Application of Multiple-Valued Logic to a Serial Decomposition of PLA's

Tsutomu SASAO

Department of Computer Science and Electronics
Kyushu Institute of Technology, Iizuka 820, Japan

Abstract: A decomposition of a programmable logic array (PLA) into two cascaded PLA's is considered. The problem is divided into two subproblems: a partition problem which finds the partition of the input variables and an encoding problem which finds an encoding of signals between two PLA's. Multiple-valued decomposition theory is used to find a good partition of the input variables. Experimental results show that the serial decomposition reduce the total PLA size by 10 to 30 percents.

I. INTRODUCTION

It is well known that an arbitrary logic function can be realized by an AND-OR two-level circuit [MUR 79]. In integrated circuits, two-level circuits are often realized as programmable logic arrays (PLA's). Because PLAs have regular structure, they are easy to design, easy to modify, and easy to test. Thus, recent VLSIs (very large scale integrations) use large PLAs in their control parts. However, the larger PLAs, the more sparse the connections in the arrays: i.e., large PLAs tend to waste silicon chip area.

One method to alleviate this problem is folding of PLAs [WOO 79]. However, folding of PLAs makes layout problems complex. Therefore, folding is not so popular in VLSI design [CHA 87]. Another method to alleviate the problem is decomposition of PLAs, i.e., to realize given functions by using several smaller PLAs. Decomposition of PLAs can be classified into two types: serial decomposition and parallel decomposition.

The first type of the decomposition, a serial decomposition is shown in Fig.1(a). In this decomposition, the input variables are partitioned into two groups, and the first PLA realizes intermediate functions and the second PLA realizes desired function. The second type of the decomposition, a parallel decomposition is shown in Fig.1(b). In this decomposition, the output functions are partitioned into two groups, and each PLA realizes each group independently. Because both types of decompositions can make total size of PLAs smaller than original ones, they are often used in modern microprocessors [PEN86, CHA87].

In this paper, we consider the serial decomposition of PLA's. The decomposition is effective when the total size of the decomposed PLA's is smaller than original one. An optimal decomposition is one with the smallest total PLA size. Because it is very difficult to obtain an optimal decomposition, we divide this problem into two subproblems: a partition problem and an encoding problem.

The first problem, the partition problem is to find a partition of the input variables. Suppose that f is the output function of an original PLA, and f is decomposed in a form $f(X)=g(h(X_1),X_2)$. We want to obtain a partition (X_1,X_2) of input variables which makes g and h as simple as possible. In this case, we assume that h takes multiple values. By this assumption, an arbitrary function can be represented in the above form. The second problem, the encoding problem is to find an encoding of signals between two PLA's. Suppose that f is decomposed as $f=g(h(X_1),X_2)$. Then, the next problem is to realize g and h by using as small PLAs as possible. Note that g has a multiple-valued input, and h takes multiple valued output. We represent the multiple-valued signal by a code with binary signals. In this case, we want to obtain an encoding which makes both PLAs as small as possible.

In this paper, we solve the above problems and formalize a decomposition method for PLAs. Experimental results show that the serial decomposition reduce the total PLA size by 10 to 30 percents.

This paper is organized as follows:

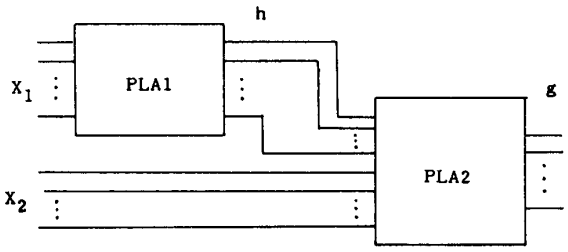
Section II introduces multiple-valued decomposition theory, which is useful to find a good partition of the input variables.

Section III shows an algorithm to find a partition of the input variables.

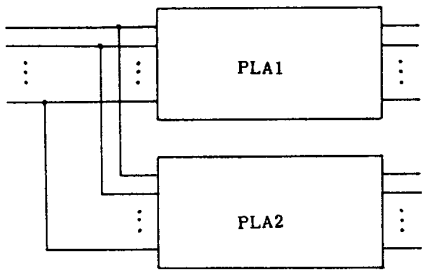
Section IV formalizes encoding problem. In this paper, only one-hot encoding and minimum length encoding are considered.

Section V shows examples for Table 2.1.

Section VI shows the experimental results.



(a) Serial Decomposition



(b) Parallel Decomposition

Fig.1.1 Decomposition of PLAs

II. Multiple-Valued Decomposition Theory

In this section, we introduce a multiple-valued decomposition theory, which is a generalization of the classical functional decomposition theory.

