# LOGIC SYNTHESIS FOR PLA WITH 2-INPUT LOGIC ELEMENTS

*Hiroaki Yoshida*[†], *Hiroaki Yamaoka*[†], *Makoto Ikeda*[†], *and Kunihiro Asada*[††]

[†]Department of Electronic Engineering, University of Tokyo
[††]VLSI Design and Education Center(VDEC), University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan

**ABSTRACT**

In this paper, we present a new logic synthesis method for PLA with 2-input logic elements. A PLA with 2-input logic elements can achieve low-power dissipation and high-speed operation by using latch sense-amplifiers and a charge sharing scheme. In addition, an arbitrary 2-input logic function is conveniently implemented in place of the conventional AND/OR planes. Therefore it can realize some classes of logic functions in a smaller circuit area. Since the proposed method makes full use of the existing multiple-valued logic minimization algorithms along with a new logic extraction technique for 2-input functions, it can be easily implemented and can handle practical circuits. The method has been implemented and the experimental results are presented.

## 1. INTRODUCTION

In the past two decades, *Programmable Logic Arrays* (PLAs) have been frequently used because of the advantages such as high-speed operation, easy to implement and modify, and accurate area and performance predictability. Recently, PLAs have emerged again as an efficient style for implementing high performance designs. For example, the IBM 1-GHz PowerPC processor used PLAs to implement control logic[1]. Khatri *et al.* proposed a VLSI design methodology using a network of PLAs[2]. Their scheme can dramatically reduce the cross-talk between the signal wires with a significant improvement of area and performance.

On the other hand, the conventional PLA implementations are relatively large in comparison to the implementation styles which realize multi-level logic. To overcome this drawback, some variant forms of PLA which implement Boolean functions efficiently have been proposed, such as three-level PLA and PLA with two-input decoders. Generally, these variants are slower.

In this paper, a logic synthesis method for PLA with 2-input logic elements is presented. This is a generalization of the method for AND-XOR-OR type sense-amplifying PLA[3]. Since our method is based on multiple-valued logic and makes full use of the existing algorithms, it can be easily implemented and can handle practical circuits.

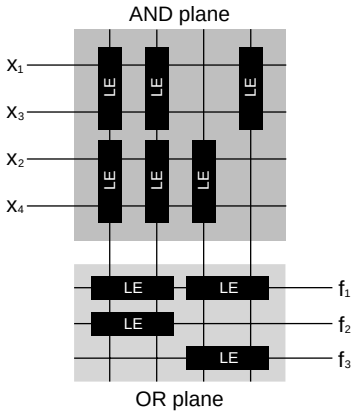


**Fig. 1**. PLA with 2-input logic elements.

## 2. PLA WITH 2-INPUT LOGIC ELEMENTS

The PLA with 2-input logic elements can achieve low-power dissipation and high-speed operation by using latch sense-amplifiers and a charge sharing scheme[4]. As illustrated in Fig. 1, some AND/OR cells can be replaced with 2-input logic cells, which realize arbitrary 2-input logic functions denoted by **LE** in the figure. Since the replacement is achieved by reconnecting some local wires, there is almost no effect on area and delay. The output of AND-plane is the Boolean AND of the outputs of logic elements, and the output of OR-plane is the Boolean OR of the outputs of logic elements. That is, the present PLA realizes LE-AND-LE-OR 4-level logic.

It is well known that the synthesis method for a PLA with input decoders, which is based on the multiple-valued minimization, can reduce the number of the product terms [5]. The PLA with 2-input logic elements can be viewed as a PLA with 2-input decoders in AND- and OR-plane, as illustrated in Fig. 2. Therefore, the present PLA can realize Boolean functions more efficiently than PLA with 2-input decoders in only AND-plane.

