CHAPTER 8

PROGRAMMABLE LOGIC DEVICES

8.1 Introduction

When implementing combinational circuits using discrete gates such as AND, OR, NAND, NOR, XOR, and XNOR, the circuit structure and the required number of gates vary from function to function. These gates are usually available in standard small-scale integrated (SSI) circuits. An integrated circuit (IC) is a large number of electronic components interconnected and packed into a single small semiconductor chip. Circuit implementation using decoders and multiplexers with added external gates is more structured than using discrete gates. Decoders and multiplexers are available in medium-scale integrated (MSI) circuits. This chapter introduces the implementation of combinational circuits using programmable logic devices (PLD), which have the highest degree of structure among the three different types of implementations. Programmable logic devices are large-scale integrated (LSI) circuits.

Three types of programmable logic devices are introduced in this chapter. They are read-only memory (ROM), programmable logic array (PLA), and programmable array logic (PAL). Each of the PLDs comprises of two arrays of gates: an AND gate array and an OR gate array.

8.2 Notations

Some notations for PLDs are shown in Figure 8.1. There are two types of connection between two perpendicular lines: programmable and non-programmable. Non-programmable connection, also known as hard-wired connection, is denoted by a dot at the intersection of the two perpendicular lines. The connection is permanent and cannot be removed. For programmable connection, a switching element is used to connect two lines, which is shown by an "×" at the intersection of these two lines. They are not connected if an "×" is absent. Fuses or transistors are used as switching elements. When fuses are the switching elements, the undesired connections can be blown out by passing a high current through them.

A different notation is also used for the inputs of programmable gates as shown in Figure 8.1. For an n-input programmable gate, only one line is drawn and connected to the gate. The n-inputs are denoted by n perpendicular lines to the single line connected to the gate, with an "×" at each intersection. A gate input can be disconnected by removing the "×". A phase splitter is used to generate the true form and the complemented form of an input.

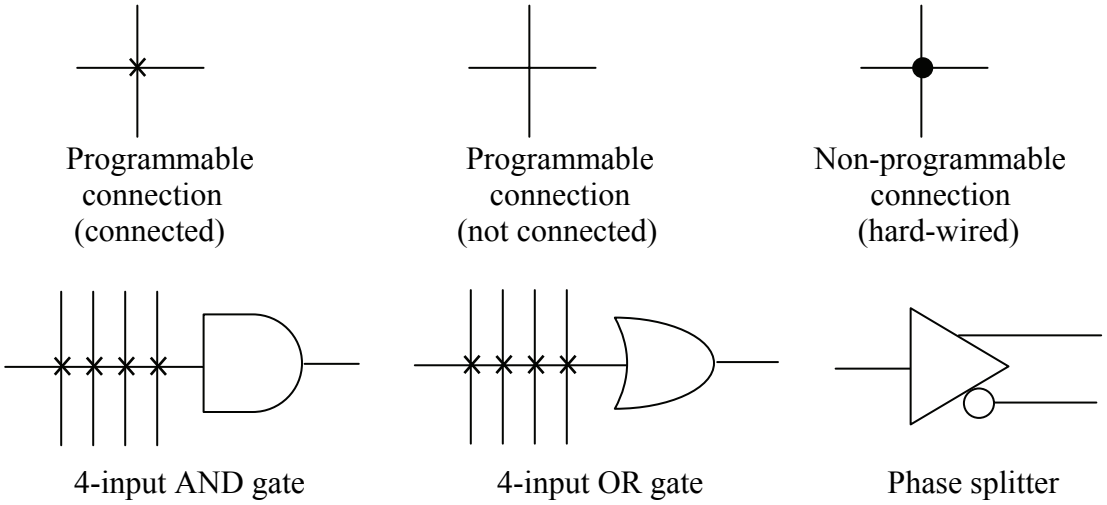


Figure 8.1 PLD notations

8.3 Read-Only-Memory (ROM)

The structure of a ROM is given in Figure 8.2. The AND gate array is fixed and non-programmable. The OR gate array is programmable. The AND gate array is in fact a decoder. The n inputs to a ROM are also the inputs to its AND array. The products generated by the AND array are connected to each and every OR gates in the OR array. Thus there are 2^n inputs connected to each OR gate in the OR array of a ROM when it has not yet been programmed. The outputs of the OR gate array are the outputs of a ROM.

