On the Optimal Design of Multiple-Valued PLA's

TSUTOMU SASAO, MEMBER, IEEE

Abstract—**This paper describes the design and analysis of three types of multiple-valued PLA's: Type 1 PLA's realize functions directly in the form of the MAX of MIN of literal functions and constants. In Type 2 PLA's, the body of the PLA is binary and the output is encoded as a multiple-valued logic value. Type 3 PLA's are the same as Type 2 PLA's except 2-bit decoders and a permutation network is used on the input. Using the number of columns required to realize a given function as a measure to compare PLA's, it is shown that Type 3 PLA's are superior to Type 2 which, in turn, are superior to Type 1.**

Index Terms—**Adder, complexity, input encoding problem, logic minimization, multiple-valued logic, output encoding problem, PLA.**

I. Introduction

ONE of the most pressing problems in present-day two-valued systems is interconnection complexity, both in-chip and between chips [8]. It is evident that multiple-valued logic (MVL) is useful for reducing interconnections. Thus, various MVL systems have been proposed for many years.

When we design multiple-valued VLSI (MV-VLSI), we encounter the same problems as in two-valued systems. The first problem is the enormous design complexity of VLSI's. As the number of the elements in a chip increases, design time increases exponentially. Because logic design of multiple-valued systems is usually much more complicated than two-valued systems, this problem is more important in MV-VLSI's. In order to reduce the design time and errors, automatic design is indispensable in MV-VLSI's. However, even in two-valued systems, automatic design of logic circuit is very difficult. Although automatic design systems for random logic circuits which produce good circuits are known [3], [4], [19], they do not always generate optimum circuits. The only two-valued circuits which are successfully designed by a complete automatic system and whose optimality is guaranteed are programmable logic arrays (PLA's) [18].

The second problem is the testability of the VLSI's. In the modern VLSI's, testing cost often dominates the total production cost [2]. In order to overcome the design complexity and testability problems, circuits having regular structure such as PLA's ROM's, and RAM's are extensively used in many of the VLSI's. For example, recent VLSI microprocessors such as BELLMAC 32A [12] and Motorola MC68020 [2] use PLA's extensively in the control part of the processors. PLA's can also be used to implement complex MVL circuits. Therefore, PLA's are the most promising vehicle for implementing complex MVL circuits.

In this paper, the author proposes three types of multiple-valued PLA's (MVPLA's): Type 1 PLA consists of literal generators (which convert multiple-valued signals into two-valued signals), a MIN array, and a MAX array. Type 2 PLA consists of literal generators, an AND array and an OR array, and output encoders (which convert two-valued signals into multiple-valued signals). Type 3 PLA consists of a permutation network and 2-bit decoders in addition to the components of Type 2 PLA. Because the AND and OR arrays are the same as those of two-valued PLA's [16], Type 2 and Type 3 PLA's are easily implemented by (static or dynamic) MOS/CMOS circuits, and they can be designed by various existing PLA tools such as MINI [7], MINI-II [17], and ESPRESSO-MV [15]. Logical capability and logical complexity analysis show that the proposed MVPLA requires much smaller arrays than previously published MVPLA's [11], [9].

II. PLA with Min and Max Arrays

A. Logical Implementation

An arbitrary p-valued logic function $f(X_1, X_2, \cdots, X_n)$ can be represented by an expression:

$$f = 0 \cdot g_0 \vee 1 \cdot g_1 \vee \cdots \vee (p-1) \cdot g_{p-1} \qquad (2.1)$$

where $\vee$ denotes the MAX operator and $\cdot$ denotes the MIN operator. g_i $(i = 0, 1, \cdots, p-1)$ is a p-valued input function which takes only two values 0 and $p-1$. They denote the input combinations such that $f(X_1, X_2, \cdots, X_n) = i$.

In (2.1), g_0 can be omitted. $(p-1)$, which is the largest value, can also be omitted from (2.1). Therefore, (2.1) can be rewritten as

$$f = 1 \cdot g_1 \vee \cdots \vee (p-2) \cdot g_{p-2} \vee g_{p-1}. \qquad (2.2)$$

Each subfunction g_i $(i = 1, 2, \cdots, p-1)$ of (2.2) can be represented as the sum-of-products (MAX-of-MIN's) expression of literal functions:

$$g_i = \bigvee X_1^{S_1} \cdot X_2^{S_2} \cdot \cdots \cdot X_n^{S_n} \qquad (2.3)$$

where $S_j \subseteq P$, and $P = \{0, 1, 2, \cdots, p-1\}$. A *literal function* (or simply *literal*) is a one-variable p-valued input two-valued output function. A literal $X_j^{S_j}$ takes a value of 0 if $X \notin S$ and a value $(p-1)$ if $X \in S$.

Note that an arbitrary function g_i has a sum-of-products expression of form (2.3).

Manuscript received August 23, 1986; revised April 2, 1987. A preliminary version of this paper was presented at the 16th International Symposium on Multiple Valued Logic, Blacksburg, VA, May 1986.

The author is with the Department of Computer Science and Electronics, Kyushu Institute of Technology, Iizuka, 820 Japan.

IEEE Log Number 8824539.

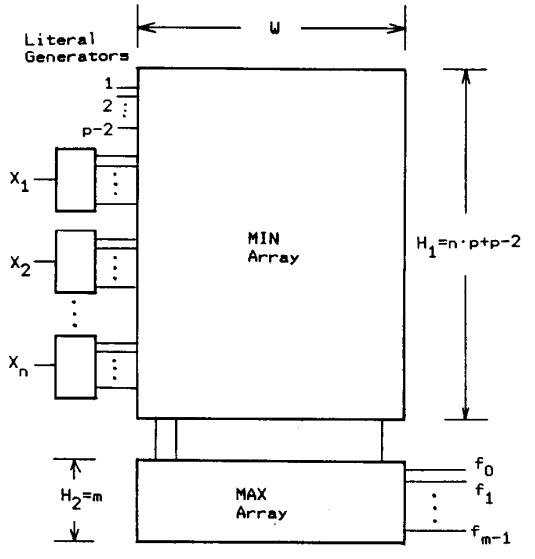


Fig. 1. p-valued PLA with MIN and MAX array (Type 1 PLA).

Fig. 1 shows an n-input m-output p-valued PLA with a MIN and MAX array. Similar PLA structures can be found in [8] and [11]. It has *n literal generators.* Each literal generator decodes p-valued variable X_i into literals (two-valued signals). There are 2^p possible literals, but we need not generate all. There are many ways to choose the set of literals. We choose *the minimum universal set of literals,* which can represent any other literal by a logical product of some of the literals in the set, and contains the minimum number of elements. As shown in the Appendix, the minimum universal set of literals contains p elements. Because each literal generator has p outputs, and $(p-2)$ different constants are used in the MIN array in Fig. 1, the number of rows in the MIN array is $H_1 = np + (p-2)$.

Theorem 2.1: Type 1 PLA realizes an arbitrary p-valued logic function. Let W be the number of columns necessary to realize an n-variable function, then $W \leq (p-1)p^{n-1}$.

