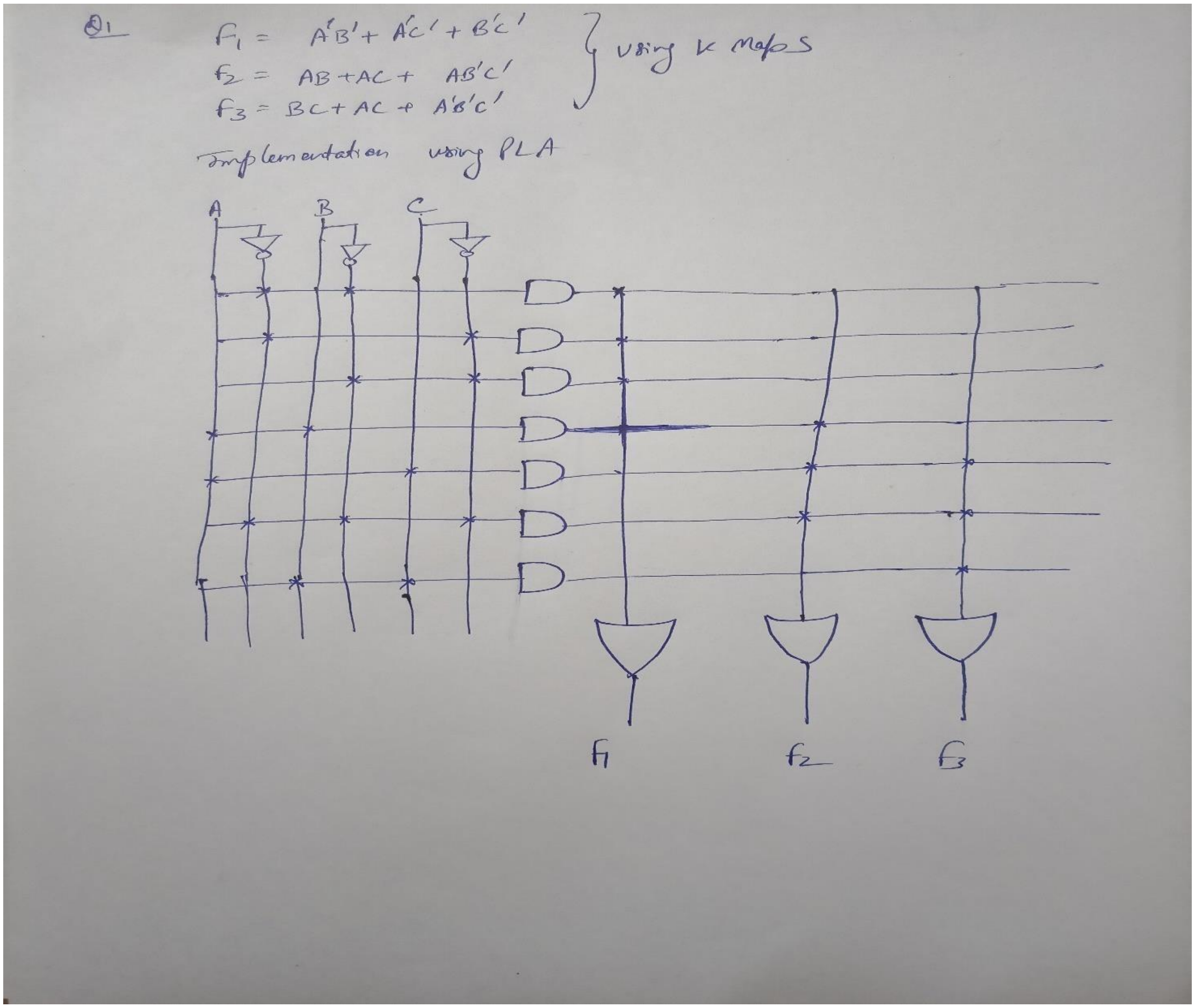
ANS 1.



