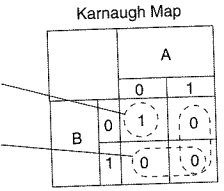
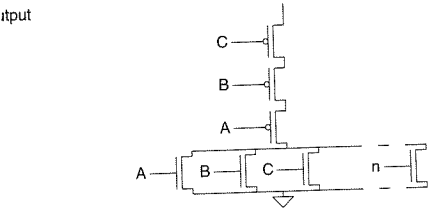
S

CMOS NOR Gate Truth Table

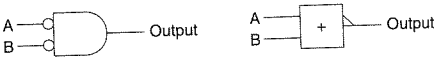
	A INPUT 0	A INPUT 1
	Z(p)	
Z(n)		**0**(n)
	Z(p)	
0(n)		**0**(n)

is used in most complementary CMOS logic gates
ynamic gates or static gates that dissipate static
id switch levels are shown in Table 1.8. By inspec-
s implements the NOR function.
NOR gate schematic is shown in Fig. 1.7(a). It is
ntroduced in Fig. 1.5(b) and Fig. 1.5(c), according





(d)



to the Karnaugh map in Table 1.7. Note that the N- and P-SWITCH combinations are the dual or complement of the combination for the NAND gate. In contrast to the NAND gate, extra inputs are accommodated in the NOR structure by adding N-SWITCHES in parallel and P-SWITCHES in series with the corresponding switch structures (Fig. 1.7d).

Some further points may be noted from this example. First, note that for all inputs there is always a path from '1' or '0' (V_{DD} or V_{SS} supplies) to the output and that the full supply voltages appear at the output. The latter feature leads to a *fully restored* logic family. This simplifies the circuit design considerably. In comparison with other forms of logic, where the pull-up and pull-down switch transistors have to be ratioed in some manner, the transistors in the CMOS gate do not have to be ratioed for the gate to function correctly. Second, there is never a path from the '1' to the '0' supplies for any combination of inputs (in contrast to single channel MOS, GaAs, or bipolar technologies). As we will learn in subsequent chapters, this is the basis for the low static power dissipation in CMOS.

1.5.5 Compound Gates

A compound gate is formed by using a combination of series- and parallel-switch structures. For example, the derivation of the switch connection diagram for the function $F = \overline{((A.B) + (C.D))}$ is shown in Fig. 1.8. The decomposition of this function and generation of the diagram may be approached as follows. For the n-side, take the uninverted expression $((A.B) + (C.D))$. The AND expressions $(A.B)$ and $(C.D)$ may be implemented by series connections of switches, as shown in Fig. 1.8(a). Now, taking these as subswitches and ORing the result requires the parallel connection of these two structures. This is shown in Fig. 1.8(b). For the p-side we invert the expression used for the n-expansion, yielding $((A + B) . (C + D))$. This suggests two OR structures, which are subsequently connected in series. This progression is evident in Fig. 1.8(c) and Fig. 1.8(d). The final step requires connecting one end of the p-structure to '1' (V_{DD}) and the other to the output. One side of the n-structure is connected to '0' (V_{SS}) and the other to the output in common with the p-structure. This yields the final connection diagram (Fig. 1.8e). The schematic icon is shown in Fig. 1.8(f), which shows that this gate may be used in a 2-input multiplexer. If $C = \overline{B}$, then $F = A$ if B is true, while $F = \overline{D}$ if B is false.

The Karnaugh map for a second function $F = \overline{((A + B + C).\ D)}$ is shown in Fig. 1.9(a). The subfunction $(A + B + C)$ is implemented as three parallel N-SWITCHES. This structure is then placed in series with an N-SWITCH with D on the input. The p-function is $(\overline{D} + \overline{A}.\overline{B}.\overline{C})$ (Fig. 1.9b). This requires three P-SWITCHES in series connected in turn in parallel with a P-SWITCH with D on the input. The completed gate is shown in Fig. 1.9(c). In general, CMOS gates may be implemented by analyzing the relevant

TABLE 1.8 2-input CMOS NOR Gate Truth Table

OUTPUT		A INPUT 0	A INPUT 1
B INPUT	0	**1**(p) Z(n)	Z(p) **0**(n)
	1	Z(p) **0**(n)	Z(p) **0**(n)

n-structure. This property is used in most complementary CMOS logic gates (but not necessarily in dynamic gates or static gates that dissipate static power). The truth table and switch levels are shown in Table 1.8. By inspection, one may see that this implements the NOR function.