Proof: It is clear that each column of Type 1 PLA realizes a product $(j) \cdot X_1^{S_1} \cdot X_2^{S_2} \cdot \cdots \cdot X_n^{S_n}$, where j is a constant such that $1 \leq j \leq p-1$. Type 1 PLA realizes an arbitrary p-valued function as a MAX-of-MIN's of the form (2.2). The number of columns is equal to the number of products in (2.2). Each subfunction g_i can be realized with at most p^{n-1} products because g_i can be represented by a canonical expression $g_i = \bigvee G(X_1, \boldsymbol{a}) X_2^{a_2} \cdot X_3^{a_3} \cdot \cdots \cdot X_n^{a_n}$, where $G(X_1, \boldsymbol{a} = g_i(X_1, a_2, a_3, \cdots, a_n), a_k \in P$ and $k = 2, \cdots, n$. Therefore, the total number of products in (2.2) is at most $(p-1)p^{n-1}$. (Q.E.D.)

The logic design of a Type 1 PLA can be done as follows.

Algorithm 2.1:

1) Obtain a minimum sum-of-products expression for g_{p-1}.

2) For each $g_k, (k = p-2, \cdots, 1)$, obtain a minimum sum-of-products expression for g_k. In this case, $g_r (k+1 \leq r \leq p-1)$ can be used as DON'T CARE sets. g_i $(i = 1, \cdots, p-1)$ are p-valued input two-valued output functions and minimization of these expressions can be done by MINI, MINI-II, or ESPRESSO-MV.

TABLE I
TRUTH TABLE FOR FOUR-VALUED ADDER

Input		Output	
X_1	X_2	Sum	Carry
0	0	0	0
0	1	1	0
0	2	2	0
0	3	3	0
1	0	1	0
1	1	2	0
1	2	3	0
1	3	0	1
2	0	2	0
2	1	3	0
2	2	0	1
2	3	1	1
3	0	3	0
3	1	0	1
3	2	1	1
3	3	2	1

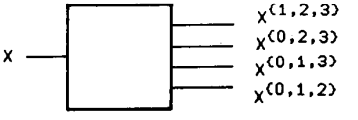


Fig. 2. Literal generator.

Example 2.1: Let us design an adder of four-valued logic shown in Table I. In this case, the minimum universal set of literals is generated by a literal generator shown in Fig. 2. By using the map shown in Fig. 3, we can obtain the minimum sum-of-products expressions for sum and carry functions as follows:

$$\text{Sum} = 1 \cdot g_1 \vee 2 \cdot g_2 \vee g_3,$$

where

$$g_1 = X_1^{\{1,3\}} \cdot X_2^{\{0,2\}} \vee X_1^{\{0,2\}} \cdot X_2^{\{1,3\}},$$

$$g_2 = X_1^{\{2\}} \cdot X_2^{\{0\}} \vee X_1^{\{1\}} \cdot X_2^{\{1\}} \vee X_1^{\{0\}} \cdot X_2^{\{2\}} \vee X_1^{\{3\}} \cdot X_2^{\{3\}},$$

and

$$g_3 = X_1^{\{3\}} \cdot X_2^{\{0\}} \vee X_1^{\{2\}} \cdot X_2^{\{1\}} \vee X_1^{\{1\}} \cdot X_2^{\{2\}} \vee X_1^{\{0\}} \cdot X_2^{\{3\}}.$$

$\text{Carry} = 1 \cdot g_4$, where

$$g_4 = X_1^{\{3\}} \cdot X_2^{\{1\}} \vee X_1^{\{2,3\}} \cdot X_2^{\{2\}} \vee X_1^{\{1,2,3\}} \cdot X_2^{\{3\}}.$$

Fig. 4 shows the PLA realizing the adder. Note that 13 products are used in this PLA. (End of example).

B. Physical Implementation

The MAX and the MIN arrays are easily implemented by bipolar technology, but they require many transistors if realized by MOS technology. Therefore, this structure is unsuitable for MOS implementation.

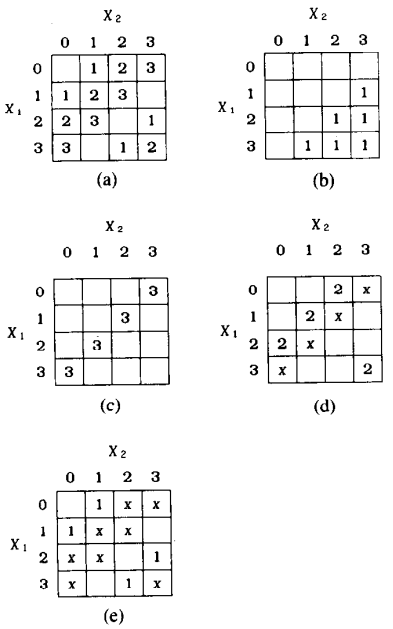


Fig. 3. Maps for adder using Type 1 PLA. (a) Sum. (b) Carry. (c) g_3. (d) $2 \cdot g_2$. (e) $1 \cdot g_1$.

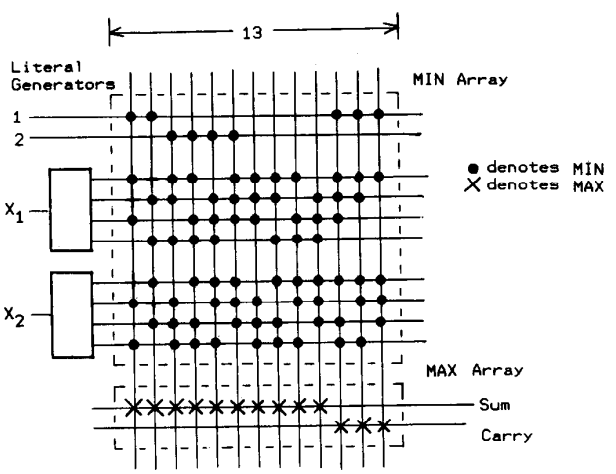


Fig. 4. Adder using Type 1 PLA.

III. PLA with AND–OR Arrays Followed by Output Encoders

A. Logical Implementation

Fig. 5 shows an n-input m-output p-valued PLA with AND-OR arrays followed by output encoders. We call this PLA a *Type 2 PLA*. Similar to Type 1 PLA, each p-valued signal X_i is converted into p literals. Then, these literals are used in the AND and the OR arrays to realize mr two-valued functions $h_0, h_1, \cdots, h_{mr-1}$, where $r = [\log_2 p]$. ($[\log_2 p]$ denotes the least integer greater than or equal to $\log_2 p$.) Finally, these two-valued signals are converted into p-valued signals by the output encoders.

For simplicity, suppose that $m = 1$ and $p = 2^r$. h_i ($i = 0, \cdots, r-1$) are p-valued input functions which take only

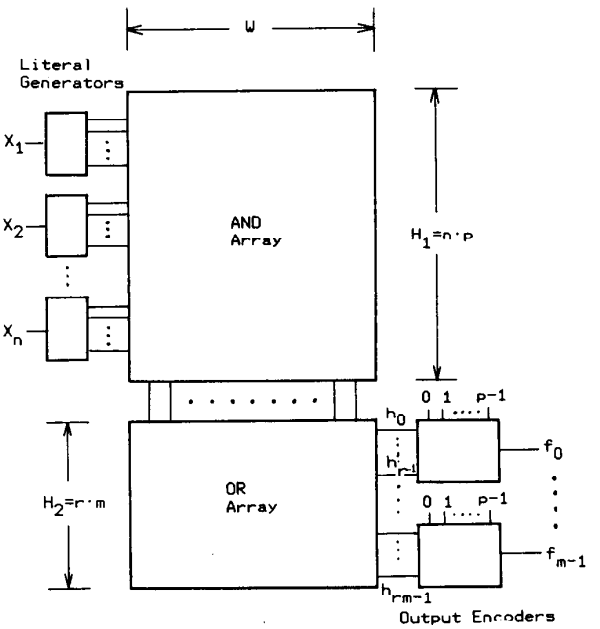


Fig. 5. p-valued PLA with output encoders (Type 2 PLA).

