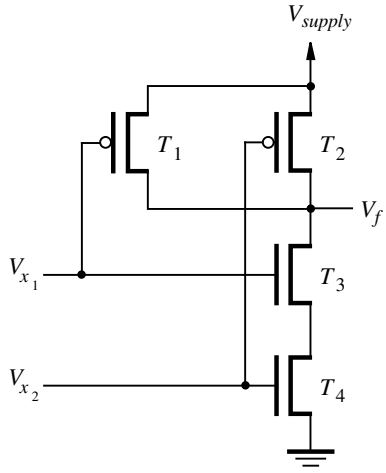


Figure A.16 Structure of a CMOS circuit.



(a) Circuit

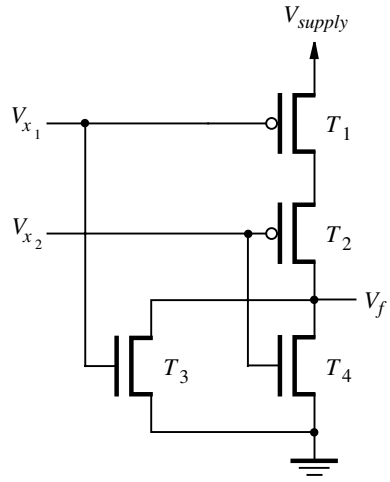
x_1	x_2	T_1	T_2	T_3	T_4	f
0	0	on	on	off	off	1
0	1	on	off	off	on	1
1	0	off	on	on	off	1
1	1	off	off	on	on	0

(b) Truth table and transistor states

Figure A.17 CMOS realization of a NAND gate.

processors and large memory chips. Second, the smaller the transistor, the faster it can be switched from one state to another. Thus, CMOS circuits can be operated at speeds in the gigahertz range.

Different CMOS circuits have been developed to operate with power supply voltages up to 15 V. The most commonly used power supplies are in the range from 1 V to 5 V.



(a) Circuit

x_1	x_2	T_1	T_2	T_3	T_4	f
0	0	on	on	off	off	1
0	1	on	off	off	on	0
1	0	off	on	on	off	0
1	1	off	off	on	on	0

(b) Truth table and transistor states

Figure A.18 CMOS realization of a NOR gate.

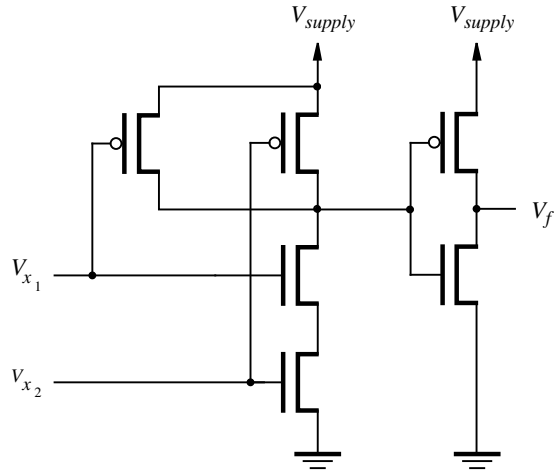


Figure A.19 CMOS realization of an AND gate.

Circuits that use lower power supply voltages dissipate much less power (power dissipation is proportional to V^2_{supply}), which means that more transistors can be placed on a chip without causing overheating. A drawback of lower power supply voltage is reduced noise immunity.

Transitions between low and high signal levels in a CMOS inverter are illustrated in more detail in Figure A.20. The blue curve, known as the *transfer characteristic*, shows the output voltage as a function of the input voltage. It indicates that a rather sharp transition

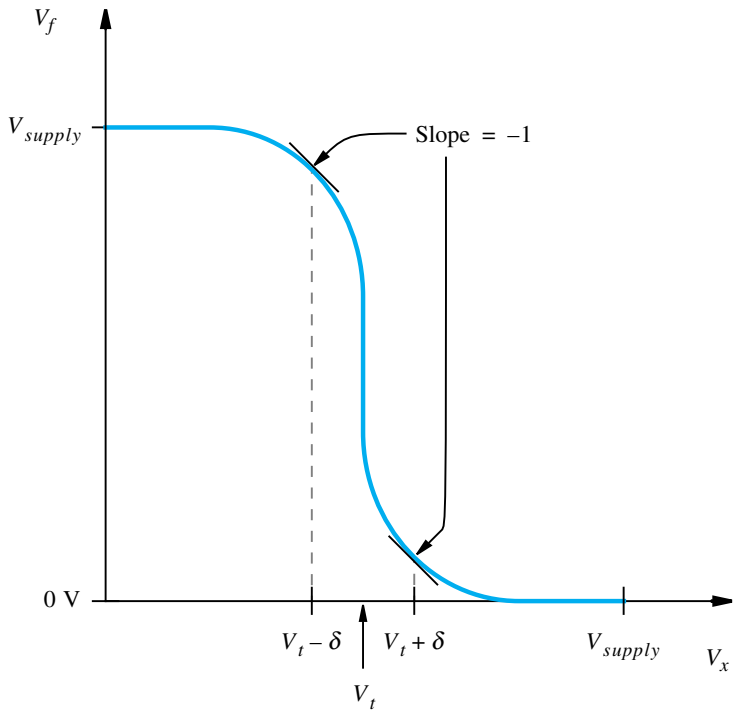


Figure A.20 The voltage transfer characteristic for the CMOS inverter.