Definition 2.1: Let $X=(x_1,x_2,\ldots,x_n)$ be input variables. The set of the variables in X is denoted by (X). $X=(X_1,X_2,\ldots,X_r)$ is called a partition of X, where $(X)=(X_1)\cup(X_2)\cup\ldots\cup(X_r)$ and $(X_i)\cap(X_j)=\phi$ $(i\neq j)$.
The number of variables in X is denoted by d(X).

Definition 2.2: For a function f(X) and a partition $X=(X_1,X_2,\ldots,X_r)$, the decomposition chart with respect to X_1 is defined as follows:

1) It has 2^{n_1} columns and 2^{n-n_1} rows, where the columns correspond to the input combinations of X1, and $n_1=d(X_1)$.
2) Each column has the distinct binary label of n_1 bits.
3) Each row has the distinct binary label of $(n-n_1)$ bits.
4) Each entry of the chart corresponds to the truth value of the function.

Example 2.1: Consider the function f(X) shown in Table 2.1. Let $X=(X_1,X_2)$ be a partition of X, where $X_1=(x_1,x_2)$ and $X_2=(x_3,x_4)$. Then, we have the decomposition chart with respect to X_1 shown in Fig.2.1.

Definition 2.3: The number of different binary patterns in the columns of the decomposition chart is called the column multiplicity and denoted by p.

Example 2.2: In Fig.2.1, the column multiplicity of the decomposition chart is 3.

Theorem 2.1: Suppose that $X=(X_1,X_2,\ldots,X_n)$ is a partition of the input variables for a given function f(X). Let p_1 and p_2 be column multiplicities with respect to X_1 and X_2, respectively. Then the column multiplicity with respect to $X_A=(X_1,X_2)$ is at most $p_1\cdot p_2$.

(Proof) Consider a decomposition chart with respect to X_1. The decomposition chart with respect to X_A is obtained by refining each column into 2^{n_2} columns. Because the number of different patterns in the 2^{n_2} columns is at most p_2, the total number of different columns is at most $p_1\cdot p_2$.

$X1=(x1,x2)$

$(x3,x4)$	00	01	11	10
00	1	1		1
01		1		1
11		1		1
10	1		1	

Fig.2.1 Function f to be realized

(Proof) Consider a decomposition chart with respect to X_1. The decomposition chart with respect to X_A is obtained by refining each column into 2^{n_2} columns. Because the number of different patterns in the 2^{n_2} columns is at most p_2, the total number of different columns is at most $p_1\cdot p_2$.

Corollary 2.1: Suppose that the column multiplicity with respect to X is p. Let $(X_B)=(X_A)\cup(x_k)$ and $(x_k)\not\subseteq(X_A)$. Then the column multiplicity with respect to X_B is at most $2\cdot p$.

Theorem 2.2: For a function f(X) and a partition $X=(X_1,X_2,\ldots,X_r)$, there exist functions g and h such that $f(X)=g(h(X_1),X_2,\ldots,X_r)$, where

$g: P\times B^{n_2}\times\ldots\times B^{n_r}\rightarrow B$,

$h: B^{n_1}\rightarrow P$,

$B=\{0,1\}$, $P=\{0,1,\ldots,p-1\}$, $n_i=d(X_i)$ $(i=1,2,\ldots,r)$ and p is the column multiplicity of the decomposition chart of f with respect to X_1.

(Proof) For $\underline{a}$, $\underline{b}\in B^{n_1}$, define an equivalence relation $\sim$ such that
$\underline{a}\sim\underline{b} \leftrightarrow f(\underline{a},X_2,X_3,\ldots,X_r)=f(\underline{b},X_2,X_3,\ldots,X_r)$.

Let $\Pi=(L_0,L_1,\ldots,L_{p-1})$ be a partition of B^{n_1} induced by $\sim$.

The function $h:B^{n_1}\rightarrow P$ is defined as $h(\underline{a})=i$ iff $a\in L_i$ $(i=0,1,\ldots,p-1)$. Note that the first variable of g takes p values, and h is a p-valued function.

Definition 2.4: For the functions f, g and h in Theorem 2.1, $f(X)=g(h(X_1),X_2,\ldots,X_r)$ is called a decomposition of f.

When $p<2^{n_1}$, the decomposition is non-trivial. If the function f has a non-trivial decomposition, f is said to be decomposable.

In the classical decomposition theory, the function is decomposable only if p=2 [ASH 57]. However, in the multiple-valued decomposition theory, the function is decomposable if $p<2^{n_1}$. Fig.2.2 shows the circuit corresponding the decomposition. The circuit H realize the function h, where h takes p different values. In general, H has multiple outputs.

Definition 2.5: Let $X=(X_1,X_2,\ldots,X_r)$ be a partition of the input variables of f. If the function is invariant under the permutation of variables in (X_1), then f is partially symmetric with respect to (X_1).

Some arithmetic functions such as adders are partially symmetric. The following Lemma shows that partially symmetric functions are decomposable.

