No man who is occupied in doing a very difficult thing, and doing it very well, loses his self-respect.

—George Bernard Shaw

Exercises

4.1 *(PALs and PLAs)* Our third word problem case study described a logical "function unit" that computed the eight combinational functions of the inputs A and B: constant 1, A OR B, A NAND B, A XOR B, A XNOR B, A AND B, A NOR B, and constant 0. Show how to implement this using a programmable logic array (with five inputs and one output). Draw the AND array and OR array, and indicate which connections must be made to implement the function. For each output from the AND array, indicate along the wire the product term it is implementing.

4.2 *(PALs and PLAs)* Show how to implement the BCD-to-seven-segment LED decoder using a PAL such as that in the P16H8 structure: 10 inputs, 8 OR gate outputs, and 7 product terms per OR gate. Use the shorthand notation developed in Section 4.1.

4.3 *(Transmission Gates)* A priority encoder circuit has inputs and outputs that are numbered from 1 to N. The operation of the circuit is defined as follows. Determine the highest-numbered input that is asserted. Only the output with the same index as the highest-numbered input that is asserted should be asserted. If none of the inputs are asserted, all of the outputs read out a 0. A truth table for a two-input priority encoder circuit is shown in Figure Ex4.3.

I_2	I_1	O_2	O_1
1	X	1	0
0	1	0	1
0	0	0	0

Figure Ex4.3 Two-input priority encoder truth table.

a. Implement a two-input priority encoder circuit using discrete inverters and NAND gates only.

b. Implement the same function as in (a), except using CMOS transmission gates and inverters. Counting transistors, which implementation is more transistor efficient?

4.4 *(Transmission Gates)* Consider the description of the following circuit, often called a function generator. The circuit has two data inputs, A and B, four control inputs, G_0, G_1, G_2, and G_3, and one output F. F is defined as

$$F = G_0\overline{A}\,\overline{B} + G_1\overline{A}B + G_2A\overline{B} + G_3AB$$

a. Describe how to "program" this function to implement the following 10 functions: A AND B, A OR B, A NAND B, A NOR B, 0, 1, A, B, A XOR B, and A XNOR B by placing the appropriate values on the G_i inputs.

b. Implement this function using CMOS transmission gates and inverters only. (*Hint*: Use the inputs A and B to control the transmission gates; consider how to route the appropriate G_i to the output F.)

4.10 *(Multiplexer Logic)* Because 32:1 multiplexers do not exist in standard component catalogs, design a two-stage multiplexer network that realizes the six-variable function

$$f(A, B, C, D, E, F) = \Sigma\, m(3, 7, 12, 14, 15, 19, 23, 27, 28, 29, 31, 35, 39, 44, 45, 46, 48, 49, 50, 52, 53, 55, 56, 57, 59)$$

a. How many TTL packages are used?

b. How many TTL packages are required to implement this function using conventional inverters and NAND gates in a two-level network?

4.11 *(Multiplexer Logic)* You are working for a company that produces clones of famous name computers. Your assignment is to copy the circuit shown in Figure Ex4.11, which is a component of the computer. Because of copyright laws, your boss is very emphatic that the new circuit not resemble the old one. Also, your boss's brother-in-law's electronics wholesale firm has a special deal on multiplexers. So the boss wants you to use only 4:1 multiplexers in your circuit.

