benches. Another practical book is *VHDL for Programmable Logic*, by Kevin Skahill of Cypress Semiconductor (Addison-Wesley, 1996).

All of the ABEL and VHDL examples in this chapter and throughout the text were compiled and in most cases simulated using Foundation 1.5 Student Edition software from Xilinx, Inc. (San Jose, CA 95124, `www.xilinx.com`). The Foundation software integrates a schematic editor, HDL editor, compilers for ABEL, VHDL and Verilog, and a simulator from Aldec, Inc. (Henderson, NV 89014, `www.aldec.com`) along with Xilinx' own specialized tools for CPLD and FPGA design and programming. It also contains an excellent on-line help system, including reference manuals for both ABEL and VHDL. The Foundation software is packaged on CD-ROM with some editions of this book.

The IEEE standard VHDL packages are important elements of any VHDL design environment. A listing of their type and function definitions is included as an appendix in some VHDL texts, but if you're really curious about them, you can find the complete source files in the library subsystem of any VHDL design system, including Foundation software.

We briefly discussed device testing in the context of ABEL test vectors. There is a large, well-established body of literature on digital device testing, and a good starting point for study is McCluskey's 1986 book. Generating a set of test vectors that completely tests a large circuit such as a PLD is a task best left to a program. At least one company's entire business is focused on programs that automatically create test vectors for PLD testing (ACUGEN Software, Inc., Nashua, NH 03063, `www.acugen.com`).

Drill Problems

4.1 Using variables NERD, DESIGNER, FAILURE, and STUDIED, write a boolean expression that is 1 for successful designers who never studied and for nerds who studied all the time.

4.2 Prove theorems T2–T5 using perfect induction.

4.3 Prove theorems T1′–T3′ and T5′ using perfect induction.

4.4 Prove theorems T6–T9 using perfect induction.

4.5 According to DeMorgan's theorem, the complement of $X + Y \cdot Z$ is $X' \cdot Y' + Z'$. Yet both functions are 1 for XYZ = 110. How can both a function and its complement be 1 for the same input combination? What's wrong here?

4.6 Use the theorems of switching algebra to simplify each of the following logic functions:

(a) $F = W \cdot X \cdot Y \cdot Z \cdot (W \cdot X \cdot Y \cdot Z' + W \cdot X' \cdot Y \cdot Z + W' \cdot X \cdot Y \cdot Z + W \cdot X \cdot Y' \cdot Z)$

(b) $F = A \cdot B + A \cdot B \cdot C' \cdot D + A \cdot B \cdot D \cdot E' + A \cdot B \cdot C' \cdot E + C' \cdot D \cdot E$

(c) $F = M \cdot N \cdot O + Q' \cdot P' \cdot N' + P \cdot R \cdot M + Q' \cdot O \cdot M \cdot P' + M \cdot R$

4.7 Write the truth table for each of the following logic functions:

(a) $F = X' \cdot Y + X' \cdot Y' \cdot Z$ (b) $F = W' \cdot X + Y' \cdot Z' + X' \cdot Z$

(c) $F = W + X' \cdot (Y' + Z)$ (d) $F = A \cdot B + B' \cdot C + C' \cdot D + D' \cdot A$

(e) $F = V \cdot W + X' \cdot Y' \cdot Z$ (f) $F = (A' + B' + C \cdot D) \cdot (B + C' + D' \cdot E')$

(g) $F = (W \cdot X)' \cdot (Y' + Z')'$ (h) $F = (((A + B)' + C')' + D)'$

(i) $F = (A' + B + C) \cdot (A + B' + D') \cdot (B + C' + D') \cdot (A + B + C + D)$

4.8 Write the truth table for each of the following logic functions:

(a) $F = X' \cdot Y' \cdot Z' + X \cdot Y \cdot Z + X \cdot Y' \cdot Z$ (b) $F = M' \cdot N' + M \cdot P + N' \cdot P$

(c) $F = A \cdot B + A \cdot B' \cdot C' + A' \cdot B \cdot C$ (d) $F = A' \cdot B \cdot (C \cdot B \cdot A' + B \cdot C')$

