多出力 스위칭函數의 設計에 관한 計算機 앨고리즘

論 文
29 —10 —4

A Computer Algorithm for Implementing the Multiple-Output Switching Functions

趙 東 燮* · 黃 熙 隆**
(Dong-Sub Cho · Hee-Yeung Hwang)

Abstract

This paper is concerned with the computer design of the multiple-output switching functions by using the improved MASK method in order to obtain the paramount prime implicants (prime implicants of the multiple-output switching function) and new algorithm to design the optimal logic network. All the given minterms for each function are considered as minterms of one switching function to simplify the design procedures. And then the improved MASK method whose memory requirement and time consuming are much less than any existing known method is applied to identify the paramount prime implicants. In selecting the irredundant paramount prime implicants, new cost criteria are generated. This design technique is suitable both for solving a problem by hand or programming it on a digital computer.

I. Introduction

One of the major areas in switching theory research has been concerned with obtaining suitable algorithms for the minimization of Boolean functions in connection with the general problem of their economic realization. A solution of the minimization problem, in general, involves consideration of two distinct steps. In the first step, all the prime implicants of the function are found, while in the second step, from this set of all the prime implicants, a minimal subset (according to some criteria of minimality) of prime implicants is selected such that their disjunction is equivalent to the function and from which none of the prime implicants can be dropped without sacrifying equivalence. Many different algorithms exist for solving the given problems.

In finding the prime implicants of a single-output switching function the algorithms described by the authors [1]—[10] is more straight-forward and simpler than that of a multiple-output switching function. Many combinational design problems have two or more output signals that are to be generated from the same set of input variables. The simplest approach to the multiple-output switching function is to consider each output function separately. Each one of the output functions is minimized and implemented without regard to the other output functions. To avoid this disadvantage, the sharing of

* 正會員 : 서울大 大學院 碩士過程
** 正會員 : 서울大 工大 電子計算機工學科 副敎授 工博
接受日字 : 1980年 8月 2日

logical capability between output functions is introduced. The method of finding the prime implicants of the multiple-output switching function (the paramount prime implicants) is more or less similar to two steps described before. But it requries a time-consuming labor to implement the optimal logic circuits.

In this paper, tabular method is developed for a more efficient generation of all the paramount prime implicants starting from the minterm-type expression represented in decimal mode. The improved MASK method is used to obtain the paramount prime implicants without generating the entire set of the paramount prime implicants. And then a new approximate minimization algorithm is developed in order to design the logic network with the cost very close to the minimum.

This algorithm contains two new and efficient cost criteria for selecting the paramount prime implicants that assure a very suitable automatic generation of the paramount prime implicants. The minimal subset of the paramount prime implicants is selected without difficulty by the improved MASK method and the cost is much close to the minimum. This paper aims to remove this disadvantage by developing an improved algorithm for the identification and selection of the paramount prime implicants. Manual application of the minimization algorithm is possible only when n is small, where n denotes the number of the functions. When n is large, the algorithm is automatically applied on digital computer. A switching function is generally represented in a digital computer by the binary codes or the decimal codes.

Ⅱ. Definitions and Theorems

In this section some basic definitions and theorems are introduced

Definition 1: $F(x_1, x_2, \cdots, x_n)$ is the canonical form of Boolean function in the n variables x_i, $i=1,2,\cdots,n$ and Φ denotes any prime implicant, *i.e.*,

$\Phi_j(x_1, x_2, \cdots, x_n)=\prod_{i=1}^{n}(x_i)^{r_{i,j}}=0, 1, 2,$ $j=1,2,\cdots,m$ where m is the number of the prime implicants.

Assumed that

$(x_i)^0=1$, $(x_i)^1=x_i$, $(x_i)^2=\bar{x}_i$ and Π denotes the conjunction of Boolean expressions.

For example, from the canonical form of Boolean function $F(x_1, x_2, \cdots, x_n)=\bar{x}_1\bar{x}_2x_3+\bar{x}_1x_2x_3+x_1x_2x_3$, two prime implicants are obtained,

$\Phi_1(x_1, x_2, x_3)=(x_1)^2(x_2)^0(x_3)^1=\bar{x}_1x_3$

$\Phi_2=(x_1, x_2, x_3)=(x)_1{}^0(x_2)^1(x_3)^1=x_2x_3$.

Definition 2: Let the irredundant set of the prime implicants be $F^*(x_1, x_2, \cdots, x_n)$ in connection with canonical form of Boolean function $F(x_1, x_2, \cdots, x_n)$. The set of the irredundant prime implicants $F^*(x_1, x_2, \cdots, x_n)$ is given as follow,

$$F^*(x_1, x_2, \cdots, x_n)=\sum_{k=1}^{s}\Phi^*_k(x_1, x_2, \cdots x_n)$$

where Φ^*_k stands for any irredundant prime implicant and s denotes the number of the irredundant prime implicants.