Q1

$f_1 = A'B' + A'C' + B'C'$

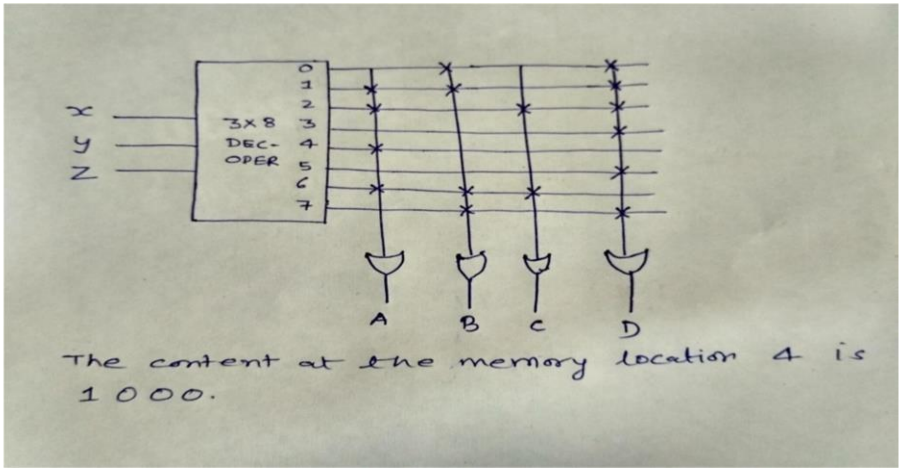
$f_2 = AB + AC + AB'C'$

$f_3 = BC + AC + A'B'C'$

} using K maps

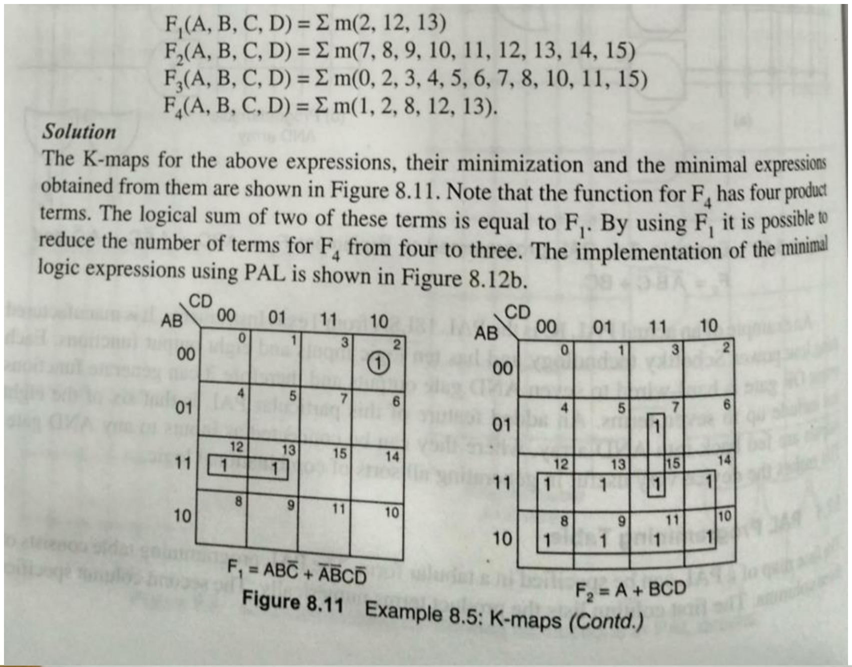
Implementation using PLA

ANS 2



The content at the memory location 4 is 1000.

ANS 3.



$F_1(A, B, C, D) = \Sigma\, m(2, 12, 13)$
$F_2(A, B, C, D) = \Sigma\, m(7, 8, 9, 10, 11, 12, 13, 14, 15)$
$F_3(A, B, C, D) = \Sigma\, m(0, 2, 3, 4, 5, 6, 7, 8, 10, 11, 15)$
$F_4(A, B, C, D) = \Sigma\, m(1, 2, 8, 12, 13).$

Solution

The K-maps for the above expressions, their minimization and the minimal expressions obtained from them are shown in Figure 8.11. Note that the function for F_4 has four product terms. The logical sum of two of these terms is equal to F_1. By using F_1 it is possible to reduce the number of terms for F_4 from four to three. The implementation of the minimal logic expressions using PAL is shown in Figure 8.12b.