Lemma 2.1: If the function f is partially symmetric with respect to (X_1), then the column multiplicity of the decomposition chart with respect to X is at most n +1.

It is well known that partially symmetric functions can be realized by decomposed circuits [HUR 85]. Multiple-valued decomposition theory is useful not only for partially symmetric functions but also non-symmetric functions. Therefore, it is a much more powerful tool than classical decomposition theory.

III. Partition Problem

The final goal of the decomposition is to minimize the total size of PLA's, but it is not easy to find such decompositions. In order to make the problem simpler, we divide the decomposition problem into two subproblems. In this section, we consider the first problem, i.e., the partition problem of the input variables.

We assume the following to make the problem simpler:

Assumption 3.1: Suppose that a function is given, and the number of the variables in X_1 is fixed. Then, the smaller the column multiplicity, the simpler the circuits for g and h.

So, in order to obtain a simple circuit, we have to obtain a partition with minimum column multiplicity. For n-variable functions, there are 2^n different partitions. When n is small (say 16), it is easy to obtain the partition with minimum column multiplicity by an exhaustive method. However, when n and n_1 are large, it is almost impossible to obtain a partition with minimum column multiplicity by brute force method. A heuristic method which finds a good partition in a reasonable computation time is desired.

The following problems arise when n becomes large:

1) Because the size of decomposition chart is proportional to 2^n, the memory size increases exponentially.
2) Because the number of possible partitions is 2^n, the number of combinations to consider increases exponentially.

To solve the above problems, we use the following methods:

1) Represent each column of the decomposition chart by a logical expression. Let the number of variables in X_1 be n_1. Then, the number of expressions in the chart is 2^{n_1}. Therefore, when n_1 is small, the decomposition chart can be represented with small memory storage. In Algorithms 3.1, we compute the column multiplicities for n_1=2 to NE. So the total number of combinations to consider is $\binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{NE}$.
   An equivalence of two columns are checked as an equivalence of two logical expressions.
2) The important thing in the decomposition is to find a subset X_1 with minimum column multiplicity, when the number of the variables in X_1 is fixed. To reduce the computation time, only a subset X_1 with minimum column multiplicity is obtained. In this case, we obtain the upper bound on the column multiplicity from Corollary 2.1.

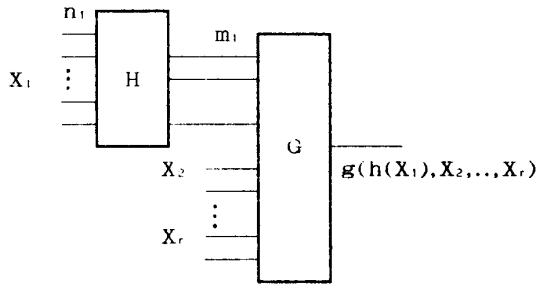


Fig.2.2 Generalized Decomposition

Algorithm 3.1: (Partition of the input variables)

1) Let UP←2.
2) For i=2 to NE do the operations 3) to 8).
3) UP←2·UP
4) For all possible partitions (X_1,X_2) such that $d(X_1)=i$, do the operations 5) to 7).
5) Calculate column multiplicity p with respect to X_1. If the lower bound on the column multiplicity is greater than UP, then stop the calculation for the column multiplicity and go to 4).
6) If (p<UP), then let $B_i \leftarrow X_1$, UP←p, and go to 4).
7) If (p=UP), then let $B_i \leftarrow B_1 \cup X_1$ and go to 4).
8) go to 2).
9) For i=2 to NE, do the operations 10).
10) For each $X_1 \in B$, calculate the sizes of PLA1 and PLA2. Use one-hot encoding to estimate the size. (Details are shown in 4.1).
11) Let X_1 be a subset with the minimum estimated PLAs.

In 1), UP shows the upper bound on the column multiplicity. In 3), Corollary 2.1 is used to obtain the upper bound. In 5), column are represented by logical expressions. In 6), B contains the sets of input variables X_1 with minimum multiplicity, for $d(X_1)=i$.

IV. Encoding Problem.

Suppose that a partition $X=(X_1,X_2,..,X_r)$ with the minimum column multiplicity is found for a given number $n_1=d(X_1)$. By Theorem 2.2, we can find functions g and h such that $f(X)=g(h(X_1),X_2,...,X_r)$. Note that $h(X_1)$ takes p different values. In order to represent a multiple-valued variable, we use a binary encoding. Then, the next problem is to specify the code for the output of h. We assign a distinct binary code for each column pattern of the decomposition chart, so that the circuits become as simple as possible. In this case, we have two alternatives to choose the code: one is a code which minimize G and the other is a code which minimize H. The problem to find the encoding which minimizes H is an optimum output encoding problem, while the problem to find the encoding which minimizes G is an optimum input encoding problem [DEM 85,86].