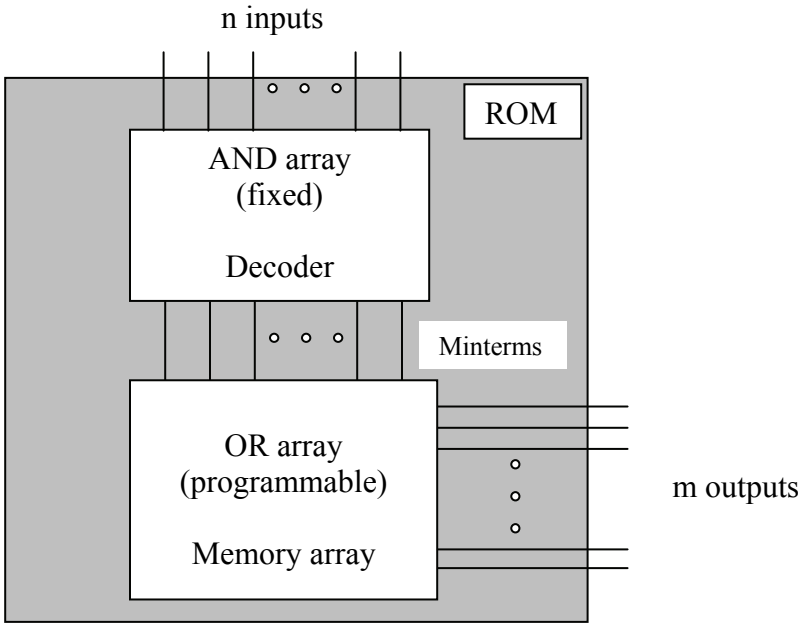


Figure 8.2 Structure of ROM.

Since the AND array of a ROM is a decoder, all the canonical products or minterms of a function are available at the AND array outputs if the AND array inputs are the variables of the function. The function can be implemented by removing unnecessary inputs to an OR gate in the OR array. In implementing functions with ROMs, functions have to be in minterm list form.

❖ Example 8.1

The following four functions are implemented using a ROM. Since f_1, f_2, f_3, and f_4 are 3-variable functions, there should be three inputs to the ROM and at least four OR gates in the OR array. The minterm list representations of the functions are given below. The implementation is shown in Figure 8.3

$f_1(A, B, C) = \Sigma\, m(0, 2, 3, 4)$
$f_2(A, B, C) = \Sigma\, m(1, 2, 3, 5, 6)$
$f_3(A, B, C) = \Sigma\, m(2, 6, 7)$
$f_4(A, B, C) = \Sigma\, m(2, 3, 5)$

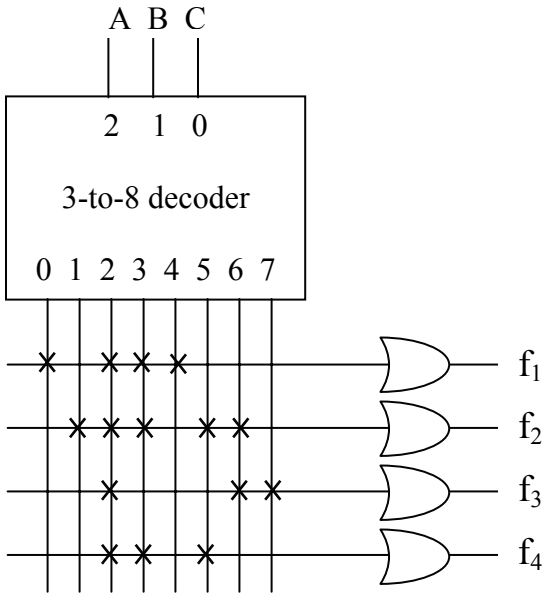


Figure 8.3 Implementation of functions by ROM.