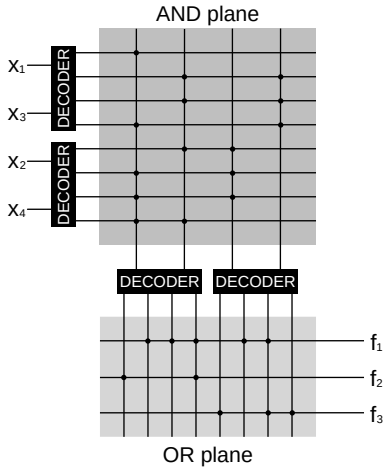


**Fig. 2**. PLA with decoders.

## 3. LOGIC SYNTHESIS FOR PLA WITH 2-INPUT LOGIC ELEMENTS

### 3.1. Definitions

Let $X_i$ be a variable taking a value from the set $P_i = \{0, \ldots, p_i - 1\}$. A **literal** $X_i^{S_i}$ represents the Boolean function

$$X_i^{S_i} = \begin{cases} 0 & \text{if } X_i \notin S_i \\ 1 & \text{if } X_i \in S_i \end{cases}$$

where $S_i$ is a subset of $P_i$. The **complement** of the literal $\overline{X}_i^{S_i}$ is the literal $X_i^{\overline{S}_i}$. A **product term** is a Boolean product of literals. A **sum-of-products** is a Boolean sum of product terms. The **supercube** of product terms $S$ and $T$ is the product term

$$X_1^{S_1 \cup T_1} X_2^{S_2 \cup T_2} \cdots X_n^{S_n \cup T_n}$$

which is the smallest product term containing both $S$ and $T$. Similarly, the supercube of a sum-of-products $F$ is the smallest product term containing every product term of $F$. The **cofactor** $S_T$ of a product term $S$ with respect to a product term $T$ is

$$S_T = \begin{cases} 0 & \text{if } S_i \cap T_i = \emptyset \ \exists i \\ X_1^{S_1 \cup \overline{T}_1} X_2^{S_2 \cup \overline{T}_2} \cdots X_n^{S_n \cup \overline{T}_n} & \text{otherwise.} \end{cases}$$

Similarly, the cofactor $F_S$ of a sum-of-products $F$ with respect to a product term $S$ is the sum of the cofactor of each product term of $F$ with respect to $S$.

### 3.2. Overall Flow

As presented in [5], the output signals of AND-plane correspond to the products of 4-valued literals. Therefore, logic elements in OR-plane can generate an arbitrary function of two product terms. Let $S$ and $T$ be product terms, then the functions to be generated by logic elements in OR-plane are categorized as follows: 1) $S$, 2) $\overline{S}$, 3) $\overline{S} \cdot \overline{T}$, 4) $S\overline{T}$, 5) $S \oplus T$ or $\overline{S \oplus T}$. These functions are referred to as the **LE-terms** of type 1–5 respectively. The expressions which can be realized by LE-PLA can be viewed as the sum of LE-terms. Note that the other functions, which are not catogorized above, are redundant because they can be expressed by the sum of LE-terms. The objective of our method is to minimize the number of product terms needed.

The basic idea of our approach is simple. It finds LE-terms contained in given Boolean functions, and then performs the minimization considering them. The next two sections describe them in detail.

### 3.3. Extraction of LE-terms

The most important step in our synthesis flow is to find LE-terms such that a given Boolean function $f$ contains them. Since LE-terms of type 1 are implicants of a function $f$, we are interested in how to find LE-terms of the other types. The remainder of this section describes how to extract LE-terms of each type.

**type 2**: The complement of a product term can be viewed as the sum of literals (*e.g.* $\overline{X_1^{\{1,2,3\}} X_2^{\{0,2\}}} = X_1^{\{0\}} + X_2^{\{1,3\}}$). This type of LE-term can easily be obtained by picking up all product terms which consist of a literal and complementing them.

**type 3**: The product of two complements of products can be viewed as the product of two sum of literals. This type of LE-term can be obtained as follows: 1) picking up all product terms which consist of one or two literals, and 2) factoring them. For factoring, we utilize the multiple-valued factorization algorithm presented in [6]. For example, consider the following sum-of-products

$$F = X_1^{\{0\}} X_3^{\{1,2\}} + X_2^{\{0,1,3\}} X_3^{\{1,2\}} + X_1^{\{1\}}.$$

By factoring and complementing, an LE-term of type 3 is obtained as

$$\begin{aligned} F &= (X_1^{\{0,1\}} + X_2^{\{0,1,3\}})(X_1^{\{2\}} + X_3^{\{1,2\}}) \\ &= \overline{X_1^{\{2,3\}} X_2^{\{2\}}} \cdot \overline{X_1^{\{0,1,3\}} X_3^{\{0,3\}}}. \end{aligned}$$