Example 1: Let us find the set of the irredundant prime implicants for

$F(x_1, x_2, \cdots, x_n)=\bar{x}_1\bar{x}_2x_3+x_1x_2\bar{x}_3+x_1x_2x_3$

The following three prime implicants are generated according to the reduction property $x_1x_2+x_1\bar{x}_2=x_1$.

$\Phi_1(x_1, x_2, x_3)=x_1x_2$

$\Phi_2(x_1, x_2, x_3)=x_2x_3$

$\Phi_3(x_1, x_2, x_3)=x_1x_3$

From above set of the prime implicants the selection of the irredundant prime implicants generates the followings.

$\Phi^*_1(x_1, x_2, x_3)=x_1x_3$,

$\Phi^*_2(x_1, x_2, x_3)=x_1x_2$.

Hence, $F^*(x_1, x_2, x_3)=x_1x_3+x_1x_2$.

Definition 3: Let C_j commonality of any prime implicant of the multiple-output switching function be C_j, $j=1,2,$

…, m.

If the number of the given switching functions is l The commonality is defined as

$$C_i(f_1,f_2,\cdots,f_l)=\sum_{i=1}^{l}(f_i)^{r_{i,j}}\quad r=0,1,4$$

$$\text{where}(f)^r=\begin{cases}1 \text{ if } r=0\\ f \text{ if } r=1\\ 1 \text{ or } f \text{ if } r=4.\end{cases}$$

For instances, the commonality, of the prime implicant $\phi_i\ f_1f_3$, can be expressed as,

$C_i(f_1,f_2,f_3)=(f_1)^1(f_2)^0(f_3)^1=f_1f_3$.

And for the case of $r=4$, $C_i(f_1,f_2,f_3)=(f_1)^1(f_2)^4(f_3)^0$ implies the don't care term f_2, thus it can be reduced to f_1 or f_1f_2.

From Definition 1 and Definition 3, the multiple-output paramount prime implicants are termed as $P_i(\phi_i, C_i)$. The irredundant paramount prime implicatns are denoted by $P_k^*(\phi_k^*, C_k^*)$.

The paramount prime implicants P_i have the following property.

Theorem 1: (The property of the paramount prime implicants) The paramount prime implicants $P_j(\phi_i, C_i)$are valid only when $\phi_i\neq 1$ or $C_j\neq 1$.

The proof of Theorem 1 is very obvious in that the paramount prime implicants must have at least one variable and one commonality from the Definition 1 and Definition 3.

Definition 4: The multiple-output switching function is given by the form of the zero cube paramount prime implicant.

Example 2: Given that the multiple-output switching function is

$$f_1(x_1,x_2,x_3)=\bar{x}_1\bar{x}_2\bar{x}_3+\bar{x}_1\bar{x}_2x_3+\bar{x}_1x_2\bar{x}_3+x_1\bar{x}_2x_3$$
$$=\sum m(0,1,2,5)$$
$$f_2(x_1,x_2,x_3)=\bar{x}_1\bar{x}_2\bar{x}_3+\bar{x}_1\bar{x}_2x_3+x_1x_2\bar{x}_3$$
$$=\sum m(0,1,6)$$
$$f_3(x_1,x_2,x_3)=\bar{x}_1\bar{x}_2\bar{x}_3+\bar{x}_1x_2\bar{x}_3+x_1\bar{x}_2x_3$$
$$=\sum m(0,2,5).$$

The zero cube paramount prime implicants are generated in connection with their commonality.

$P_1(\bar{x}_1\bar{x}_2\bar{x}_3,\ f_1f_2f_3)$

$P_2(\bar{x}_1\bar{x}_2x_3,\ f_1f_2)$

$P_3(\bar{x}_1x_2\bar{x}_3,\ f_1f_3)$

$P_4(x_1\bar{x}_2x_3,\ f_1f_3)$

$P_5(x_1x_2\bar{x}_3,\ f_2)$

Thus, the multiple-output switching function of this example can be represented as shown below.

MOSF (Multiple-Output Switching Function)
$=P_1+P_2+P_3+P_4+P_5$

From the algebraic expression of the zero cube paramount prime implicant of Example 2, a new table called the intersection table is introduced as in Table 1. The relations between minterms and output functions are mapped into this table. In Section III all the higher cube paramount prime implicants will be generated from the intersection table which contains the zero cube paramount prime implicants listed in ascending order of its decimal value.

Table 1. The intersection table of Example 2

Cube	P_j	Covered Minterm	Φ_j			C_j		
			x_1	x_2	x_3	f_1	f_2	f_3
0	P_1	0	2	2	2	1	1	1
	P_2	1	2	2	1	1	1	0
	P_3	2	2	1	2	1	0	1
	P_4	5	1	2	1	1	0	1
	P_5	6	1	1	2	0	1	0

Using above definitions, the MASK method [1] is improved for the multiple-output switching function. The paramount prime implicant, P_j, has the commonality part which characterizes it. The left part ϕ_i is identified by using the MASK method for the single-output switching function and the right side C_j is particularly generated only for each paramount prime implicant.

