EXERCISES

2.1 *(Truth Tables and Gate Logic)* Derive a gate-logic implementation for the three functions in the truth table of Figure 2.3. Try to come up with as small an implementation as you can in terms of gate count.

2.2 *(Gate Logic)* Draw schematics for the following functions in terms of AND, OR, and NOT gates.

(a) $X(Y + Z)$
(b) $XY + XZ$
(c) $\overline{X(Y + Z)}$
(d) $\bar{X} + \overline{\bar{Y}\bar{Z}}$
(e) $W(X + YZ)$

2.3 *(Gate Logic)* Draw the schematics for the following functions using NOR gates and inverters only:

(a) $\overline{[\bar{X} + \overline{(Y + Z)}]}$
(b) $\overline{[\overline{(\bar{X} + \bar{Y})} + \overline{(\bar{X} + \bar{Z})}]}$

2.4 *(Gate Logic)* Draw the schematics for the following functions using NAND gates and inverters only:

(a) $\overline{[X(\overline{\bar{Y}\bar{Z}})]}$
(b) $XY + XZ$

2.5 *(Gate Logic)* Design a hall light circuit to the following specification. There is a switch at either end of a hall that controls a single light. If the light is off, changing the position of either switch causes the light to turn on. Similarly, if the light is on, changing the position of either switch causes the light to turn off. Write your assumptions, derive a truth table, and describe how to implement this function in terms of logic gates.

2.6 *(Laws and Theorems of Boolean Algebra)* Prove the following simplification theorems using the first eight laws of Boolean algebra:

(a) $(X + Y)(X + \bar{Y}) = X$
(b) $X(X + Y) = X$
(c) $(X + \bar{Y})Y = XY$
(d) $(X + Y)(\bar{X} + Z) = XZ + \bar{X}Y$

2.7 *(Laws and Theorems of Boolean Algebra)* Verify that

(a) OR and AND are duals of each other
(b) NOR and NAND are duals of each other
(c) XNOR and XOR are duals of each other
(d) XNOR is the complement of XOR:

$$\overline{(X\bar{Y} + \bar{X}Y)} = \bar{X}\bar{Y} + XY$$

2.8 *(Laws and Theorems of Boolean Algebra)* Prove, using truth tables, that:

(a) $XY + YZ + X\bar{Z} = YZ + X\bar{Z}$
(b) $(A + \bar{B})B = AB$
(c) $(A + B)(\bar{A} + C) = AC + \bar{A}B$
(d) $ABC + \bar{A}BC + \bar{A}\bar{B}C + \bar{A}B\bar{C} + \bar{A}\bar{B}\bar{C} = BC + \bar{A}\bar{B} + \bar{A}\bar{C}$

2.9 *(Laws and Theorems of Boolean Algebra)* Prove, using the theorems of Boolean algebra, that $BC + \bar{A}\bar{B} + \bar{A}\bar{C} = ABC + \bar{A}$.

2.10 *(Laws and Theorems of Boolean Algebra)* Use DeMorgan's law to compute the complement of the following Boolean expressions:

(a) $A(B + CD)$
(b) $ABC + B(\bar{C} + \bar{D})$
(c) $\bar{X} + \bar{Y}$
(d) $X + Y\bar{Z}$
(e) $(X + Y)Z$
(f) $X + \overline{(YZ)}$
(g) $X(Y + Z\bar{W} + \bar{V}S)$

2.11 *(Laws and Theorems of Boolean Algebra)* Form the complement of the following functions:

(a) $f(A,B,C,D) = [A + \overline{BCD}][\overline{AD} + B(\bar{C} + A)]$
(b) $f(A,B,C,D) = \bar{A}BC + (\bar{A} + B + D)(AB\bar{D} + \bar{B})$

2.12 *(Laws and Theorems of Boolean Algebra)* Using Boolean algebra, verify that the schematic of Figure Ex. 2.12 implements an XOR function.

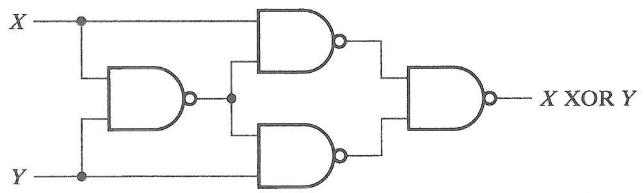

Figure Ex. 2.12 **XOR implemented by NAND gates.**

2.13 *(Laws and Theorems of Boolean Algebra)* Demonstrate that a 2-input NOR gate is a universal logic element. You can do this by showing how they can be used to make: NOT, AND, OR, and XOR gates. Remember that each input of the NOR gates must be used, it can not be left unconnected. Is an XOR gate a universal logic element? Why or why not? What about a 2-input NAND gate?

