(d) $f(x_1, \ldots, x_6) = \sum$ (1, 3, 5, 7, ...

(e) $f(x_1, \ldots, x_6) =$ 37, 38, 39, 44, 45, 46, 47, ...

(f) $f(x_1, \ldots, x_6) = \sum$ (5, 7, 13, 15, 16, 18, 21, 23, 24, 26, 29, 31, 32, ...
45, 47, 53, 55, 56, 57, 58, 59)

(g) $f(x_1, \ldots, x_6) = \sum$ (0, 1, 4, 7, 8, 10, 12, 17, 19, 21, 23, 33, 35, 37, 39, 41, 43,
52, 54, 60, 62) + $\sum_d$ (2, 3, 9, 15, 28, 29, 30, 31, 45, 47, 53,
55, 61, 63)

6. (a) Derive the Quine–McCluskey tabular procedure for minimizing a switching function in the product-of-sums form.
   (b) Apply this procedure to minimize the following functions:
   (1) $f(x_1, \ldots, x_5) = \prod$ (0, 1, 2, 3, 8, 9, 10, 11, 17, 19, 21, 23, 25, 27, 29, 31)
   (2) $f(x_1, \ldots, x_5) = \prod$ (0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 25, 27, 29, 31)
   (3) $f(x_1, \ldots, x_6) = \prod$ (4, 5, 6, 7, 8, 9, 10, 11, 16, 19, 24, 25, 41, 43, 45, 47, 49, 51,
53, 55) · $\prod_d$ (1, 3, 12, 14, 17, 27, 48, 50, 57, 59, 61, 63)

1.6 Function Minimization of Multiple-Output Circuits

So far we have discussed the minimization of the single switching function by the Karnaugh method and the Quine–McCluskey method. In practice, many logic design problems involve designing switching circuits to have more than one output. For these problems, which are too large or complicated to be solved by the Karnaugh method, it is necessary to turn to a tabular method for either hand computation or digital-computer usage. The multiple-switching-function minimization criterion is as follows:

1. Each minimized function should have as many terms in common with those in other minimized functions as possible.
2. Each minimized function should have a minimum number of product (sum) terms and no product (sum) term of it can be replaced by a product (sum) term with few variables.

The procedure for minimizing a set of m switching functions according to the above minimization criterion using the Quine–McCluskey method is described below.

Step 1 Find the prime implicants of the m switching functions. In finding the prime implicants of the m switching functions, we use a single table to find the prime implicants of the m functions simultaneously by using the tag. Suppose that we want to minimize the set of three output functions

$$f_1(x_1, x_2, x_3, x_4) = \sum (1, 2, 3, 5, 7, 8, 9, 12, 14) \quad (1.6.1)$$

$$f_2(x_1, x_2, x_3, x_4) = \sum (0, 1, 2, 3, 4, 6, 8, 9, 10, 11) \quad (1.6.2)$$

$$f_3(x_1, x_2, x_3, x_4) = \sum (1, 3, 5, 7, 8, 9, 12, 13, 14, 15) \quad (1.6.3)$$

The single table that can be used to find the prime implicants of the three functions simultaneously is shown in Fig. 1.6.1(a). Notice that the first column of this table includes the minterms (represented in decimal numbers) of all three functions. They are grouped into groups as usual according to the number of 1's in their binary representations, which are shown in the second column. The third column consists of a tag for each minterm (row) for indicating which of the output functions include the minterm. Each symbol of the tag corresponds to one of the output functions and is either 0 or 1, depending on whether the corresponding minterm is not included in the output function or is included in the output function. For example, the tag of the first row is

f_1	f_2	f_3
0	1	0

which indicates that the minterm 0 is included in the output function f_2 but is not included in the output functions f_1 and f_3.

The first, second, and third reductions of the implicants to form prime implicants are the same as those in the single-output technique. The rule for obtaining the tag of each row of the tables of Fig. 1.6.1(b), (c), and (d) is as follows:

Symbol of the Tag of Row i in the (k − 1)th Reduction Table	*Symbol of the Tag of Row j in the (k − 1)th Reduction Table*	*Symbol of the Tag of the Row in the kth Reduction Table Obtained from Rows i and j of the (k − 1)th Reduction Table*
0	0	0
0	1	0
1	0	0
1	1	1

Notice that there is a 1 in a new tag *only* where there are 1's in *both* the original implicants. The tag of the implicant (0, 1) in the first row of the table in Fig. 1.6.1(b) is

f_1	f_2	f_3
0	1	0

This is because only the symbols of the tags of the minterms 0 and 1 under f_2 are both 1. The reasoning behind this tag-formation rule is as follows. The new 2^i-cell implicant corresponds to a product term which is included in those functions which include *both* the 2^{i-1}-cell implicants used in forming the product term. It should be noted that when the tag of a new 2^i-cell implicant is (0 0 . . . 0), which means that none of the output functions includes this new implicant, this new implicant should, therefore, not be included. For example, in the table of Fig. 1.6.1(a), minterms 4 and 5 are combinable. Since their tags are (0 1 0) and (1 0 1), the resulting tag is (0 0 0); thus implicant (4, 5) does not appear in the table of Fig. 1.6.1(b). For the same reason, implicant (11, 15) is not included in the same table.