Data can also be stored in ROMs. The n inputs of a ROM serve as the addresses to the 2^n words stored in the ROM. The range of addresses is from 0 to $(2^n - 1)$. If there are m outputs, then each word has m bits. The size of such a ROM is $2^n \times m$. It is called a 2^n word by m-bit ($2^n \times m$) ROM. If the 8×4 ROM in Figure 8.3 is a memory, the data in a certain location can be retrieved by applying the address of this location to the inputs of the ROM. For example, if ABC = 011, $f_1f_2f_3f_4 = 1101$. The word at the location with an

address of 3 is 1101. The eight 4-bit words stored in the ROM at addresses 0 to 7 are 1000, 0100, 1111, 1101, 1000, 0101, 0110, and 0010 respectively.

❖ Example 8.2

The implementation of a seven-segment display using a ROM is illustrated in this example. A seven-segment display consists of seven light-emitting diodes (LEDs) as shown in Figure 8.4(a). It is used to display a decimal digit by illuminating some selected segments. As shown in Figure 8.4(b), $b_3b_2b_1b_0$ is the four binary input combination for a decimal digit. The seven outputs of the circuit will drive the seven segments. An illuminated segment is driven by an output signal of 1. The circuit also includes an output E. It is asserted when the inputs are not a valid decimal digit. The letter "E" will also be displayed if the input combination is not a decimal digit.

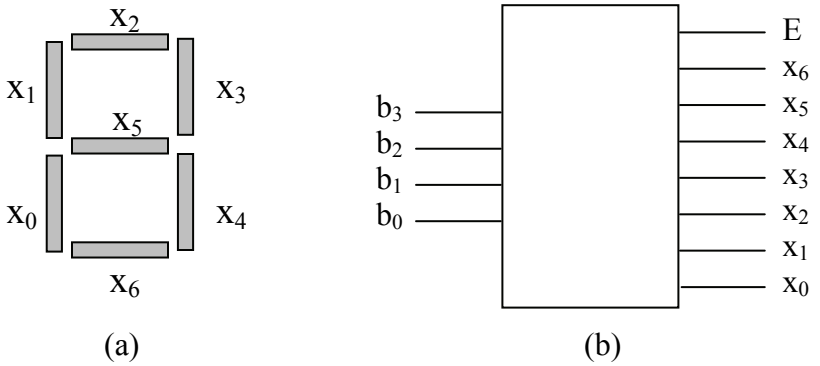


Figure 8.4 (a) Structure of 7-segment display. (b) Block diagram for a 7-segment display circuit.

Table 8.1 is the truth table for the circuit that generates the outputs to drive the seven segments. When the inputs $b_3b_2b_1b_0 = 0000$, a decimal 0 will be displayed. All segments should be lit except x_5. Therefore $x_0 = x_0 = x_1 = x_2 = x_3 = x_4 = x_6 = 1$, $x_5 = 0$. E = 0 because 0000 is a valid input combination. The values at the outputs for all other input combinations can be derived in a similar manner using the display shown in Figure 8.5 for all the ten decimal digits and the letter E. The segments not driven by a value of 1 are removed from the display. The implementation is shown in Figure 8.6 using the minterms in Table 8.1.

Figure 8.5 Seven-segment display of decimal digits and the letter E.

Table 8.1 Truth table for 7-segment display circuit.