(e) $F = X \cdot Y \cdot (X' \cdot Y \cdot Z + X \cdot Y' \cdot Z + X \cdot Y \cdot Z' + X' \cdot Y' \cdot Z)$ (f) $F = M \cdot N + M' \cdot N' \cdot P'$

(g) $F = (A + A') \cdot B + B \cdot A \cdot C' + C \cdot (A + B') \cdot (A' + B)$ (h) $F = X \cdot Y' + Y \cdot Z + Z' \cdot X$

4.9 Write the canonical sum and product for each of the following logic functions:

(a) $F = \Sigma_{X,Y}(1,2)$ (b) $F = \prod_{A,B}(0,1,2)$

(c) $F = \Sigma_{A,B,C}(2,4,6,7)$ (d) $F = \prod_{W,X,Y}(0,1,3,4,5)$

(e) $F = X + Y' \cdot Z'$ (f) $F = V' + (W' \cdot X)'$

4.10 Write the canonical sum and product for each of the following logic functions:

(a) $F = \Sigma_{X,Y,Z}(0,3)$ (b) $F = \prod_{A,B,C}(1,2,4)$

(c) $F = \Sigma_{A,B,C,D}(1,2,5,6)$ (d) $F = \prod_{M,N,P}(0,1,3,6,7)$

(e) $F = X' + Y \cdot Z' + Y \cdot Z'$ (f) $F = A'B + B'C + A$

4.11 If the canonical sum for an *n*-input logic function is also a minimal sum, how many literals are in each product term of the sum? Might there be any other minimal sums in this case?

4.12 Give two reasons why the cost of inverters is not included in the definition of "minimal" for logic minimization.

4.13 Using Karnaugh maps, find a minimal sum-of-products expression for each of the following logic functions. Indicate the distinguished 1-cells in each map.

(a) $F = \Sigma_{X,Y,Z}(1,3,5,6,7)$ (b) $F = \Sigma_{W,X,Y,Z}(1,4,5,6,7,9,14,15)$

(c) $F = \prod_{W,X,Y}(0,1,3,4,5)$ (d) $F = \Sigma_{W,X,Y,Z}(0,2,5,7,8,10,13,15)$

(e) $F = \prod_{A,B,C,D}(1,7,9,13,15)$ (f) $F = \Sigma_{A,B,C,D}(1,4,5,7,12,14,15)$

4.14 Find a minimal product-of-sums expression for each function in Drill 4.13 using the method of Section 4.3.6.

4.15 Find a minimal product-of-sums expression for the function in each of the following figures and compare its cost with the previously found minimal sum-of-products expression: (a) Figure 4-27; (b) Figure 4-29; (c) Figure 4-33.

4.16 Using Karnaugh maps, find a minimal sum-of-products expression for each of the following logic functions. Indicate the distinguished 1-cells in each map.

(a) $F = \Sigma_{A,B,C}(0,1,2,4)$ (b) $F = \Sigma_{W,X,Y,Z}(1,4,5,6,11,12,13,14)$

(c) $F = \prod_{A,B,C}(1,2,6,7)$ (d) $F = \Sigma_{W,X,Y,Z}(0,1,2,3,7,8,10,11,15)$

(e) $F = \Sigma_{W,X,Y,Z}(1,2,4,7,8,11,13,14)$ (f) $F = \prod_{A,B,C,D}(1,3,4,5,6,7,9,12,13,14)$

4.17 Find a minimal product-of-sums expression for each function in Drill 4.16 using the method of Section 4.3.6.

4.18 Find the complete sum for the logic functions in Drill 4.16(d) and (e).

4.19 Using Karnaugh maps, find a minimal sum-of-products expression for each of the following logic functions. Indicate the distinguished 1-cells in each map.