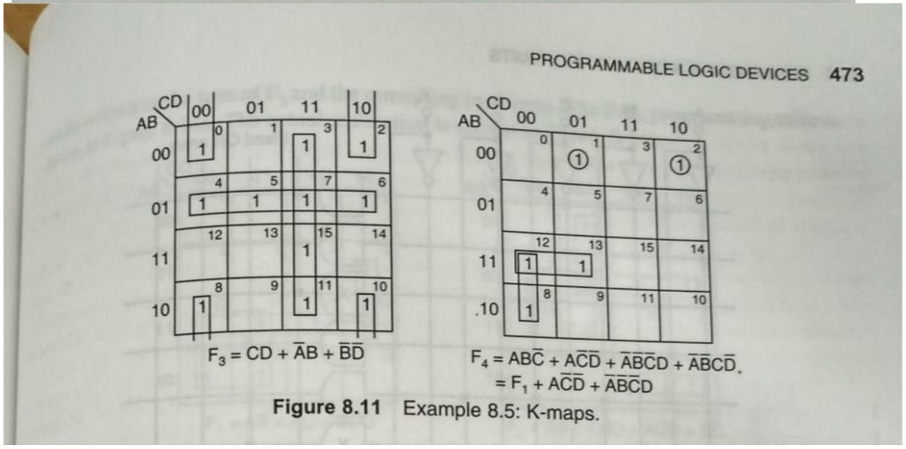
AB \ CD	00	01	11	10
00	0	1	3	2 ①
01	4	5	7	6
11	12 1	13 1	15	14
10	8	9	11	10

$F_1 = AB\bar{C} + \bar{A}\bar{B}C\bar{D}$

AB \ CD	00	01	11	10
00	0	1	3	2
01	4	5	7 1	6
11	12 1	13 1	15 1	14 1
10	8 1	9 1	11 1	10 1

$F_2 = A + BCD$

Figure 8.11 Example 8.5: K-maps *(Contd.)*



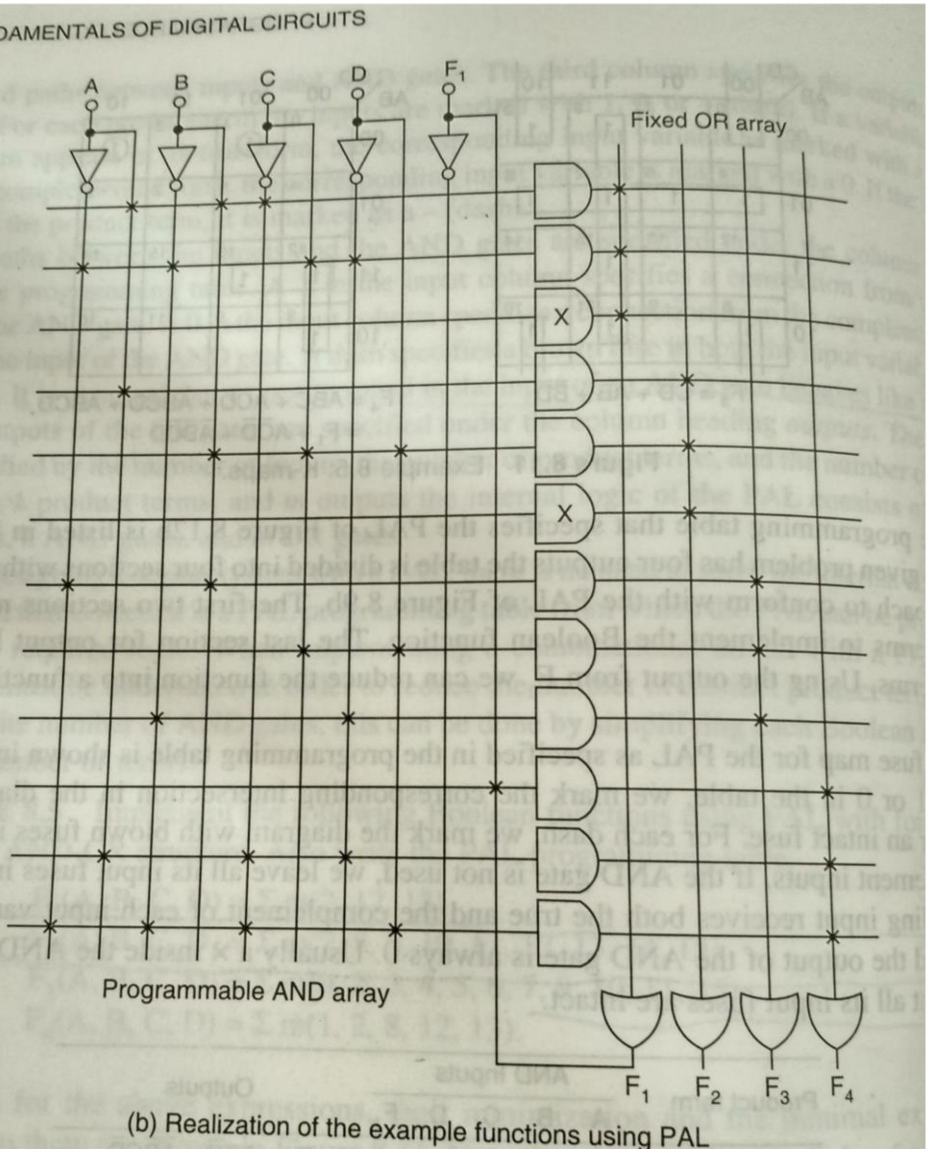
AB \ CD	00	01	11	10
00	0 1	1	3 1	2 1
01	4 1	5 1	7 1	6 1
11	12	13	15 1	14
10	8 1	9	11 1	10 1