Decimal digit	$b_3\ b_2\ b_1\ b_0$	$E\ x_6\ x_5\ x_4\ x_3\ x_2\ x_1\ x_0$
0	0 0 0 0	0 1 0 1 1 1 1 1
1	0 0 0 1	0 0 0 1 1 0 0 0
2	0 0 1 0	0 1 1 0 1 1 0 1
3	0 0 1 1	0 1 1 1 1 1 0 0
4	0 1 0 0	0 0 1 1 1 0 1 0
5	0 1 0 1	0 1 1 1 0 1 1 0
6	0 1 1 0	0 1 1 1 0 1 1 1
7	0 1 1 1	0 0 0 1 1 1 0 0
8	1 0 0 0	0 1 1 1 1 1 1 1
9	1 0 0 1	0 1 1 1 1 1 1 0
Invalid	1 0 1 0	1 1 1 0 0 1 1 1
Invalid	1 0 1 1	1 1 1 0 0 1 1 1
Invalid	1 1 0 0	1 1 1 0 0 1 1 1
Invalid	1 1 0 1	1 1 1 0 0 1 1 1
Invalid	1 1 1 0	1 1 1 0 0 1 1 1
Invalid	1 1 1 1	1 1 1 0 0 1 1 1

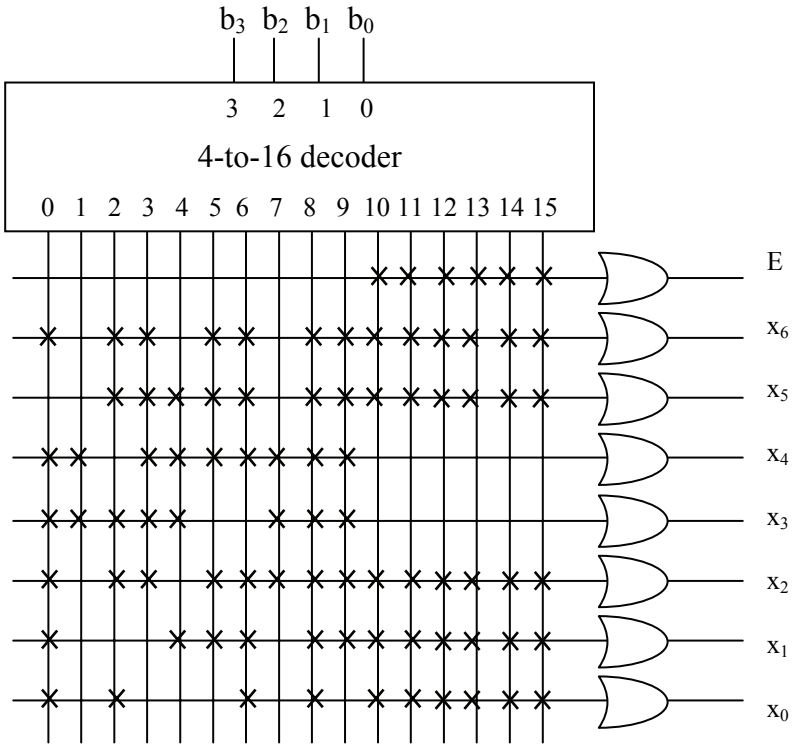


Figure 8.6 Implementation of a 7-segment display circuit using ROM.

8.4 Programmable Logic Array (PLA)

Implementation of functions using ROMs becomes less economical when the number of variables increases. Every time the number of variables increases by one, the size of the AND gate array is doubled. Many of the AND array outputs may not be needed and are wasted. The number of AND gates in an AND array can be minimized by using programmable logic arrays (PLA). The structure of PLA is shown in Figure 8.7. The AND array in a PLA is programmable. The number of product terms generated is unrelated to the number of inputs. The OR array is the same as that of a ROM. All the products generated from the AND array are connected to each and every one of the gates in the OR array. Because the AND array is programmable and the products generated by the AND array can be shared by all the OR gates, minimization technique for multiple-output circuits should be used to reduce the total number of products in the simplest sum-of-products expressions.