Theorem 2: Given that 2^n minterms are reducible to Φ_i, the commonality C_i is found by the following algorithm.

$$C_i(f_1,f_2,\cdots,f_l)=\prod_{i=1}^{l}((f_i)^{\prod_{j=1}^{2^n}r_{i,j}}$$

Proof: The commonality of 2^n zero cube paramount prime implicants are

$$C_1(f_1,f_2,\cdots,f_l)=\prod_{i=1}^{l}(f_i)^{r_{i,1}}$$

$$C_2(f_1,f_2,\cdots,f_l)=\prod_{i=1}^{l}(f_i)^{r_{i,2}}$$

$$C_{2^n}(f_1,f_2,\cdots,f_l)=\prod_{i=1}^{l}(f_i)^{r_{i,2^n}}.$$

To extract the possibility of commonality, first each function is checked. And then combine

them by using Definition 3.

$$C_i(f_1,f_2,\cdots,f_1)=((f_1)^{r_{1,1}\cdot r_{1,2}\cdots r_{1,2^n}}).$$
$$((f_2)^{r_{2,2}\cdot r_{2,2}\cdots r_{2,2^n}})\cdots$$
$$((f_1)^{r_{1,1}\cdot r_{1,2}\cdots r_{1,2^n}})=\prod_{i=1}^{1}((f_i)^{\prod_{j=1}^{2^n} r_{i,j}})\quad \text{Q.E.D.}$$

Example 3: Determine the commonality between P_1 and P_2 in Table 1. Their commonalities are,

$$C_1(f_1,f_2,f_3)=(f_1)^1(f_2)^1(f_3)^1$$
$$C_2(f_1,f_2,f_3)=(f_1)^1(f_2)^1(f_3)^0.$$

Using Theorem 2, C^* may be found to be,

$$C^*(f_1,f_2,f_3)=(f_1)^{1\cdot1}(f_2)^{1\cdot1}(f_3)^{1\cdot0}=(f_1)^1(f_2)^1(f_3)^0$$
$$=f_1f_2.$$

Definition 5: If the given set of the zero cube paramount prime implicants is arranged in ascending order of binary representation of ϕ_j regardless of the commonality C_i, the lowest and the highest terms are denoted by LM and HM, respectively.

Computer algorithm for finding the paramount prime implicants is developed from Theorem 1 and Theorem 2.

Theorem 3: (Computer algorithm for finding the paramount prime implicants) The paramount prime implicant which eliminates n variables can be identified from the given set of the zero cube prime implicants if and only if the followings are hold.

(1) LM .AND. HM=LM where AND operation is bit-by-bit operation.

(2) Let the result of LM .EX—OR. HM be MASK, which shows the eliminated variable positions. Check if the value of OR-masking between MASK and LM through HM is same as HM. The number of the same values must be 2^n.

(3) C_j satisfy Theorem 1.

Theorem 3—(1) and (2)are proved in reference [1] and Theorem 3—(3)is derived from Theorem 1.

In selection of the set of the irredundant paramount prime implicants two cost criteria are used. They can be simply calculated by examing the relation between Φ_j and C_i. The theorem connected with cost is given below.

Theorem 4: (Selection of the irredundant paramount prime implicants with the minimum cost)

The minimum cost is demanded when the paramount prime implicant covers the entries of the commonality in the zero cube paramount prime implicants as many as possible. And if the commonalities covered by each paramount prime implicants are same, the paramount prime implicant which has the minimum number of the remaining commonalities should be selected.

Proof: The zero cube paramount prime implicants have the relations between the given canonical minimizing the hardware implementing cost, these relations are corresponded to the connecting wires with the constraint of two level logic network. Hence, without the loss of generality, the maximum covering which has the maximum relations all through C_j part of P_j is preferable in order to minimize the hardware cost and the number of AND gates required in the first level. Cost is defined as the number of the relations covered by the corresponding paramount prime implicant. And if there are the paramount prime implicants which have the same cost, as is the case with the single output switching function, the selection of the paramount prime implicant should be pointed on the maximum covering of the given minterms, which reduce the number of the input variables and the AND gates in the first level. This is easily done by evaluating the number of the remaining relations of C_j without counting the entries covered before in estimating cost. The number of the remaining commonalities is termed as subcost. Thus the minimum subcost yields the lower hardware cost.

Ⅲ. Generation of the paramount prime implicants

To obtain the multiple-output switching function, abbreviated to MOSF, the improved MASK method in Section II can be applied to the intersection table. The intersection table consists of the zero cube paramount prime implicants Φ_j and their commonality C_i. Let the

MOSF be the following,

$f_1(x_1,x_2,x_3,x_4)=\sum m(0,2,8,10)$

$f_2(x_1,x_2,x_3,x_4)=\sum m(0,1,2,11)$

$f_3(x_1,x_2,x_3,x_4)=\sum m(0,1,2,4,8)$.