**type 4**: The extraction of LE-terms of type 4 is based on the following theorem:

**Theorem 3.1** *Let $S = X_1^{S_1} X_2^{S_2} \cdots X_n^{S_n}$ and $T$ be cubes and $F$ be a sum-of-products. Then,*

$$\overline{F}_S \subseteq T \Longleftrightarrow F \supseteq S\overline{T}$$

*where $F_S$ is the cofactors of $F$ with respect to a cube $S$.*

**Given:** a sum-of-products $F$
**Procedure** cube_gen
  $C = \{\}$
  **for** each product term $X_1^{S_1}X_2^{S_2}\cdots X_n^{S_n}$ in $F$
    **for** each $1 \le i \le n$
      **for** all subset $T$ of $\overline{S}_i$
        $C = C \cup \{X_1^{S_1}X_2^{S_2}\cdots X_i^{S_i \cup T}\cdots X_n^{S_n}\}$
      **end for**
    **end for**
  **end for**
  **return** $C$
**end Procedure**

**Fig. 3**. Cube generation procedure.

This theorem states that once a cube $S$ is given, a cube $T$ which satisfies a condition $F \supseteq S\overline{T}$ is obtained. Since we may be interested in the smallest cube $T$, the supercube of $\overline{F}_S$ can be used as $T$. Since there are $2^{4n}$ possible cubes where $n$ is the number of variables, we cannot examine all of them practically. To overcome this difficulty, we have developed a heuristic technique shown in Fig. 3. The number of the cubes generated by this technique is reduced to at most $15mn$ where $m$ is the number of the cubes in a given sum-of-products. To illustrate how to obtain LE-terms of this type, consider the sum-of-products

$$F = X_1^{\{0\}}X_2^{\{3\}}X_3^{\{0,1\}} + X_1^{\{0\}}X_2^{\{1\}} + X_1^{\{1,2\}}X_2^{\{1,3\}}$$

and let the cube $S$ be $X_1^{\{0,1,2\}}X_2^{\{1,3\}}$. The supercube of $\overline{F}_S$ is calculated as follows.

$$supercube(\overline{F}_S) = X_1^{\{0\}}X_2^{\{3\}}X_3^{\{2,3\}}$$

Finally, we have an LE-term of type 4

$$X_1^{\{0,1,2\}}X_2^{\{1,3\}} \cdot \overline{X_1^{\{0\}}X_2^{\{3\}}X_3^{\{2,3\}}}.$$

**type 5**: In the same way as type 4, LE-terms of type 5 can be extracted by using the following theorem and corollary.

**Theorem 3.2** *Let $S = X_1^{S_1}X_2^{S_2}\cdots X_n^{S_n}$ and $T$ be cubes and $F$ be a sum-of-products. Then,*

$$\overline{F}_S \subseteq T \subseteq F_{\overline{X}_1^{S_1}}F_{\overline{X}_2^{S_2}}\cdots F_{\overline{X}_n^{S_n}} \Longleftrightarrow F \supseteq S \oplus T$$

*where $F_S$ and $F_{X_k^{S_k}}$ are the cofactors of $F$ with respect to a cube $S$ and a literal $X_k^{S_k}$ respectively.*

**Corollary 3.1** *Let $S = X_1^{S_1}X_2^{S_2}\cdots X_n^{S_n}$ and $T$ be cubes and $F$ be a sum-of-products. Then,*

$$\overline{F_{\overline{X}_1^{S_1}}F_{\overline{X}_2^{S_2}}\cdots F_{\overline{X}_n^{S_n}}} \subseteq T \subseteq F_S \Longleftrightarrow F \supseteq \overline{S \oplus T}$$

*where $F_S$ and $F_{X_k^{S_k}}$ are the cofactors of $F$ with respect to a cube $S$ and a literal $X_k^{S_k}$ respectively.*