We use PLA1 to realize H and PLA2 to realize G. We consider only two types of encodings: one is one hot encoding and the other is minimum length encoding.

4.1 One Hot Encoding.

Definition 4.1: The one-hot code $(\alpha_0, \alpha_1, \ldots, \alpha_{p1})$ representing the value i (i=0,1,...,p-1) is (1,,..,1,0,1,..,1), where only (i+1)-th bit is 0.

From here, we will show that the cascaded PLA's shown in Fig.4.1 realizes the function $f(X)=g(h(X_1),X_2,...,X_r)$.

Definition 4.2: The functions $\alpha_i(X_1)$(i=0,1,...,p-1) of one-hot code for the decomposition $f(X)=g(h(X_1),X_2,...,X_r)$ are defined as

$$\alpha_i(\underline{a}) = \begin{cases} 0 & \text{if } \underline{a} \in L_i \\ 1 & \text{otherwise} \end{cases}$$

where $(L_0,L_1,...,L_{\mu-1})$ is a partition of B^{n_1} defined in the proof of Theorem 2.2.

Note that $\bigvee_{i=0}^{p-1} \overline{\alpha_i} = 1$ and $\overline{\alpha_i} \wedge \overline{\alpha_j} = 0$ $(i \neq j)$.

Definition 4.3: Let P=(0,1,...,p-1) be a set of p integers, S be a subset of P, and X be a variable which takes a value in P. A literal function (or literal for short) X^S is a one-variable p-valued input two-valued output function such that

$$X^S = \begin{cases} 1 & \text{if } X \in S \\ 0 & \text{otherwise} \end{cases}$$

Suppose that PLA1 realizes functions $\alpha_i(X_1)$, (i=0,1,...,p-1). Then, these p lines realizing α_i represent a p-valued variable, because for any input of X1, only one line out of p lines is 0 and other lines are 1. Let Y1 be a variable which takes p values. Y can be related to X as follows:

$$Y_1 = i \leftrightarrow \overline{\alpha_i}(X_1) = 1$$

Therefore, $\overline{\alpha_i}(X_1)$ corresponds to a literal Y_1^i.

Lemma 4.1: Suppose that PLA1 realize $\alpha_i(X_1)$, (i=0,1,...,p-1).
Let Y_1 be a variable which takes p values.
Then, an arbitrary literal $Y_1^{T_1}$, $T_1 \subseteq P$, can be realized in a column of PLA2.
(Proof) The p lines realizing α_i (i=0,1,...,p-1) represent a p-valued variable Y_1. And, α_i corresponds to Y_1. Note that

$$Y_1^{T_1} = \bigvee_{i \in T_1} Y_1^i = \bigwedge_{i \in (P-T_1)} \overline{Y_1^i} .$$

It is easy to see that

$$\bigwedge_{i \in (P-T_1)} \overline{Y_1^i} \text{ corresponds to } \bigwedge_{i \in (P-T_1)} \alpha_i$$

Therefore, $Y_1^{T_1}$ can be realized by the product of $\alpha_i(X_1)$. (Q.E.D.)

Definition 4.4: Let $\underline{a}=(a_1,a_2,\ldots,a_n)$ be a constant in B^n. $X^{\underline{a}}$ is said to be a minterm of X, and $\overline{X^{\underline{a}}}$ is said to be a maxterm of X.

Lemma 4.2: An arbitrary literal X^S ($S \subseteq B^n$) can be represented by the products of some maxterms of X:

$$X^S = \bigwedge_{\underline{a} \in (B^n - S)} \overline{X^{\underline{a}}} .$$

Theorem 4.1: Let $f(X)=g(h(X_1),X_2,\ldots,X_r)$ be a decomposition of f. Then $g(Y_1,X_2,X_3,\ldots,X_r)$ can be written as a sum-of-products expression:

$$g(Y_1,X_2,\ldots,X_r) = \bigvee_{(T_1,S_2,\ldots,S_r)} Y_1^{T_1} X_2^{S_2} \cdot \ldots \cdot X_r^{S_r} . \quad \text{----- (4.1)}$$

If α_i (i=0,1,...,p-1) and all the maxterms of X_j (j=2,..,r) are produced then an arbitrary product which has the form $Y_1^{T_1} Y_2^{T_2} \cdot \ldots \cdot Y_r^{T_r}$ can be realized in a column of PLA2.
(Proof) By Lemmas 4.1 and 4.2. (Q.E.D.)

Fig. 4.1 shows a decomposed PLA using one-hot encoding. Note that, in one-hot encoding, complemented inputs for α_i (i=0,1,...,p-1) are unnecessary in PLA2. From Theorem 4.1, the minimum PLA2 corresponds to minimum sum-of-products expression for g. Minimization of the expressions for multiple-valued input two-valued functions can be done by MINI, MINI-II [SAS 84], QM, ESPRESSO-MV, or ESPRESSO-EXACT [RUD 87].