The intersection table of above MOSF is as shown in Table 2.

Table 2. The intersection table of the MOSF given by

$f_1=\sum m(0,2,8,10)$

$f_2=\sum m(0,1,2,11)$

$f_3=\sum m(1,2,4,8)$

Cube	P_j	Covered Minterm	ϕ_j				C_j		
			x_1	x_2	x_3	x_4	f_1	f_2	f_3
0	P_1	0	2	2	2	2	1	1	0
	P_2	1	2	2	2	1	0	1	1
	P_3	2	2	2	1	2	1	1	1
	P_4	4	2	1	2	2	1	1	1
	P_5	4	2	1	2	2	0	0	1
	P_6	10	1	2	1	2	1	0	0
	P_7	11	1	2	1	1	0	1	0

Referring to Theorems in Section II, the higher cube paramount prime implicants are identified in any subset of the zero cube paramount prime implicants by the following steps.

Step 1: Read ϕ_j and C_j of the zero cube paramount prime implicants.

Step 2: Select the subset of the zero cube paramount prime implicants.

Step 3: Applying Theorem 1,2 and 3 in Section II, the higher cube paramount prime implicant is generated from the chosen set.

Step 4: If all the paramount prime implicants are identified, go to Step 5. Otherwise, go to Step 2.

Step 4: Stop.

The flow chart for computer program is described in more detail in Fig. 1.

According to the flow chart there can be 3 possible paramount prime implicants as in Table 3. They are listed in order of generation by improved MASK method. And the corresponding C_j parts satisfy Theorem 2. On the left side of the paramount prime implicant table the P_j are divided in the order of the cube. The third column from the left signifies minterms covered by P_j. Later these are used to evaluate the cost

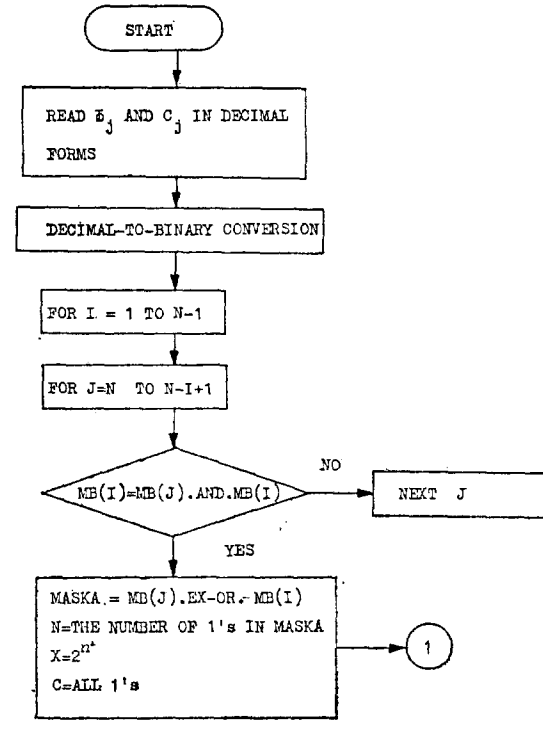

Fig. 1. Flow chart for generating the paramount prime implicants.

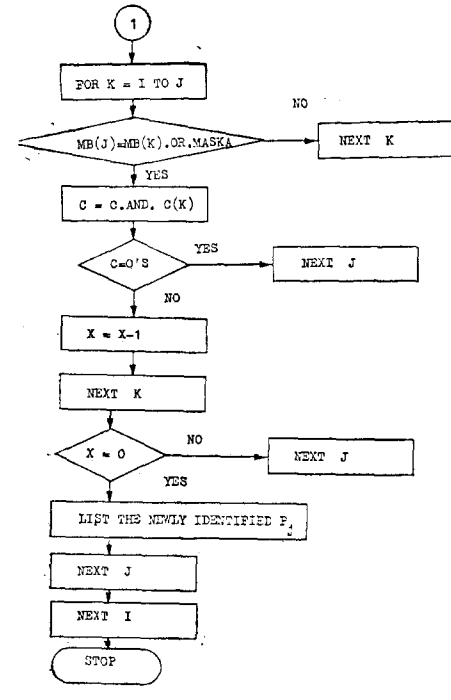

Fig. 1. (Continued)

and the subcost.