**Given:** a sum-of-products $F$
**Procedure** LE-PLA
  $X = \{\}$
  Extract LE-terms of type 2 and type 3, and put them into $X$.
  $C$ = cube_gen($F$)
  **for** each $S = X_1^{S_1}X_2^{S_2}\cdots X_n^{S_n} \in C$
    $T = supercube(\overline{F}_S)$
    $X = X \cup \{S\,\overline{T}\}$
    **if** $S$ satifies $\overline{F}_S \subseteq F_{\overline{X}_1^{S_1}}F_{\overline{X}_2^{S_2}}\cdots F_{\overline{X}_n^{S_n}}$ **then**
      Simplify $\overline{F}_S$ with $F_{\overline{X}_1^{S_1}}F_{\overline{X}_2^{S_2}}\cdots F_{\overline{X}_n^{S_n}}$ as a *don't care* set and obtain $T$.
      $X = X \cup \{S \oplus T\}$
    **end if**
  **end for**
  $G = 0$
  **for** each LE-term $T_k$ in $X$
    Create a new variable $P_k$ to represent $T_k$.
    $G = G + (T_k \oplus P_k)$
  **end for**
  Simplify $F$ with $G$ as a *don't care* set and obtain $F_{LE}$.
  Replace $P_k$ in $F_{LE}$ with $T_k$.
  **return** $F_{LE}$
**end Procedure**

**Fig. 4**. Synthesis procedure.

A cube $S$ is also generated by the procedure shown in Fig. 3. Since it is impractical to enumerate all of the cubes which satisfies the above conditions, we simplify $\overline{F}_S$ with $F_{\overline{X}_1^{S_1}}F_{\overline{X}_2^{S_2}}\cdots F_{\overline{X}_n^{S_n}}$ as a *don't care* set, and obtain a cube $T$.

### 3.4. Synthesis with Extracted LE-terms

Our synthesis procedure is similar to the method presented in [3]. The details of this procedure are shown in Fig. 4. In the procedure, we utilize the technique used in Boolean division [7] to synthesize a given Boolean function with extracted LE-terms. For example, suppose a Boolean function such as

$$F = X_1^{\{0\}}X_3^{\{0,3\}} + X_2^{\{0,3\}}X_3^{\{2,3\}} + X_1^{\{2,3\}} + X_3^{\{1\}}$$

and the extracted LE-terms are

$$\overline{X_1^{\{0,1\}}X_3^{\{0,2,3\}}} \text{ and } X_1^{\{1,2,3\}} \cdot \overline{X_1^{\{0,1\}}X_2^{\{1,2\}}X_3^{\{2,3\}}}.$$

We create new variables $P_1$ and $P_2$ to represent each LE-terms, and form the don't care set

$$G = (X_1^{\{0,1\}}X_3^{\{0,2,3\}} \oplus P_1) + (X_1^{\{1,2,3\}} \cdot \overline{X_1^{\{0,1\}}X_2^{\{1,2\}}X_3^{\{2,3\}}} \oplus P_2).$$

By simplifying $F$ with $G$ as a don't care set, we obtain the synthesized expression

$$\begin{aligned} F_{LE} &= X_1^{\{0,2,3\}}X_3^{\{0,3\}} + P_2 \\ &= X_1^{\{0,2,3\}}X_3^{\{0,3\}} + X_1^{\{1,2,3\}} \cdot \overline{X_1^{\{0,1\}}X_2^{\{1,2\}}X_3^{\{2,3\}}}. \end{aligned}$$

**Table 1**. Experimental results.

| *circuit* | *AND-OR* | | *LE-AND-OR* | | *AND-LE-OR* | | *LE-AND-LE-OR* | |
|---|---|---|---|---|---|---|---|---|
| | *#products* | *time[sec]* | *#products* | *time[sec]* | *#products* | *time[sec]* | *#products* | *time[sec]* |
| Z5xp1 | 65 | 0.1 | 53 | 0.1 | 62 | 0.6 | 52 | 2.0 |
| add6 | 355 | 0.5 | 37 | 0.1 | 325 | 93.2 | 37 | 1.5 |
| addm4 | 200 | 0.4 | 109 | 0.3 | 193 | 4.9 | 99 | 13.4 |
| adr4 | 75 | 0.1 | 17 | 0.0 | 69 | 0.7 | 17 | 0.1 |
| dist | 123 | 0.1 | 75 | 0.1 | 120 | 2.3 | 70 | 4.8 |
| f51m | 77 | 0.1 | 51 | 0.1 | 69 | 0.4 | 48 | 2.0 |
| l8err | 52 | 0.1 | 39 | 0.0 | 49 | 0.6 | 38 | 1.0 |
| m181 | 42 | 0.1 | 30 | 0.1 | 40 | 0.2 | 28 | 14.6 |
| mlp4 | 128 | 0.2 | 97 | 0.1 | 124 | 1.3 | 92 | 25.6 |
| rd73 | 127 | 0.0 | 37 | 0.0 | 113 | 5.0 | 34 | 1.4 |
| root | 57 | 0.1 | 42 | 0.0 | 52 | 0.9 | 40 | 1.4 |
| sqr6 | 49 | 0.0 | 42 | 0.0 | 49 | 0.1 | 40 | 0.3 |
| *total* | 1350 | 1.8 | 629 | 0.9 | 1265 | 110.2 | 595 | 68.1 |

