排他的論理和を実現可能な2線式PLAのための論理合成手法*

吉田 浩章　　山岡 寛明　　池田 誠　　浅田 邦博†

東京大学 大学院 工学系研究科 電子工学専攻‡
東京大学 大規模集積システム設計教育研究センター (VDEC) §
〒 113-8656 東京都文京区本郷 7-3-1
E-mail: hiroaki@silicon.u-tokyo.ac.jp

概要　排他的論理和を実現可能な 2 線式 PLA はラッチ型センスアンプおよび寄生容量間の電荷分配を利用することによって低消費電力・高速動作を実現可能にしている。本稿では、この排他的論理和を実現可能な 2 線式 PLA のための論理合成手法である全数探索手法およびヒューリスティック手法を提案する。また、2 つの手法の比較および実際のレイアウトにおけるシミュレーションを行い、本手法の有効性について示す。

Abstract　An XOR-Based Dual-Rail PLA can achieve low-power dissipation and high-speed operation by using latch sense-amplifiers and a charge sharing scheme. In this paper, we propose the two logic synthesis methods for XOR-based dual-rail PLA: *exhaustive method and heuristic method*. We also present experimental results.

1　はじめに

近年、半導体プロセスの微細化により VLSI の大規模化が進むに従って設計は複雑さを増しており、短期間に信頼性のある回路を設計することは非常に難しくなってきている。Programmable Logic Array(PLA) はその規則的な構造のため設計や面積・性能の見積もりが容易であることや、AND-OR の二段論理で構成されるため高速に動作する、などといった理由から現在でもしばしば利用される。しかしこのような利点がある一方で、ある論理関数に対しては多段論理で構成された設計に比べて回路規模が爆発的に増大してしまうという欠点がある。

排他的論理和を実現可能な 2 線式 PLA はセンスアンプを用いた回路方式を採用することによって高速な動作を実現しており、また従来の PLA における AND セルや OR セルの代わりに 2 入力排他的論理和セルを挿入することが可能になっている [1] [2]。そのため積和形の論理式だけでなく排他的論理和を含んだ論理式をそのまま実現することが可能となっている。

本稿ではこの排他的論理和を実現可能な 2 線式 PLA のための 2 つの論理合成手法である全数探索手法およびヒューリスティック手法を提案する。今回提案する手法では既存の 2 段論理最小化アルゴリズムを最大限利用するようになっており、大規模な回路を扱うことが可能になっている。まず 2 章で排他的論理和を実現可能な 2 線式 PLA について簡単な説明を行い、3 章では排他的論理和を実現可能な 2 線式 PLA のための論理合成手法を提案する。4 章で提案手法を用いた実験結果を示した後、最後に 5 章で本稿をまとめる。

2　排他的論理和を実現可能な2線式PLA

従来の PLA は回路規模が小さい場合には高速に動作するが、回路規模が大きい場合には寄生容量が増大し遅延時間や消費電力が大きくなってしまう。排他的論理和を実現可能な 2 線式 PLA はラッチ型センスアンプおよび寄生容量間の電荷分配を利用することによって大きな回路規模において高速動作および低消費電力を実現している。図 1 にその基本セルを示す。

この PLA では図 1 に示すように AND セルや OR セルと同様に排他的論理和セルを用いることが可能

---

* Logic Synthesis for XOR-Based Dual-Rail PLA
† Hiroaki Yoshida, Hiroaki Yamaoka, Makoto Ikeda and Kunihiro Asada
‡ Dept. of Electronic Engineering, Univ. of Tokyo
§ VLSI Design and Education Center, Univ. of Tokyo