$F_3 = CD + \bar{A}B + \bar{B}\bar{D}$

AB \ CD	00	01	11	10
00	0	1 ①	3	2 ①
01	4	5	7	6
11	12 1	13 1	15	14
10	8 1	9	11	10

$F_4 = AB\bar{C} + A\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D + \bar{A}\bar{B}C\bar{D}.$
$= F_1 + A\bar{C}\bar{D} + \bar{A}\bar{B}\bar{C}D$

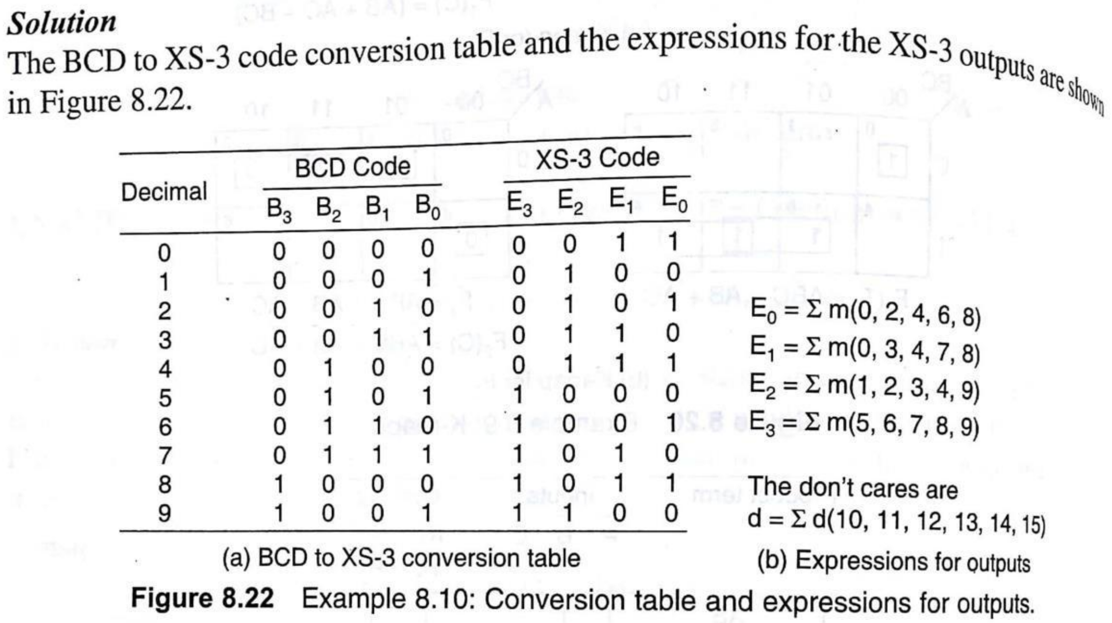
Figure 8.11 Example 8.5: K-maps.



(b) Realization of the example functions using PAL

Ans 4 8 OR gates and each OR gate is having 32 inputs.