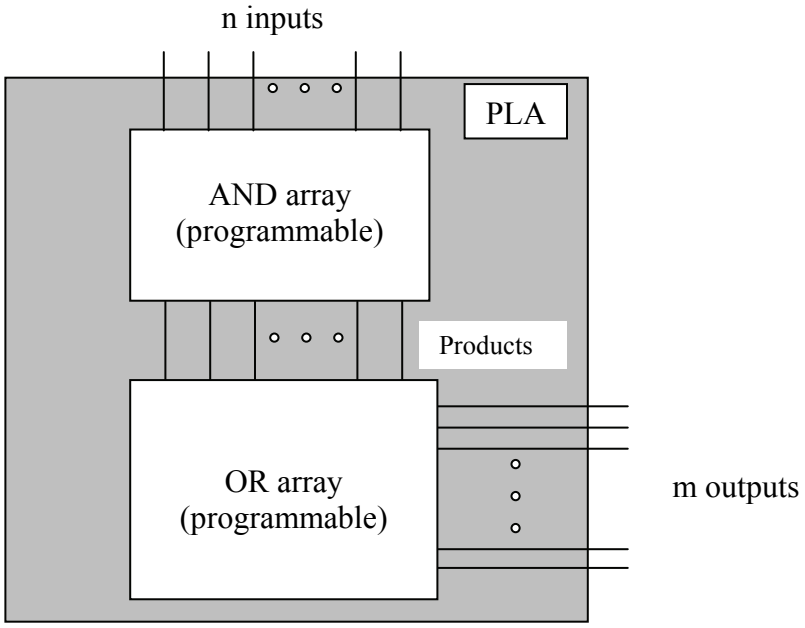


Figure 8.7 Structure of PLA.

❖ Example 8.3

The four 3-variable functions implemented by a ROM in Figure 8.3 are implemented using a PLA in this example. The simplest sum-of-products expressions for the functions are

$f_1(A, B, C) = \Sigma\, m(0, 2, 3, 4) = B'C' + A'B$
$f_2(A, B, C) = \Sigma\, m(1, 2, 3, 5, 6) = BC' + B'C + A'B$
$f_3(A, B, C) = \Sigma\, m(2, 6, 7) = BC' + AB$
$f_4(A, B, C) = \Sigma\, m(2, 3, 5) = A'B + AB'C$

To determine the size of the AND array, the number of distinct products in all the sum-of-products expressions is counted. There are six of them: B'C', A'B, BC', B'C, AB, and A'BC. They are generated by programming the AND array. The implementation is given in Figure 8.8.

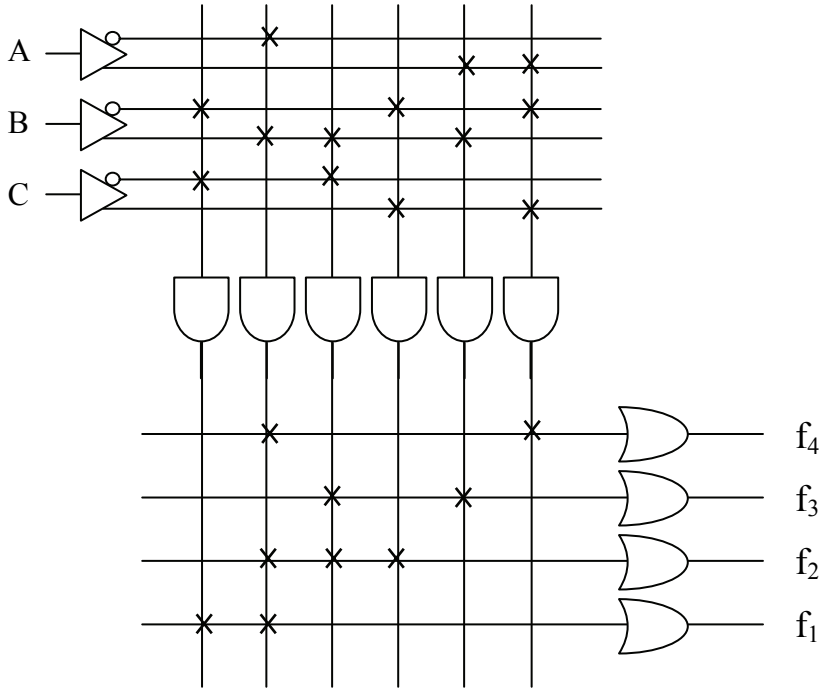


Figure 8.8 Implementation of functions by PLA.