Table 3. The paramount prime implicant table

Cube	P_j	Covered Minterm	Φ_j				C_j		
			x_1	x_2	x_3	x_4	f_3	f_1	f_2
0	P_1	0	2	2	2	2	1	1	0
	P_2	1	2	2	2	1	0	1	1
	P_3	2	2	2	1	2	1	1	0
	P_4	4	2	1	2	2	0	0	1
	P_5	8	1	2	2	2	1	0	1
	P_6	10	1	2	1	2	1	0	0
	P_7	11	1	2	1	1	0	1	0
2	P_8	0,2,8,10	0	2	0	2	1	0	0
1	P_9	0,2	2	2	0	2	1	1	0
	P_{10}	0,1	2	2	2	0	0	1	0

Ⅳ. Selection of the irredundant paramount prime implicant

In Section III, all the possible paramount prime implicants are identified and construct the paramount prime implicant table used in this section. From this paramount prime implicant table, the irredundant paramount prime implicants are selected with the optimal cost by using Theorem 4. To computerize the selection procedure, the following steps are given.

Step 1: Evaluate the cost and the subcost for each paramount prime implicant.

Step 2: Using Theorem 4 the irredundant paramount prime implicant is selected.

Step 3: If all the cost are 0, go to Step 4. Otherwise, go to Step 1.

Step 4: Stop.

The program evaluaitng the cost and the subcost is shown in Fig. 2, and this subroutine is named CSG (Cost and Subcost Generation).

The detailed flow chart for the computer programming is shown in Fig. 3.

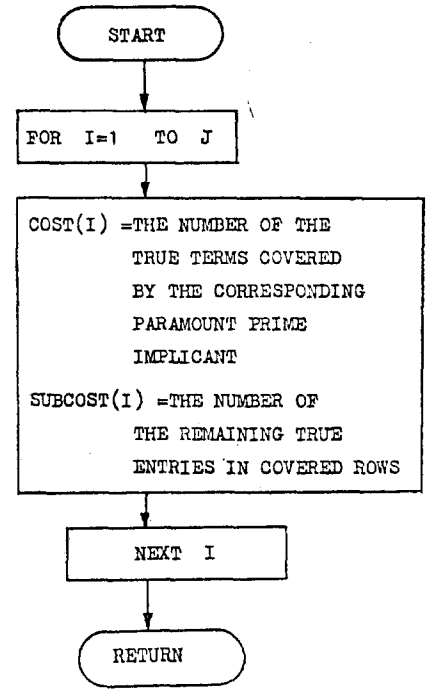

Fig. 2. The subroutine CSG (Cost and Subcost Generation).

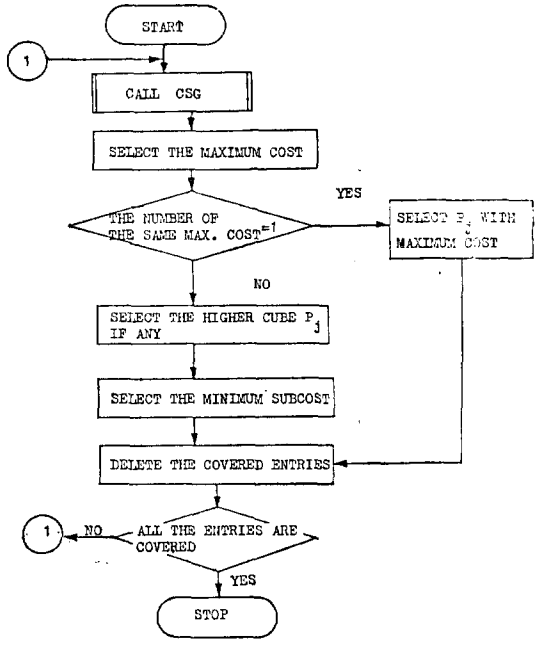

Fig. 3. Flow chart for selecting the irredundant paramount prime implicants.

Table 4. The selection of the paramount prime implicant by using cost and subcost

Cube	P_j	Covered Minterm	Φ_j				C_j			Cost	Subcost
			x_1	x_2	x_3	x_4	f_1	f_2	f_3		
		0	2	2	2	2	1	1	0	2	0
		1	2	2	2	1	0	1	1	2	0
		2	2	2	1	2	1	1	0	2	0
		4	2	1	2	2	0	0	1	1	0
		8	1	2	2	2	1	0	1	2	0
		10	1	2	1	2	1	0	0	1	0
		11	1	2	1	1	0	1	0	1	0
2		0, 2, 8, 10	0	2	0	2	1	0	0	4	3
1		0, 2	2	2	0	2	1	1	0	4	0
		0, 1	2	2	2	0	0	1	0	2	2

By using procedure in Fig. 3, the first cost and subcost table is constructed on the right side of the paramount prime implicant table. The symbol 'V' means that P_1* covers the checked entries. These are illustrated in Table 4. P_1* a selected has the maximum cost and minimum subcost satisfying Theorem 4. If one P_4* is selected has the maximum cost and minimum subcost satisfying Theorem 4. If one P_4* is selected the covered entries in intersection table must be deleted for the next evaluaion of the cost and the subcost. Table 5 shows the following cost and subcost table and the selected P_4*. In Table 5, the number located on the symbol 'V' denotes the covering sequence.

For instance '$\overset{3}{\mathrm{V}}$' means that the corresponding entries are covered by selecting P_3*.