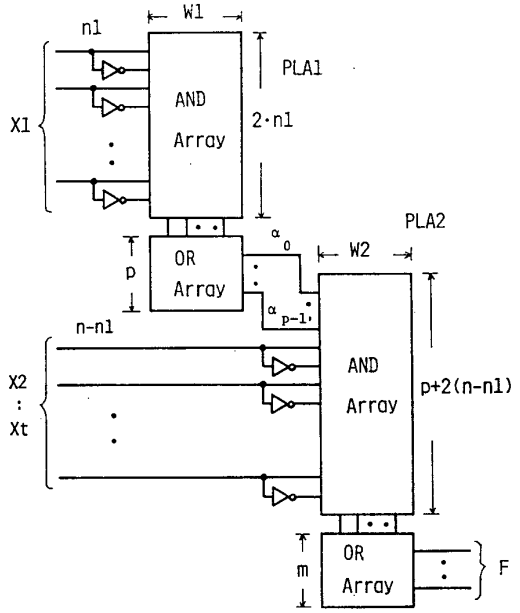


Fig.4.1 Realization using one-hot encoding

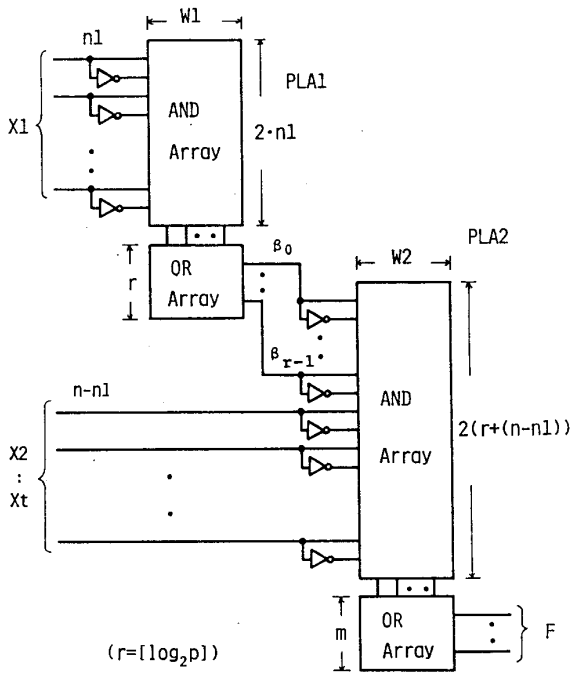


Fig.4.2 Realization using minimum-length encoding

4.2 Minimum Length Encoding.

The minimum length encoding uses an r-bit code to represent p different values, where $r=\lceil \log_2 p \rceil$. It uses fewer connection lines between PLA1 and PLA2 than one-hot encoding. Let Y be a variable which takes p values. Suppose that r-bit code $(\beta_0, \beta_1, \ldots, \beta_{r-1})$ is used to represent Y_1.

Theorem 4.2: Let $f(X)=g(h(X_1),X_2,\ldots,X_r)$ be a decomposition of f.
Let $g_e(\beta_0, \beta_1, \ldots, \beta_{r-1}, X_2, \ldots, X_r)$ be a function representing $g(Y_1,X_2,\ldots,X_r)$ by using above encoding. Then, g_e can be represented by a sum-of-products expression:

$$g_e(\beta_0, \beta_1, \ldots, \beta_{r-1}, X2, \ldots, Xr)$$
$$= \bigvee \beta_0^{T_0} \cdot \beta_1^{T_1} \cdots \beta_{r-1}^{T_{r-1}} X_2^{S_2} \cdots X_r^{S_r} \quad \text{---(4.2)}$$

Fig.4.2 shows a decomposed PLA using minimum length encoding. Note that each product in (4.2) corresponds to each column of PLA2. The number of products of (4.2) is equal to or greater than that of (4.1). In the case of minimum length encoding, different encodings produce expressions with different complexity. It is not easy to find an encoding which minimize the number of products in (4.2). We use the following simple heuristic to reduce the number of products in PLA1. Unfortunately, it does not always produce the optimum solution as will be shown in the example of 5.2.2.

Heuristic 4.1: The most frequently occur column pattern in the decomposition chart is assigned to the code (0,0,...,0), and the next most frequently occur patterns are assigned to codes (1,0,..,0), (0,1,0,...,0),...,(0,....,1), and so on. The more frequent the pattern occurs in the decomposition chart, the more the number of 0's in the code.

Table 2.1 4-variable function

x1	x2	x3	x4	f
0	0	0	0	1
0	0	0	1	0
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	1
0	1	1	0	0
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	0
1	0	1	1	1
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

V. Example

In this part, we show the idea of the decomposition by using the function f shown in Table 2.1.