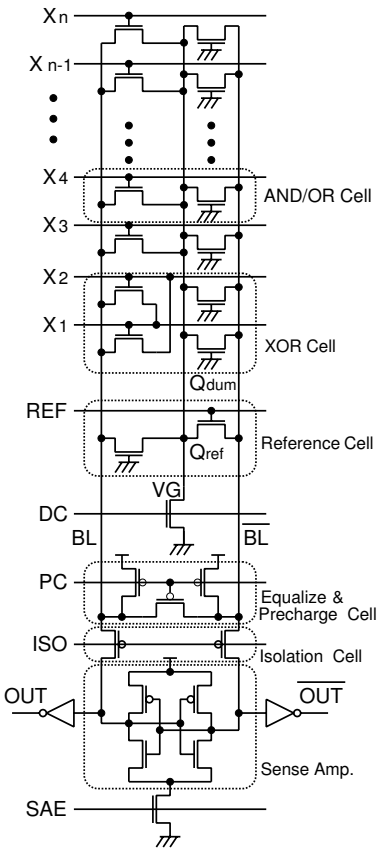

図 1. 排他的論理和を実現可能な 2 線式 PLA の基本セル

になっているため、

$$f = \sum_k p_k \tag{1}$$
$$(p_k = c_k \oplus d_k, \overline{c_k \oplus d_k} \text{または } c_k)$$
$$c_k, d_k = \prod_l q_l \quad (q_l = l_l \oplus m_l \text{または } l_l) \tag{2}$$

で与えられる論理式を実現可能である。ここで l_l および m_l はリテラル、つまりブール変数もしくはその否定である。

3 論理合成手法

3.1 用語の定義

ここでは本稿で用いられる用語の定義を行う。

ある変数 x に対して、$x, \overline{x}$ を**リテラル**と呼ぶ。また論理関数 f に対して、ある入力変数 x を 0 または 1 に固定した関数 $f_x, f_{\overline{x}}$ を f の x に関する**コファクタ (cofactor)** と呼ぶ。同様に、論理関数 f のリテラル $l_1, l_2, \ldots, l_n$ からなる積項 c に関するコファクタ f_c は、リテラル l_k が変数の肯定ならばその変数を 1 に、否定ならばその変数を 0 に固定したもので与えられる。

3.2 手法の概略

2.2 節で説明したように排他的論理和を実現可能な 2 線式 PLA は AND 平面および OR 平面の両方で排他的論理和セルを用いることで (1) 式および (2) 式で与えられる論理式を実現可能であるが、回路の構造上 AND 平面で自由に排他的論理和を行うことは難しい。この理由から本研究では OR 平面においてのみ排他的論理和を行うものとし、与えられた積和形の論理式を次式の形に合成する問題について考える。

$$f = \sum_k p_k \tag{3}$$
$$(p_k = c_k \oplus d_k, \overline{c_k \oplus d_k} \text{または } c_k)$$

ここで c_k および d_k は積項である。例えば次のような積和形の論理式

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

を (3) 式の形に合成すると、次のように変形することが可能である。

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

前者の論理式を PLA で実現する場合には積項が 4 個必要であるのに対して、後者では積項は 3 個となる。今回の手法では図 2 に示すように、まず与えられた積和形の論理式に含まれる排他的論理和項の抽出を行い、次にそれらを用いて (3) 式の形に合成を行う。

3.3 排他的論理和項の抽出

与えられた積和形の論理式 f が包含する $c \oplus d$ の形の論理式を求める場合、すべての可能な組み合わせは変数の数を n とすると全部で 3^{2n} 個あり、それらをすべて調べることは入力が多い場合には不可能に近い。そこで今回提案する手法では排他的論理和項中の積項 c が与えられているものとし、もう片方の積項 d を求めることによって排他的論理和項の抽出を行う。次に 2 つの補題を示す。これらは定理 3.1 を証明するために必要である。

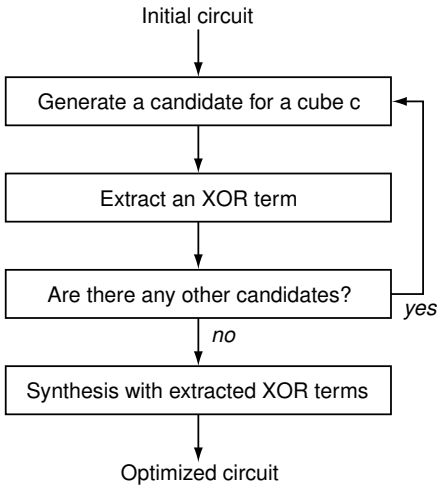

図 2. 提案手法の流れ

補題 3.1 c および d を積項、f を論理関数とすると、

$$f_c \supseteq \overline{d} \Longleftrightarrow f \supseteq c\overline{d} \tag{4}$$