two values 0 and $p-1$. The r-tuple of two-valued signals $(h_0, h_1, \cdots, h_{r-1})$ represents the binary encoding of the p-valued signal, i.e., $(0, 0, \cdots, 0)$ represents 0, $(0, 0, \cdots, p-1)$ represents 1, $\cdots$, and $(p-1, p-1, \cdots, p-1)$ represents $2^r - 1 = p - 1$. In Fig. 5, each column in the AND array realizes the product $X_1^{S_1} \cdot X_2^{S_2} \cdot \cdots X_n^{S_n}$ and each row of the OR array realizes the function h_j $(0 \le j \le r-1)$. The output encoder accepts $(h_0, h_1, \cdots, h_{r-1})$, and generates a p-valued signal according to the binary encoding. The output encoder can be realized by a p-input multiplexer shown in Fig. 6(a), and it is denoted by the symbol shown in Fig. 6(b).

Theorem 3.1: Let W be the number of columns necessary to realize an arbitrary p-valued function in a Type 2 PLA, then $W \le [\log_2 p] \cdot p^{n-1}$.

Proof: The number of columns of a Type 2 PLA is equal to the total number of the products in the functions $h_0, h_1, \cdots, h_{r-1}$. It is clear that each function h_j can be realized by at most p^{n-1} products. Hence, we need at most $r \cdot p^{n-1}$ products to represent an arbitrary function.(Q.E.D.)

Example 3.1: Let us design the adder of four-valued logic shown in Table I. Suppose that the four-valued output signal is represented by a pair of two-valued signals as shown in Table II. Then the function to be realized by the arrays can be represented as Table III. By using the maps shown in Fig. 7, we can obtain the minimum sum-of-products expressions for s_1, s_0, and c_0 as follows:

$$s_1 = X_1^{\{2,3\}} \cdot X_2^{\{0\}} \vee X_1^{\{1,2\}} \cdot X_2^{\{1\}} \vee X_1^{\{0,1\}} \cdot X_2^{\{2\}} \vee X_1^{\{0,3\}} \cdot X_2^{\{3\}},$$

$$s_0 = X_1^{\{1,3\}} \cdot X_2^{\{0,2\}} \vee X_1^{\{0,2\}} \cdot X_2^{\{1,3\}},$$

$$c_0 = X_1^{\{3\}} \cdot X_2^{\{1\}} \vee X_1^{\{2,3\}} \cdot X_2^{\{2\}} \vee X_1^{\{1,2,3\}} \cdot X_2^{\{3\}}.$$

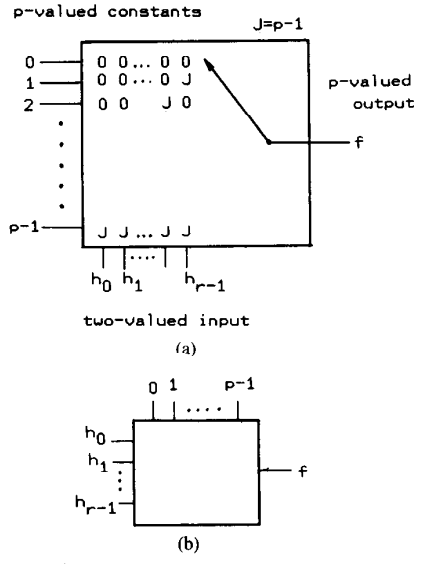


Fig. 6. Output encoder. (a) p-input multiplexer. (b) Symbol for output encoder.

TABLE II
OUTPUT ENCODING FOR FOUR-VALUED ADDER

4-valued signal	2-valued signals
0	0 0
1	0 3
2	3 0
3	3 3

TABLE III
TRUTH TABLE FOR FOUR-VALUED ADDER

Input		Output			
		Sum		Carry	
X_1	X_2	S_1	S_0	C_1	C_0
0	0	0	0	0	0
0	1	0	3	0	0
0	2	3	0	0	0
0	3	3	3	0	0
1	0	0	3	0	0
1	1	3	0	0	0
1	2	3	3	0	0
1	3	0	0	0	3
2	0	3	0	0	0
2	1	3	3	0	0
2	2	0	0	0	3
2	3	0	3	0	3
3	0	3	3	0	0
3	1	0	0	0	3
3	2	0	3	0	3
3	3	3	0	0	3

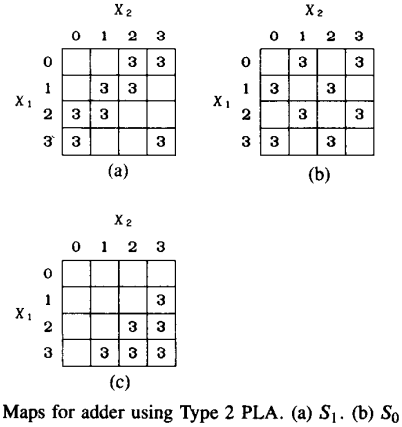


Fig. 7. Maps for adder using Type 2 PLA. (a) S_1. (b) S_0. (c) C_0.

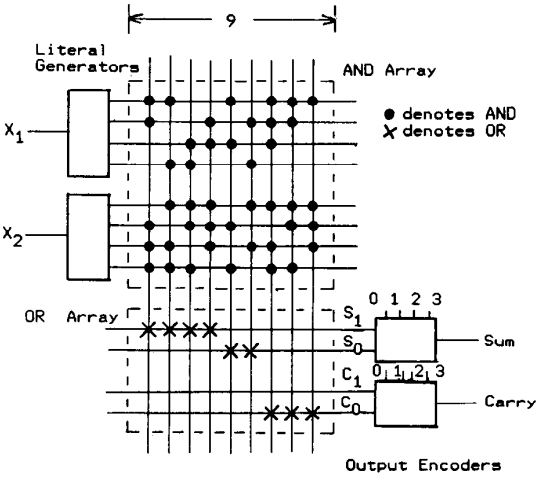


Fig. 8. Adder using Type 2 PLA.

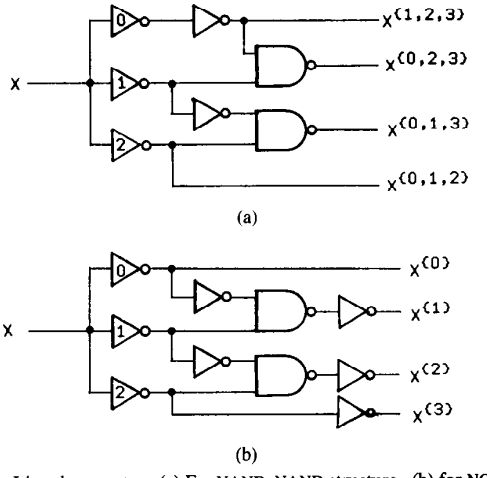


Fig. 9. Literal generator. (a) For NAND-NAND structure. (b) for NOR-NOR structure.

Fig. 8 shows the PLA realizing the adder. Note that nine products are used in this PLA. (End of example).