Ans 5.



Solution

The BCD to XS-3 code conversion table and the expressions for the XS-3 outputs are shown in Figure 8.22.

Decimal	BCD Code				XS-3 Code			
	B_3	B_2	B_1	B_0	E_3	E_2	E_1	E_0
0	0	0	0	0	0	0	1	1
1	0	0	0	1	0	1	0	0
2	0	0	1	0	0	1	0	1
3	0	0	1	1	0	1	1	0
4	0	1	0	0	0	1	1	1
5	0	1	0	1	1	0	0	0
6	0	1	1	0	1	0	0	1
7	0	1	1	1	1	0	1	0
8	1	0	0	0	1	0	1	1
9	1	0	0	1	1	1	0	0

(a) BCD to XS-3 conversion table

$E_0 = \Sigma\, m(0, 2, 4, 6, 8)$

$E_1 = \Sigma\, m(0, 3, 4, 7, 8)$

$E_2 = \Sigma\, m(1, 2, 3, 4, 9)$

$E_3 = \Sigma\, m(5, 6, 7, 8, 9)$

The don't cares are

$d = \Sigma\, d(10, 11, 12, 13, 14, 15)$

(b) Expressions for outputs

Figure 8.22 Example 8.10: Conversion table and expressions for outputs.

5

$E_0 = \overline{B_0}$

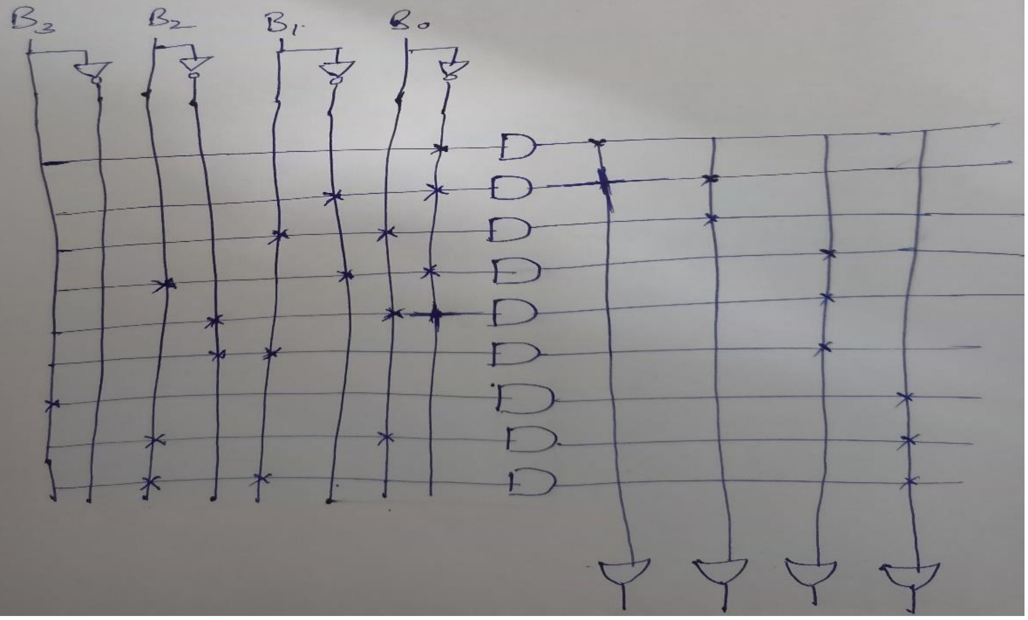
$E_1 = \overline{B_1}\,\overline{B_0} + B_1 B_0$

$E_2 = B_2 \overline{B_1}\,\overline{B_0} + \overline{B_2} B_0 + \overline{B_2} B_1$

$E_3 = B_3 + B_2 B_0 + B_2 B_1$

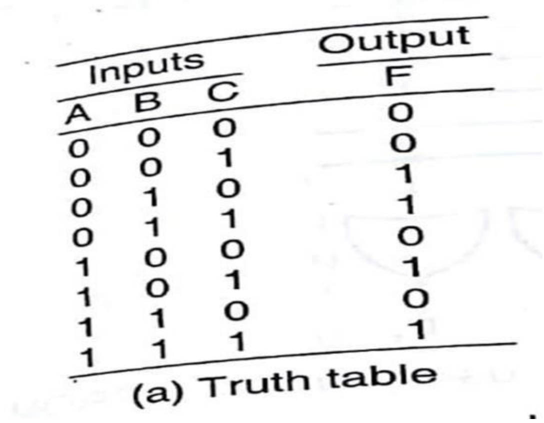
By solving K maps

Implementation using PLA



Ans 6. 16K * 4 = 64K RAM is of 64K. So, 64 * 8 = 512K. Hence, 512/64 = 8.