## 4. EXPERIMENTAL RESULTS

The method described in the paper has been implemented as a part of ESPRESSO-MV[8]. Table 1 shows the results on the `math` PLA benchmark circuits. In the table, *LE-AND-OR*, *AND-LE-OR*, and *LE-AND-LE-OR* correspond to PLAs which have logic elements in AND-plane, OR-plane, and both planes respectively. The input variable assignments for LE-AND-OR and LE-AND-LE-OR type PLAs were performed by using the heuristic algorithm[5] which is implemented in ESPRESSO-MV. The results show that LE-AND-LE-OR type PLA can realize the Boolean functions in the least product terms among the four types of PLAs. As for the other circuits such as `indust` and `random`, our method and ESPRESSO-MV are both inefficient. This may indicate that PLA with logic elements, including PLA with input encoders, is suitable for implementing mathematical functions.

## 5. CONCLUSIONS AND FUTURE WORKS

In this paper, we present a logic synthesis method for PLA with 2-input logic elements. Our method utilizes the existing algorithms such as multiple-valued minimization and factoring along with a new logic extraction techniques. The experimental results show that the present PLA can efficiently implement Boolean functions. Since our method doesn't take account of the sharing of the product terms between output functions, further improvements can be made. In the future, we plan to develop an algorithm to take it into account.

## 6. ACKNOWLEDGEMENT

The authors would like to thank Prof. Masahiro Fujita at Univ. of Tokyo for helpful discussions.

## 7. REFERENCES

[1] S. Posluszny, N. Aoki, D. Boerstler, J. Burns, S. Dhong, U. Ghoshal, P. Hofstee, D. LaPotin, K. Lee, D. Meltzer, H. Ngo, K. Nowka, J. Silberman, O. Takahashi, and I. Vo, "Design Methodology for a 1.0 GHz Microprocessor," in *Proc. IEEE Int. Conf. Computer Design*, pp. 17–23, Oct. 1998.

[2] S. P. Khatri, R. K. Brayton, and A. Sangiovanni-Vincentelli, "Cross-talk Immune VLSI Design using a Network of PLAs Embedded in a Regular Layout Fabric," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, pp. 412–418, Nov. 2000.

[3] H. Yoshida, H. Yamaoka, M. Ikeda, and K. Asada, "Logic Synthesis for AND-XOR-OR type Sense-Amplifying PLA," in *Proc. IEEE Int. Conf. VLSI Design & Asia South Pacific Design Automation Conf.*, pp. 166–171, Jan. 2002.

[4] H. Yamaoka, M. Ikeda, and K. Asada, "A High-Speed PLA Using Array Logic Circuits with Latch Sense Amplifiers and a Charge Sharing Scheme," in *Proc. IEEE Asia South Pacific Design Automation Conf.*, pp. 3–4, Jan. 2001.

[5] T. Sasao, "Input Variable Assignment and Output Phase Optimization of PLA's," *IEEE Trans. Computer*, vol. C-28, no. 9, pp. 879–894, Oct. 1984.

[6] L. Lavagno, S. Malik, R. K. Brayton, and A. Sangiovanni-Vincentelli, "MIS-MV: Optimization of Multi-level Logic with Multiple-valued Inputs," in *Proc. IEEE/ACM Int. Conf. Computer-Aided Design*, pp. 560–563, Nov. 1990.

[7] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang, "MIS: A Multiple-Level Logic Optimization System," *IEEE Trans. Conputer-Aided Design*, vol. CAD-6, no. 6, pp. 1062–1081, Nov. 1987.

[8] Richard L. Rudell and A. Sangiovanni-Vincentelli, "Multiple-Valued Minimization for PLA Optimization," *IEEE Trans. Conputer-Aided Design*, vol. CAD-6, no. 5, pp. 727–750, Sept. 1987.