

4.11 MULTIPLEXERS

A multiplexer is a combinational circuit that selects binary information from one of many input lines and directs it to a single output line. The selection of a particular input line is controlled by a set of selection lines. Normally, there are 2^n input lines and n selection lines whose bit combinations determine which input is selected.

A two-to-one-line multiplexer connects one of two 1-bit sources to a common destination, as shown in Fig. 4.31. The circuit has two data input lines, one output line, and one selection line S . When $S = 0$, the upper AND gate is enabled and I_0 has a path to the output. When $S = 1$, the lower AND gate is enabled and I_1 has a path to the output. The multiplexer acts like an electronic switch that selects one of two sources. The block diagram of a multiplexer is sometimes depicted by a wedge-shaped symbol, as shown in Fig. 4.31(b). It suggests visually how a selected one of multiple data sources is directed into a single destination. The multiplexer is often labeled “MUX” in block diagrams.

A four-to-one-line multiplexer is shown in Fig. 4.32. Each of the four inputs, I_0 through I_3 , is applied to one input of an AND gate. Selection lines S_1 and S_0 are decoded to select a particular AND gate. The outputs of the AND gates are applied to a single OR gate that provides the one-line output. The function table lists the input that is passed to the output for each combination of the binary selection values. To demonstrate the operation of the circuit, consider the case when $S_1S_0 = 10$. The AND gate associated with input I_2 has two of its inputs equal to 1 and the third input connected to I_2 . The other three AND gates have at least one input equal to 0, which makes their outputs equal to 0. The output of the OR gate is now equal to the value of I_2 , providing a path from the selected input to the output. A multiplexer is also called a *data selector*, since it selects one of many inputs and steers the binary information to the output line.

The AND gates and inverters in the multiplexer resemble a decoder circuit, and indeed, they decode the selection input lines. In general, a 2^n -to-1-line multiplexer is constructed from an n -to- 2^n decoder by adding 2^n input lines to it, one to each AND gate. The outputs of the AND gates are applied to a single OR gate. The size of a