(a) $F = \Sigma_{W,X,Y,Z}(0,1,3,5,14) + d(8,15)$ (b) $F = \Sigma_{W,X,Y,Z}(0,1,2,8,11) + d(3,9,15)$

(c) $F = \Sigma_{A,B,C,D}(1,5,9,14,15) + d(11)$ (d) $F = \Sigma_{A,B,C,D}(1,5,6,7,9,13) + d(4,15)$

(e) $F = \Sigma_{W,X,Y,Z}(3,5,6,7,13) + d(1,2,4,12,15)$

4.20 Repeat Drill 4.19, finding a minimal product-of-sums expression for each logic function.

4.21 For each logic function in the two preceding exercises, determine whether the minimal sum-of-products expression equals the minimal product-of-sums expression. Also compare the circuit cost for realizing each of the two expressions.

4.22 For each of the following logic expressions, find all of the static hazards in the corresponding two-level AND-OR or OR-AND circuit, and design a hazard-free circuit that realizes the same logic function.

(a) $F = W \cdot X + W'Y'$ (b) $F = W \cdot X' \cdot Y' + X \cdot Y' \cdot Z + X \cdot Y$

(c) $F = W' \cdot Y + X' \cdot Y' + W \cdot X \cdot Z$ (d) $F = W' \cdot X + Y' \cdot Z + W \cdot X \cdot Y \cdot Z + W \cdot X' \cdot Y \cdot Z'$

(e) $F = (W + X + Y) \cdot (X' + Z')$ (f) $F = (W + Y'+Z') \cdot (W' + X' + Z') \cdot (X'+Y+Z)$

(g) $F = (W + Y + Z') \cdot (W + X' + Y + Z) \cdot (X' + Y') \cdot (X + Z)$

4.23 Write a set of ABEL test vectors for prime-number detector in Table 4-17.

4.24 Write a structural VHDL program (entity and architecture) for the alarm circuit in Figure 4-19.

4.25 Repeat Drill 4.24, using the dataflow style of description.

4.26 Repeat Drill 4.24, using a process and a behavioral description style.

4.27 Write a structural VHDL program corresponding to the NAND-gate based logic circuit in Figure 5-17.

Exercises

4.28 Design a non-trivial-looking logic circuit that contains a feedback loop but has an output that depends only on its current input.

4.29 Prove the combining theorem T10 without using perfect induction, but assuming that theorems T1–T9 and T1′–T9′ are true.

4.30 Show that the combining theorem T10 is just a special case of consensus (T11) used with covering (T9).

4.31 Prove that $(X + Y') \cdot Y = X \cdot Y$ *without* using perfect induction. You may assume that theorems T1–T11 and T1′–T11′ are true.

4.32 Prove that $(X + Y) \cdot (X' + Z) = X \cdot Z + X' \cdot Y$ *without* using perfect induction. You may assume that theorems T1–T11 and T1′–T11′ are true.

4.33 Show that an n-input AND gate can be replaced by $(n-1)$ 2-input AND gates. Can the same statement be made for NAND gates? Justify your answer.

4.34 How many physically different ways are there to realize $V \cdot W \cdot X \cdot Y \cdot Z$ using four 2-input AND gates (4/4 of a 74x08)? Justify your answer.

4.35 Use switching algebra to prove that tying together two inputs of an $(n+1)$-input AND or OR gate gives it the functionality of an n-input gate.

4.36 Prove DeMorgan's theorems (T13 and T13′) using finite induction.

4.37 Which logic symbol more closely approximates the internal realization of a TTL NOR gate, Figure 4-4(c) or (d)? Why?

4.38 Use the theorems of switching algebra to rewrite the following expression using as few inversions as possible (complemented parentheses are allowed):

$$B' \cdot C + A \cdot C \cdot D' + A' \cdot C + E \cdot B' + E \cdot (A + C) \cdot (A' + D')$$

4.39 Prove or disprove the following propositions:

(a) Let A and B be switching-algebra *variables*. Then $A \cdot B = 0$ and $A + B = 1$ implies that $A = B'$.

(b) Let X and Y be switching-algebra *expressions*. Then $X \cdot Y = 0$ and $X + Y = 1$ implies that $X = Y'$.

4.40 Prove Shannon's expansion theorems. (*Hint:* Don't get carried away; it's easy.)

generalized Shannon-expansion theorems

4.41 Shannon's expansion theorems can be generalized to "pull out" not just one but i variables so that a logic function can be expressed as a sum or product of 2^i terms. State the generalized Shannon expansion theorems.

