Logic Synthesis for AND-XOR-OR type Sense-Amplifying PLA

Hiroaki Yoshida Hiroaki Yamaoka Makoto Ikeda Kunihiro Asada

Department of Electronic Engineering
VLSI Design and Education Center(VDEC)
University of Tokyo
7-3-1 Hongo, Bunkyo-ku, Tokyo 113-8656, Japan
E-mail: {hiroaki,yamaoka,ikeda,asada}@silicon.u-tokyo.ac.jp

Abstract

In this paper, a new logic synthesis method for an AND-XOR-OR type sense-amplifying PLA is proposed. An AND-XOR-OR type sense-amplifying PLA can achieve low-power dissipation and high-speed operation by using latch sense-amplifiers and a charge sharing scheme. In addition, 2-input XOR 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 two-level logic minimization algorithms, it can handle large circuits such as 64-input Boolean function. The method has been implemented and the experimental results are presented. The experimental results show that some classes of Boolean functions can become much smaller and hence we can obtain significantly faster circuits than conventional PLAs with a small area penalty.

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, 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 AND-XOR-OR type sense-amplifying PLAs, is presented. An AND-XOR-OR type sense-amplifying PLA can achieve low-power dissipation and high-speed operation by using latch sense-amplifiers and a charge sharing scheme[3]. In addition, some AND/OR cells in a PLA can be replaced with 2-input XOR cells. Since our method makes full use of the existing two-level minimization algorithms, they can handle large circuits.

The rest of this paper is organized as follows. In the next section, we briefly review AND-XOR-OR type sense-amplifying PLA. Section 3 describes our logic synthesis methods for AND-XOR-OR type sense-amplifying PLA. Experimental results are presented in Section 4. Finally, in Section 5, we conclude our work and discuss about future works.

2. AND-XOR-OR type Sense-Amplifying PLA

The AND-XOR-OR type sense-amplifying PLA is effective in terms of high-speed operation and low-power dissipation, especially for large inputs. Figure 1 shows the basic cell of AND-XOR-OR type sense-amplifying PLA. As illustrated in Figure 1, some AND/OR cells can be replaced with 2-input XOR cells. Since an XOR operation is achieved by reconnecting some wires, there is almost no effect on area and delay. An AND-XOR-OR type sense-amplifying PLA can implement Boolean expressions as follows:

$$f = \sum_k p_k \tag{1}$$

$$(p_k = c_k \oplus d_k \ or \ \overline{c_k \oplus d_k} \ or \ c_k \ or \ \overline{c_k})$$

$$c_k = \prod_l q_l \quad (q_l = l_l \oplus m_l \ or \ l_l) \tag{2}$$

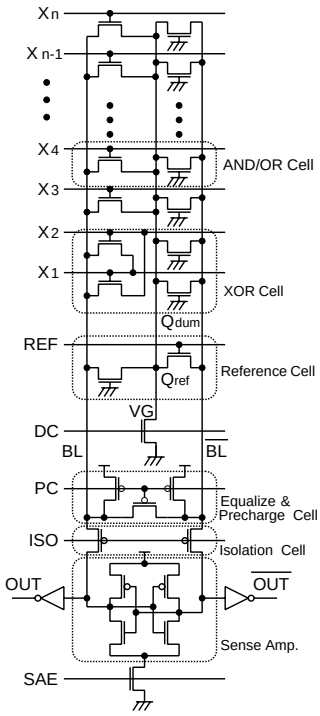

Figure 1. Basic cell of AND-XOR-OR type sense-amplifying PLA

$$d_k = \prod_m r_m \quad (r_m = l_m \oplus m_m \; or \; l_m) \tag{3}$$

where l_l and m_l are literals. Due to the PLA structure, an XOR operation is basically permitted only between two adjacent signals. To simplify the problem, we concentrate our attention on the Boolean expressions such as

$$f = \sum_k p_k \tag{4}$$

$$(p_k = c_k \oplus d_k \; or \; \overline{c_k \oplus d_k} \; or \; c_k \; or \; \overline{c_k})$$

where c_k and d_k are cubes, *i.e.*, XOR operations are performed only in OR plane. Note that XOR operations in AND plane are a kind of 2-input decoder and hence we can handle them in a similar way to the synthesis for PLA with 2-input decoders[4].