5.1 Original PLA realization.

The size of the standard PLA is $C_1=(2n+m)\cdot W$. From the map of the function, 6 products are necessary to represent f. Fig.5.1 shows the PLA with the size $C_1=54$, where n=4, m=1, and W=6

5.2 Decomposed PLA realization

Let $X=(X_1,X_2,X_3)$ be a partition of $X=(x_1,x_2,x_3,x_4)$, where $X_1=(x_1,x_2)$ $X_2=(x_3)$, and $X_3=(x_4)$. Note that only three different patterns appear in the column of Fig.2.1, i.e., the column multiplicity of the decomposition chart is 3. Consider the decomposition $f(X)=g(h(X_1),X_2,X_3)$. The function h shown in Table 5.1 is a two-valued input three-valued output function. And the function $g(Y_1,X_2,X_3)$ shown in Table 5.2 is multiple-valued input two-valued output function, where Y_1 takes three values and X_2 and X_3 takes two values.

5.2.1 One-Hot Encoding

In this case, we use three-bit code $(\alpha_1, \alpha_2, \alpha_3)$ to represent the three-valued variable Y_1. Table 5.3 shows relation between Y_1 and $(\alpha_1, \alpha_2, \alpha_3)$. PLA1 converting (x_1,x_2) into $\alpha_1, \alpha_2, \alpha_3)$ has only two products when the output phase is optimized [SAS 84]. Fig.5.2 is the map of the function g represented by Y_1, x_3, and x_4. PLA2 realizing the function g has only three products. Fig.5.3 shows the decomposed PLA's. The total size of the decomposed PLA's obtained from Fig.4.1 is $C_2=(2\cdot n_1+ p)\cdot W_1+(2(n-n_1)+p+m)\cdot W_2$.
In this case n=4, m=1, n_1=2, p=3, W_1=2, and W_2=3. So, we have C_2=38.

5.2.2 Minimum Length Encoding

In this case, we use two binary variables, β_0 and β_1 to represent the three-valued variable Y_1. Table 5.4 shows relation between Y_1 and (β_0, β_1). PLA1 converting (x_1,x_2) into (β_0, β_1) has two products.

Fig.5.4 shows the function g represented by variables β_0, β_1, x_3, and x_4. Note that PLA1 never produces the pattern (1,1). So, the entries for these inputs are don't cares. PLA2 realizing the function g has four products. Fig.5.5 shows the decomposed PLA's. The total size of the decomposed PLAs given by Fig.4.2 is
$C_2=(2\cdot n_1+r)\cdot W_1+(2(n-n_1+r)+m)W_2$. In this case, n=4, m=1, n_1=2, r=2, W_1=2, W_2=4. So C_2=48
If we use the encoding shown in Table 5.5, then the number of products in PLA2 becomes to three. So, the total size is C_2=39.

VI. Experimental Results

The partition and the encoding algorithms were programmed in FORTRAN, and implemented on a PC98XA, a personal computer using an 8-Mega Hertz INTEL 80286 microprocessor. For simplicity, we used single-output functions to show the idea of the multiple-valued decomposition theory. However, the decomposition of multiple-output functions can be formulated similarly. Our program can treat multiple-output functions.

We investigated many industrial and arithmetic PLA's [BRA 84],[RUD 87], [SAS 86b]. Most PLAs had non-trivial decompositions. 10 out of 23 decomposable PLA's are more than 10% smaller than

Table 5.1 Function Y1=h(X1)

x1	x2	Y1
0	0	0
0	1	1
1	0	1
1	1	2

Table 5.2 Function g(Y1,X2,X3)

Y1	X2	X3	g
0	0	0	1
0	0	1	0
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	1
2	0	0	0
2	0	1	0
2	1	0	1
2	1	1	0

Table 5.3 One-hot Encoding

Y1	$\alpha 0$	$\alpha 1$	$\alpha 2$
0	0	1	1
1	1	0	1
2	1	1	0

Table 5.4 Minimum Length Encoding

Y1	$\beta 0$	$\beta 1$
0	0	1
1	0	0
2	1	0

Table 5.5 Another Minimum Length Encoding

Y1	$\beta 0$	$\beta 1$
0	0	1
1	0	0
2	1	1

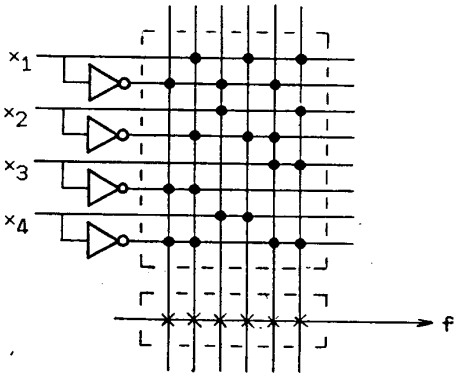


Fig.5.1 Original PLA

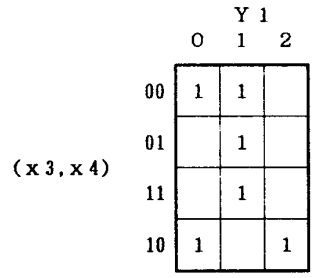


Fig.5.2 Function g represented by Y1, x3, and x4.