B. Physical Implementation

The AND and the OR arrays are easily implemented by both bipolar and MOS technology. When we realize a large PLA, dynamic CMOS circuit is the most attractive technology [12]. In this case, a NOR-NOR structure is used to implement the AND and the OR arrays. When we need an extremely low-power system, a static CMOS circuit is also feasible [14]. In this case, a NAND-NAND structure is used to implement the AND and the OR arrays to take advantage of the n-channel device in the serial device path.

For example, when $p = 4$, Type 2 PLA can be implemented as follows:

1) The literal generator is implemented as shown in Fig. 9. For the NAND-NAND structure, which is logically equivalent to the AND-OR structure, we use Fig. 9(a). While, for the NOR-NOR structure, which is logically equivalent to the OR-AND structure,

TABLE IV
INVERTERS WITH VARIOUS THRESHOLDS

Input	Output			
	0	1	2	
0	3	3	3	3
1	0	3	3	-
2	0	0	3	-
3	0	0	0	0

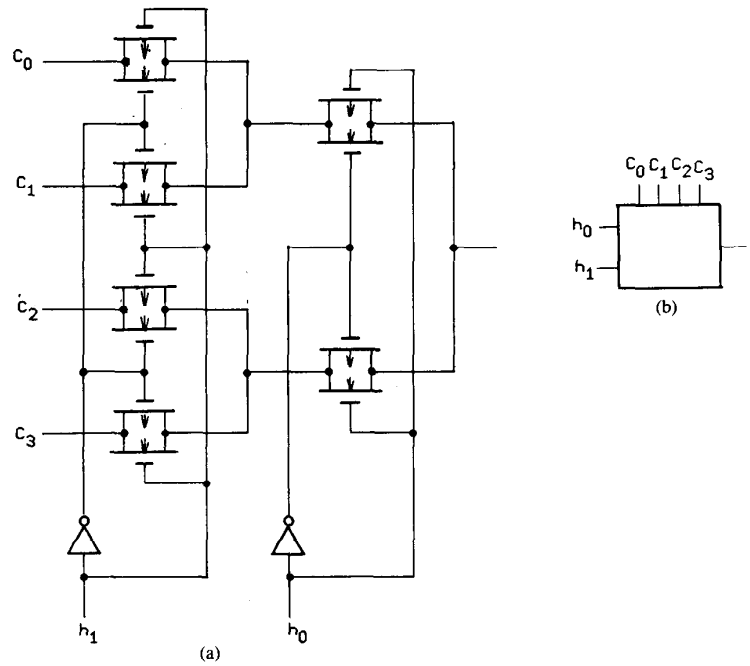


Fig. 10. Output encoder (four-valued) (a) CMOS realization. (b) Logic symbol.

every binary signal must be complemented and so we use Fig. 9(b). The input–output relations of the inverters having different thresholds are shown in Table IV. These inverters can be realized either by an ion implantation technique for n-MOS [10] or CMOS [22] or by voltage divider circuits using transistors [13].

2) The output encoder using a CMOS circuit is shown in Fig. 10(a) and denoted by the symbol shown in Fig. 10(b). For an NAND-NAND structure, we set $C_0 = 0, C_1 = 1, C_2 = 2$, and $C_3 = 3$, while for a NOR-NOR structure, we set $C_0 = 3, C_1 = 2, C_2 = 1$, and $C_3 = 0$.

C. Comparison of Type 1 PLA to Type 2 PLA

Table V compares Type 1 PLA's to Type 2 PLA's. Because a Type 2 PLA is easily implemented by MOS/CMOS technology, it is more suitable for VLSI than Type 1 PLA. Bounds on the number of distinct functions realized by both types of PLA's are derived in the Appendix, and summarized in the table. Although the result suggests that Type 2 PLA's usually require smaller arrays than Type 1 PLA's, it needs further study to verify it. Indeed, there is a function whose Type 1 PLA realization requires smaller arrays than Type 2 PLA (see Addendum of [21] distributed at ISMVL-84). Comparison of the complexities of these PLA's is quite interesting. Similar study can be found in [1].

In the next section, we will consider the design of Type 2 PLA's in detail.

IV. Output Encoding Problem

A. Optimal Output Encoding for Adder

In Type 2 PLA's, a p-valued output signal is represented by an r-tuple of two-valued signals $(h_0, h_1, \cdots, h_{r-1})$, where the natural binary encoding is used to relate the p-valued signal to binary signals, i.e., $(0, 0, \cdots, 0)$ represents 0, $(0, 0, \cdots, p-1)$ represents $1, \cdots$, and $(p-1, p-1, \cdots, p-1)$ represents $2^r - 1 = p - 1$. However, if another encoding is used to represent the p-valued signal, the size of the array can often be reduced.

Example 4.1: In realizing the adder in Example 3.1, we assign a pair of two-valued signals to represent a four-valued signal as shown in Table II. However, if we assign signals as shown in Table VI, we have Table VII. By using maps shown in Fig. 11, we have the minimum sum-of-products expression for h_1, h_0, and c_0 as follows:

$$h_1 = X_1^{\{0,2\}} \cdot X_2^{\{0,2\}} \vee X_1^{\{1,3\}} \cdot X_2^{\{1,3\}},$$

$$h_0 = X_1^{\{0,1\}} \cdot X_2^{\{0\}} \vee X_1^{\{0,3\}} \cdot X_2^{\{1\}} \vee X_1^{\{2,3\}} \cdot X_2^{\{2\}} \vee X_1^{\{1,2\}} \cdot X_2^{\{3\}},$$

$$C_0 = X_1^{\{2,3\}} \cdot X_2^{\{2\}} \vee X_1^{\{1,2\}} \cdot X_2^{\{3\}} \vee X_1^{\{3\}} \cdot X_2^{\{1,3\}}.$$

TABLE V
COMPARISON OF TYPE 1 PLA TO TYPE 2 TO REALIZE p-VALUED FUNCTIONS

		Type 1 PLA	Type 2 PLA
Structure		Literal generators Min-array, Max-array	Literal generators AND-array, OR-array Output encoders
Signals in array		p-valued	two-valued
Technology		Bipolar static	MOS/CMOS/Bipolar static/dynamic
Array size to realize aribitrary functions	H_1	$np+p-2$	$n \cdot p$
	H_2	m	rm
	W *	$\leq (p-1) \cdot p^{n-1}$	$\leq r \cdot p^{n-1}$
Number of* realizable functions	U B	$(p-1)^W \cdot t^{nW}$	$(p-1)^W \cdot t^{nW}$ +
	L B	$(p-1)^W \cdot t^{(n-u)W}$	$(p-1)^W \cdot t^{(n-u)W}$ +
Design method		g_0 can be omitted. $g_i(i=1,2,\ldots,p-2)$ are minimized by using g_s as don't care sets. $(i+1 \leq s \leq p-1)$.	$g_i(i=0,1,\ldots,p-1)$ are realized by $h_0, h_1, \ldots, h_{r-1}$. These expressions are minimized simultaneously. Good optput encoding reduce size.

n: number of the input variables
m: number of the output functions
W: number of the columns of PLA
*: When m=1
UB: Upper Bound, LB: Lower Bound
+: $p=2^r$, $p \geq 4$
$t=2^p-1$, $r=\lfloor \log_2 p \rfloor$, $W=p^u$.
[a] denotes the least integer equal to or greater than a