The resulting 2-input NOR gate schematic is shown in Fig. 1.7(a). It is composed from sections introduced in Fig. 1.5(b) and Fig. 1.5(c), according

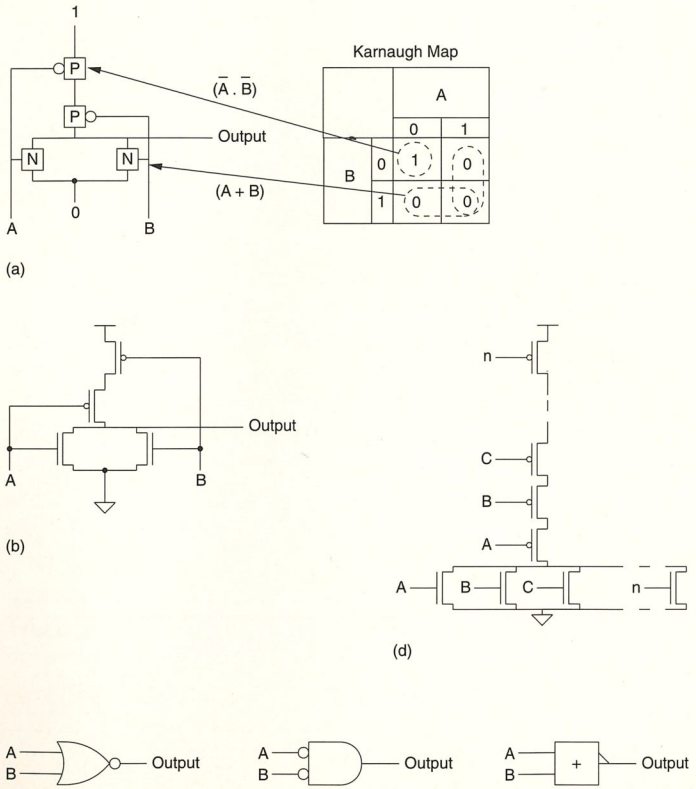


FIGURE 1.7 A CMOS NOR

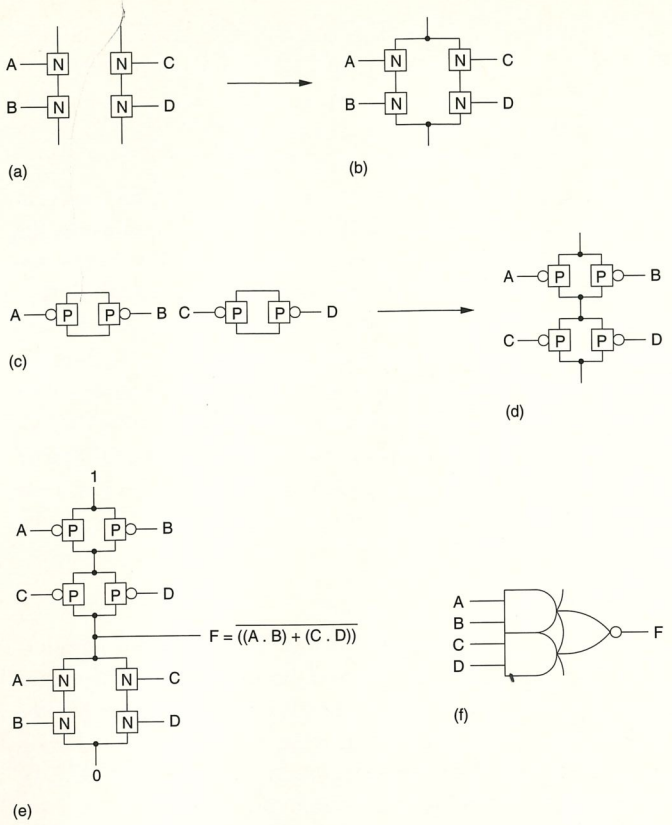


FIGURE 1.8 CMOS compound gate for function $F = \overline{((A.B) + (C.D))}$

Karnaugh map for both n- and p-logic structures and subsequently generating the required series and parallel combinations of transistors.