8.5 Programmable Array Logic (PAL)

PAL is a device with a programmable AND gate array identical to that of PLA. However, the OR gate array is fixed and not programmable. Every product generated by the AND array is connected to one and only gate in the OR array. The structure of PAL is shown in Figure 8.9. The diagram in Figure 8.10 shows a PAL with twelve gates in the AND array and four gates in the OR array. The numbers of inputs to the four OR gates are 2, 2, 4, and 4.

❖ Example 8.4

The PAL in Figure 8.10 is used to implement the following 4-variable functions.

X = A'C + C'D
Y = A'C'D' + AD + AB'C
Z = BD + A'C + A'B'D' + ABC' + ACD

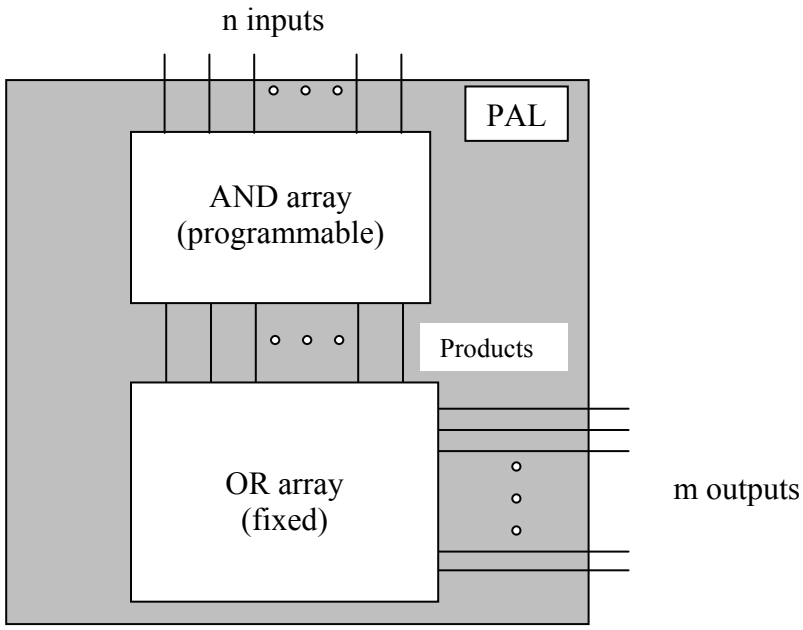


Figure 8.9 Structure of PAL.