TABLE VI
OPTIMUM OUTPUT ENCODING OF ADDERS FOR TYPE 2 PLA

(a) Encoding for Sum

4-valued signal	2-valued signals
0	3 3
1	0 3
2	3 0
3	0 0

(b) Encoding for Carry

4-valued signal	2-valued signals
0	0 0
1	0 3
2	3 0
3	3 3

TABLE VII
TRUTH TABLE FOR ADDER (OUTPUT ENCODING OPTIMUM)

Input X_1	Input X_2	Output Sum h_1	Output Sum h_0	Output Carry C_1	Output Carry C_0
0	0	3	3	0	0
0	1	0	3	0	0
0	2	3	0	0	0
0	3	0	0	0	0
1	0	0	3	0	0
1	1	3	0	0	0
1	2	0	0	0	0
1	3	3	3	0	3
2	0	3	0	0	0
2	1	0	0	0	0
2	2	3	3	0	3
2	3	0	3	0	3
3	0	0	0	0	0
3	1	3	3	0	3
3	2	0	3	0	3
3	3	3	0	0	3

Fig. 12 shows the Type 2 PLA for this function. Note that the first two terms of C_0 are shared with h_0. In this PLA, only seven columns are used to realize the function. In this case, we need to permute the connection of constants in the output encoder to obtain the proper output values. (End of Example).

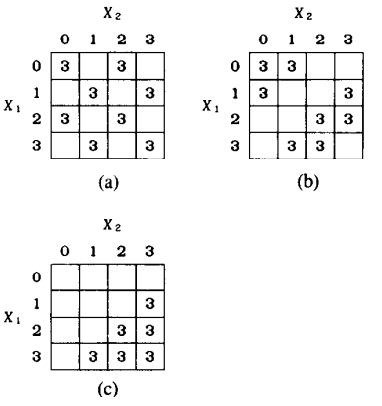


Fig. 11. Maps for adder using Type 2 with optimal output encoding. (a) h_1. (b) h_0. (c) C_0.

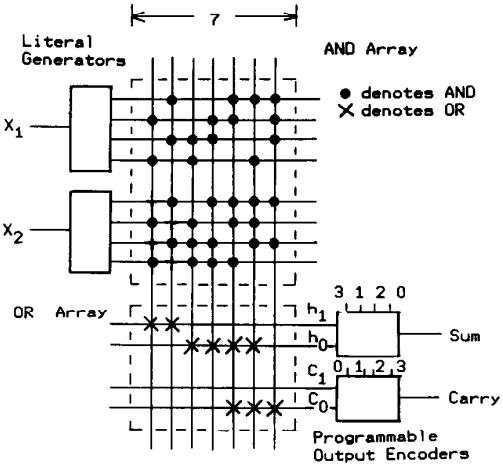


Fig. 12. Adder using Type 2 PLA with optimal output encoding.

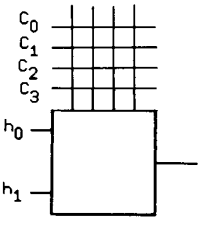


Fig. 13. Programmable output encoder (four-valued).

B. Optimal Output Encoding Problem for MVPLA

As was illustrated in the Example 4.1, different output encodings derive PLA's with different complexities. Suppose that we can use *programmable output encoders* shown in Fig. 13. In such a case, we can use any output encoding for each output.

Definition 4.1: The optimal output encoding of a Type 2 PLA is a set of encodings which makes the size of the arrays minimum.

For a p-valued single-output function, there are $p!$ different ways of encodings. The exhaustive way to find an optimum output encoding requires $p!$ minimization. For $p = 4$, the

TABLE VIII
ESSENTIALLY DIFFERENT OUTPUT ENCODINGS

Value	#1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
0	00	00	00	03	03	03	03	03	03	33	33	33
1	03	03	33	00	00	30	30	33	33	00	03	03
2	30	33	03	30	33	00	33	00	30	03	00	30
3	33	30	30	33	30	33	00	30	00	30	30	00

TABLE IX
AVERAGE NUMBER OF PRODUCTS IN FOUR-VALUED PLA'S

	Type 1 PLA	Type 2 PLA	
		Original Encoding	Optimum Encoding
n=2	6.4	6.1	5.3
n=3	20.0	19.1	17.4
n=4	67.4	64.4	61.0
n=5	251.5	244.3	235.3

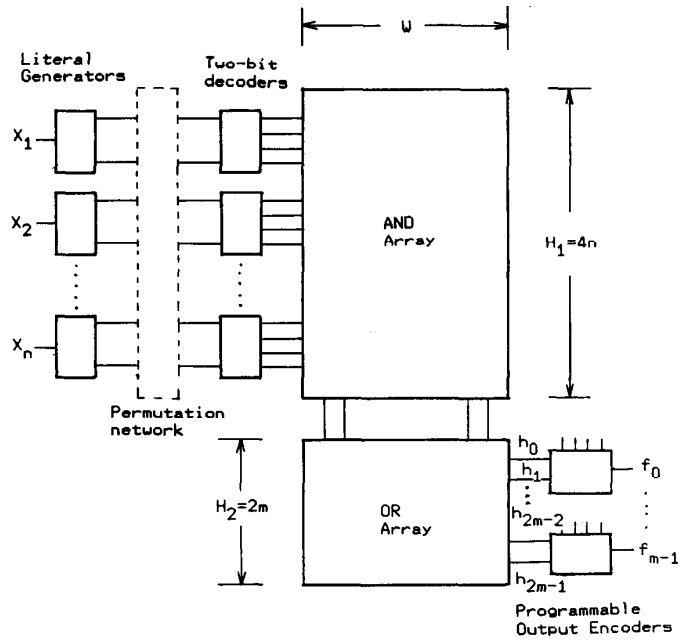


Fig. 14. Four-valued PLA with 2-bit decoders and programmable output encoders.

number is 4! = 24. This value can be reduced to 12 by considering the permutation of functions h_0 and h_1. Table VIII lists the 12 essentially different output encodings.

As for the optimum output encoding problem for an m-output function, the exhaustive search requires $(p!/[\log p])^m$ minimizations, which is impractical for large problems. A heuristic method for optimal output phase assignment [17] can be used to obtain a good encoding if not optimal. The method decides which to realize, either the function or its complement for each output, so as to make the arrays as small as possible. Therefore, this method selects a good encoding out of 2^{mr} possible encodings. The encoding shown in Table VI was obtained by this method. This encoding has been verified to be optimum by the exhaustive examination by using a computer.

C. Computer Simulation

Table IX compares the numbers of products for Type 1 PLA's, Type 2 PLA's with output encodings nonoptimized, and Type 2 PLA's with optimum output encodings. Randomly generated functions were used to compare the complexities of PLA's. For each function the number of input combinations which are mapped into i ($i = 0, 1, 2, 3$) is equal to 4^{n-1}, where n is the number of the input variables. The optimum output encodings were obtained by the exhaustive method. Minimization of the expressions were done by QM (modified Quine-McClusky method [19]) for $n = 2$ and 3 and by MINI2 for $n = 4$ and 5. When $n = 5$, output encoding optimum PLA's require on the average 3.7 percent fewer products than output encoding original PLA's. In most cases, Type 1 PLA's require more products than Type 2 PLA's with output encoding nonoptimized.

V. MULTIPLE-VALUED PLA WITH 2-BIT DECODERS

In a two-valued PLA with two-bit decoders, the size of the arrays can be reduced by considering the assignment of the input variables to the decoders [16], [17]. In an MVPLA having the structure shown in Fig. 14, the size of the array can be reduced by using the similar technique. We call this PLA a *Type 3 PLA*.