Often the function required might require the output of the gate to be inverted or one or more of the inputs to be inverted. For instance, if you required a 4-input AND gate, you could implement this with a 4-input NAND

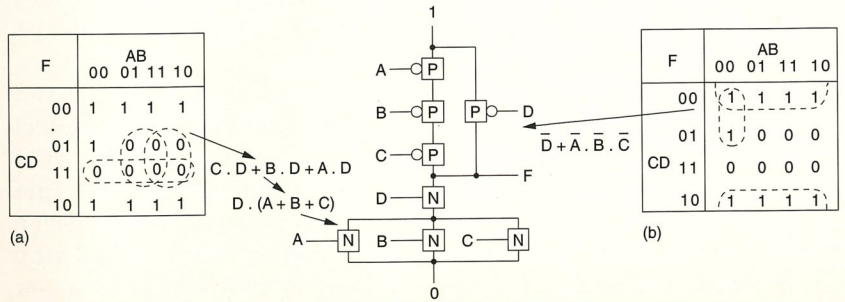
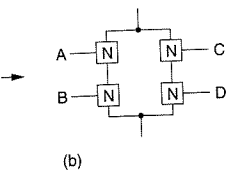
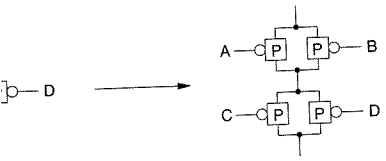


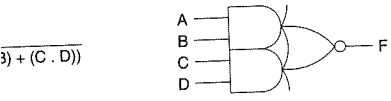
FIGURE 1.9 CMOS compound gate for function



(b)

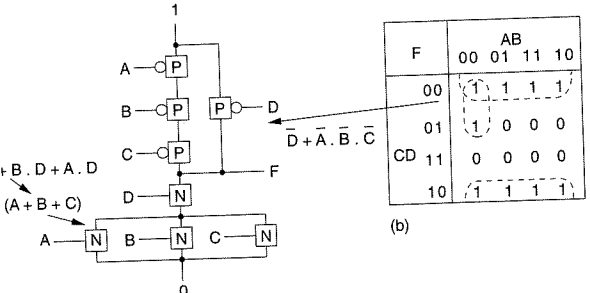


(d)



(f)

n- and p-logic structures and subsequently generat-
ınd parallel combinations of transistors.
required might require the output of the gate to be
e of the inputs to be inverted. For instance, if you
gate, you could implement this with a 4-input NAND



F	AB 00	01	11	10
CD 00	1	1	1	1
01	1	0	0	0
11	0	0	0	0
10	1	1	1	1

(b)

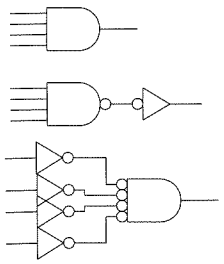


FIGURE 1.10 Various implementations of a CMOS 4-input AND gate

gate and an inverter, or by DeMorgan's theorem, one 4-input NOR gate and four input inverters. Figure 1.10 shows these options. Obviously, in isolation, the former is the most compact implementation. In a larger logic system one may optimize the gates depending on the speed and density required.

Exercises

1. Design CMOS logic gates for the following functions:
   - **a.** $Z = \overline{A.B.C.D}$
   - **b.** $Z = \overline{A + B + C + D}$
   - **c.** $Z = \overline{((A.B.C) + D)}$
   - **d.** $Z = \overline{(((A.B) + C).D)}$
   - **e.** $Z = \overline{(A.B) + C.(A + B)}$
2. Use a combination of CMOS gates to generate the following functions:
   - **a.** $Z = A$ (buffer)
   - **b.** $Z = A.\ \overline{B} + \overline{A}.B$ (XOR)
   - **c.** $Z = A.B + \overline{A}\ .\ \overline{B}$ $(XNOR)$
   - **d.** $Z = A.\ \overline{B}.\overline{C} + \overline{A}.\overline{B}\ .C + \overline{A}.\overline{C}\ .B + A.B.C$

   (SUM function in binary adder)
3. Design the following logic functions:
   - **a.** A 2:4 decoder defined by
     $Z0 = \overline{A0}\ .\ \overline{A1}$
     $Z1 = A0.\ \overline{A1}$
     $Z2 = \overline{A0}\ .A1$
     $Z3 = A0.A1$
   - **b.** A 3:2 priority encoder defined by
     $Z0 = \overline{A0}\ .(A1 + \overline{A2})$
     $Z1 = \overline{A0}\ .\ \overline{A1}$

1.5.6 Multiplexers

Complementary switches may be used to select between a number of inputs,