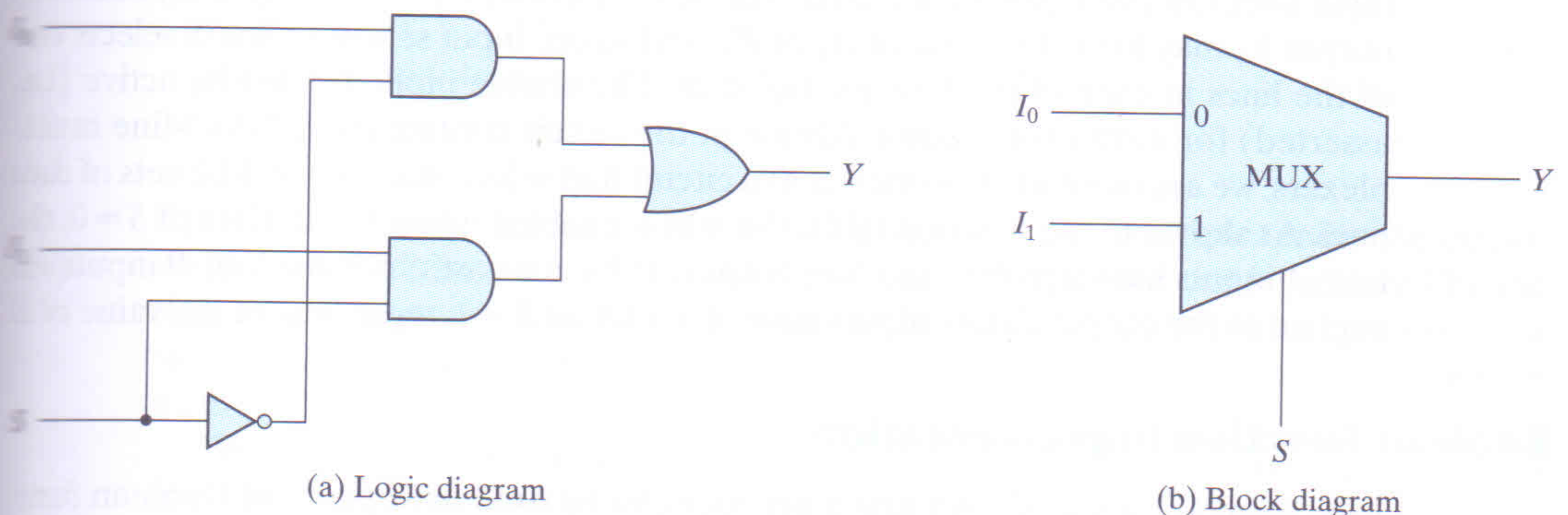


FIGURE 4.31

Two-to-one-line multiplexer

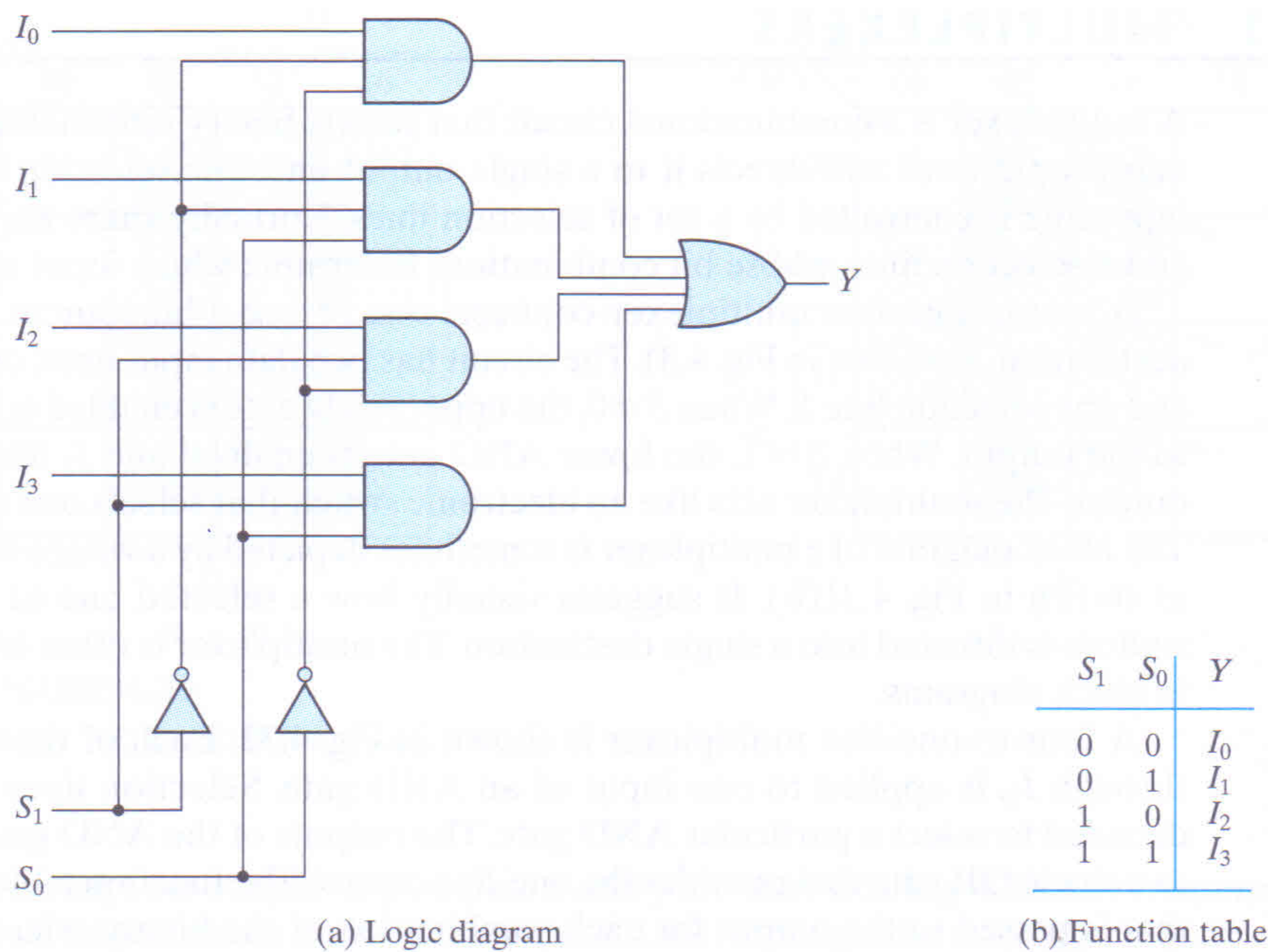


FIGURE 4.32
Four-to-one-line multiplexer

multiplexer is specified by the number 2^n of its data input lines and the single output line. The n selection lines are implied from the 2^n data lines. As in decoders, multiplexers may have an enable input to control the operation of the unit. When the enable input is in the inactive state, the outputs are disabled, and when it is in the active state, the circuit functions as a normal multiplexer.