The given MOSF is as follow.

M.O.S.F.$=P_1^*+P_2^*+P_3^*+P_4^*+P_5^*+P_6^*$.

For the exact solution above function must be rearranged to eliminate the redundant commonality discussed in following Section V.

Table 5. The selection of the paramount prime implicants

Cube	P_j	Covered Minterm	Φ_j				C_j			COST	SUB-COST	COST	SUB-COST	COST	SUB-COST	COST	SUB-COST
			x_1	x_2	x_3	x_4	f_1	f_2	f_3								
0	P_1	0	2	2	2	2	$1\overset{1}{\mathrm{V}}$	$1\overset{1}{\mathrm{V}}$	0	2	0	0	0	0	0	0	0
	P_2	1	2	2	2	1	0	$1\overset{3}{\mathrm{V}}$	$1\overset{3}{\mathrm{V}}$	2	0	2	0	2	0	P_3*	
	P_3	2	2	2	1	2	$1\overset{1}{\mathrm{V}}$	$1\overset{1}{\mathrm{V}}$	0	2	0	0	0	0	0	0	0
	P_4	4	2	1	2	2	0	0	$1\overset{4}{\mathrm{V}}$	1	0	1	0	1	0	1	$0P_4$*
	P_5	8	1	2	2	2	$1\overset{2}{\mathrm{V}}$	0	$1\overset{5}{\mathrm{V}}$	2	0	2	0	1	0	1	$0P_5$*
	P_6	10	1	2	1	2	$1\overset{2}{\mathrm{V}}$	0	0	1	0	1	0	0	0	0	0
	P_7	11	1	2	1	1	0	$1\overset{6}{\mathrm{V}}$	0	1	0	1	0	1	0	1	$0P_6$*
2	P_8	0, 2, 8, 10	0	2	0	2	1	0	0	4	3	2	1	P_2*			
1	P_9	0, 2	2	2	0	2	1	1	0	4	0	P_1*					
	P_{10}	0, 1	2	2	2	0	0	1	0	2	2	1	1	1	1	0	0

5. Consideration of the including relations in commonality

In this section, the elimination of the wire connections which arises in the multiple-output switching functions is discussed. This results in the reduction of the input lines of OR gates in the second level. First, let the selected paramount prime implicants P_1* and P_2* be given by the followings,

$P_1^*(\Phi_1, C_1)=P_1(x_1x_2,\ f_1f_2)$

$P_2^*(\Phi_2,\ C_2)=P_2(x_1x_2x_3, f_1f_3)$.

From above two paramount prime implicants, Φ_1 and Φ_2 have the including relation such that $\Phi_1 \supset \Phi_2$. This relation is valid only for the func-

tion f_1. So function f_1 has not the terms x_1x_2 and $x_1x_2x_3$, but x_1x_2. Finally they can be reduced to

$P_1^*(x_1x_2,\ f_1f_2)$

$P_2^*(x_1x_2x_3,\ f_3)$.

Computer programming for applying this including property $\Phi_1 \supset \Phi_2$ is straightforward. Their relations are checked by the operation

ϕ_1 .AND. $\Phi_2=\phi_2$.

Graphic representation of wire connection is shown in Fig. 4.

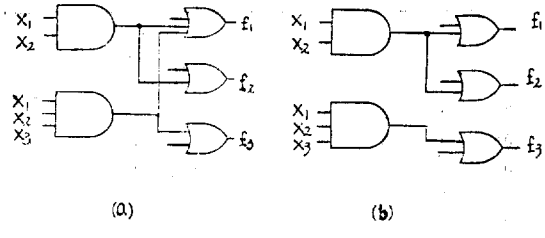

Fig. 4. Elimination of wire connections by the including relation.

In order to find the including relations, P_k are listed in increasing order of k with their C_k^* located on the right. And then the covered minterms are checked as shown in Table 6. The two entries encircled twice have the including relation and would rather be eliminated for decreasing the design cost. From the Table 6. the solution of MOSF can be written as:

$$f_1=(x_1)^0(x_2)^2(x_3)^0(x_4)^2=\bar{x}_2\bar{x}_4$$

$$f_2=(x_1)^2(x_2)^2(x_3)^0(x_4)^2+(x_1)^2(x_2)^2(x_3)^2(x_4)^0+(x_1(^1(x_2)^2(x_3)^1(x_4)^1$$
$$=\bar{x}_1\bar{x}_2\bar{x}_4+\bar{x}_1\bar{x}_2\bar{x}_3+x_1\bar{x}_2x_3x_4$$

$$f_3=(x_1)^2(x_2)^2(x_3)^2(x_4)^0+(x_1)^2(_2)^1(x_3)^2(x_4)^2+(x_1)^1(x_2)^2(x_3)^2(x_4)^2$$
$$=\bar{x}_1\bar{x}_2\bar{x}_3+\bar{x}_1x_2\bar{x}_3\bar{x}_4+x_1\bar{x}_2\bar{x}_3\bar{x}_4$$