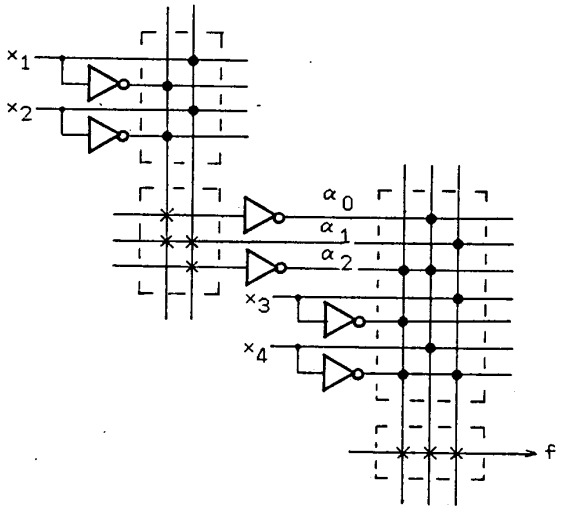


Fig.5.3 Decomposed PLA (One-hot encoding)

Table 6.1 Decompositon Result

PLA Name	Original PLA				Decomposed PLA					Size (%)
					PLA1			PLA2		
	n	m	W	C1	n1	p	W1	W2	C2	C1/C2
NXCPLA	9	23	43	1763	4	8	8	31	1399	0.79
NAPLA	12	10	16	544	5	4	6	14	476	0.88
NAPLA1	12	7	10	310	3	3	4	8	260	0.84
NWCOND	11	2	31	744	3	4	5	23	556	0.75
BOTH	10	39	81	4779	4	9	9	64	3993	0.84
GARY	15	11	107	4387	3	5	5	99	3817	0.87
AUG1	16	8	54	2160	4	5	10	40	1610	0.75
ROT8	8	5	57	1197	3	5	8	35	788	0.66
LG8MOD	8	5	38	798	4	5	5	35	695	0.87

n: number of inputs
m: number of outputs
W: number of columns of original PLA
C1: size of the original PLA =(2n+m)W
n1: number of inputs for PLA1
p: number of outputs of PLA1
W1: number of columns of PLA1
C2: size of decomposed PLA's =(2n1+p)W1+(2n-2n1+p+m)W2

the original ones. Table 6.1 shows the selected results for the nine PLA's. It is often possible to reduce the size by optimizing the output phase, but no output phase optimization were done for the PLA's in Table 6.1. In most cases, one-hot encodings produced smaller PLAs than minimum length encodings. So, only the results for one-hot encodings are shown in Table 6.1.

For arithmetic functions and special functions with strong symmetry properties (i.e., ADR4, RD53, RD73, RD84, SYM9), the decomposed PLA's are much smaller than the original ones. We can extend Theorem 2.2 to treat the decomposition $f(X)=g(h_1(X_1),h_2(X_2),\ldots,h_r(X_r))$ [SAS 81].

For example, RD84 (also called WGT8 [SAS 86a] or SAO1 [SAS 86b]) can be decomposed as $f(X)= g(h_1(X_1),h_2(X_2))$, where $X_1=(x_1,x_2,x_3,x_4)$ and $X_2=(x_5,x_6,x_7,x_8)$. In this case, h_1 and h_2 are two-valued input 5-valued output functions, while $g(Y_1,Y_2)$ is a 5-valued input two-valued output function. The size of the original PLA is 5100, while size of the decomposed PLA's is only 400 when we use one hot encoding and 396 when we use minimum length encoding [SAS 86a].

VII. Conclusion and Future Work

In this paper, we formulated a serial decomposition problem and presented a method to realize logic functions by cascaded PLAs. We decomposed various PLAs, and obtained the following results:

1) 10 PLAs out of 23 decomposable PLAs are more than 10% smaller then original PLAs.
2) In most cases, one-hot encoding produced smaller circuits than minimum-length encodings.

The present method has several points to be improved:

3) The algorithm for the partition of the input variables needs exponential computation time, although we have some methods to speed up. We need a heuristic method to find a good partition more efficiently.
4) In this experiment, minimum-length encodings produced larger circuits than one-hot encodings. One reason for this is, we did not optimize the minimum-length encodings. We need a better heuristic than Heuristic 4.1 to find a better encoding.