Example 5.1: Suppose that the adder shown in Table I is

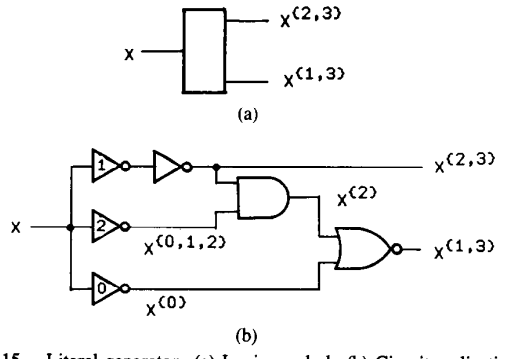


Fig. 15. Literal generator. (a) Logic symbol. (b) Circuit realization.

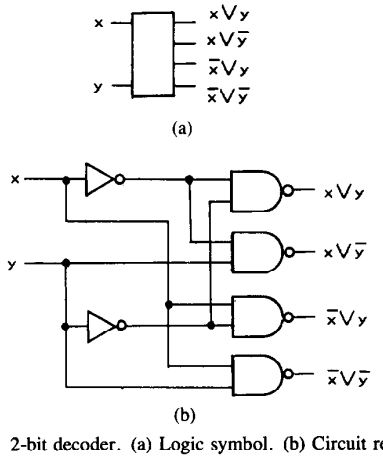


Fig. 16. 2-bit decoder. (a) Logic symbol. (b) Circuit realization.

realized by the PLA having a structure shown in Fig. 14. In this case, each literal generator generates two literals as shown in Fig. 15. In addition, we use the 2-bit decoder shown in Fig. 16. Between the literal generators and the 2-bit decoders, we use a permutation network. Now, introduce four independent two-valued variables y_1, y_2, y_3, and y_4, to represent X_1 and X_2.

Let $y_1 = X_1^{\{2,3\}}, y_2 = X_1^{\{1,3\}}, y_3 = X_2^{\{2,3\}}$, and $y_4 = X_2^{\{1,3\}}$.

Then, $\bar{y}_1 = X_1^{\{0,1\}}, \bar{y}_2 = X_1^{\{0,2\}}, \bar{y}_3 = X_2^{\{0,1\}}$, and $\bar{y}_4 = X_2^{\{0,2\}}$

By using the new variables y_1, y_2, y_3, and y_4, the adder can be represented as shown in Table X. Next, introduce two super variables $Y_1 = (y_1, y_3)$ and $Y_2 = (y_2, y_4)$. Then, S_1, S_0, and C_0 can be represented by maps shown in Fig. 17(a)–(c). From these maps, we have the minimum sum-of-products expression:

$$S_1 = Y_1^{\{03,30\}} \cdot Y_2^{\{00,03,30\}} \vee Y_1^{\{00,33\}} \cdot Y_2^{\{33\}},$$

$$S_0 = Y_2^{\{03,30\}},$$

$$C_0 = Y_1^{\{33\}} \vee Y_1^{\{03,30\}} \cdot Y_2^{\{33\}}.$$

The PLA realizing these functions requires only five products.

By optimizing the output encodings, the size of the PLA is

TABLE X
TRUTH TABLE OF ADDER FOR TYPE 3 PLA

X_1		X_2		Sum		Carry	
y_1	y_2	y_3	y_4	S_1	S_0	C_1	C_0
0	0	0	0	0	0	0	0
0	0	0	3	0	3	0	0
0	0	3	0	3	0	0	0
0	0	3	3	3	3	0	0
0	3	0	0	0	3	0	0
0	3	0	3	3	0	0	0
0	3	3	0	3	3	0	0
0	3	3	3	0	0	0	3
3	0	0	0	3	0	0	0
3	0	0	3	3	3	0	0
3	0	3	0	0	0	0	3
3	0	3	3	0	3	0	3
3	3	0	0	3	3	0	0
3	3	0	3	0	0	0	3
3	3	3	0	0	3	0	3
3	3	3	3	3	0	0	3

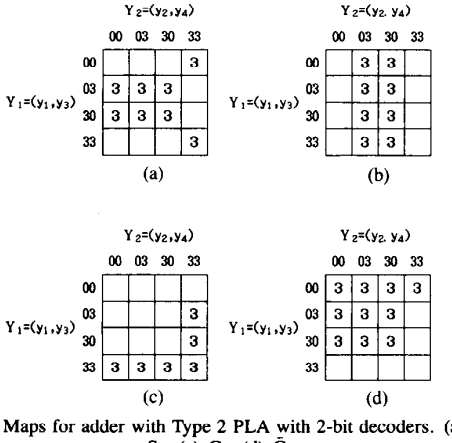


Fig. 17. Maps for adder with Type 2 PLA with 2-bit decoders. (a) S_1. (b) S_0. (c) C_0. (d) $\bar{C}_0$.

TABLE XI
OPTIMUM OUTPUT ENCODING FOR ADDER FOR TYPE 3 PLA

(a) Encoding for Sum

4-valued signal	2-valued signals	
0	0	0
1	0	3
2	3	0
3	3	3

(b) Encoding for Carry

4-valued signal	2-valued signals	
0	0	3
1	0	0
2	3	3
3	3	0

further reduced. When we use the output encodings shown in Table XI, we have the PLA with only four columns. In this case, $\bar{C}_0$ is realized instead of C_0. As shown in Fig. 17(d), $\bar{C}_0$ can be written as

$$\bar{C}_0 = Y_1^{\{03,30\}} \cdot Y_2^{\{00,03,30\}} \vee Y_1^{\{00\}}.$$

Fig. 18 shows the PLA realizing a four-valued adder. Note that the first term of $\bar{C}_0$ is shared with S_1.(End of example).

Table XII compares the size of the four-valued adders for three different PLA's. Type 2 PLA's require smaller arrays than Type 1 PLA's. And Type 3 PLA's require smaller arrays than Type 2 PLA's although Type 3 PLA's require additional hardware such as a permutation network and 2-bit decoders.

VI. Conclusion and Comparison to Other Methods

In this paper, the author proposed three types of PLA's. Because these PLA's use the minimum universal set of literals, they require the minimum number of literal lines.

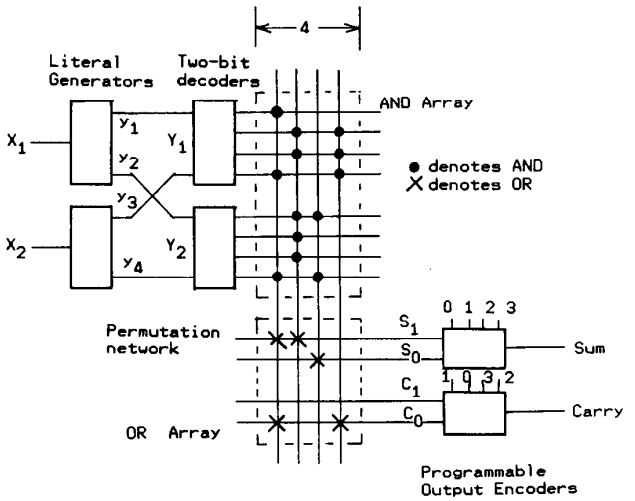


Fig. 18.