ANS 7.



Inputs			Output
A	B	C	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

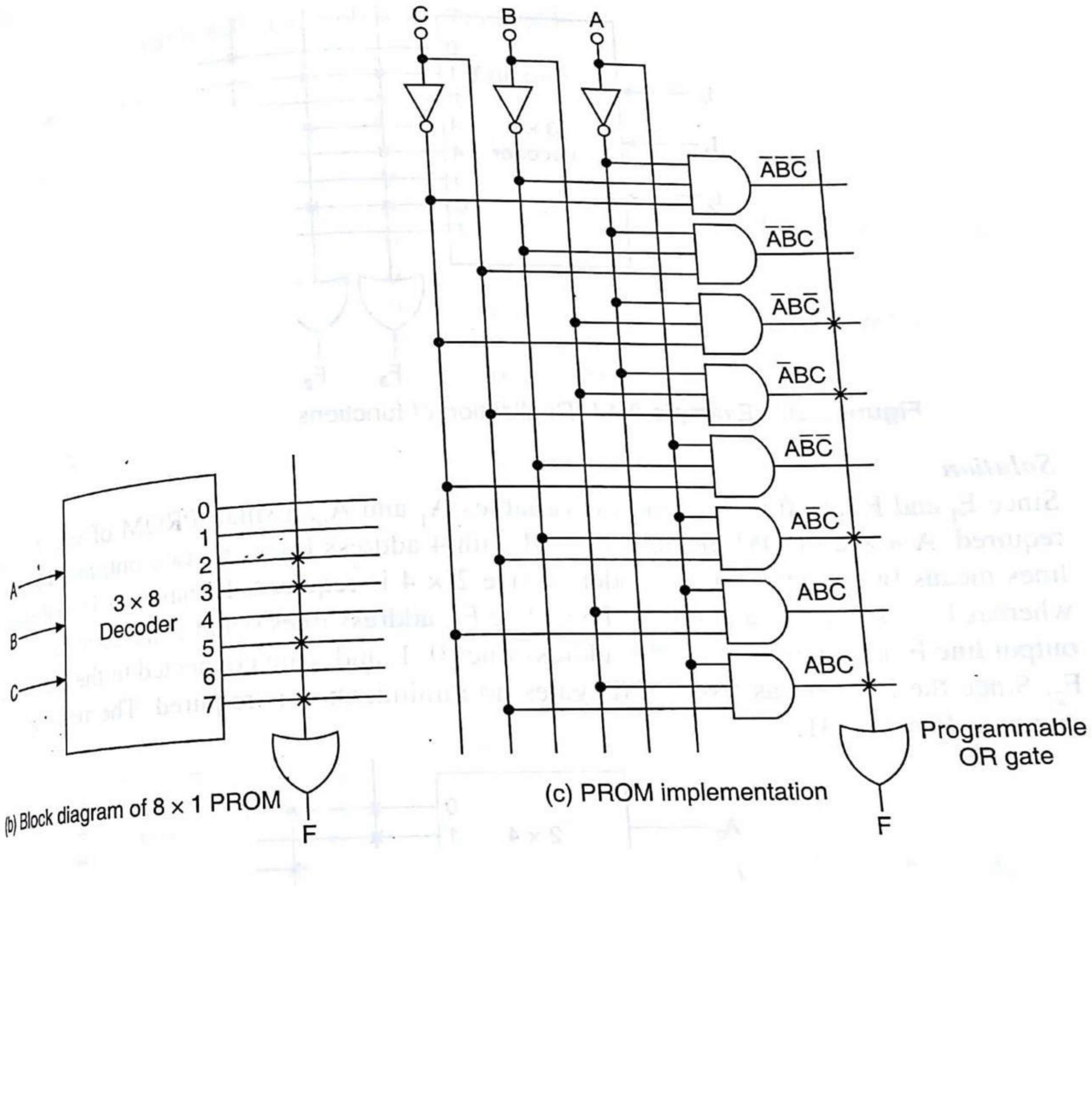
(a) Truth table

Solution

From the truth table shown in Figure 8.29a, we observe that the logic function is