Multiplexer circuits can be combined with common selection inputs to provide multiple-bit selection logic. As an illustration, a quadruple 2-to-1-line multiplexer is shown in Fig. 4.33. The circuit has four multiplexers, each capable of selecting one of two input lines. Output Y_0 can be selected to come from either input A_0 or input B_0 . Similarly, output Y_1 may have the value of A_1 or B_1 , and so on. Input selection line S selects one of the lines in each of the four multiplexers. The enable input E must be active (i.e., asserted) for normal operation. Although the circuit contains four 2-to-1-line multiplexers, we are more likely to view it as a circuit that selects one of two 4-bit sets of data lines. As shown in the function table, the unit is enabled when $E = 0$. Then, if $S = 0$, the four A inputs have a path to the four outputs. If, by contrast, $S = 1$, the four B inputs are applied to the outputs. The outputs have all 0's when $E = 1$, regardless of the value of S .

Boolean Function Implementation

In Section 4.9, it was shown that a decoder can be used to implement Boolean functions by employing external OR gates. An examination of the logic diagram of a multiplexer reveals that it is essentially a decoder that includes the OR gate within

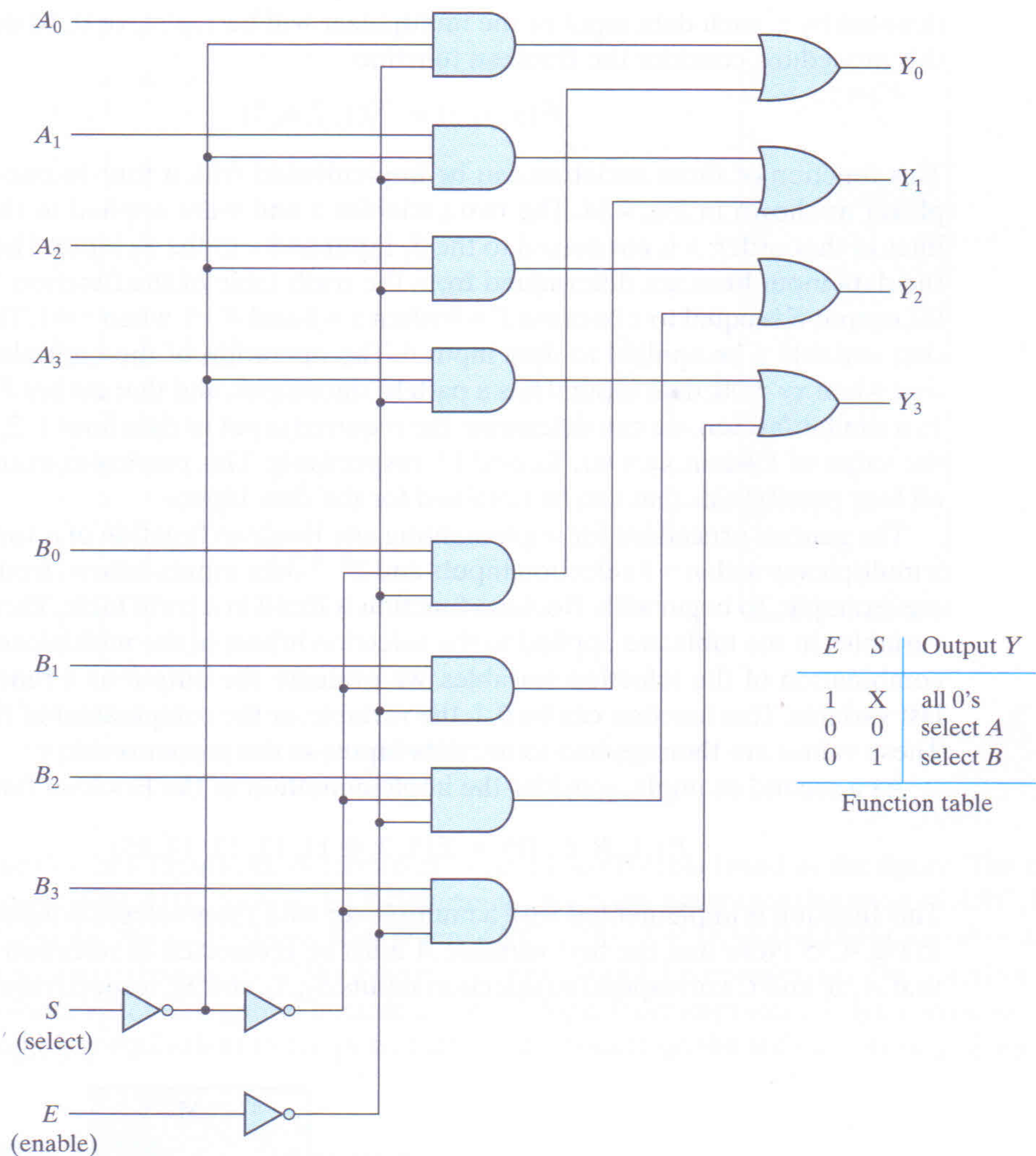


FIGURE 4.33
Quadruple two-to-one-line multiplexer

the unit. The minterms of a function are generated in a multiplexer by the circuit associated with the selection inputs. The individual minterms can be selected by the data inputs, thereby providing a method of implementing a Boolean function of n variables with a multiplexer that has n selection inputs and 2^n data inputs, one for each minterm.

We will now show a more efficient method for implementing a Boolean function of n variables with a multiplexer that has $n - 1$ selection inputs. The first $n - 1$ variables of the function are connected to the selection inputs of the multiplexer. The remaining single variable of the function is used for the data inputs. If the single variable is