4.42 Show how the generalized Shannon expansion theorems lead to the canonical sum and canonical product representations of logic functions.

Exclusive OR (XOR) gate

4.43 An *Exclusive OR (XOR) gate* is a 2-input gate whose output is 1 if and only if exactly one of its inputs is 1. Write a truth table, sum-of-products expression, and corresponding AND-OR circuit for the Exclusive OR function.

4.44 From the point of view of switching algebra, what is the function of a 2-input XOR gate whose inputs are tied together? How might the output behavior of a real XOR gate differ?

4.45 After completing the design and fabrication of an SSI-based digital system, a designer finds that one more inverter is required. However, the only spare gates in the system are a 3-input OR, a 2-input AND, and a 2-input XOR. How should the designer realize the inverter function without adding another IC?

4.46 Any set of logic-gate types that can realize any logic function is called a *complete set* of logic gates. For example, 2-input AND gates, 2-input OR gates, and inverters are a complete set, because any logic function can be expressed as a sum of products of variables and their complements, and AND and OR gates with any number of inputs can be made from 2-input gates. Do 2-input NAND gates form a complete set of logic gates? Prove your answer.

complete set

4.47 Do 2-input NOR gates form a complete set of logic gates? Prove your answer.

4.48 Do 2-input XOR gates form a complete set of logic gates? Prove your answer.

4.49 Define a two-input gate, other than NAND, NOR, or XOR, that forms a complete set of logic gates if the constant inputs 0 and 1 are allowed. Prove your answer.

4.50 Some people think that there are *four* basic logic functions, AND, OR, NOT, and *BUT*. Figure X4.50 is a possible symbol for a 4-input, 2-output *BUT gate*. Invent a useful, nontrivial function for the BUT gate to perform. The function should have something to do with the name (BUT). Keep in mind that, due to the symmetry of the symbol, the function should be symmetric with respect to the A and B inputs of each section and with respect to sections 1 and 2. Describe your BUT's function and write its truth table.

BUT
BUT gate

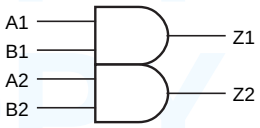


Figure X4.50

4.51 Write logic expressions for the Z1 and Z2 outputs of the BUT gate you designed in the preceding exercise, and draw a corresponding logic diagram using AND gates, OR gates, and inverters.

4.52 Most students have no problem using theorem T8 to "multiply out" logic expressions, but many develop a mental block if they try to use theorem T8′ to "add out" a logic expression. How can duality be used to overcome this problem?

4.53 How many different logic functions are there of n variables?

4.54 How many different 2-variable logic functions F(X,Y) are there? Write a simplified algebraic expression for each of them.

4.55 A *self-dual logic function* is a function F such that $F = F^D$. Which of the following functions are self-dual? (The symbol ⊕ denotes the Exclusive OR (XOR) operation.)

self-dual logic function
⊕

(a) $F = X$

(b) $F = \Sigma_{X,Y,Z}(0,3,5,6)$

(c) $F = X \cdot Y' + X' \cdot Y$

(d) $F = W \cdot (X \oplus Y \oplus Z) + W' \cdot (X \oplus Y \oplus Z)'$

(e) A function F of 7 variables such that F = 1 if and only if 4 or more of the variables are 1

(f) A function F of 10 variables such that F = 1 if and only if 5 or more of the variables are 1

4.56 How many self-dual logic functions of n input variables are there? (*Hint:* Consider the structure of the truth table of a self-dual function.)

4.57 Prove that any n-input logic function $F(X_1, \ldots, X_n)$ that can be written in the form $F = X_1 \cdot G(X_2, \ldots, X_n) + X_1' \cdot G^D(X_2, \ldots, X_n)$ is self-dual.

4.58 Assuming that an inverting gate has a propagation delay of 5 ns, and that a non-inverting gate has a propagation delay of 8 ns, compare the speeds of the circuits in Figure 4-24(a), (c), and (d).