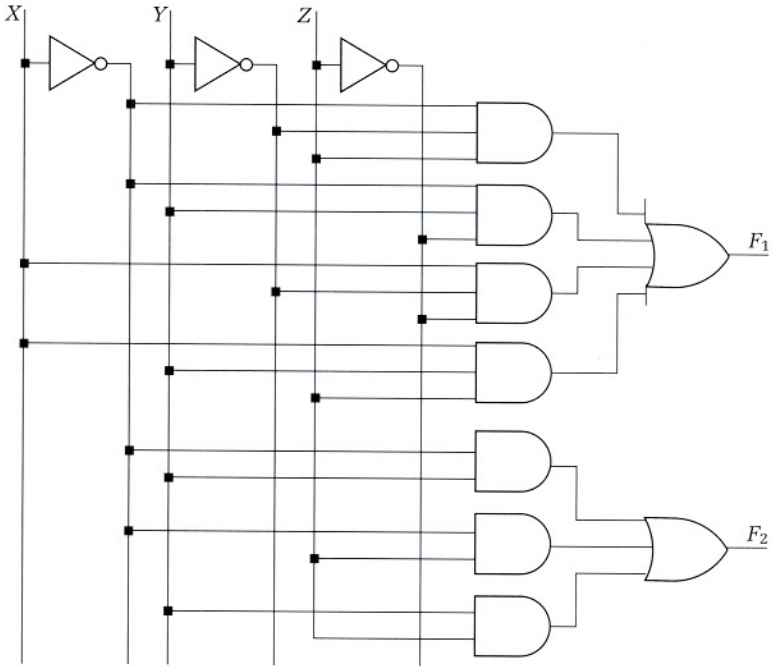


Figure Ex4.11 Circuit to be re-engineered with multiplexers.

4.12 *(Decoder Implementation)* Demonstrate how to implement a 6:64 decoder using generic 2:4 and 4:16 decoders.

4.13 *(Decoder Implementation)* We have seen how to implement decoders using AND gates and NOR gates. Show how to implement the truth table of the 74138 4:16 decoder, including the three-input enable logic, G_1, $\overline{G}_{2A}$, $\overline{G}_{2B}$, using only NAND gates and inverters.

4.14 *(Decoder Implementation)* A decoder together with an OR gate connected to its output terminals can be used in the synthesis of combinational networks.

a. Implement the function $f(A, B, C, D) = \overline{A}\overline{B}D + \overline{A}BD + A\overline{C}\overline{D} + AC\overline{D}$ (not necessarily in minimized form) using one 4:16 decoder and a very large fan-in OR gate.

b. Compare the resulting number of ICs with a solution using discrete gates only.

4.15 *(Implementation Methods)* Given the following function in sum of products form (not necessarily minimized):

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

Implement the function F using

a. An 8:1 multiplexer.

b. A 4:16 decoder with a 16-input OR gate.

c. A 16-word ROM.

d. A PLA-like structure using the notation of Section 4.1.

4.16 *(Implementation Methods)* Given a four-input Boolean function, $f(A, B, C, D) = \Sigma\, m(0, 3, 5, 7, 11, 12, 13, 15)$

a. Implement the function using a 16:1 multiplexer.

b. Implement using an 8:1 multiplexer (use D, $\overline{D}$ as MUX data inputs and A, B, C as MUX control inputs).

c. Implement the function using a 4:1 multiplexer. (*Hint:* Place A and B on the select inputs. Assume $\overline{C}$, $\overline{D}$ are available and use an OR gate to form one of the inputs to the multiplexer.)

d. Implement the function using a 4:16 decoder and an OR gate.

4.17 *(Implementation Methods)* Given the function $f(A, B, C, D) = (\overline{A} + B)\ (\overline{A} + C + D)\ (A + \overline{C} + D)$ in minimized product of sums form and the don't-care set $D = \{M_0, M_2, M_9, M_{10}\}$, do the following:

a. Write f in canonical product of sums form.

b. Write f in canonical sum of products form.

c. Write f in minimized sum of products form.

d. Show how to implement f with a single three-stack by three-input AND-OR-Invert gate.

e. Show how to implement f with an 8:1 multiplexer.

4.18 *(Implementation Methods)* Given the three functions X, Y, and Z, defined by $X(A, B, C, D) = \Sigma\, m(1, 2, 3, 5, 7, 9, 11, 13, 15)$, $Y(A, B, C, D) = \Pi\, M(2, 3, 4, 5, 6, 7, 8, 10, 12, 14)$, and $Z(A, B, C, D) = \Sigma\, m(0, 1, 2, 3, 5, 7)$,

a. Find the minimum sum of products form for each of these functions. How many unique product terms are there in your answer?

b. Find an alternative sum of products form for X, Y, and Z that minimizes the number of unique product terms to implement all three functions simultaneously. How many unique product terms do you find in this implementation?

c. Show how to implement your solution to part (b) in a PLA structure.