2.14 *(Block Diagrams)* Given the truth table for the half adder, show that the composition of two half adders and an OR gate, as in Figure 2.19, yield the same truth table as the full adder.

2.15 *(Waveform Verification)* This chapter has described two different gate-level implementations for a full-adder circuit: direct implementation from the Boolean equations derived in Section 2.2 and hierarchical implementation via cascaded half adders, as in Figure 2.19. Would you expect the waveform behaviors of these implementations to be identical? Justify your answer.

2.16 *(Block Diagrams)* Using the formulas for the two full-adder outputs, derive expressions for the three outputs of the 2-bit adder of Figure 2.20 in terms of the 2-bit inputs A and B.

2.17 *(Boolean Simplification)* Simplify the following functions using the theorems of Boolean algebra. Write the particular law or theorem you are using in each step. For each function, by how many literals did you reduce its representation?

- **(a)** $f(X,Y) = XY + X\bar{Y}$
- **(b)** $f(X,Y) = (X + Y)(X + \bar{Y})$
- **(c)** $f(X,Y,Z) = \bar{Y}Z + \bar{X}YZ + XYZ$
- **(d)** $f(X,Y,Z) = (X + Y)(\bar{X} + Y + Z)(\bar{X} + Y + \bar{Z})$
- **(e)** $f(W,X,Y,Z) = X + XYZ + \bar{X}YZ + \bar{X}Y + WX + W\bar{X}$

2.18 *(Boolean Simplification)* Consider the function:

$$f(A,B,C,D) = (AD + \bar{A}C)[\bar{B}(C + B\bar{D})].$$

- **(a)** Draw its schematic using AND, OR, and NOT gates.
- **(b)** Using Boolean algebra, put the function into its minimized form and draw the resulting schematic.

2.19 *(Canonical Forms)* Consider the function:

$$f(A,B,C,D) = \Sigma\ m(0,1,2,7,8,9,10,15).$$

- **(a)** Write this as a Boolean expression in canonical minterm form.
- **(b)** Rewrite the expression in canonical maxterm form.
- **(c)** Write the complement of f in "little m" notation and as a canonical minterm expression.
- **(d)** Write the complement of f in "big M" notation and as a canonical maxterm expression.

2.20 *(Canonical Forms)* Consider the function:

$$f(A,B,C,D) = \Sigma\ m(1, 2, 3, 5, 8, 13).$$

- **(a)** Write this as a Boolean expression in canonical minterm form.
- **(b)** Rewrite the expression in canonical maxterm form.
- **(c)** Write the complement of f in "little m" notation and as a canonical minterm expression.
- **(d)** Write the complement of f in "big M" notation and as a canonical maxterm expression.

2.21 *(Canonical Forms)* Consider the function:

$$f(A,B,C) = AB + \bar{B}\bar{C} + A\bar{C}$$

(a) Express the function in canonical sum-of-products form. Use "little m" notation.
(b) Express the complement of the function in canonical product-of-sums form. Use "big M" notation.

2.22 *(Canonical Forms)* Given the following function in sum-of-products form (not necessarily minimized):

$$F(A,B,C,D) = \bar{A}BC + AD + AC$$

Re-express the function in:

(a) Canonical product-of-sums form. Use $\Pi\, M$ notation.
(b) Minimized product-of-sums form.
(c) $\bar{F}$ in minimized product-of-sums form.
(d) $\bar{F}$ in minimized sum-of-products form.
(e) Implement F and $\bar{F}$ using NAND gates only. You may assume that literals and their complements are available.
(f) Implement F and $\bar{F}$ using NOR gates only. You may assume that literals and their complements are available.

2.23 *(Canonical Forms and Boolean Simplification)* Given the following function in product-of-sums form, not necessarily minimized:

$$F(W,X,Y,Z) = (W + \bar{X} + \bar{Y})(\bar{W} + \bar{Z})(W + Y)$$

(a) Express the function in the canonical sum-of-products form. Use "little m" notation.
(b) Re-express the function in minimized sum-of-products form.
(c) Express $\bar{F}$ in minimized sum-of-products form.
(d) Re-express $\bar{F}$ in minimized product-of-sums form.

2.24 *(Boolean Simplification)* Using K-maps, find the following:

(a) Minimum sum-of-products form for the function and its complement given in Exercise 2.19.
(b) Minimum product-of-sums form for the function and its complement given in Exercise 2.19.

2.25 *(Boolean Simplification)* Use Karnaugh maps (K-maps) to simplify the following functions in sum-of-products form. How many literals appear in your minimized solutions?

