```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY shifter IS
     PORT ( w        : IN    STD_LOGIC_VECTOR(3 DOWNTO 0) ;
            Shift    : IN    STD_LOGIC ;
            y        : OUT   STD_LOGIC_VECTOR(3 DOWNTO 0) ;
            k        : OUT   STD_LOGIC ) ;
END shifter ;

ARCHITECTURE Behavior OF shifter IS
BEGIN
     PROCESS (Shift, w)
     BEGIN
          IF Shift = '1' THEN
               y(3) <= '0' ;
               y(2 DOWNTO 0) <= w(3 DOWNTO 1) ;
               k <= w(0) ;
          ELSE
               y <= w ;
               k <= '0' ;
          END IF ;
     END PROCESS ;
END Behavior ;
```

Figure 6.59 Structural VHDL code that specifies the shifter circuit in Figure 6.56.

PROBLEMS

Answers to problems marked by an asterisk are given at the back of the book.

6.1 Show how the function $f(w_1, w_2, w_3) = \sum m(0, 2, 3, 4, 5, 7)$ can be implemented using a 3-to-8 binary decoder and an OR gate.

6.2 Show how the function $f(w_1, w_2, w_3) = \sum m(1, 2, 3, 5, 6)$ can be implemented using a 3-to-8 binary decoder and an OR gate.

***6.3** Consider the function $f = \overline{w}_1\overline{w}_3 + w_2\overline{w}_3 + \overline{w}_1 w_2$. Use the truth table to derive a circuit for f that uses a 2-to-1 multiplexer.

6.4 Repeat problem 6.3 for the function $f = \overline{w}_2\overline{w}_3 + w_1 w_2$.

***6.5** For the function $f(w_1, w_2, w_3) = \sum m(0, 2, 3, 6)$, use Shannon's expansion to derive an implementation using a 2-to-1 multiplexer and any other necessary gates.

6.6 Repeat problem 6.5 for the function $f(w_1, w_2, w_3) = \sum m(0, 4, 6, 7)$.

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.numeric_std.all ;

ENTITY shifter IS
    PORT ( w       : IN    UNSIGNED(3 DOWNTO 0) ;
           Shift   : IN    STD_LOGIC ;
           y       : OUT   UNSIGNED(3 DOWNTO 0) ;
           k       : OUT   STD_LOGIC ) ;
END shifter ;

ARCHITECTURE Behavior OF shifter IS
BEGIN
    PROCESS (Shift, w)
    BEGIN
        IF Shift = "1" THEN
            y <= w SRL 1 ;
            k <= w(0) ;
        ELSE
            y <= w ;
            k <= "0" ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Figure 6.60 Behavioral VHDL code that specifies the shifter circuit in Figure 6.56.

6.7 Consider the function $f = \overline{w}_2 + \overline{w}_1\overline{w}_3 + w_1w_3$. Show how repeated application of Shannon's expansion can be used to derive the minterms of f.

6.8 Repeat problem 6.7 for $f = w_2 + \overline{w}_1\overline{w}_3$.

6.9 Prove Shannon's expansion theorem presented in section 6.1.2.

***6.10** Section 6.1.2 shows Shannon's expansion in sum-of-products form. Using the principle of duality, derive the equivalent expression in product-of-sums form.

6.11 Consider the function $f = \overline{w}_1\overline{w}_2 + \overline{w}_2\overline{w}_3 + w_1w_2w_3$. Give a circuit that implements f using the minimal number of two-input LUTs. Show the truth table implemented inside each LUT.

***6.12** For the function in problem 6.11, the cost of the minimal sum-of-products expression is 14, which includes four gates and 10 inputs to the gates. Use Shannon's expansion to derive a multilevel circuit that has a lower cost and give the cost of your circuit.

6.13 Consider the function $f(w_1, w_2, w_3, w_4) = \sum m(0, 1, 3, 6, 8, 9, 14, 15)$. Derive an implementation using the minimum possible number of three-input LUTs.