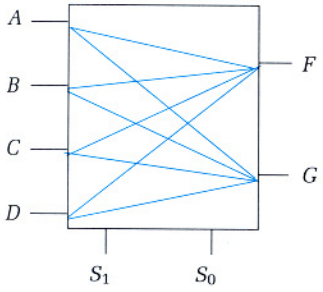
4.19 *(Implementation Methods)* Consider the implementation of a circuit with four data inputs, A, B, C, D, two outputs, F and G, and two control inputs S_1 and S_0. The block diagram and functional truth table are shown in Figure Ex4.19.

a. Assume that S_1, $\overline{S}_1$, S_0, and $\overline{S}_0$ are available. Implement F and G using AND and OR gates only.

b. Assume that S_1, $\overline{S}_1$, S_0, and $\overline{S}_0$ are available. Draw the schematic implementing the functions using NAND gates and non-inverting tri-state buffers with active low enables. (Your solution should not be a simple restatement of your answer for part (a).)

c. Assume that S_1, $\overline{S}_1$, S_0, and $\overline{S}_0$ are available. Draw the schematics that implement F and G using open-collector NAND gates and conventional inverters only.

d. Implement F and G using transmission gates only.



S_1	S_0	F	G
0	0	A	D
0	1	B	C
1	0	C	B
1	1	D	A

Figure Ex4.19 Specification for Exercise 4.19.

4.20 *(ROM-based Implementation)* Design a schematic for a read-only memory subsystem of size 65536 words by 8 bits wide, using 2764 8K by 8-bit ROMs.

a. Use a single 3:8 decoder and inverters.

b. Use a single 2:4 decoder and inverters. Is there a clever way to make use of the output enable inputs to the ROM as well as the chip select lines?

4.21 *(Word Problems)* A logic network has four inputs ($Input_0$, $Input_1$, $Input_2$, $Input_3$) and two outputs ($Output_0$, $Output_1$). At least one of the inputs is always asserted high. If a given input line has a logic 1 applied to it, the output signals will encode its index in binary. For example, if $Input_2$ is asserted, the output reads $Output_1 = 1$, $Output_0 = 0$. If two or more inputs are at logic 1, the output will be set according to which input has the highest index ($Input_3 > Input_2 > Input_1 > Input_0$).

a. Fill in the truth table for this function.

b. Fill in K-maps for $Output_1$ and $Output_0$, and find the Boolean expression for the minimum sum of products implementation.

4.22 *(Word Problems)* You are to implement a combinational multiplier. It has two 2-bit inputs and a 4-bit output. The first 2-bit input is represented by the variables *A*, *B*; the second 2-bit input is represented by *C*, *D*. The outputs are *W*, *X*, *Y*, *Z*, from the most significant bit to the least.

a. Complete a truth table that describes the functional behavior of the multiplier.

b. Find the minimum sum of products forms for the outputs using the K-map method.

4.23 *(Word Problems)* An *n*-input majority function asserts its output whenever more than half of its inputs are asserted. You are to implement a seven-input majority function, which will assert its output whenever four or more of its inputs are asserted.

Don't panic just because this is a seven-variable function. Build it up as a multilevel function whose subfunctions each have less than six variables. As a block diagram, it looks like Figure Ex4.23. Circuits #1 and #2 tally the number of their inputs that are asserted, providing the count in binary on the outputs (*V*, *Y* are the most significant bits; *W*, *Z* are the least significant bits). Based on these second-level inputs, *Q* determines if more than four or more of the original inputs are 1.

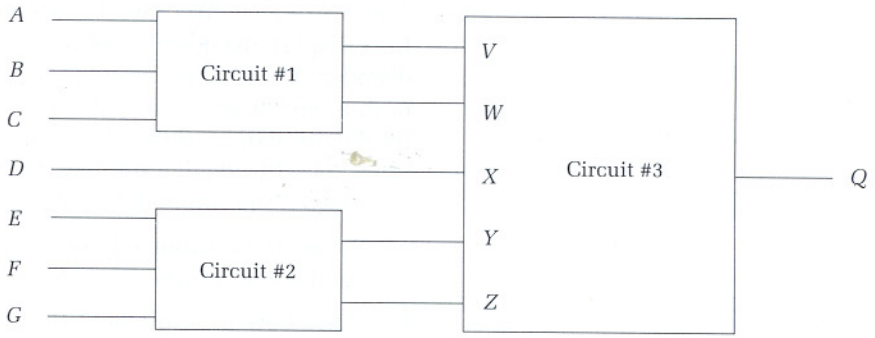


Figure Ex4.23 Block diagram for the tally circuit.

a. Find the minimized sum of products form for circuit #1 (circuit #2 is identical). The functions V and W should look familiar. What do they implement?

b. Complete a five-variable truth table for circuit #3.

c. Find the minimum sum of products form for Q using the K-map method.

d. Find the minimum product of sums form for Q using the K-map method.