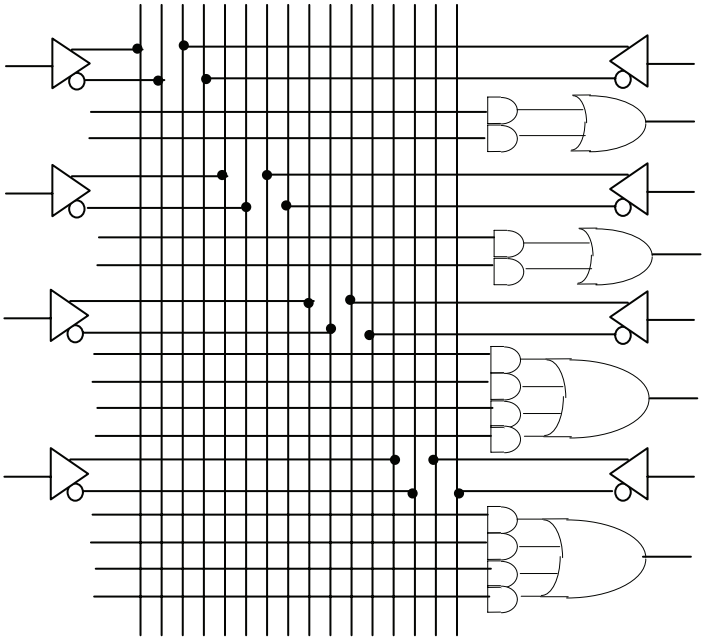


Figure 8.10 Logic diagram of PAL.

The implementation is shown in Figure 8.11. The number of products for each function is mapped to one OR gate. It is seen that X can be implemented by one of the 2-input OR gates. Y has three products and cannot be realized by a 2-input OR gate. A 4-input OR gate should be used. The fourth input of this OR gate is 0. By leaving all the inputs to an AND gate intact, the output of the AND gate is 0. Instead of placing an "×"

at every intersection, a single "×" is placed inside the AND gate symbol. Z has five products and there is no OR gate with five inputs. Fortunately, the unused 2-input OR gate can be used to generate BD + A'C, which is denoted as P. P is connected to an input pin, which is programmed in the AND array and becomes a product connected to an input of the unused 4-input OR gate, leaving the other three AND gates for A'B'D', ABC', and ACD.

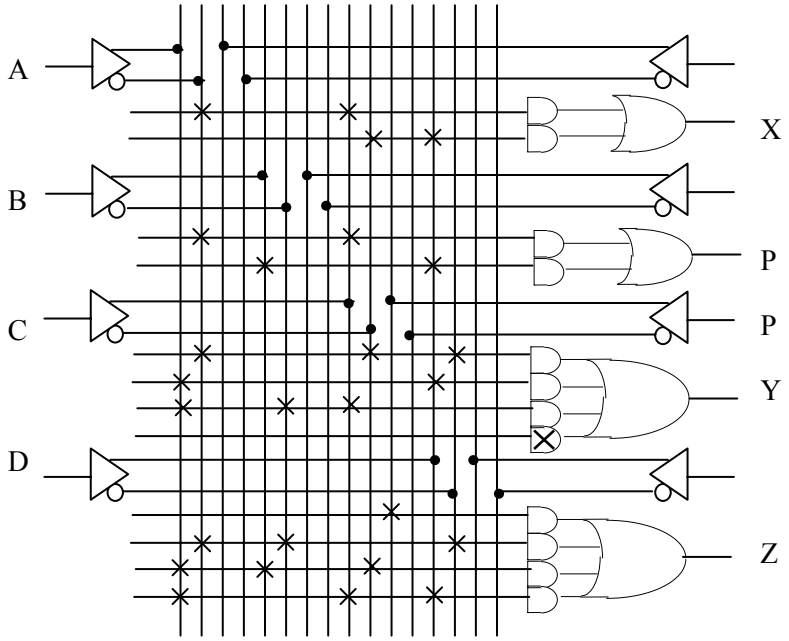


Figure 8.11 Implementation of the functions in Example 8.4 using PAL.

PROBLEMS

1. Design a circuit to convert a 4-bit BCD code to a 4-bit excess-3 code using a 16-word by 4-bit (16 x 4) ROM.

2. Implement the following expressions by a 16 x 4 ROM.

   W = ACD + BD' + C'D'
   X = AB'D + A'C' + BC + C'D'
   Y = A'C' + ACD
   Z = CD + A'C' + AB'D + BD' + C'D'

3. Implement the following functions using a PLA with four inputs, seven gates in the AND array, and four gates in the OR array.