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;
USE ieee.numeric_std.all ;

ENTITY barrel IS
    PORT ( w  : IN   UNSIGNED(3 DOWNTO 0) ;
           s  : IN   UNSIGNED(1 DOWNTO 0) ) ;
           y  : OUT  UNSIGNED(3 DOWNTO 0) ) ;
END barrel ;

ARCHITECTURE Behavior OF barrel IS
BEGIN
    PROCESS (s, w)
    BEGIN
        CASE s IS
            WHEN "00" =>
                y <= w ;
            WHEN "01" =>
                y <= w ROR 1 ;
            WHEN "10" =>
                y <= w ROR 2 ;
            WHEN OTHERS =>
                y <= w ROR 3 ;
        END CASE ;
    END PROCESS ;
END Behavior ;
```

Figure 6.61 VHDL code that specifies the barrel shifter circuit in Figure 6.57.

***6.14** Give two examples of logic functions with five inputs, $w_1, \ldots, w_5$, that can be realized using 2 four-input LUTs.

6.15 For the function, f, in Example 6.27 perform Shannon's expansion with respect to variables w_1 and w_2, rather than w_1 and w_4. How does the resulting circuit compare with the circuit in Figure 6.52?

6.16 Actel Corporation manufactures an FPGA family called Act 1, which has the multiplexer-based logic block illustrated in Figure P6.1. Show how the function $f = w_2\overline{w}_3 + w_1w_3 + \overline{w}_2w_3$ can be implemented using only one Act 1 logic block.

6.17 Show how the function $f = w_1\overline{w}_3 + \overline{w}_1w_3 + w_2\overline{w}_3 + w_1\overline{w}_2$ can be realized using Act 1 logic blocks. Note that there are no NOT gates in the chip; hence complements of signals have to be generated using the multiplexers in the logic block.

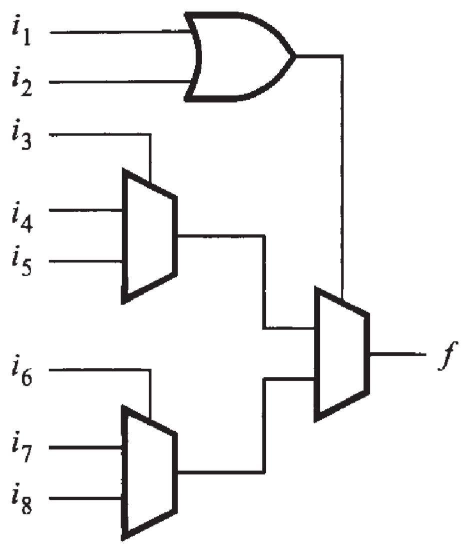

Figure P6.1 The Actel Act 1 logic block.

***6.18** Consider the VHDL code in Figure P6.2. What type of circuit does the code represent? Comment on whether or not the style of code used is a good choice for the circuit that it represents.

6.19 Write VHDL code that represents the function in problem 6.1, using one selected signal assignment.

6.20 Write VHDL code that represents the function in problem 6.2, using one selected signal assignment.

6.21 Using a selected signal assignment, write VHDL code for a 4-to-2 binary encoder.

6.22 Using a conditional signal assignment, write VHDL code for an 8-to-3 binary encoder.

6.23 Derive the circuit for an 8-to-3 priority encoder.

6.24 Using a conditional signal assignment, write VHDL code for an 8-to-3 priority encoder.

6.25 Repeat problem 6.24, using an if-then-else statement.

6.26 Create a VHDL entity named *if2to4* that represents a 2-to-4 binary decoder using an if-then-else statement. Create a second entity named *h3to8* that represents the 3-to-8 binary decoder in Figure 6.17, using two instances of the *if2to4* entity.

6.27 Create a VHDL entity named *h6to64* that represents a 6-to-64 binary decoder. Use the treelike structure in Figure 6.18, in which the 6-to-64 decoder is built using five instances of the *h3to8* decoder created in problem 6.26.

6.28 Write VHDL code for a BCD-to-7-segment code converter, using a selected signal assignment.

***6.29** Derive minimal sum-of-products expressions for the outputs a, b, and c of the 7-segment display in Figure 6.25.

```
LIBRARY ieee ;
USE ieee.std_logic_1164.all ;

ENTITY problem IS
    PORT ( w              : IN    STD_LOGIC_VECTOR(1 DOWNTO 0) ;
           En             : IN    STD_LOGIC ;
           y0, y1, y2, y3 : OUT   STD_LOGIC ) ;
END problem ;