in output voltage takes place when the input voltage passes through the value of about $V_{supply}/2$. There is a *threshold* voltage, V_t, and a small value δ such that $V_{out} \approx V_{supply}$ if $V_{in} < V_t - \delta$ and $V_{out} \approx 0$ if $V_{in} > V_t + \delta$. This means that the input signal need not be exactly equal to the nominal value of either 0 or V_{supply} to produce the correct output signal. There is room for some error, called *noise,* in the input signal that will not cause adverse effects. The amount of noise that can be tolerated is called the *noise margin*. This margin is $V_{supply} - (V_t + \delta)$ volts when the logic value of the input is 1, and it is $V_t - \delta$ when the logic value of the input is 0. CMOS circuits have excellent noise margins.

In this section, we have introduced the basic features of CMOS circuits. For a more detailed discussion of this technology the reader may consult References [1] and [8].

A.5.2 PROPAGATION DELAY

Logic circuits do not switch instantaneously from one state to another. Speed is measured by the rate at which state changes can take place. A related parameter is *propagation delay,* which is defined in Figure A.21. When a state change takes place at the input, a delay is encountered before the corresponding change at the output is observed. This propagation delay is usually measured between the 50-percent points of the transitions, as shown in the

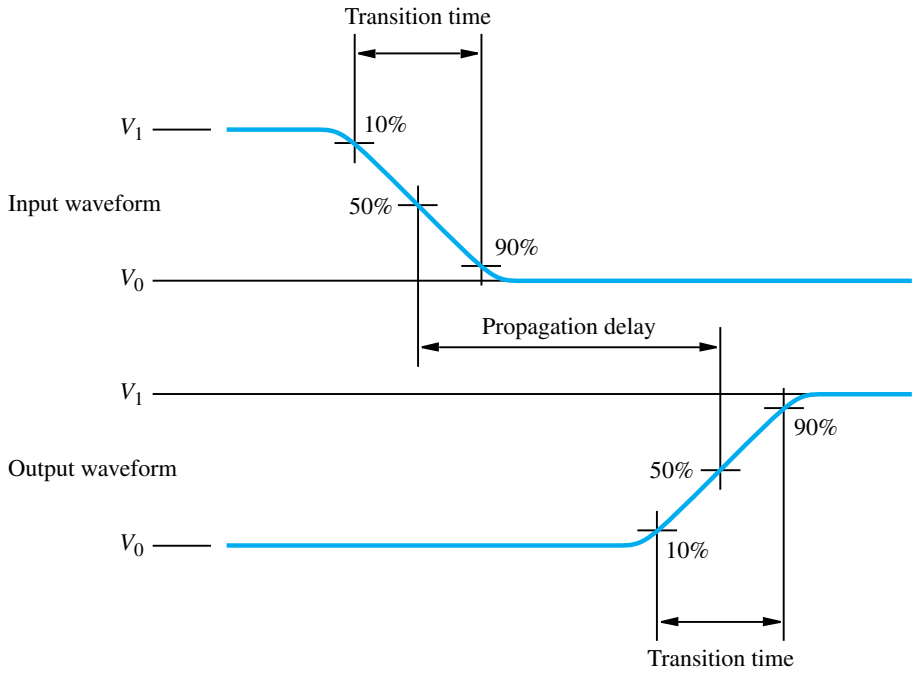


Figure A.21 Definition of propagation delay and transition time.

figure. Another important parameter is the *transition time,* which is normally measured between the 10- and 90-percent points of the signal swing, as shown. The maximum speed at which a logic circuit can be operated decreases as the propagation delay through different paths within that circuit increases. The delay along any path in a logic circuit is the sum of individual gate delays along this path.

A.5.3 FAN-IN AND FAN-OUT CONSTRAINTS

The number of inputs to a logic gate is called its *fan-in*. The number of gate inputs that the output of a logic gate drives is called its *fan-out*. Practical circuits do not allow large fan-in and fan-out because they both have an adverse effect on the propagation delay and hence the speed of the circuit.

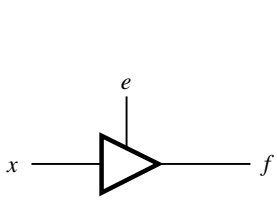
Each transistor in a CMOS gate contributes a certain amount of capacitance. As the capacitance increases, the circuit becomes slower and its signal levels and noise margins become worse. Therefore, it is necessary to limit the fan-in and fan-out, typically to a number less than ten. If the number of desired inputs exceeds the maximum fan-in, it is necessary to use an additional gate of the same type. Figure A.9*a* shows how two gates of the same type can be cascaded. If the number of outputs that have to be driven by a particular gate exceeds the acceptable fan-out, it is possible to use two copies of that gate.