TABLE XII
COMPARISON OF SIZE OF ADDERS

		Type 1 PLA	Type 2 PLA		Type 3 PLA	
			encoding original	encoding optimum	encoding original	encoding optimum
One-figure adder X_0 +) Y_0 / Z_1 Z_0	H_1	10	8	8	8	8
	H_2	2	4	4	4	4
	W	13	9	7	5	4
	S	156	108	84	60	48
Two-figure adder X_1X_0 +) Y_1Y_0 / $Z_2Z_1Z_0$	H_1	18	16	16	16	16
	H_2	3	6	6	6	6
	W	79	63	52	17	14
	S	1659	1386	1144	374	308

$S = W \cdot (H_1 + H_2)$

The array structure for four-valued logic proposed by [9] uses 14 literals for each input. On the other hand, the method in this paper uses only four literals. Thus, the height of the MIN array in this paper is about 29 percent of [9]. The four-valued PLA which can be obtained by extending [11] uses only four literals $\{X^{\{0\}}, X^{\{1\}}, X^{\{2\}}, X^{\{3\}}\}$. However, the MIN array using these literals cannot be minimized effectively, because each column of the MIN array realizes the product $X_1^{S_1}X_2^{S_2}\cdots X_n^{S_n}$, where $X_j^{S_j}$ may take only five different patterns, i.e., $X^{\{0\}}, X^{\{1\}}, X^{\{2\}}, X^{\{3\}}$, and $X^{\{0,1,2,3\}}$. On the other hand, the Type 1 PLA proposed in this paper uses the minimum universal set of literals

$$\{X^{\{1,2,3\}}, X^{\{0,2,3\}}, X^{\{0,1,3\}}, X^{\{0,1,2\}}\},$$

and the MIN array can usually be minimized into the half of [11] or even smaller, because each column of the MIN array realizes the product of literal functions $X_j^{S_j}$, where the literal may take 15 different patterns. Indeed, Table 2 of [9] implies that the number of columns for Type 1 PLA is, on the average, 50–60 percent of [11]. Thus, the Type 1 PLA proposed in this paper usually requires much smaller arrays than previously published ones [11], [9].

Although, the Type 1 PLA is easy to implement by bipolar technology, it is unsuitable for MOS realization. The Type 2 PLA is suitable for MOS/CMOS implementation. Computer simulation shows that Type 2 PLA's are, on the average, smaller than Type 1 PLA's.

Type 3 PLA's are suitable for adders although they need additional hardware such as permutation network and 2-bit decoders.

APPENDIX

A. Minimum Universal Set of Literals for MVPLA's

Definition A.1: Let $L = \{X^{S_0}, X^{S_1}, \cdots, X^{S_{k-1}}\}$ be a set of literals of X. L is said to be *universal* if any other literal of X can be represented by an AND (or a MIN) operation among the elements in L. L is said to be *minimum* if k is the minimum.

Theorem A.1: $L = \{X^{S_0}, X^{S_1}, \cdots, X^{S_{p-1}}\}$ is the minimum universal set of literals of X, where $S_i = \{\bar{i}\} = P - \{i\}, i = 0, 1, \cdots, p-1$ and $P = \{0, 1, \cdots, p-1\}$.

Proof: L is universal: It is sufficient to show that any literals X^A of X can be represented by the AND operation of the elements in L. X^A can be represented by

$$X^A = \bigwedge_{i \in B} X^{S_i}, \qquad \text{where } B = \bar{A} = P - A,$$

$$\text{and } P = \{0, 1, \cdots, p-1\}. \quad \text{(A.1)}$$

L is the minimum: There are 2^p literals including constant zero and constant $p-1$. In order to represent all the literals in the forms of (A.1), we need at least p different elements. To show that L is unique, assume, on the contrary, that there is another set $L' \neq L$ which is a minimum universal set. Thus, there is i such that $X^{S_i} \notin L'$. Since L' is universal, it realizes S_i itself (as the products of literals in L'). That is,

$$X^{S_i} = X^{S_1'} \cdot X^{S_2'} \cdot \cdots \cdot X^{S_m'} \qquad \text{where } X^{S_j'} \in L'.$$

Because $S_i = S_1' \cap S_2' \cap \cdots \cap S_m'$ and $S_i = P - \{i\}$, every S_j' contains all the elements in $P - \{i\}$, and at least one S_j',

does not contain the element i. But, if follows that $S'_{j'} = S_i$, a contradiction. It must be that L is unique. (Q.E.D.)

B. On the Number of Functions Realized by a PLA with W Columns

Theorem A.2: Let $A(n, p, W)$ be the number of distinct n-variable p-valued single-output functions realized by a Type 1 PLA with W columns. When $W = p^u$, $(p-1)^W \cdot t^{W(n-u)} \leq A(n, p, W) \leq (p-1)^W \cdot t^{nW}$, where $t = 2^p - 1$.

Proof: Lower Bound: Consider a function f which can be represented by the expression

$$f(X_1, X_2, \cdots, X_n) = \bigvee_{a \in P^u} (X_1^{a_1} \cdot X_2^{a_2} \cdot \cdots X_u^{a_u}) \cdot (i) X_{u+1}^{S_{u+1}} \cdots X_n^{S_n} \quad \text{(A.2)}$$

where $a = (a_1, \cdots, a_u), i = 1, 2, \cdots, p-1$, and $S_k \subseteq P (k = u+1, \cdots, n)$. Note that the total number of products in (A.2) is $W = p^u$. The number of distinct nonzero functions realized by

$$(i) X_{u+1}^{S_{u+1}} \cdot \cdots \cdot X_n^{S_n} \quad \text{(A.3)}$$

is $(p-1) \cdot t^{(n-u)}$, because for each set $(S_{u+1}, \cdots, S_n)$ and for each i, there exists a unique function, and there are $t = 2^p - 1$ possible ways to choose a subset S_i of P. The total number of distinct nonzero functions realized by (A.2) is at least $(p-1)^W \cdot t^{W(n-u)}$, because for each a every combination of i and $S_k (k = u+1, \cdots, n)$ in (A.3) will make different functions.

Upper Bound: f can be represented as a MAX-of-MIN's expression:

$$f(X_1, X_2, \cdots, X_n) = \bigvee_{(S_1, S_2, \cdots, S_n)} (i) \cdot X_1^{S_1} \cdot X_2^{S_2} \cdot \cdots \cdot X_n^{S_n} \quad \text{(A.4)}$$

where $f(S_1, S_2, \cdots, S_n) = i$ is a constant and $S_k \subseteq P (k = 1, 2, \cdots, n)$. It is clear that the number of distinct functions realized by (A.4) with W products is at most $(p-1)^W \cdot t^{nW}$. (Q.E.D.)

Lemma A.1: Let $B(n, p, W)$ be the number of distinct n-variable p-valued input two-valued output functions represented by the expression

$$f(X_1, X_2, \cdots, X_n) = \bigvee_{(S_1, S_2, \cdots, S_n)} X_1^{S_1} \cdot X_2^{S_2} \cdot \cdots \cdot X_n^{S_n} \quad \text{(A.5)}$$

with W products. When $W = p^u$, $t^{W(n-u)} \leq B(n, p, W) \leq t^{Wn}$, where $t = 2^p - 1$.

Proof: Lower Bound: Consider the expression which has the following form:

$$f(X_1, X_2, \cdots, X_n) = \bigvee_{a \subseteq P^u} X_1^{a_1} \cdot X_2^{a_2} \cdot \cdots \cdot X_u^{a_u} \cdot X_{u+1}^{S_{u+1}} \cdot \cdots \cdot X_n^{S_n} \quad \text{(A.6)}$$

where $a = (a_1, a_2, \cdots, a_u)$, and $S_k \subseteq P$ and $k = u+1, \cdots, n$.