(a) $f(X,Y,Z) = \Pi\, M(0,1,6,7)$
(b) $f(W,X,Y,Z) = \Pi\, M(1,3,7,9,11,15)$
(c) $f(A,B,C,D) = \Sigma\, m(0,2,4,6)$

2.26 *(Boolean Simplification)* Determine the minimized realization of the following functions in the sum-of-products form:

(a) $f(W,X,Y,Z) = \Sigma\, m(0,2,8,9) + \Sigma\, d(1,3)$
(b) $f(W,X,Y,Z) = \Sigma\, m(1,7,11,13) + \Sigma\, d(0,5,10,15)$

(c) $f(A,B,C,D) = \Sigma\, m(1,2,11,13,14,15) + \Sigma\, d(0,3,6,10)$.
(d) $f(A,B,C,D) = \Pi\, M(2,5,6,8,9,10) * \Pi\, D(4,11,12)$.

2.27 *(Boolean Simplification)* Use the K-map method to find the minimized product-of-sums expressions for the following Boolean functions:

(a) $f(A,B,C) = A \oplus B \oplus C$
(b) $f(A,B,C) = AB + BC + AC$
(c) $f(A,B,C,D) = \Sigma\, m(1,3,5,7,9) + \Sigma\, d(6,12,13)$
(d) $f(A,B,C,D) = \Pi\, M(0,1,6,7)$
(e) $f(A,B,C,D) = \Sigma\, m(0,2,4,6)$

2.28 *(Boolean Simplification)* Simplify the following expressions using the laws and theorems of Boolean algebra:

(a) $S(A,B,C) = \bar{A}\bar{B}C + \bar{A}B\bar{C} + A\bar{B}\bar{C} + ABC$
(b) $F(A,B,C) = \bar{A}\bar{B}\bar{C} + \bar{A}\bar{B}C + A\bar{B}\bar{C} + A\bar{B}C + AB\bar{C} + ABC$
(c) $G(A,B,C,D) = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}C\bar{D} + A\bar{B}\bar{C}\bar{D} + A\bar{B}CD$
$+ AB\bar{C}\bar{D} + ABCD$

2.29 *(Boolean Simplification)* Use K-maps on the expressions of Problem 2.28. Show your work in K-map form. From 2.28 (a) through (c):

(a) Find the minimized sum-of-products form.
(b) Find the minimized product-of-sums form.
(c) Find the minimized sum-of-products form of the function's complement.
(d) Find the minimized product-of-sums form of the function's complement.

2.30 *(Laws and Theorems of Boolean Algebra)* Simplify the following expressions using the laws and theorems of Boolean algebra:

(a) $W(A,B,C) = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + A\bar{B}C$
(b) $X(A,B,C) = \bar{A}\bar{B}\bar{C} + \bar{A}BC + A\bar{B}\bar{C} + ABC$
(c) $Y(A,B,C,D) = \bar{A}\bar{B}\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D} + \bar{A}\bar{B}CD$
$+ A\bar{B}\bar{C}\bar{D} + A\bar{B}C\bar{D}$

2.31 *(Karnaugh Map Method)* Use K-maps on the expressions of Exercise 2.30. Show your work in K-map form.

(a) Find the minimized sum-of-products form.
(b) Find the minimized product-of-sums form.
(c) Find the minimized sum-of-products form of the function's complement.
(d) Find the minimized product-of-sums form of the function's complement.

2.32 *(Karnaugh Map Method)* There may be more than one true minimum equivalent form for a given Boolean expression. Demonstrate this by drawing a 4-variable K-map that has two different minimized forms for the same Boolean expression, each with the same number of terms and literals.

2.33 *(Karnaugh Map Method)* Using a 4-variable K-map, fill it with 1s and 0s to find a function that illustrates the following points. Write the expressions for each of the requested forms and count the number of terms and literals for each one:

(a) The minimized sum-of-products and product-of-sums forms have the same number of terms and literals.
(b) The minimized sum-of-products form has fewer terms and literals than the minimized product-of-sums form.
(c) The minimized product-of-sums form has fewer terms and literals than the minimized sum-of-products form.

2.34 *(Incompletely Specified Functions)* Use K-maps to derive minimal expressions for the functions for C_0, C_1, C_3, C_4, C_5, and C_6 in Figure 2.33.

2.35 *(Incompletely Specified Functions)* Use K-maps to derive minimal sum-of-product expressions for d30 and d31 from the calendar example of Section 1.4.1. Make sure to take advantage of don't care information.

2.36 *(Encoding)* Use K-maps to derive minimal sum-of-product expressions for d30 and d31 from the calendar example of Section 1.4.1 but with a different encoding for the months. In this case, have the months start with January = 0000 and December = 1011. Make sure to take advantage of don't care information. Is this encoding better or worse than the original one?