が成立する。

証明 $\Longrightarrow$: コファクタの定義より

$$f \supseteq cf_c \supseteq c\overline{d}$$

$\Longleftarrow$: $f \supseteq c\overline{d}$ の両辺のコファクタをとることにより

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

補題 3.2 $c = l_1 l_2 \cdots l_n$ および d を積項、f を論理関数とすると、

$$f_{\overline{l}_1} f_{\overline{l}_2} \cdots f_{\overline{l}_n} \supseteq d \Longleftrightarrow f \supseteq \overline{c}d \tag{5}$$

が成立する。

証明 $\Longrightarrow$: コファクタの定義より任意の k に対して

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

が成立する。これらの和をとることにより

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

$\Longleftarrow$: *(6)* 式に $c = l_1 l_2 \cdots l_n$ を代入すると

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

が得られる。両辺のコファクタをとると任意の k に対して

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

が成立する。これらの積をとることにより

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

これらの補題より次の定理が導かれる。

定理 3.1 $c = l_1 l_2 \cdots l_n$ および d を積項、f を論理関数とすると、

$$\overline{f}_c \subseteq d \subseteq f_{\overline{l}_1} f_{\overline{l}_2} \cdots f_{\overline{l}_n} \Longleftrightarrow f \supseteq c \oplus d \tag{6}$$

が成立する。ここで f_c および f_{l_k} はそれぞれ積項 c およびリテラル l_k に関するコファクタを表す。

証明 補題 *3.1* および *3.2* より明らかである。 $\square$

系 3.1 $c = l_1 l_2 \cdots l_n$ および d を積項、f を論理式とすると、

$$\overline{f_{\overline{l}_1} f_{\overline{l}_2} \cdots f_{\overline{l}_n}} \subseteq d \subseteq f_c \Longleftrightarrow f \supseteq \overline{c \oplus d} \tag{7}$$

が成立する。ここで f_c および f_{l_k} はそれぞれ積項 c およびリテラル l_k に関するコファクタを表す。

これらの定理を用いると、与えられた積項 c から $f \supseteq c \oplus d$ または $f \supseteq \overline{c \oplus d}$ を満たす積項 d を求めることができる。例えば、次のような積和形の論理式

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

を考える。このとき $c = x_1 x_2$ とすると、$\overline{f}_c = x_3 x_4$ および $f_{\overline{x}_1} f_{\overline{x}_2} = x_3 x_4$ となり $d = x_3 x_4$ が求まる。よって $x_1 x_2 \oplus x_3 x_4$ が f に包含されることがわかる。

3.4 全数探索手法

3.3 節の手法を利用して合成を行うためには、どのようにして積項 c の候補を決定するかが問題になる。ここで提案する全数探索手法ではすべての可能な積項をその候補としたものである。また積項 d についてもすべての可能なものを考慮している。この場合の合成の手続きの詳細を図 3 に示す。

この手続きでは抽出した排他的論理和項を考慮して最終的に合成を行う際に、Boolean division で用いられる手法 [3] を利用している。例えば、次のような積和形の論理式

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

で、排他的論理和項 $x_1 x_2 \oplus x_3$ が求まっているとする。この排他的論理和項を表す新しい変数を p とす

Given: a sum-of-products expression f
Procedure Exhaustive 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
  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

図 3. 全数探索手法の手続き

ると、$x_1x_2 \oplus x_3$ と p は常に同じ値にならなければならないので、ドントケア g は

$$g = (x_1x_2 \oplus x_3) \oplus p$$

となる。最後にこのドントケアを用いて f の簡単化を行うと、

$$f_{XOR} = p + \overline{x}_1\overline{x}_2 = x_1x_2 \oplus x_3 + \overline{x}_1\overline{x}_2$$

となり、最終的な論理式が得られる。2 段論理最小化アルゴリズムは排他的論理和項 p を 1 項と数えてしまうため、この手法では必要以上に排他的論理和項が最終的な論理式に採用される可能性があり、注意しなくてはならない。

3.5 ヒューリスティック手法

3.4 節で提案した全数探索手法ではすべての可能な積項を候補としており、その候補の数は入力数を n とすると 3^n となり大きな入力数の場合には実用的ではない。