4.59 Find the minimal product-of-sums expressions for the logic functions in Figures 4-27 and 4-29.

4.60 Use switching algebra to show that the logic functions obtained in Exercise 4.59 equal the AND-OR functions obtained in Figures 4-27 and 4-29.

4.61 Determine whether the product-of-sums expressions obtained by "adding out" the minimal sums in Figure 4-27 and 4-29 are minimal.

4.62 Prove that the rule for combining 2^i 1-cells in a Karnaugh map is true, using the axioms and theorems of switching algebra.

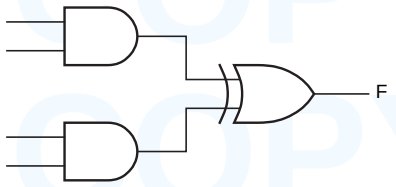
irredundant sum

4.63 An *irredundant sum* for a logic function F is a sum of prime implicants for F such that if any prime implicant is deleted, the sum no longer equals F. This sounds a lot like a minimal sum, but an irredundant sum is not necessarily minimal. For example, the minimal sum of the function in Figure 4-35 has only three product terms, but there is an irredundant sum with four product terms. Find the irredundant sum and draw a map of the function, circling only the prime implicants in the irredundant sum.

4.64 Find another logic function in Section 4.3 that has one or more nonminimal irredundant sums, and draw its map, circling only the prime implicants in the irredundant sum.

4.65 Draw a Karnaugh map and assign variables to the inputs of the AND-XOR circuit in Figure X4.65 so that its output is $F = \Sigma_{W,X,Y,Z}(6, 7, 12, 13)$. Note that the output gate is a 2-input XOR rather than an OR.

Figure X4.65



4.66 A 3-bit "comparator" circuit receives two 3-bit numbers, $P = P_2P_1P_0$ and $Q = Q_2Q_1Q_0$. Design a minimal sum-of-products circuit that produces a 1 output if and only if $P > Q$.

4.67 Find minimal multiple-output sum-of-products expressions for $F = \Sigma_{X,Y,Z}(0,1,2)$, $G = \Sigma_{X,Y,Z}(1,4,6)$, and $H = \Sigma_{X,Y,Z}(0,1,2,4,6)$.

4.68 Prove whether or not the following expression is a minimal sum. Do it the easiest way possible (algebraically, not using maps).

$$F = T' \cdot U \cdot V \cdot W \cdot X + T' \cdot U \cdot V' \cdot X \cdot Z + T' \cdot U \cdot W \cdot X \cdot Y' \cdot Z$$

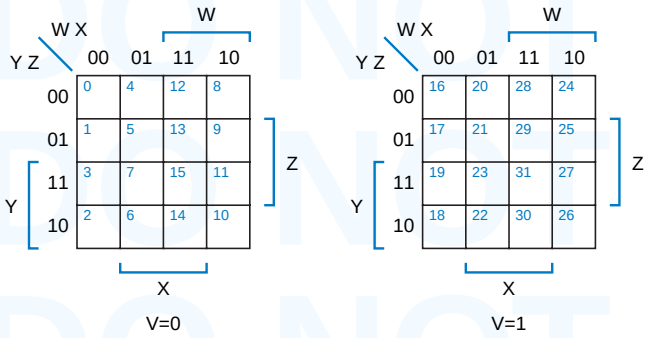


Figure X4.72

4.69 The text states that a truth table or equivalent is the starting point for traditional combinational minimization methods. A Karnaugh map itself contains the same information as a truth table. Given a sum-of-products expression, it is possible to write the 1s corresponding to each product term directly on the map without developing an explicit truth table or minterm list, and then proceed with the map-minimization procedure. In this way, find a minimal sum-of-products expression for each of the following logic functions:

(a) $F = X' \cdot Z + X \cdot Y + X \cdot Y' \cdot Z$

(b) $F = A' \cdot C' \cdot D + B' \cdot C \cdot D + A \cdot C' \cdot D + B \cdot C \cdot D$

(c) $F = W \cdot X \cdot Z' + W \cdot X' \cdot Y \cdot Z + X \cdot Z$

(d) $F = (X' + Y') \cdot (W' + X' + Y) \cdot (W' + X + Z)$

(e) $F = A \cdot B \cdot C' \cdot D' + A' \cdot B \cdot C' + A \cdot B \cdot D + A' \cdot C \cdot D + B \cdot C \cdot D'$

4.70 Repeat Exercise 4.69, finding a minimal product-of-sums expression for each logic function.

4.71 Derive the minimal product-of-sums expression for the prime BCD-digit-detector function of Figure 4-37. Determine whether or not the expression algebraically equals the minimal sum-of-products expression and explain your result.