For example, consider the conventional PLA shown in Figure 2 which realizes the following Boolean expressions

$$\begin{aligned} f_1 &= x_1x_2\overline{x}_4 + \overline{x}_1x_3x_4 + x_2\overline{x}_3x_4 + \overline{x}_2x_3x_4 \\ f_2 &= x_1x_2\overline{x}_3 + \overline{x}_1x_3 + \overline{x}_2x_3 + \overline{x}_1\overline{x}_2. \end{aligned}$$

The expressions can be transformed into

$$\begin{aligned} f_1 &= (x_1x_2 \oplus x_3x_4) + \overline{x}_1x_2\overline{x}_3x_4 \\ f_2 &= (x_1x_2 \oplus x_3) + \overline{x}_1\overline{x}_2 \end{aligned}$$

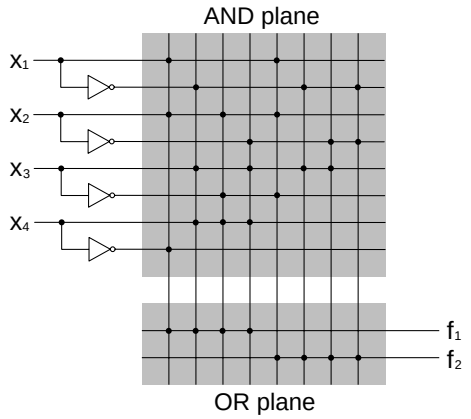

Figure 2. Example of conventional PLA

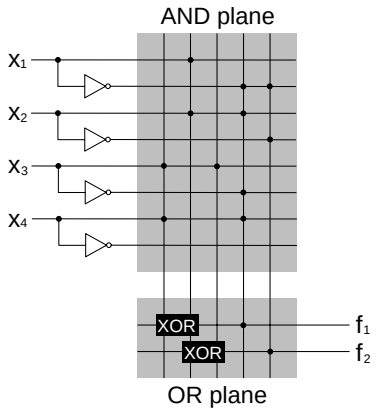

Figure 3. Example of AND-XOR-OR type sense-amplifying PLA

and Figure 3 shows the corresponding AND-XOR-OR type sense-amplifying PLA realization.

3. Logic Synthesis for AND-XOR-OR type Sense-Amplifying PLA

3.1. Definitions

The **cofactor** f_l of a Boolean function f with respect to a literal $l = x_i$ or $l = \overline{x}_i$ is the Boolean function f with the fixed value indicated by the literal. The **cofactor** f_C of a Boolean function f with respect to a cube C is the Boolean function f with the fixed value indicated by the literals of C.

In addition to the above definitions, we employ the following definition: An **XOR term** is defined as a logic expression which combines two product terms by exclusive-OR or its complement, *i.e.*, $c \oplus d$ or $\overline{c \oplus d}$ where c and d are product terms.

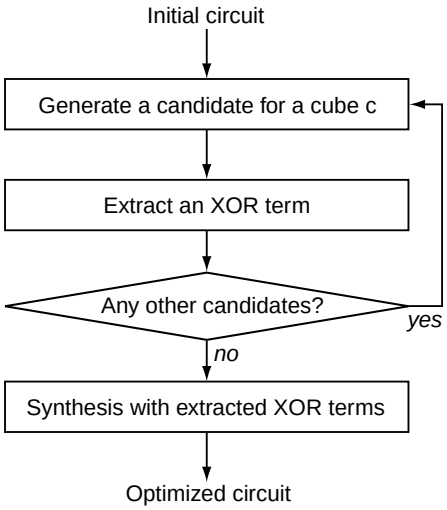

Figure 4. Proposed synthesis flow

3.2. Overall Flow

The expression given in Equation (4) can be viewed as the sum of product terms and XOR terms. The objective of our approach is to minimize the number of product terms needed, where an XOR term requires two product terms.

The basic idea of our approach is simple. It finds XOR terms contained in a given Boolean function, and then performs the two-level minimization considering them. Since there are 3^{2n} XOR terms where n is the number of inputs, it is impractical to check whether each of them is contained in a given Boolean function f. Therefore, we first generate a cube c and then calculate a cube d such that d satisfies a condition $f \supseteq c \oplus d$ or $f \supseteq \overline{c \oplus d}$.

The basic synthesis flow, shown in Figure 4, consists of three steps: 1) generation of candidates, 2) extraction of XOR terms, and 3) synthesis with extracted XOR terms.