It is important to point out that there is a difference between single-function minimization and multiple-function minimization in determining prime implicants in

	Decimal Number	Binary Representation of Each Term	f_1 f_2 f_3
Index 0	0	0 0 0 0	0 1 0 ✓
Index 1	1	0 0 0 1	1 1 1 ✓
	2	0 0 1 0	1 1 0 ✓
	4	0 1 0 0	0 1 0 ✓
	8	1 0 0 0	1 1 1 ✓
Index 2	3	0 0 1 1	1 1 1 ✓
	5	0 1 0 1	1 0 1 ✓
	6	0 1 1 0	0 1 0 ✓
	9	1 0 0 1	1 1 1 ✓
	10	1 0 1 0	0 1 0 ✓
	12	1 1 0 0	1 0 1 ✓
Index 3	7	0 1 1 1	1 0 1 ✓
	11	1 0 1 1	0 1 0 ✓
	13	1 1 0 1	0 0 1 ✓
	14	1 1 1 0	1 0 1 ✓
Index 4	15	1 1 1 1	0 0 1 ✓

(a) Tag

Decimal Numbers	First Reduction	f_1 f_2 f_3
✓ 0, 1	0 0 0 –	0 1 0 ✓
✓ 0, 2	0 0 – 0	0 1 0 ✓
0, 4	0 – 0 0	0 1 0 ✓
0, 8	– 0 0 0	0 1 0 ✓
1, 3	0 0 – 1	1 1 1 A
1, 5	0 – 0 1	1 0 1 ✓
1, 9	– 0 0 1	1 1 1 B
2, 3	0 0 1 –	1 1 0 C
2, 6	0 – 1 0	0 1 0 ✓
2, 10	– 0 1 0	0 1 0 ✓
4, 6	0 1 – 0	0 1 0 ✓
8, 9	1 0 0 –	1 1 1 D
8, 10	1 0 – 0	0 1 0 ✓
8, 12	1 – 0 0	1 0 1 E
3, 7	0 – 1 1	1 0 1 ✓
3, 11	– 0 1 1	0 1 0 ✓
5, 7	0 1 – 1	1 0 1 ✓
5, 13	– 1 0 1	0 0 1 ✓
9, 11	1 0 – 1	0 1 0 ✓
9, 13	1 – 0 1	0 0 1 ✓
10, 11	1 0 1 –	0 1 0 ✓
12, 13	1 1 0 –	0 0 1 ✓
12, 14	1 1 – 0	1 0 1 F
7, 15	– 1 1 1	0 0 1 ✓
13, 15	1 – 1 1	0 0 1 ✓
14, 15	1 1 1 –	0 0 1 ✓

(b) Tag

Fig. 1.6.1 Example of minimizing a set of ... using the Quine–McClus...

Decimal Numbers	*Second Reduction*	$f_1\ f_2\ f_3$	
0, 1, 2, 3	0 0 – –	0 1 0	✓
0, 1, 8, 9	– 0 0 –	0 1 0	✓
0, 2, 4, 6	0 – – 0	0 1 0	✓
0, 2, 8, 10	– 0 – 0	0 1 0	✓
1, 3, 5, 7	0 – – 1	1 0 1	H
1, 3, 9, 11	– 0 – 1	0 1 0	✓
1, 5, 9, 13	– – 0 1	0 0 1	K
2, 3, 10, 11	– 0 1 –	0 1 0	✓
8, 9, 10, 11	1 0 – –	0 1 0	✓
8, 9, 12, 13	1 – 0 –	0 0 1	J
5, 7, 13, 15	– 1 – 1	0 0 1	K
12, 13, 14, 15	1 1 – –	0 0 1	L