Note that both f_2 and f_3 have the common term $\bar{x}_1\bar{x}_2\bar{x}_3$. Fig. 5. illustrates the hardware implementation of this MOSF

Table 6. Elimination of the including relation

P_4^*	P_5^*				C_k^*			Covered Minterm						
	x_1	x_2	x_3	x_4	f_1	f_2	f_3	0	1	2	4	8	10	11
P_1^*	2	2	0	2	1	1	0	v		v				
P_2^*	0	2	0	2	1	0	0	v		v		v	v	
P_3^*	2	2	2	1	0	1	1		v					
P_4^*	2	1	2	2	0	0	1				v			
P_5^*	1	2	2	2	1	0	1					v		
P_6^*	1	2	1	1	0	1	0							v

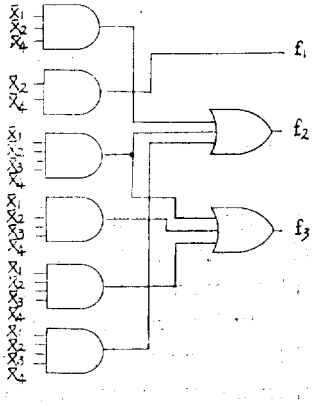

Fig. 5. Hardware implementation with respect to Table 6.

6. Consideration of the incompletely specified MOSF

In MOSF design, the incompletely specified minterms can also be included. So far MOSF are implemented without taking into account the incompletely specified minterms, but the procedures presented in Section Ⅲ and Ⅳ can also be applied to this section in a same way. The great differences are the notation on the intersection table and generation of the cost of the paramount prime implicants that cover the don't care or incompletely specified entries. From the

Definition 3 in Section II, the don't care entries are denoted as the number 4. And they are treated as true minterm when selecting the paramount prime implicants. But they need not be counted in evaluating cost and subcost, since they will not influence overall system cost. As an example of the incompletely specified MOSF, the followings are given:

$f_1(x_1,x_2,x_3,x_4)=\sum m(2,8,10)+d(0)$

$f_2(x_1,x_2,x_3,x_4)=\sum m(0,1,11)+d(2)$

$f_3(x_1,x_2,x_3,x_4)=\sum m(1,4,8)+d(0)$

The solution of this example is illustrated in Table 7 and Table 8. From the inclusion table (Table 7) the given incompletely specified MOSF can be written by using sum-of-product representation;

$f_1=\bar{x}_2\bar{x}_4$

$f_2=\bar{x}_1\bar{x}_2\bar{x}_3+x_1\bar{x}_2x_3x_4$

$f_3=\bar{x}_1\bar{x}_2\bar{x}_3+\bar{x}_2\bar{x}_3\bar{x}_4+\bar{x}_1\bar{x}_3\bar{x}_4$.

Table 7. The prime implicant table for incompletely specified MOSF

Cube	P_j	Covered Minterm	ϕ_j				C_j			COST	SUB COST	COST	SUB COST	COST	SUB COST	COST	SUB COST	COST	SUB COST
			x_1	x_2	x_3	x_4	f_1	f_2	f_3										
0	P_1	0	2	2	2	2	4	11 v	4	1	0	0	0	0	0	0	0	0	0
	P_2	1	2	2	2	1	0	11 v	11 v	2	0	0	0	0	0	0	0	0	0
	P_3	2	2	2	1	2	12 v	4	0	1	0	1	0	0	0	0	0	0	0
	P_4	4	2	1	2	2	0	0	14 v	1	0	1	0	1	0	1	0	0	0
	P_5	8	1	2	2	2	12 v	0	13 v	2	0	2	0	1	0	0	0	0	0
	P_6	10	1	2	1	2	12 v	0	0	1	0	1	0	0	0	0	0	0	0
	P_7	11	1	2	1	1	0	15 v	0	1	0	1	0	1	0	1	0	1	$0P_5$*
2	P_8	0, 2, 8, 10	9	2	0	2	1	0	0	3	2	3	1	P_2*					
1	P_9	0, 8	0	2	2	2	1	0	1	2	1	2	0	1	0	P_3*			
	P_{10}	0, 4	2	0	2	2	0	0	1	1	1	1	0	1	0	1	0	P_4*	
	P_{11}	0, 2	2	2	0	2	1	1	0	2	0	1	0	0	0	0	0	0	0
	P_{12}	0, 1	2	2	2	0	0	1	1	3	0	P_1*							

Table 8. Solution of Table 7

P_k*	ϕ_k*				C_k*			Covered minterm						
	x_1	x_2	x_3	x_4	f_1	f_2	f_3	0	1	2	4	8	10	11
P_1*	2	2	2	0	0	1	1	v	v					
P_2*	0	2	0	2	1	0	0	v		v		v	v	
P_3*	0	2	2	2	1	0	1	v				v		
P_4*	2	0	2	2	0	0	1	v			v			
P_5*	1	2	1	1	0	1	0							v

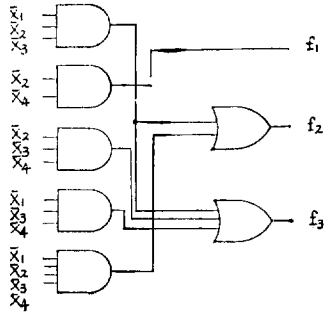

Fig. 6. Hardware implementation of the incompletely specified MOSF in Section VI.