3.3. Extraction of XOR Terms

The most important step in the synthesis flow is to find XOR terms $c \oplus d$ or $\overline{c \oplus d}$ such that a given Boolean function f contains them. We assume that one of the cubes in an XOR term, c, is given. Then we are particularly interested in the condition which a cube d satisfies.

Theorem 3.1 *Let $c = l_1 l_2 \cdots l_n$ and d be cubes and f be a Boolean function. Then,*

$$\overline{f}_c \subseteq d \subseteq f_{\overline{l}_1} f_{\overline{l}_2} \cdots f_{\overline{l}_n} \Longleftrightarrow f \supseteq c \oplus d \qquad (5)$$

where f_c and f_{l_k} are the cofactors of f with respect to a cube c and a literal l_k, respectively.

Proof *See the Appendix.* □

Corollary 3.1 *Let $c = l_1 l_2 \cdots l_n$ and d be cubes and f be a Boolean function. Then,*

$$\overline{f_{\overline{l}_1} f_{\overline{l}_2} \cdots f_{\overline{l}_n}} \subseteq d \subseteq f_c \Longleftrightarrow f \supseteq \overline{c \oplus d} \qquad (6)$$

where f_c and f_{l_k} are the cofactors of f with respect to a cube c and a literal l_k, respectively.

Once a cube c is given, we can obtain all cubes d which satisfies a condition $f \supseteq c \oplus d$ or $f \supseteq \overline{c \oplus d}$ by using the Theorem 3.1 and Corollary 3.1. For example, consider a sum-of-product expression

$$f = x_1 x_2 \overline{x}_4 + \overline{x}_1 x_3 x_4 + x_2 \overline{x}_3 x_4 + \overline{x}_2 x_3 x_4$$

and let the cube c be $c = x_1 x_2$. By calculating $\overline{f}_c$ and $f_{\overline{x}_1} f_{\overline{x}_2}$, we have

$$\begin{aligned} \overline{f}_c &= x_3 x_4 \\ f_{\overline{x}_1} f_{\overline{x}_2} &= x_3 x_4 \end{aligned}$$

and then obtain $d = x_3 x_4$ (since $\overline{f}_c \subseteq d \subseteq f_{\overline{x}_1} f_{\overline{x}_2}$). Thus we find that f contains $x_1 x_2 \oplus x_3 x_4$.

3.4. Exact Method

To find XOR terms which are contained in a given Boolean function f, candidates for cube c must be determined first. Exact method uses all possible cubes as candidates for cube c. Since all XOR terms are extracted and considered, we can obtain a Boolean expression with minimum product terms. The detailed description of the procedure is given in Figure 5.

In the procedure, we utilize the technique used in Boolean division [5] to synthesize a given Boolean function with extracted XOR terms. For example, suppose a Boolean function such as

$$f = x_1 x_2 \overline{x}_3 + \overline{x}_1 x_3 + \overline{x}_2 x_3 + \overline{x}_1 \overline{x}_2$$

and the extracted XOR terms are $\overline{x}_1 \oplus x_2 \overline{x}_3$, $x_1 \overline{x}_3 \oplus \overline{x}_2$, and $x_1 x_2 \oplus x_3$. We create new variables p_1, p_2, and p_3 to represent each XOR terms, and form the don't care set

$$\begin{aligned} g = \; & (\overline{x}_1 \oplus x_2 \overline{x}_3 \oplus p_1) + (x_1 \overline{x}_3 \oplus \overline{x}_2 \oplus p_2) \\ & + (x_1 x_2 \oplus x_3 \oplus p_3). \end{aligned}$$

The sum of all prime implicants of f with g as a don't care set is as follows:

$$f_{PRIME} = x_1 x_2 \overline{x}_3 + \overline{x}_1 x_3 + \overline{x}_2 x_3 + \overline{x}_1 \overline{x}_2 + p_1 + p_2 + p_3.$$

Finally, we solve the weighted covering problem, where the weights of p_1, p_2, and p_3 are 2 and the weights of the others are 1, and obtain the synthesized Boolean expression

$$f_{XOR} = p_3 + \overline{x}_1 \overline{x}_2 = (x_1 x_2 \oplus x_3) + \overline{x}_1 \overline{x}_2.$$