$$F = \Sigma\, m(2, 3, 5, 7) = \bar{A}B\bar{C} + \bar{A}BC + A\bar{B}C + ABC$$

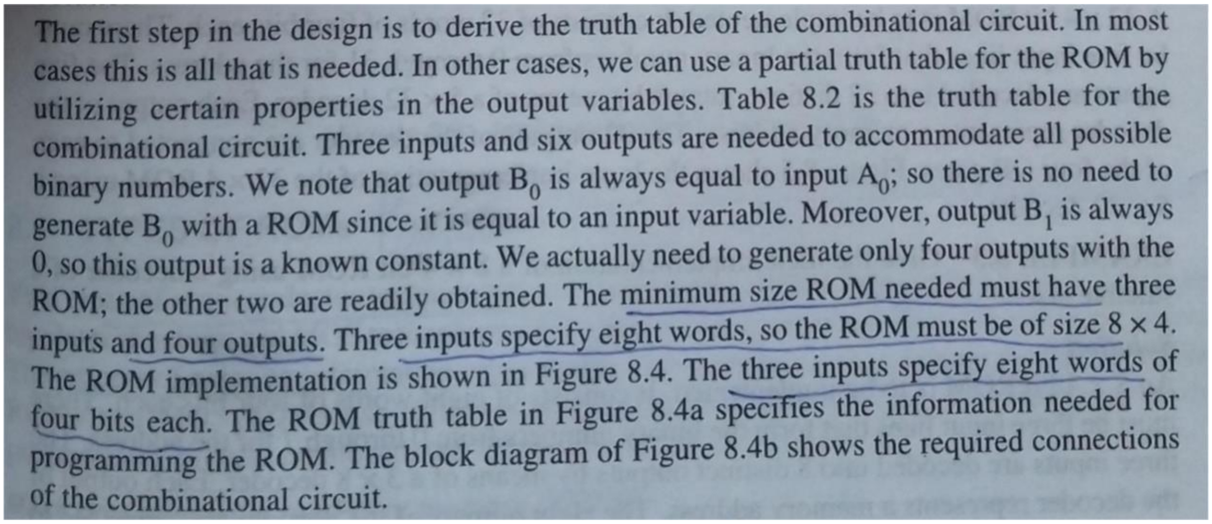
No simplification of the expression is to be done for realization with a PROM. Eight AND gates (one for each minterm) and one OR gate (to obtain the SOP form of the single output) are required for its implementation. Figure 8.29c shows the programmed PROM in the simplified connection format of a PLA. To realize F, address lines 2, 3, 5 and 7 are connected to F. Logic 1 or a 0 is stored at every address combination corresponding to a combination of the input variables for which the function equals a 1 or a 0. An 8 × 1 PROM means a PROM with 8 address lines and one output. So a 3 × 8 decoder is required as shown in Figure 8.29b.



(b) Block diagram of 8 × 1 PROM

(c) PROM implementation

ANS 8.



The first step in the design is to derive the truth table of the combinational circuit. In most cases this is all that is needed. In other cases, we can use a partial truth table for the ROM by utilizing certain properties in the output variables. Table 8.2 is the truth table for the combinational circuit. Three inputs and six outputs are needed to accommodate all possible binary numbers. We note that output B_0 is always equal to input A_0; so there is no need to generate B_0 with a ROM since it is equal to an input variable. Moreover, output B_1 is always 0, so this output is a known constant. We actually need to generate only four outputs with the ROM; the other two are readily obtained. The minimum size ROM needed must have three inputs and four outputs. Three inputs specify eight words, so the ROM must be of size 8 × 4. The ROM implementation is shown in Figure 8.4. The three inputs specify eight words of four bits each. The ROM truth table in Figure 8.4a specifies the information needed for programming the ROM. The block diagram of Figure 8.4b shows the required connections of the combinational circuit.