Note that the number of products in (A.6) is $W = p^u$. The number of distinct nonzero functions realized by

$$X_{u+1}^{S_{u+1}} \cdot \cdots \cdot X_n^{S_n} \quad \text{(A.7)}$$

is $t^{(n-u)}$. Thus, the total number of distinct nonzero functions represented by (A.6) is at least $t^{W(n-u)}$, because for each $(S_{u+1}, \cdots, S_n)$ in (A.7), we have distinct function.

Upper Bound: It is clear that the number of distinct functions realized by (A.5) with W products is at most t^{nW}. (Q.E.D)

Theorem A.3: Let $C(n, p, W)$ be the number of distinct p-valued single-output functions realized by a p-valued Type 2 PLA with W columns. When $W = p^u$, $p = 2^r$ and $p \geq 4$, $t^{W(n-u)} \cdot (p-1)^W \leq C(n, p, W) \leq t^{Wn} \cdot (p-1)^W$.

Proof: An n-input and r-output function realized by the body of a Type 2 PLA can be represented as

$$f(X_1, X_2, \cdots, X_n, X_{n+1}) = \bigvee_{(S_1, \cdots, S_{n+1})} X_1^{S_1} \cdot X_2^{S_2} \cdot \cdots \cdot X_n^{S_n} \cdot X_{n+1}^{S_{n+1}}, \quad \text{(A.8)}$$

where X_{n+1} denotes a variable for the outputs and it takes r values. It is clear that (A.8) also represents an $(n+1)$-input single-output function. Therefore, the number of the functions realized by a Type 2 PLA with W columns is equal to the number of $(n+1)$-variable functions represented by an expression (A.8) with W products.

Because the last variable takes r values, the number of different literals is $2^r - 1 = p - 1$. Similar to the proof of Lemma A.1, we have: lower bound: $t^{W(n-u)} \cdot (p-1)^W$, and upper bound: $t^{Wn} \cdot (p-1)^W$. (Q.E.D.)

References

[1] E. A. Bender, J. T. Butler, and H. G. Kerkhoff, "Comparing the sum with the max for use in four-valued PLA's," in *Proc. ISMVL-85,* May 1985, pp. 30–35.

[2] R. G. Daniels and W. C. Bruce, "Built-in self-test trends in Motorola microprocessors," *IEEE Design Test,* pp. 64–71, Apr. 1985.

[3] J. A. Darringer and W. H. Joyer, Jr., "A new look at logic synthesis," in *Proc. 17th Design Automat. Conf.* 1980, pp. 543–549.

[4] A. J. de Geus and W. Cohen, "Optimization of combinational logic using a rule based expert system," *J. IEEE Design Test,* Aug. 1985, pp. 22–32.

[5] H. Fleisher, "The implementation and use of multivalued logic in a VLSI environment," in *Proc. ISMVL-83,* May 1983, pp. 138–143.

[6] D. A. Freitas and K. W. Current, "CMOS circuit for quaternary encoding and decoding," in *Proc. ISMVL-84,* May 1984, pp. 164–168.

[7] S. J. Hong, R. G. Cain, and D. L. Ostapko, "MINI: A heuristic approach for logic minimization," *IBM J. Res. Develop.,* pp. 443–458, Sept. 1974.

[8] S. L. Hurst, "Multiple-valuc logic: Its status and its future," *IEEE Trans. Comput.,* vol. C-33, pp. 1160–1179, Dec. 1984.

[9] M. Imme and C. A. Papachristou, "Simplification of MVL functions and implementation via a VLSI array structure," in *Proc. ISMVL-85,* May 1985, pp. 242–248.

[10] M. Kameyama, T. Hanyu, M. Esashi, T. Higuchi, and T. Ito, "Implementation of quaternary NMOS integrated circuits for pipelined image processing," in *Proc. ISMVL-85,* May 1985, pp. 226–232.

[11] H-L Kuo and K-Y Fang, "The multiple-valued programmable logic array and its application in modular design" in *Proc. ISMVL-85,* May 1985, pp. 10–18.

[12] H. F. Law and M. Shoji, "PLA design of the BELLMAC-32A microprocessor," in *Proc. ICCC-82,* 1982, pp. 161–164.

[13] E. J. McCluskey, "Logic design of MOS ternary logic," in *Proc. ISMVL-80,* June 1980, pp. 1–5.

[14] S. Powell, E. Iodice, and E. Friedman, "An automated, low power, high speed complementary PLA design system for VLSI application," in *Proc. ICCD-84,* 1984, pp. 314–319.

[15] R. Rudell and A. L. M. Sangiovanni-Vincentelli, "ESPRESSO-MV: Algorithms for multiple-valued logic minimization," in *Proc. 1985 Custom Integrated Circuits Conf.*, May 1985, pp. 230-234.

[16] 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.

[17] ——, "Input variable assignment and output phase optimization of PLA's," *IEEE Trans. Comput.*, vol. C-33, pp. 879-894, Oct. 1984.

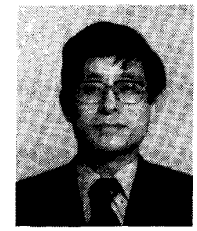
[18] ——, *Programmable Logic Array: How to make and How to Use.* (in Japanese) Tokyo, Japan, NIKKAN KOGYOU, May, 1986.

[19] T. Sasao, "MACDAS: Multi-level AND-OR circuit synthesis using two-variable function generators," in *Proc. 23rd Design Automat. Conf.*, June 1986, pp. 86-93.

[20] W. R. Smith, III, "Minimization of multivalued functions," in *Computer Science and Multiple-Valued Logic*, D. C. Rine Ed. New York: North Holland, 1977.

[21] P. Tirumalai and J. T. Butler, "On the realization of multiple-valued logic functions using CCD PLA's," in *Proc. ISMVL-84*, May 1984, pp. 33-42, and Addendum distributed at ISMVL-84.

[22] C. Zukeran, C. Afuso, M. Kameyama, and T. Higuchi, "Design of new low-power quaternary CMOS logic circuits based on multiple ion implants," in *Proc. ISMVL-85*, May 1985, pp. 84-90.



Tsutomu Sasao (S'72-M'77) was born in Osaka, Japan on January 26, 1950. He received the B.E., M.E., and Ph.D. degrees in electronic engineering from Osaka University, Osaka, Japan, in 1972, 1974, and 1977, respectively.

From 1977 to 1988, he was with Osaka University. He is currently with Kyushu Institute of Technology, Iizuka, Japan. His research interests includes logic design and switching theory. He specializes in the design of PLA and application of multiple-valued logic to the design automation. From February 1982, he spent a year at the IBM T. J. Watson Research Center, where he developed a PLA minimization system. He has published four books on switching theory and logical design in Japanese including a book on PLA design and test.

Dr. Sasao is a member of the Institute of Electronics and Communication Engineers of Japan. He received the NIWA Memorial Award in 1979, and a Distinctive Contribution Award from IEEE Computer Society MVL-TC for the paper presented at ISMVL 1986. He has served as Asia Area Program Chairman of the International Symposium on Multiple-valued Logic held in 1984, and is a member of the Executive Committee of the IEEE Computer Society Technical Committee on Multiple-Valued Logic.