例えば次のような (3) 式の形の論理式

$$f = x_1x_2 \oplus x_3x_4 + \overline{x}_1x_2\overline{x}_3x_4$$

を積和形に展開し最小化を行うと次のようになる。

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

最後の式を見てわかるように積項 c の候補となるべき x_1x_2 もしくは x_3x_4 が式中に残っている。このように大きな入力数の場合には与えられた論理式中の積項および積項からリテラルを 1 つ除いたものを候補とすることで、その候補の数を論理式のリテラル数と積項数の和にまですることが可能になる。このヒューリスティック手法を図 4 に示す。

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 $\overline{f}_c \subseteq f_{\overline{l}_1} f_{\overline{l}_2} \cdots f_{\overline{l}_n}$ **then**
      Simplify $\overline{f}_c$ with $f_{\overline{l}_1} f_{\overline{l}_2} \cdots f_{\overline{l}_n}$ as a *don't care* set and obtain d.
      if d is a valid cube **then**
        $X = X \cup \{c \oplus d\}$
      endif
    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

図 4. ヒューリスティック手法の手続き

図 4 では (6) 式の包含関係を満たす積項 d を求める際に 2 段論理最小化アルゴリズムを用いている。この手法ではほとんどのステップにおいて 2 段論理最小化アルゴリズムを利用しており、そのため既存の高性能な 2 段論理最小化アルゴリズムを用いることで高速に計算を行うことが可能である。

3.6 多出力関数への拡張

PLA では AND 平面で生成された積項を OR 平面で共有することが可能なので、これを考慮して合成を行うことによって全体として積項の数を減らすことができる。排他的論理和を実現可能な 2 線式 PLA では OR 平面で排他的論理和をとる際には基本的に

その 2 つの信号は隣り合ったものである必要があるため、積項の共有は不可能である。しかし、図 5 に示すように配線を工夫することで積項の共有が可能になる。これまで提案した手法を多出力関数において積項の共有を考慮して行うためには、まずすべての関数に対して排他的論理和項を求めた後、それらを考慮して合成を行うことによって可能になる。

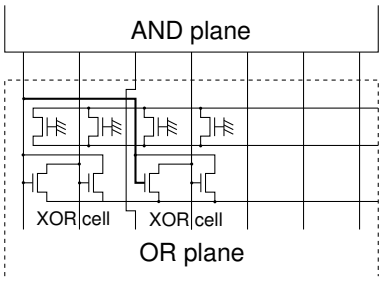

図 5. 排他的論理和を実現可能な 2 線式 PLA における積項の共有

4 実験結果

まず 5 入力のすべての論理関数に対して ESPRESSO[4] および両手法を適用した。5 入力の論理関数は全部で 2^{2^5} 個あるが、ここではいくつかの変数を入れ替えたものおよびいくつかの変数の否定を取ったものについては区別する必要がないため、それらを同値とした NP 同値類を対象にしている。

図 6, 7 に示すグラフは提案手法によって合成された回路の積項数が ESPRESSO よりも少ないものを数えたものである。またグラフにおいて横軸はその関数の最小項の数、つまり真理値表における 1 の数に対応している。積項が減ったものの中で全数探索手法・ヒューリスティック手法共に平均で 81%、最大で 92% の積項数の削減が実現された。

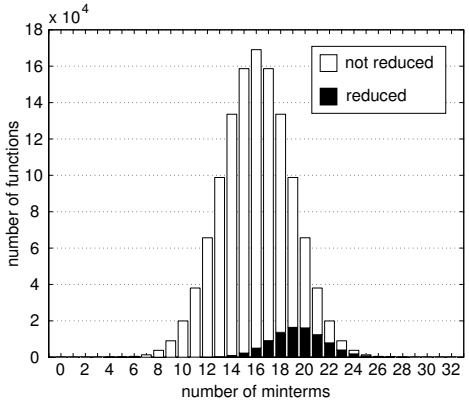

図 6. 実験結果 (全数探索手法)