ARCHITECTURE Behavior OF problem IS
BEGIN
    PROCESS (w, En)
    BEGIN
        y0 <= '0' ; y1 <= '0' ; y2 <= '0' ; y3 <= '0' ;
        IF En = '1' THEN
            IF w = "00" THEN y0 <= '1' ;
            ELSIF w = "01" THEN y1 <= '1' ;
            ELSIF w = "10" THEN y2 <= '1' ;
            ELSE y3 <= '1' ;
            END IF ;
        END IF ;
    END PROCESS ;
END Behavior ;
```

Figure P6.2 Code for problem 6.18.

6.30 Derive minimal sum-of-products expressions for the outputs d, e, f, and g of the 7-segment display in Figure 6.25.

6.31 Design a shifter circuit, similar to the one in Figure 6.56, which can shift a four-bit input vector, $W = w_3w_2w_1w_0$, one bit-position to the right when the control signal *Right* is equal to 1, and one bit-position to the left when the control signal *Left* is equal to 1. When *Right* = *Left* = 0, the output of the circuit should be the same as the input vector. Assume that the condition *Right* = *Left* = 1 will never occur.

6.32 Figure 6.21 shows a block diagram of a ROM. A circuit that implements a small ROM, with four rows and four columns, is depicted in Figure P6.3. Each X in the figure represents a switch that determines whether the ROM produces a 1 or 0 when that location is read.
(a) Show how a switch (X) can be realized using a single NMOS transistor.
(b) Draw the complete 4×4 ROM circuit, using your switches from part (*a*). The ROM should be programmed to store the bits 0101 in row 0 (the top row), 1010 in row 1, 1100 in row 2, and 0011 in row 3 (the bottom row).
(c) Show how each (X) can be implemented as a programmable switch (as opposed to providing either a 1 or 0 permanently), using an EEPROM cell as shown in Figure 3.64. Briefly describe how the storage cell is used.

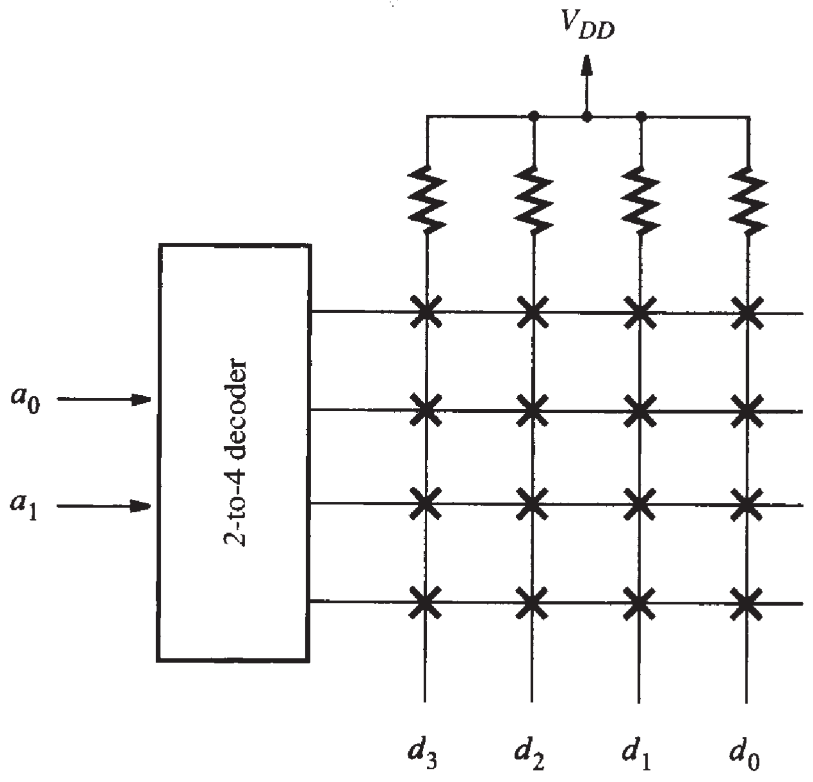

Figure P6.3 A 4 × 4 ROM circuit.

6.33 Show the complete circuit for a ROM using the storage cells designed in Part (a) of problem 6.33 that realizes the logic functions

$$d_3 = a_0 \oplus a_1$$

$$d_2 = \overline{a_0 \oplus a_1}$$

$$d_1 = a_0 a_1$$

$$d_0 = a_0 + a_1$$

REFERENCES

1. C. E. Shannon, "Symbolic Analysis of Relay and Switching Circuits," *Transactions AIEE* 57 (1938), pp. 713–723.
2. Actel Corporation, "MX FPGA Data Sheet," *http://www.actel.com.*
3. QuickLogic Corporation, "pASIC 3 FPGA Data Sheet," *http://www.quicklogic.com.*