4.72 A Karnaugh map for a 5-variable function can be drawn as shown in Figure X4.72. In such a map, cells that occupy the same relative position in the V = 0 and V = 1 submaps are considered to be adjacent. (Many worked examples of 5-variable Karnaugh maps appear in Sections 7.4.4 and 7.4.5.) Find a minimal sum-of-products expression for each of the following functions using a 5-variable map:

5-variable Karnaugh map

(a) $F = \Sigma_{V,W,X,Y,Z}(5,7,13,15,16,20,25,27,29,31)$

(b) $F = \Sigma_{V,W,X,Y,Z}(0,7,8,9,12,13,15,16,22,23,30,31)$

(c) $F = \Sigma_{V,W,X,Y,Z}(0,1,2,3,4,5,10,11,14,20,21,24,25,26,27,28,29,30)$

(d) $F = \Sigma_{V,W,X,Y,Z}(0,2,4,6,7,8,10,11,12,13,14,16,18,19,29,30)$

(e) $F = \prod_{V,W,X,Y,Z}(4,5,10,12,13,16,17,21,25,26,27,29)$

(f) $F = \Sigma_{V,W,X,Y,Z}(4,6,7,9,11,12,13,14,15,20,22,25,27,28,30) + d(1,5,29,31)$

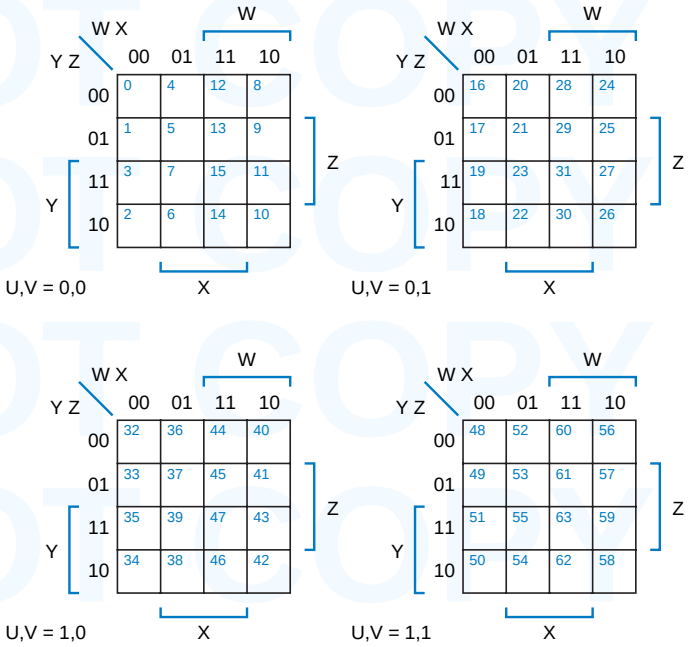


Figure X4.74

4.73 Repeat Exercise 4.72, finding a minimal product-of-sums expression for each logic function.

6-variable Karnaugh map

4.74 A Karnaugh map for a 6-variable function can be drawn as shown in Figure X4.74. In such a map, cells that occupy the same relative position in adjacent submaps are considered to be adjacent. Minimize the following functions using 6-variable maps:

(a) $F = \Sigma_{U,V,W,X,Y,Z}(1,5,9,13,21,23,29,31,37,45,53,61)$

(b) $F = \Sigma_{U,V,W,X,Y,Z}(0,4,8,16,24,32,34,36,37,39,40,48,50,56)$

(c) $F = \Sigma_{U,V,W,X,Y,Z}(2,4,5,6,12\text{–}21,28\text{–}31,34,38,50,51,60\text{–}63)$

4.75 There are $2n$ m-subcubes of an n-cube for the value $m = n - 1$. Show their text representations and the corresponding product terms. (You may use ellipses as required, e.g., 1, 2, …, n.)