Given: a sum-of-products expression f
Procedure Exact method
  $X = \{\}$
  for all possible product term $c = x_1 x_2 \cdots x_n$
    for each product term d which satisfies a condition
      $\overline{f}_c \subseteq d \subseteq f_{\overline{x}_1} f_{\overline{x}_2} \cdots f_{\overline{x}_n}$
      $X = X \cup \{c \oplus d\}$
    end for
  end for
  $g = 0$
  for each XOR term $c_k \oplus d_k$ in X
    Create a new variable p_k to represent $c_k \oplus d_k$.
    $g = g + (c_k \oplus d_k \oplus p_k)$
  end for
  Compute all prime implicants of f with g as a *don't care* set.
  Assign weights of 2 to the prime implicants corresponding to XOR terms and weights of 1 to the others.
  Solve the weighted covering problem and obtain f_{XOR}.
  Replace p_k in f_{XOR} with $c_k \oplus d_k$.
  Output f_{XOR}.
end Procedure

Figure 5. Procedure of exact method

3.5. Heuristic Method

In the exact method, all possible cubes are examined. However, there are 3^n candidates where n is the number of inputs, and hence the method cannot handle Boolean functions with large inputs. To overcome this difficulty, we have developed a heuristic technique. To illustrate how to find the good candidates heuristically, consider the following the expressions

$$\begin{aligned} f &= (x_1x_2 \oplus x_3x_4) + \overline{x}_1x_2\overline{x}_3x_4 \\ &= x_1x_2\overline{x}_3 + x_1x_2\overline{x}_4 + \overline{x}_1x_3x_4 + \\ &\qquad \overline{x}_2x_3x_4 + \overline{x}_1x_2\overline{x}_3x_4 \\ &= x_1x_2\overline{x}_4 + \overline{x}_1x_3x_4 + x_2\overline{x}_3x_4 + \overline{x}_2x_3x_4. \end{aligned}$$

As can be easily seen, the favorable candidates x_1x_2 and x_3x_4 appear even in the last expression. From this observation, the product terms themselves and the cubes given by removing a literal from each product term in a given Boolean expression f can be used as candidates for a cube c. The number of the candidates is reduced to the sum of the number of literals and the number of cubes in a given Boolean expression. The procedure based on this technique, heuristic method, is described in Figure 6.

In the procedure, the cube d and the final expression f_{XOR} are calculated by using two-level minimization algorithms. Since the method makes full use of two-level minimization algorithms, we can solve the problems efficiently by using powerful minimization methods like ESPRESSO[6].

Given: a sum-of-products expression f
Procedure Heuristic method
  $C = \{\}$
  for each product term c in f
    $C = C \cup \{c\}$
    for each literal l in c
      $C = C \cup \{c - l\}$
    end for
  end for
  $X = \{\}$
  for each $c = l_1 l_2 \cdots l_n \in C$
    if c satifies $\overline{f}_c \subseteq f_{l_1} f_{l_2} \cdots f_{l_n}$ **then**
      Simplify $\overline{f}_c$ with $f_{l_1} f_{l_2} \cdots f_{l_n}$ as a *don't care* set and obtain d.
      if d is a valid cube **then**
        $X = X \cup \{c \oplus d\}$
      end if
    end if
  end for
  $g = 0$
  for each XOR term $c_k \oplus d_k$ in X
    Create a new variable p_k to represent $c_k \oplus d_k$.
    $g = g + (c_k \oplus d_k \oplus p_k)$
  end for
  Simplify f with g as a *don't care* set and obtain f_{XOR}.
  Replace p_k in f_{XOR} with $c_k \oplus d_k$.
  Output f_{XOR}.
end Procedure

Figure 6. Procedure of heuristic method

Note that two-level logic minimizers may count p as a single product term and hence the XOR terms tend to appear in the final Boolean expression. In this sense, the cost of the output of the two-level minimizer is under-estimated. This can be avoided by modifying the minimizer to count p as two product terms.

3.6. Extension to Multi-output Functions

In the conventional PLAs, the product terms generated in AND plane can be shared in some outputs. Therefore we can reduce the number of product terms by considering all outputs at the same time. Since an AND-XOR-OR type sense-amplifying PLA has its own restriction, product terms cannot be shared when two or more XOR operations need the same product term. However, if one allows an irregular interconnect as shown in Figure 7, the sharing can be performed. The method presented in this paper can be naturally extended to such a circuit. First we extract XOR terms for all outputs, then perform two-level minimization considering them.