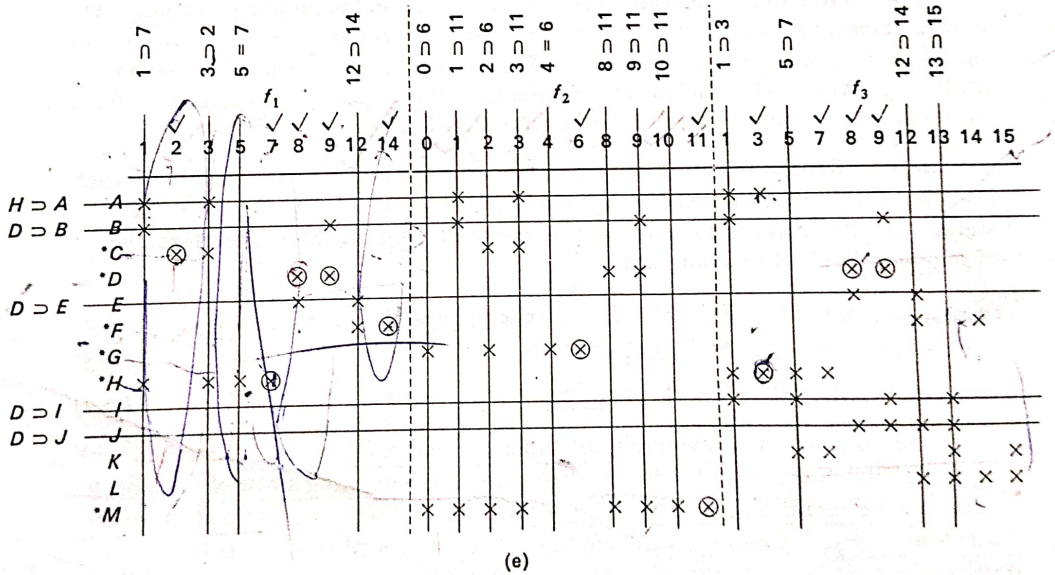
Tag

(c)

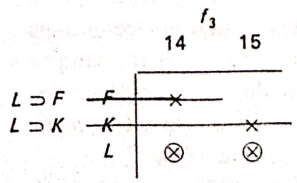
Decimal Numbers	*Third Reduction*	$f_1\ f_2\ f_3$	
0, 1, 2, 3, 8, 9, 10, 11	– 0 – –	0 1 0	M

Tag

(d)



(e)



(f)

the function reduction table. For example, in the first reduction column of Fig. 1.6.1(b), the implicants (0, 2) and (1, 3) are combined into the implicant (0, 1, 2, 3) whose tag is (0 1 0) which is the same as that of the implicant (0, 2), but *not* that of the implicant (1, 3); hence the implicant (0, 2) is "checked out", but the implicant (1, 3) is not. The rule for checking out implicants in a multiple-function reduction table is: an implicant is checked out (√) if and only if its tag is identical to that of the newly-formed implicant.

Step 2 Construct the multiple-output prime-implicant table as shown in Fig. 1.6.1(e).

The row dominance, row $I \supset$ row J, in a multiple-output prime-implicant table is defined as: row $I \supset$ row J for every output function. The definition of column dominance for the multiple-output case is the same as that for the single-output case.

Step 3 Remove all the dominating columns from the table.

Step 4 Remove all the dominated rows from the table.

Step 5 Remove the essential prime implicants from the table and include them in their respective function. For over example, with reference to Fig. 1.6.1(e), we find that $f_1 = C + D + F + H$; $f_2 = G + M$, and two terms (essential prime implicants) of f_3, which are D and H. The remaining terms of f_3 are to be determined from the remaining prime-implicant table shown in Fig. 1.6.1(f).

Step 6 Repeat Steps 3–5 as many times as needed until every minterm of each function is covered by an essential prime implicant. Fig. 1.6.1(f) shows that $L \supset F$ and $L \supset K$, and thus rows F and K can be removed from the table. Therefore, the set of minimized output functions are

$$f_1(x_1, x_2, x_3, x_4) = C + D + F + H = x_1'x_2'x_3 + x_1x_2'x_3' + x_1x_2x_4' + x_1'x_4 \qquad (1.6.4)$$

$$f_2(x_1, x_2, x_3, x_4) = G + M = x_1'x_4' + x_2' \qquad (1.6.5)$$

$$f_3(x_1, x_2, x_3, x_4) = D + H + L = x_1x_2'x_3' + x_1'x_4 + x_1x_2 \qquad (1.6.6)$$

A multiple-output prime-implicant table is *semicyclic*† if it is semicyclic with respect to each of the output functions. The criterion in selecting a row of the table to include in the minimal sum is to select a low-cost row that is contained in several output functions. The reasoning behind this is that in the final realization, the gate that realizes this row "will be shared" by the realizations of several minimized output functions; hence the complexity of the circuitry will be reduced. After such a row is selected and included in the minimal sum, we can use the reduction techniques described above to remove rows and columns from the table. Just as in the single-output case, this entire process must be repeated for each row that could replace the original selected row, and the final minimal sum is obtained by comparing the costs of the expressions that result from each arbitrary choice of a selected row.

It is important to point out that the cost of the minimized multiple-output circuit obtained by the method described in this section is in general much lower compared

†The definition of a semicyclic table for the single output case was ... which includes that of a cyclic table as a special case.