4.76 There is just one m-subcube of an n-cube for the value $m = n$; its text representation is xx…xx. Write the product term corresponding to this cube.

4.77 The C program in Table 4-9 uses memory inefficiently because it allocates memory for a maximum number of cubes at each level, even if this maximum is never used. Redesign the program so that the `cubes` and `used` arrays are one-dimensional arrays, and each level uses only as many array entries as needed. (*Hint:* You can still allocate cubes sequentially, but keep track of the starting point in the array for each level.)

4.78 As a function of m, how many times is each distinct m-cube rediscovered in Table 4-9, only to be found in the inner loop and thrown away? Suggest some ways to eliminate this inefficiency.

4.79 The third `for`-loop in Table 4-9 tries to combine all m-cubes at a given level with all other m-cubes at that level. In fact, only m-cubes with x's in the same positions can be combined, so it is possible to reduce the number of loop iterations by using a more sophisticated data structure. Design a data structure that segregates the cubes at a given level according to the position of their x's, and determine the maximum size required for various elements of the data structure. Rewrite Table 4-9 accordingly.

4.80 Estimate whether the savings in inner-loop iterations achieved in Exercise 4.80 outweighs the overhead of maintaining a more complex data structure. Try to make reasonable assumptions about how cubes are distributed at each level, and indicate how your results are affected by these assumptions.

4.81 Optimize the `Oneone` function in Table 4-8. An obvious optimization is to drop out of the loop early, but other optimizations exist that eliminate the `for` loop entirely. One is based on table look-up and another uses a tricky computation involving complementing, Exclusive ORing, and addition.

4.82 Extend the C program in Table 4-9 to handle don't-care conditions. Provide another data structure, `dc[MAX_VARS+1][MAX_CUBES]`, that indicates whether a given cube contains only don't-cares, and update it as cubes are read and generated.

4.83 (*Hamlet circuit.*) Complete the timing diagram and explain the function of the circuit in Figure X4.83. Where does the circuit get its name?

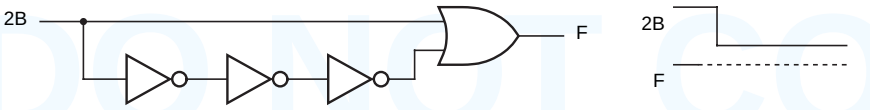


Figure X4.83

4.84 Prove that a two-level AND-OR circuit corresponding to the complete sum of a logic function is always hazard free.

4.85 Find a four-variable logic function whose minimal sum-of-products realization is not hazard free, but for which there exists a hazard-free sum-of-products realization with fewer product terms than the complete sum.

4.86 Starting with the `WHEN` statements in the ABEL program in Table 4-14, work out the logic equations for variables `X4` through `X10` in the program. Explain any discrepancies between your results and the equations in Table 4-15.

4.87 Draw a circuit diagram corresponding to the minimal two-level sum-of-products equations for the alarm circuit, as given in Table 4-12. On each inverter, AND gate, and OR gate input and output, write a pair of numbers ($t0, t1$), where $t0$ is the test number from Table 4-25 that detects a stuck-at-0 fault on that line, and $t1$ is the test number that detects a stuck-at-1 fault.

4.88 Write a dataflow-style VHDL program (entity and architecture) corresponding to the full-adder circuit in Figure 5-86.

4.89 Using the entity that you designed in Exercise 4.88, write a structural VHDL program for a 4-bit ripple adder using the structure of Figure 5-87.

4.90 Using the entity that you defined in Exercise 4.88, write a structural VHDL program for a 16-bit ripple adder along the lines of Figure 5-87. Use a `generate` statement to create the 16 full adders and their signal connection.

4.91 Rewrite the prime-number-detector architecture of Table 4-63 using a `while` statement.