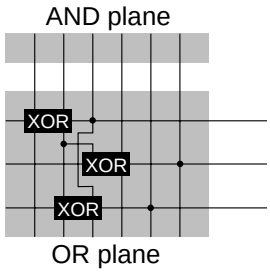

Figure 7. Product term sharing

4. Experimental Results

The method described in the paper has been implemented and interfaced with the ESPRESSO. First we applied it to all 5-input Boolean functions. Since we are not interested in the functions which are the negation of some variables and the permutation of some variables in a Boolean function, the Boolean functions in the NP-equivalence class are examined. Black bars in Figure 8 and 9 show the number of the functions which had smaller product terms than the circuits produced by ESPRESSO. Note that the results of our method cannot be worse than that of ESPRESSO because our method uses ESPRESSO internally. In the figures, the horizontal axis denotes the number of minterms, and the vertical axis denotes the number of functions. The results show that the quality of the heuristic method is comparable to that of the exact method. The CPU time needed to perform the computation for the heuristic method is 12 seconds while the exact method needs more than 20 hours.

Next, we applied the heuristic method to example functions with large inputs. Table 1 compares the results of the heuristic method and ESPRESSO. In the table, *#XORs* is the number of the XOR terms in the resulting circuit and *CPUtime* is the CPU time in seconds. As can be seen from the results, the heuristic method can reduce the number of product terms significantly when the good XOR terms are found. Though the CPU time is larger than ESPRESSO, it is still reasonable, because the size of the circuit example4 is an average size in practical applications.

Finally, we designed the several layouts of the the circuit example4 in Table 1, using ROHM's three metal layer 0.35μm CMOS process technology. For comparison, the standard-cell based design was synthesized using Synopsys Design Compliler, and placed and routed by Avant! Apollo [1]. The results are shown in Table 2. The delays for all implementations were estimated by using Avant! HSPICE. The results show that the AND-XOR-OR type sense-amplifying PLA realization can achieve two times faster operation than the conventional PLA with an area increase of about 6%. The standard-cell based design is much smaller in area however the delay time is almost two times slower than the present PLA.

[1]The VLSI layouts in this study are designed with Avant! CAD tools.

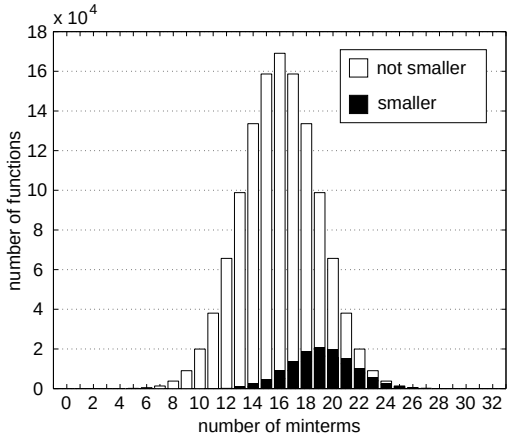

Figure 8. Exact method

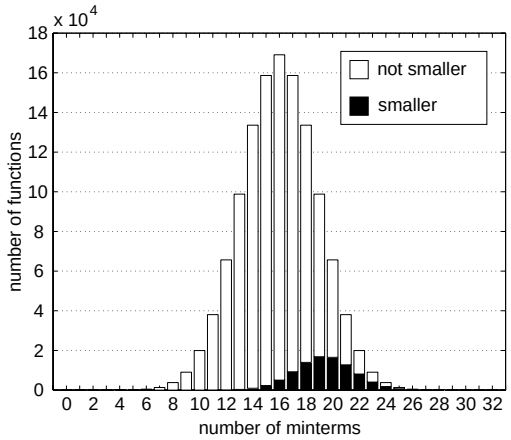

Figure 9. Heuristic method

5. Conclusions and Future Works

In this paper, a logic synthesis method for AND-XOR-OR type sense-amplifying PLA is presented. Especially, the heuristic method makes full use of the existing two-level minimization algorithms, it can handle large circuits. Experimental results show that some classes of Boolean functions can become much smaller and hence the resulting circuits can operate much faster than the conventional PLA with a small area penalty.

It is well known that PLAs with two-input decoders requires a smaller area than conventional PLAs. Further improvements can be made by combining an AND-XOR-OR type sense-amplifying PLA with two-input decoders. An extension to AND-XOR-OR type sense-amplifying PLA with two-input decoders is currently being investigated.