2.37 *(Multilevel Logic)* Factor the following sum-of-products expressions:

(a) $ABCD + ABDE$
(b) $ACD + BC + ABE + BD$
(c) $AC + ADE + BC + BDE$
(d) $AD + AE + BD + BE + CD + CE + AF$
(e) $ACE + ACF + ADE + ADF + BCE + BCF + BDE + BDF$

2.38 *(Multilevel Logic)* Write down the function represented by the circuit network in Figure Ex. 2.38 in a multilevel factored form using AND, OR, and NOT operations only—that is, no NAND or NOR operations.

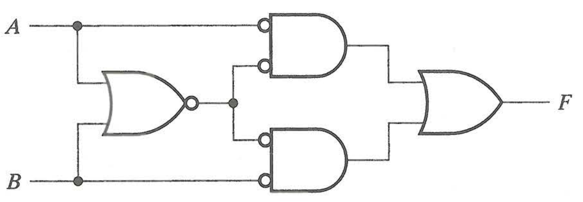

Figure Ex. 2.38

Derive the simplest Boolean expression you can (minimum number of literals and fewest gates) for the function represented by this schematic. You may use any kind of logic operators described in this chapter.

2.39 *(Multilevel Logic)* Derive the multilevel logic realization of the 2-bit adder of Figure 2.20. Compare it to a two-level logic realization of the same function. List the different types of gates (function and fanin) for each of the two implementations. Which has fewer gates? Which has fewer wires? Which is likely to be faster (do this by showing the relative delays along the worst-case paths from an input to an output)? This is similar to the comparison for the full adder at the end of Section 2.7.

2.40 *(Combinational Logic Design)* Write truth tables for the three functions described by the following specifications:

(a) A 2-bit-wide *shifter* takes two input signals, i_0 and i_1, and shifts them to two outputs, o_0 and o_1, under the control of a shift signal. If this signal *SHIFT* is false, then the outputs are equal to their corresponding inputs. If *SHIFT* is true, then o_1 is equal to i_0, and o_0 is set a 0.

(b) A 1-bit *demultiplexer* takes an input signal *IN* and "routes" it to one of two outputs, o_0 and o_1, under the control of a single *SELECT* signal. If *SELECT* is 0, then o_0 has the value of *IN* and o_1 is a 0. If *SELECT* is 1, then o_1 has the value of *IN* and o_0 is a 0.

(c) A 2-bit *multiplexer* takes two input signals, i_0 and i_1, and "routes" one of them to the single output *OUT* under the control of a 1-bit select signal. If the *SELECT* signal is false, then *OUT* is equal to i_0. If *SELECT* is true, then *OUT* is equal to i_1.

2.41 *(Boolean Simplification)* Write sum-of-products expressions for the truth tables of Exercise 2.40. Minimize them using K-maps.

2.42 *(Gates)* Given the Boolean expressions of Exercise 2.41, draw logic schematics using AND, OR, and INVERT gates that implement those functions.

2.43 *(Combinational Logic Design)* Consider a 4-input Boolean function that is asserted whenever exactly two of its inputs are asserted.

(a) Construct its truth table.

(b) What is the function in sum-of-products form, using "little *m*" notation?

(c) What is the function in product-of-sums form, using "big *M*" notation?

(d) Use the Karnaugh map method to simplify the function in sum of products form.

2.44 *(Combinational Logic Design)* In this chapter, we've examined the BCD increment-by-1 function (see Figure 2.32). Now consider a binary increment-by-1 function defined over the 4-bit binary numbers 0000 through 1111.

(a) Fill in the truth table for the function.
(b) Fill in the four 4-variable K-maps, and find the minimized sum-of-products for each output function.
(c) Repeat the process for the minimized product-of-sums form. Which leads to the simpler implementation in terms of the number of literals?

2.45 *(Combinational Logic Design)* Consider a four-input function that outputs a 1 whenever an odd number of its inputs are 1.

(a) Fill in the truth table for the function.
(b) Fill in the K-map to find the minimum sum-of-products expression for the function. What is it? Can the function be minimized using the K-map method?
(c) Can you think of a more economical way to implement this function if XOR gates are allowed? (*Warning:* It will be very tedious to try to simplify this function using Boolean algebra, so think about the question first!)

2.46 *(Combinational Logic Design)* In this chapter, we've examined a 2-bit binary adder circuit. Now consider a 2-bit binary subtractor, defined as follows. The inputs A, B and C, D form the two 2-bit numbers N_1 and N_2. The circuit will form the difference $N_1 - N_2$ on the output bits F (most significant) and G (least significant). Assume that the circuit never sees an input combination in which N_1 is less than N_2. The output bits are don't cares in these cases.

(a) Fill in the 4-variable truth table for F and G.
(b) Fill in the K-map for the minimum sum-of-products expression for the functions F and G.
(c) Repeat to find the minimum product-of-sums expression for F and G.