7. Conclusions

In order to synthesize a two level multiple-output logical network this paper presents a computer algorithm which leads to the optimal cost. Optimality here means minimization of the number of the gates and the number of the connections with the constraint of two level logic network. No fan-in or fan-out is limited. Finally, the advantages of the presented computer algor-

ithm are concluded as follows:

(1) The identification of the paramount prime implicant can be easily done by the improved MASK method described in Section II.

(2) The incompletely specified MOSF can be also synthesized by using the same algorithm.

(3) The proposed computer program can be applied to the single-output switching functions if modified.

(4) Two new cost criteria, cost and subcost, result in the minimal design cost.

(5) As the input variables increase, computer program will be run more efficiently.

(6) The FORTRAN version of this program can handle the 5 input variables and 10 outputs MOSF

Acknowledgment

The authors wish to express gratitude to Prof. Park Young Moon of Seoul National University for his helpful discussions and valuable suggestions in improving the presentation of this paper.

Appendix-Computer Program

The FORTRAN version of the proposed algorithm is listed. For the execution of the computer program, the following example taken from reference [11] is solved.

$f_1(A,B,C,D)=\sum m(2,4,10,11,12,13)$

$f_2(A,B,C,D)=\sum m(4,5,10,11,13)$

$f_3(A,B,C,D)=\sum m(1,2,3,10,11,12)$

It is noted that this program cannot handle more than 5 input variables and 10 output functions. If designer wants to treat more input variables and output functions, some parts of the program should be changed in relation to problems.

References

1. Cho Dong Sub and Hwang Hee Yeung; "On the minimization of the switching function by the MASK method," KIEE, vol. 28, No. 11, pp.801~808, Nov. 1979.
2. Hwang Hee Yeung; "A new approach to the minimization of switching functions by the simple table method," KIEE, vol. 28, No. 6, pp.61~77, June 1979.
3. E.J. McCluskey; "Minimization of Boolean functions," Bell Syst. Tech. J., vol. 35, pp. 1417~1424, Nov. 1956.
4. W.V. Quine; "A way to simplify truth functions," Amer. Math. Mon. vol. 62, pp.627~631, Nov. 1955.
5. Zosimo Arevalo and J.G. Bredeson; "A method of simplify a Boolean function into a near minimal sum-of-products for programmable logic arrays," IEEE Trans. Computer, vol. C-27, pp.1028~1038, Nov. 1978.
6. H.A. Curtis; "Simplified decomposition of Boolean functions," IEEE Trans. Comput., vol. C-25, pp.1033~1076, Oct. 1976.
7. Sureshander; "Minimization of switching functions. A fast technique," IEEE Trans. Comput.(Corresp.), vol. C-24, pp.753~756, July 1975.
8. B. Reusch; "Generation of prime implicants from subfunctions and a unifying approach to the covering problem," IEEE Trans. Comput., vol. C-24, pp.924~930, Sept. 1975.
9. V. V. Rhyne, P. Noe, M. Mckinney and U. W. Pooch; "A new technique for the fast minimization of switching functions," IEEE Trans. Comput., vol C-26, pp.757~764, Aug. 1977.
10. B.L. Hulme and R.B. Worrell; "A prime implicant algorithm with factoring," IEEE Trans. Comput., vol. C-24, pp.1129~1131, Nov. 197 1975.
11. F.J. Hill and G.R. Peterson; "Introduction to Switching Theory and Logical Design," Wiley, New York, 1974.
12. John B. Peatman; "The Design of Digital System," McGraw Hill, New York, 1972.
13. V.T. Rhyne; "Fundamentals of Digital Systems Design," Prentice-Hall, Englewood Cliffs, 1973.
14. M.M. Mano; "Computer Logic Design," Prentice-Hall, Englewood Cliffs, NJ. 1972.
15. Saburo Muroga, Logic Design and Switching

Theory, John Wiley, New York, 1979.
16. Givone, Introduction to Switching Circuit Theory, McGraw Hill, New York, 1970.
17. T.C. Bartee; "Computer designs of multiple-output logical networks," IRE Trans. Electron. Comput., pp.21~30, March 1961.
18. P.R. Schneider, D.L. Dietmeyer; "An algorithm for synthesis of multiple-output combinational logic," IEEE Trans. Comput., vol. C-17, pp.117~128, Feb. 1968.

※ **Program** 생략