Table 1. Comparisons between ESPRESSO and the proposed method.

circuit	*#inputs*	*ESPRESSO*		*Proposed(Heuristic method)*		
		#product terms	*CPU time*	*#product terms*	*#XORs*	*CPU time*
example1	8	11	0.0	6	1	0.1
example2	16	87	0.1	39	5	7.4
example3	32	144	0.8	99	3	14.0
example4	64	220	16.5	136	3	254.9

Table 2. Comparisons between several implementation styles of the circuit example4

type	*#transistors*	*Physical dimension*	*Delay time*
Standard cell	1154	$190\mu m \times 190\mu m$	1255ps
Conventional PLA	3940	$997\mu m \times 554\mu m$	1505ps
Dual-rail PLA	8226	$1865\mu m \times 505\mu m$	720ps
AND-XOR-OR type sense-amplifying PLA	5052	$1159\mu m \times 505\mu m$	698ps

Appendix: Proof of Theorem 3.1

Theorem 3.1 *Let $c = l_1 l_2 \cdots l_n$ and d be cubes and f be a Boolean functions. Then,*

$$\overline{f}_c \subseteq d \subseteq f_{\overline{l}_1} f_{\overline{l}_2} \cdots f_{\overline{l}_n} \Longleftrightarrow f \supseteq c \oplus d$$

where f_c and f_{l_k} are the cofactors of f with respect to a cube c and a literal l_k, respectively.

Proof *First, we prove the following lemmas.*
Lemma A.1 *Let c and d be cubes and f be a Boolean function. Then,*

$$f_c \supseteq \overline{d} \Longleftrightarrow f \supseteq c\overline{d}.$$

Proof $\Longrightarrow$*: By the property of cofactor,*

$$f \supseteq c f_c \supseteq c\overline{d}.$$

$\Longleftarrow$*: By calculating the cofactors of the both side of $f \supseteq c\overline{d}$ with respect to c, we have*

$$f_c \supseteq \overline{d}. \qquad \square$$

Lemma A.2 *Let $c = l_1 l_2 \cdots l_n$ and d be cubes and f be a Boolean function. Then,*

$$f_{\overline{l}_1} f_{\overline{l}_2} \cdots f_{\overline{l}_n} \supseteq d \Longleftrightarrow f \supseteq \overline{c}d.$$

Proof $\Longrightarrow$*: By the property of cofactor, for any k,*

$$f \supseteq \overline{l}_k f_{\overline{l}_k} \supseteq \overline{l}_k f_{\overline{l}_1} f_{\overline{l}_2} \cdots f_{\overline{l}_n}.$$

Thus, we have

$$\begin{aligned} f &\supseteq (\overline{l}_1 + \overline{l}_2 + \cdots + \overline{l}_n) f_{\overline{l}_1} f_{\overline{l}_2} \cdots f_{\overline{l}_n} \\ &= \overline{c} f_{\overline{l}_1} f_{\overline{l}_2} \cdots f_{\overline{l}_n} \supseteq \overline{c}d. \end{aligned}$$

$\Longleftarrow$*: Since $c = l_1 l_2 \cdots l_n$, we have*

$$f \supseteq \overline{c}d = \overline{l}_1 d + \overline{l}_2 d + \cdots + \overline{l}_n d.$$

By calculating the cofactor of the both sides, for any k,

$$f_{\overline{l}_k} \supseteq d.$$

Thus, we have

$$f_{\overline{l}_1} f_{\overline{l}_2} \cdots f_{\overline{l}_n} \supseteq d. \qquad \square$$

Since $c \oplus d = c\overline{d} + \overline{c}d$, the theorem can be proven to be true straightforwardly by Lemma A.1 and Lemma A.2. $\square$

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*, pages 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*, pages 412–418, Nov. 2000.

[3] 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.*, pages 3–4, Jan. 2001.

[4] T. Sasao. Input Variable Assignment and Output Phase Optimization of PLA's. *IEEE Trans. Computer*, C-28(9):879–894, Oct. 1984.

[5] R. K. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang. MIS: A Multiple-Level Logic Optimization System. *IEEE Trans. Conputer-Aided Design*, CAD-6(6):1062–1081, Nov. 1987.

[6] R. K. Brayton, G. D. Hachtel, C. T. McMullen, and A. L. M. Sangiovanni-Vincentelli. *Logic Minimization Algorithms for VLSI Synthesis*. Kluwer Academic Puclishers, Boston, 1984.