4.24 *(Word Problems)* You are to design a converter that maps a 4-bit binary code into a 4-bit Gray code. The 4-bit Gray code sequence is defined as follows: 0000, 0001, 0011, 0010, 0110, 0111, 0100, 1100, 1101, 1111, 1110, 1010, 1011, 1001, 1000. Give the truth table, and show how to implement this code converter as a ROM circuit and as a PLA circuit.

4.25 *(Case Studies)* How close can the rods be placed on the conveyor belt without invalidating the solution to Case Study 1?

4.26 *(Case Studies)* Verify that the *misII* equations for the BCD-to-seven-segment LED decoder really do map onto the same on-set as the *espresso* equations. This can be accomplished by expanding the equations into two-level sum of products form and filling in four-variable K-maps from the equations thus derived. How do the K-maps compare with those of Figure 4.71?

4.27 *(Case Studies)* We wish to extend the BCD-to-seven-segment LED display decoder to become a hexadecimal LED display decoder. The LED representations of the hex digits 0, 1, 2, 3, 4, 5, 6, 7, 8, and 9 are exactly the same as for the equivalent BCD digits. Figure Ex4.27 shows how the segment displays should be illuminated to denote the hex digits *A* (1010), *B* (1011), *C* (1100), *D* (1101), *E* (1110), and *F* (1111).

a. Obtain the minimized sum of products implementations for the display inputs.

b. Show how to implement the logic for the extended design as a PLA with four inputs and seven outputs. Draw the AND array and OR array, and indicate which connections must be made to implement the function. For each output from the AND array, indicate along the wire the product term it is implementing.

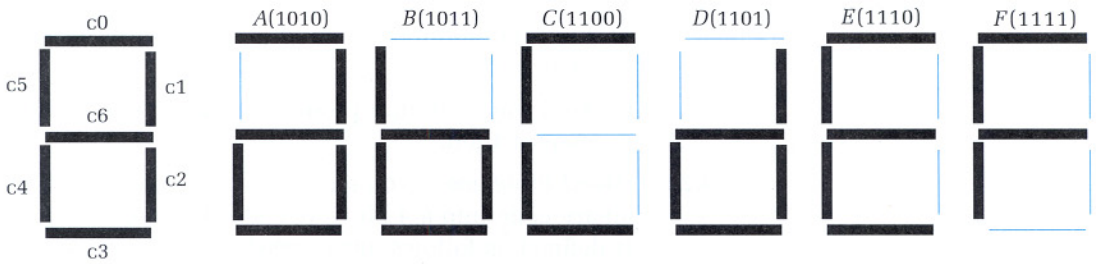


Figure Ex4.27 Hexadecimal displays.

4.28 *(Case Studies)* Design to the gate level an arithmetic and logic function unit based on the circuit of Case Study 3. The modifications to the specification are as follows. We add one additional input *Cin* and one extra output *Cout*. When the control inputs $C_0C_1C_2 = 000$, *F* generates the function *A* plus *B* plus *Cin* and *Cout* is the carry-out result from the binary addition. The rest of the control signal settings generate the same functions as before. (*Cout* is a don't care for these.)

4.29 *(Case Studies)* In the function table for the barrel shifter (Figure 4.82), the input data rotates from right to left. How does the implementation change if the data are rotated from left to right instead? Your answer should consider both the logic gate and transmission gate implementation styles.

4.30 *(Tally Circuit)* Examine the truth table for the two-input Tally function (Figure 4.23) very carefully. Is there a way to implement the three outputs without using an XOR gate? (*Hint*: Think of a way to implement the One output in terms of the Zero and Two functions.) How does this revised gate-level implementation compare with the switch implementation?