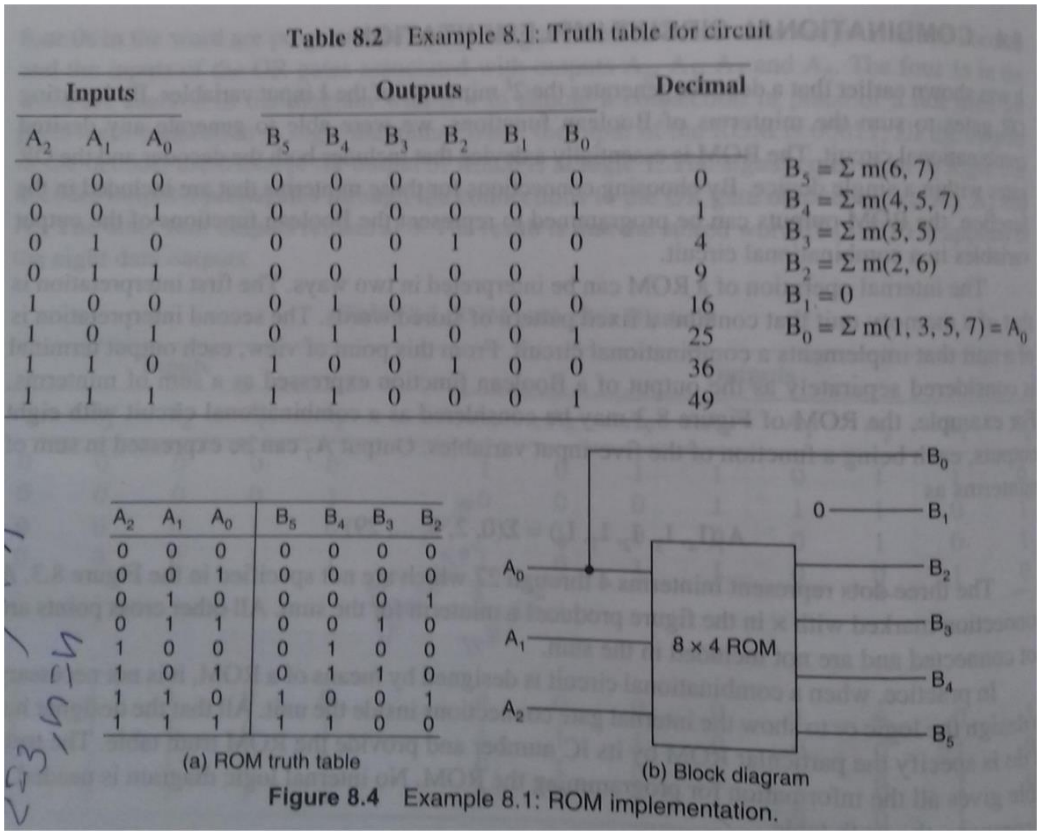


Table 8.2 Example 8.1: Truth table for circuit

Inputs			Outputs						Decimal
A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	
0	0	0	0	0	0	0	0	0	0
0	0	1	0	0	0	0	0	1	1
0	1	0	0	0	0	1	0	0	4
0	1	1	0	0	1	0	0	1	9
1	0	0	0	1	0	0	0	0	16
1	0	1	0	1	1	0	0	1	25
1	1	0	1	0	0	1	0	0	36
1	1	1	1	1	0	0	0	1	49

$B_5 = \Sigma\, m(6, 7)$

$B_4 = \Sigma\, m(4, 5, 7)$

$B_3 = \Sigma\, m(3, 5)$

$B_2 = \Sigma\, m(2, 6)$

$B_1 = 0$

$B_0 = \Sigma\, m(1, 3, 5, 7) = A_0$

A_2	A_1	A_0	B_5	B_4	B_3	B_2
0	0	0	0	0	0	0
0	0	1	0	0	0	0
0	1	0	0	0	0	1
0	1	1	0	0	1	0
1	0	0	0	1	0	0
1	0	1	0	1	1	0
1	1	0	1	0	0	1
1	1	1	1	1	0	0

(a) ROM truth table

A_0
A_1
A_2
8 × 4 ROM
0
B_0
B_1
B_2
B_3
B_4
B_5

(b) Block diagram

Figure 8.4 Example 8.1: ROM implementation.