denoted by z , each data input of the multiplexer will be $z, z', 1$, or 0 . To demonstrate this procedure, consider the Boolean function

$$F(x, y, z) = \Sigma(1, 2, 6, 7)$$

This function of three variables can be implemented with a four-to-one-line multiplexer as shown in Fig. 4.34. The two variables x and y are applied to the selection lines in that order; x is connected to the S_1 input and y to the S_0 input. The values for the data input lines are determined from the truth table of the function. When $xy = 00$, output F is equal to z because $F = 0$ when $z = 0$ and $F = 1$ when $z = 1$. This requires that variable z be applied to data input 0. The operation of the multiplexer is such that when $xy = 00$, data input 0 has a path to the output, and that makes F equal to z . In a similar fashion, we can determine the required input to data lines 1, 2, and 3 from the value of F when $xy = 01, 10$, and 11 , respectively. This particular example shows all four possibilities that can be obtained for the data inputs.

The general procedure for implementing any Boolean function of n variables with a multiplexer with $n - 1$ selection inputs and 2^{n-1} data inputs follows from the previous example. To begin with, Boolean function is listed in a truth table. Then first $n - 1$ variables in the table are applied to the selection inputs of the multiplexer. For each combination of the selection variables, we evaluate the output as a function of the last variable. This function can be 0, 1, the variable, or the complement of the variable. These values are then applied to the data inputs in the proper order.

As a second example, consider the implementation of the Boolean function

$$F(A, B, C, D) = \Sigma(1, 3, 4, 11, 12, 13, 14, 15)$$

This function is implemented with a multiplexer with three selection inputs as shown in Fig. 4.35. Note that the first variable A must be connected to selection input S_2 so that A, B , and C correspond to selection inputs S_2, S_1 , and S_0 , respectively. The values