$f_1(A, B, C, D) = \Sigma\, m(3, 5, 7, 12, 13, 14)$
$f_2(A, B, C, D) = \Sigma\, m(0, 1, 2, 3, 5, 12, 14)$
$f_3(A, B, C, D) = \Sigma\, m(0, 2, 3, 8, 10) + d\,(7, 9, 11)$
$f_4(A, B, C, D) = \Sigma\, m\,(0, 1, 5, 7, 13, 15) + d\,(2, 3, 4)$

Figure P8.1

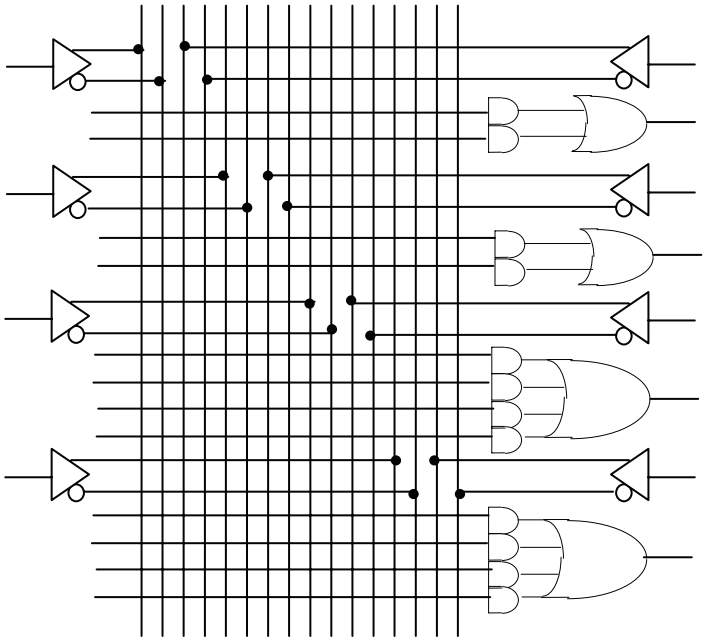
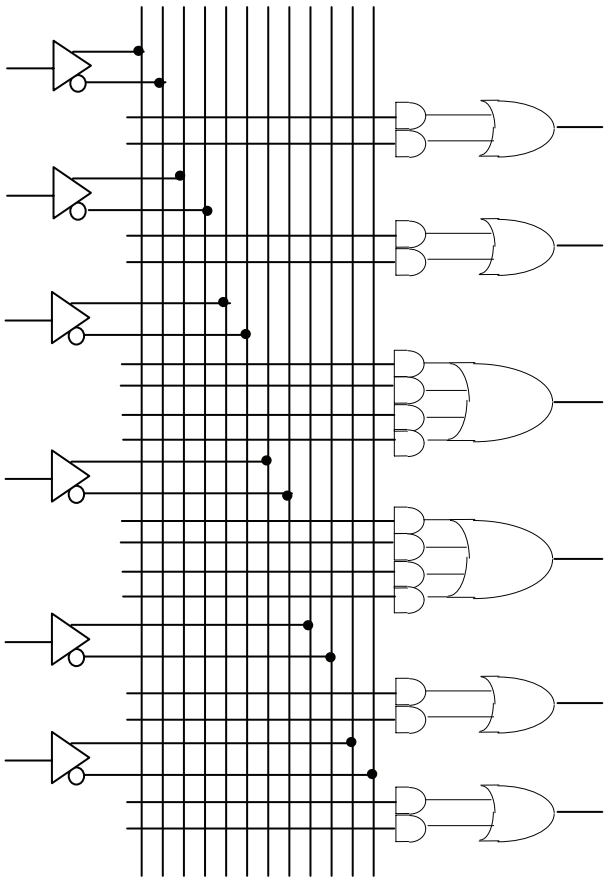


Figure P8.2



4. Use the PLA in Problem 3 to implement the functions in Problem 2.

5. Implement the functions in Problem 3 using the PAL in Figure P8.1.

6. Implement W, X, Y, Z in Problem 2 and the following expression using the PAL in Figure P8.2.

$$V = (A \oplus B \oplus C) + A'$$