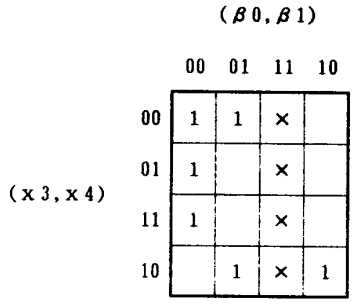


Fig.5.4 Function g represented by β0, β1, x3, and x4

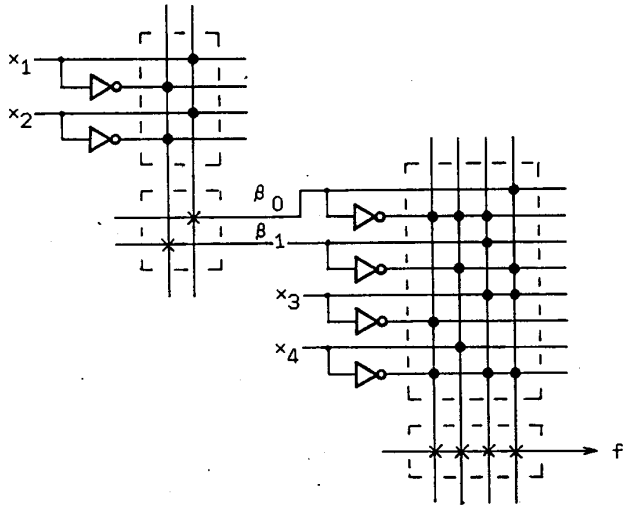


Fig.5.5 Decomposed PLA (Minimum length encoding)

[Acknowledgement]
The author thank Mr. M. Higashida for programming and discussion.
Preliminary versions of this paper were presented as [SAS 86c] and [SAS 87].

[REFERENCES]

[ASH 57] R.L.Ashenhurst, "The decomposition of switching functions,"in Proceedings of an International Symposium on the Theory of Switching, pp.74-116, April 1957.
[BRA 84] R.K.Brayton, G.D.Hachtel, C.T.McMullen and A.L.M.Sangiovanni-Vincentelli,Logic Minimization Algorithms for VLSI Synthesis, Kluwer Academic Publishers, Boston, 1984.
[CHA 87] H.H.Chao, et. al.,"Designing the Micro/370, " IEEE Design and Test, pp.32-40, June 1987.
[DEM 85] G. De Micheli, R.K.Brayton and A.Sangiovanni-Vincentelli,"Optimal state assignment for finite state machines,"IEEE Trans on CAD ,Vol. CAD-4, No.3, pp.269-285, July 1985.
[DEM 86] G. De Micheli,"Symbolic design of combinational and sequential logic circuits implemented by two-level logic macros," IEEE Trans on CAD,Vol. CAD-5, No.4, pp.597-616 Oct.1986.
[HUR 85] S.L.Hurst, D.M.Miller and J.C.Muzio, Spectral Techniques in Digital Logic, Academic Press, 1985, p.217.
[MUR 79] S.Muroga, Logical Design and Switching Theory, Wiley-Interscience, New York, 1979.
[PEN 86] J.M.Pendleton et.al, "A 32-bit microprocessor for smalltalk," IEEE Journal of Solid-State Circuits, Vol.SC-21, No.5, pp.74-749, Oct. 1986.
[RUD 87] R.Rudell and A.L.M.Sangiovanni-Vincentelli,"Multiple-valued minimization for PLA optimization," IEEE Trans on CAD,Vol. CAD-6, No.5, pp.727-750, Sept. 1987.
[SAS 81] T.Sasao,"Multiple-valued decomposition of generalized Boolean functions and the complexity of programmable logic arrays," IEEE Trans. Comput. Vol. C-30, pp.635-643, Sept. 1981.
[SAS 84] T.Sasao,"Input variable assignment and output phase optimization of PLA's," IEEE Trans. Comput. vol.C-33, No.10, pp.879-894, Oct.1984.
[SAS 86a] T.Sasao, Programmable Logic Array: How to use and how to make, (in Japanese) Nikkan Kougyo Shinbun Pub., May 1986.
[SAS 86b] T.Sasao,"MACDAS:Multi-level AND-OR Circuit synthesis using two-variable function generators,"23-rd DAC,pp.86-93, June 1986.
[SAS 86c] T.Sasao, "On the generalization of functional decomposition and its application, " (in Japanese), Workshop on FTC, Hokkaido, Sept. 1986.
[SAS 87] T.Sasao, "Functional decomposition of PLA's", The International Workshop on Logic Synthesis, Research Triangle Park, May 1987.
[WOO 79] R.A.Wood, "A high density programmable logic array chip", IEEE Trans. Comput. Vol. C-38, pp.602-608, Sept. 1979.