A.5.4 TRI-STATE BUFFERS

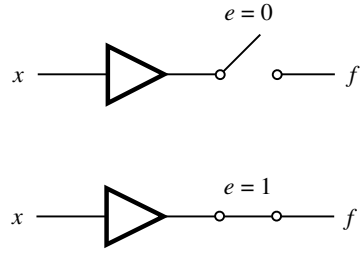
In the logic gates discussed so far, it is not possible to connect the outputs of two gates together. This would make no sense from the logic point of view because if one gate generated an output value of 1 and the other an output of 0, it would be uncertain what the combined output signal would be. More importantly, in CMOS circuits, the gate that generates the output of 1 establishes a direct path from the output terminal to V_{supply}, while the gate that generates 0 establishes a path to ground. Thus, the two gates would provide a short circuit across the power supply, which would damage the gates.

Yet, in the design of computer systems, there are many cases where an input signal to a circuit may be derived from one of a number of different sources. This can be done using multiplexer logic circuits, which are discussed in Section A.10. It can also be done using special gates called *tri-state buffers*. A tri-state buffer has three states. Two of the states produce the normal 0 and 1 signals. The third state places the output terminal of the buffer into a high-impedance state in which the output is electrically disconnected from the input it is supposed to drive.

Figure A.22 depicts a tri-state buffer. The buffer has two inputs and one output. The *enable* input, e, controls the operation of the buffer. When $e = 1$, the output f has the same logic value as the input x. When $e = 0$, the output is placed in the high-impedance state, Z. An equivalent circuit is shown in Figure A.22*b*. The triangular symbol in this figure represents a noninverting driver. This is a circuit that performs no logic operation



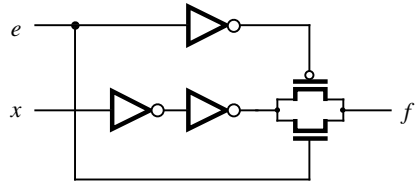
(a) Symbol



(b) Equivalent circuit

e	x	f
0	0	Z
0	1	Z
1	0	0
1	1	1

(c) Truth table



(d) Implementation

Figure A.22 Tri-state buffer.

because its output merely replicates the input signal. Its purpose is to provide additional electrical driving capability. When combined with the output switch shown in the figure, it behaves according to the truth table given in Figure A.22*c*. This table describes the required tri-state behavior. Figure A.22*d* shows a circuit implementation of the tri-state buffer. One NMOS and one PMOS transistor are connected in parallel to implement the switch, which is connected to the output of the driver. Because the two transistor types require complementary control signals at their gate inputs, an inverter is used as shown. When $e = 0$, both transistors are turned off, resulting in an open switch. When $e = 1$, both transistors are turned on, resulting in a closed switch.

The driver circuit may be required to drive a number of inputs of other gates whose combined capacitance exceeds the drive capability of an ordinary logic gate circuit. To provide a sufficient drive capability, the driver circuit needs larger transistors. Hence, the two cascaded NOT gates that realize the driver are implemented with transistors of larger size than in regular logic gates.

The reader may wonder why is it necessary to use the PMOS transistor in the output switch because from the logic function point of view the same behavior could be achieved using just the NMOS transistor. The reason is that these transistors have to "pass" the logic value generated by the driver circuit to the output f, and it turns out that NMOS transistors pass the logic value 0 well but the logic value 1 poorly, while PMOS transistors pass 1 well and 0 poorly. The parallel arrangement of NMOS and PMOS transistors passes both 1s and 0s well. For a more detailed discussion of this issue and tri-state buffers in general, the reader may consult Reference [1].

A.6 FLIP-FLOPS

The majority of applications of digital logic require the storage of information. For example, a circuit that controls a combination lock must remember the sequence in which the digits are dialed in order to determine whether to open the lock. Another important example is the storage of programs and data in the memory of a digital computer.

The basic electronic element for storing binary information is called a *latch*. Consider the two cross-coupled NOR gates in Figure A.23*a*. Let us examine this circuit, starting with the situation in which R = 1 and S = 0. Simple analysis shows that $Q_a = 0$ and $Q_b = 1$. Under this condition, both inputs to gate G_a are equal to 1. Thus, if R is changed to 0, no change will take place at the outputs Q_a and Q_b. If S is set to 1 with R equal to 0, Q_a and Q_b will become 1 and 0, respectively, and will remain in this state after S is returned to 0. Hence, this logic circuit constitutes a memory element, or a latch, that remembers which of the two inputs S and R was most recently equal to 1. A truth table for this latch is given in Figure A.23*b*. Some typical waveforms that characterize the latch are shown in Figure A.23*c*. The arrows in Figure A.23*c* indicate the cause-effect relationships among the signals. Note that when the R and S inputs change from 1 to 0 at the same time, the resulting state is undefined. In practice, the latch will assume one of its two stable states at random. The input valuation R = S = 1 is not used in most applications of such latches.

Because of the nature of the operation of the preceding circuit, the S and R lines are referred to as the *set* and *reset* inputs. Since the valuation R = S = 1 is normally not used,