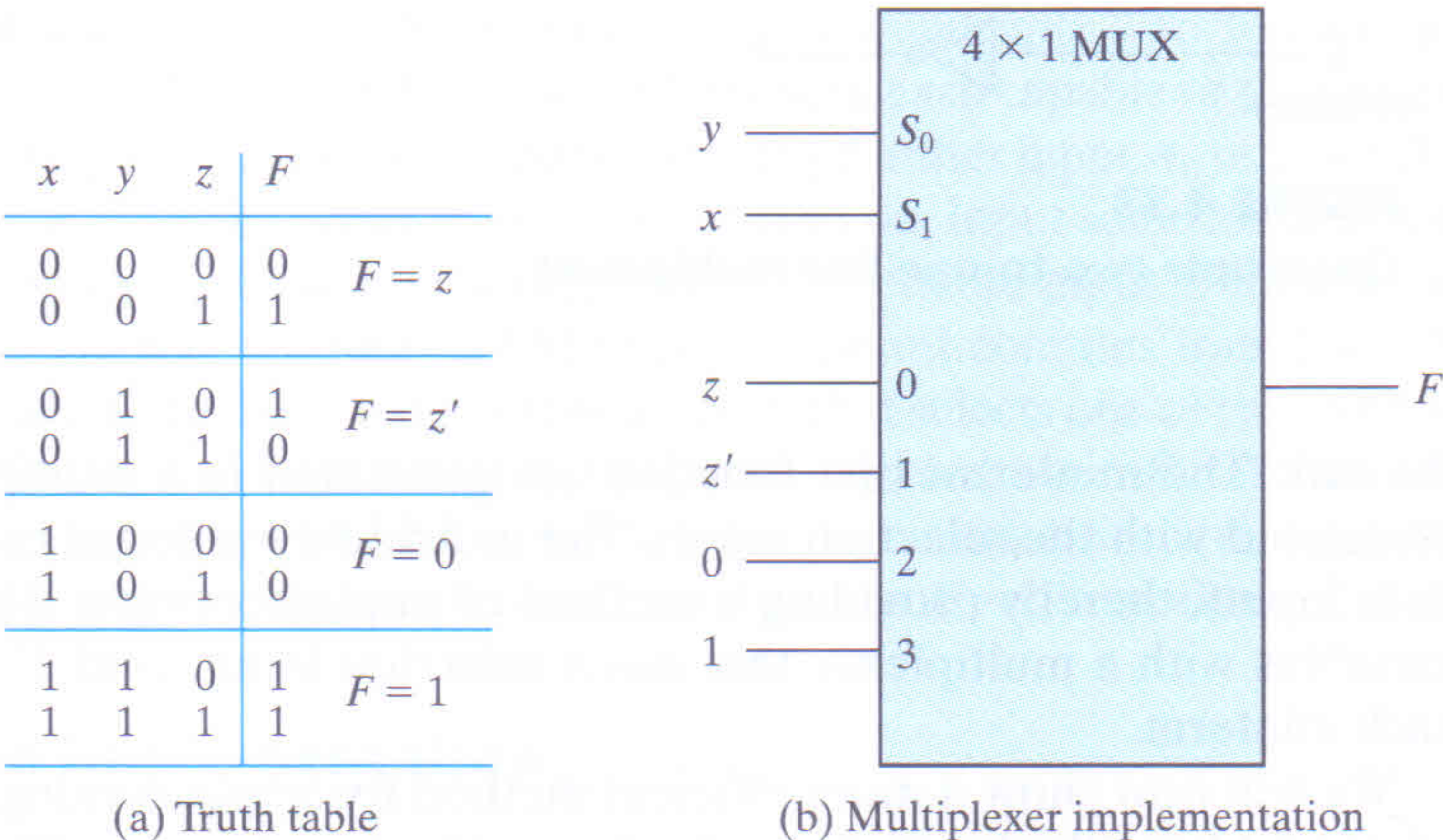


FIGURE 4.34
Implementing a Boolean function with a multiplexer

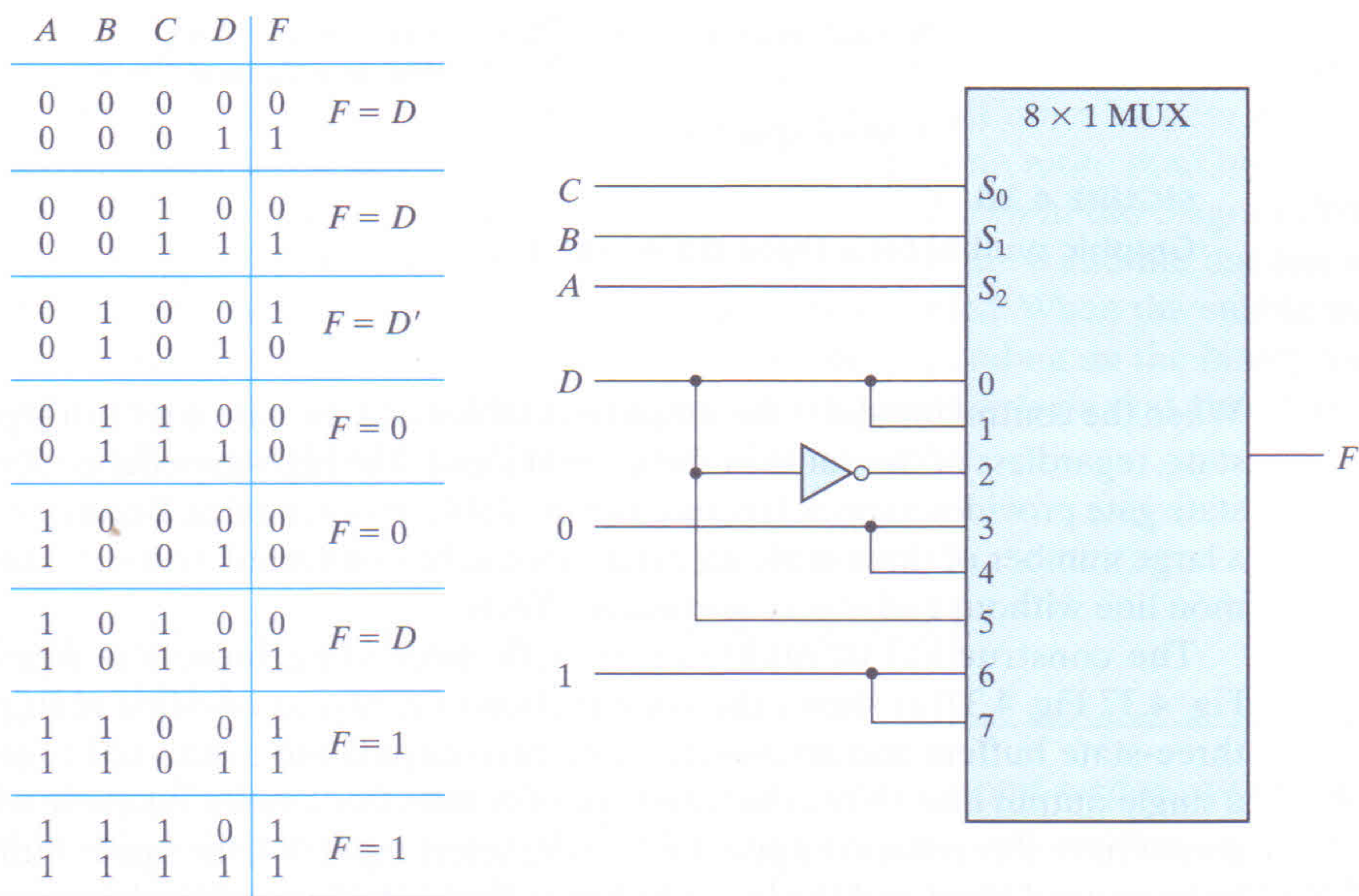


FIGURE 4.35
Implementing a four-input function with a multiplexer

for the data inputs are determined from the truth table listed in the figure. The corresponding data line number is determined from the binary combination of ABC . For example, the table shows that when $ABC = 101$, $F = D$, so the input variable D is applied to data input 5. The binary constants 0 and 1 correspond to two fixed signal values. When integrated circuits are used, logic 0 corresponds to signal ground and logic 1 is equivalent to the power signal, depending on the technology (e.g., 3 V).

Three-State Gates

A multiplexer can be constructed with three-state gates—digital circuits that exhibit three states. Two of the states are signals equivalent to logic 1 and logic 0 as in a conventional gate. The third state is a *high-impedance* state in which (1) the logic behaves like an open circuit, which means that the output appears to be disconnected, (2) the circuit has no logic significance, and (3) the circuit connected to the output of the three-state gate is not affected by the inputs to the gate. Three-state gates may perform any conventional logic, such as AND or NAND. However, the one most commonly used is the buffer gate.

The graphic symbol for a three-state buffer gate is shown in Fig. 4.36. It is distinguished from a normal buffer by an input control line entering the bottom of the symbol. The buffer has a normal input, an output, and a control input that determines the state of the output. When the control input is equal to 1, the output is enabled and the gate behaves like a conventional buffer, with the output equal to the normal input.