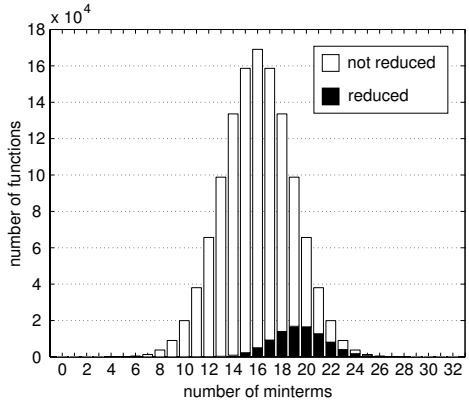

図 7. 実験結果 (ヒューリスティック手法)

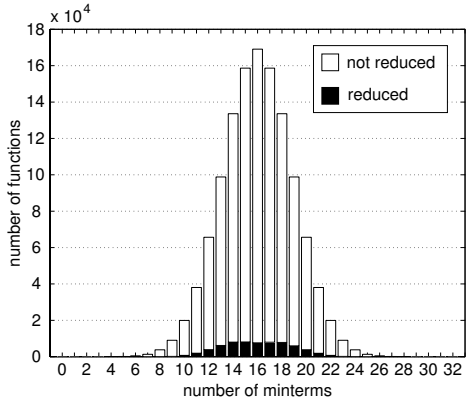

図 8. 実験結果 (全数探索手法および出力極性最適化)

排他的論理和を実現可能な 2 線式 PLA は否定の出力を持っているため、出力極性最適化を行うことによりさらなる積項の削減が可能である。図 8 は全数探索手法および ESPRESSO の両手法において出力極性最適化を考慮した結果である。

次により大規模な回路に対してヒューリスティック手法および ESPRESSO を適用し、その比較を行った。例題に対する結果を表 1 に示す。表中において *#product terms* は最終的な積項の数、*#XORs* は最終的に PLA 中で使用される排他的論理和セルの数に対応する。

最後に表 1 中の回路 example4 のレイアウトを作成し、従来の PLA および CMOS スタンダードセル方式との比較を行った。プロセスは ROHM 社の $0.35\mu m$ の CMOS 、ポリ 2 層、メタル 3 層を使用した。また CMOS スタンダードセル方式は Synopsys 社の Design Compiler で論理合成を、Avanti 社の Apollo で配置配線を用いて作成した。回路シミュレーション結果を表 2 に示す。

表 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

表 2. Comparisons between several implementations of a circuit example4. (Process technology: $0.35\mu m$ CMOS, 2-Poly, 3-Metal)

type	*#transistors*	*Physical dimension*	*Delay time*
Standard cell	811	$152\mu m \times 133\mu m$	3280ps
Conventional PLA	3940	$997\mu m \times 554\mu m$	1505ps
Dual-rail PLA	8226	$1865\mu m \times 505\mu m$	720ps
XOR-based dual-rail PLA	5052	$1159\mu m \times 505\mu m$	698ps

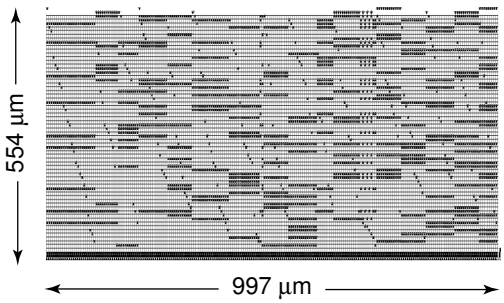

図 9. レイアウト (従来の PLA)

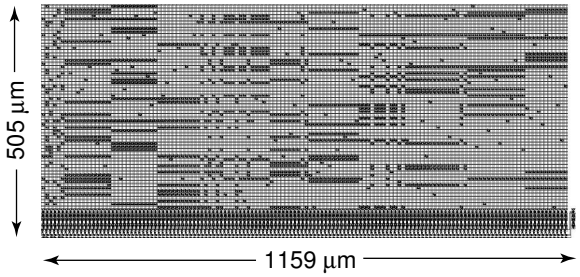

図 10. レイアウト (排他的論理和を実現可能な 2 線式 PLA)

5 まとめ

本稿ではこの排他的論理和を実現可能な 2 線式 PLA のための論理合成手法である全数探索手法およびヒューリスティック手法を提案した。今回提案した手法は一部の論理関数に対して大幅に積項数を削減することが可能であり、場合によっては従来の PLA と同等の面積で 2 倍以上の高速な演算を行う回路を出力することがあることを示した。またヒューリスティック手法を利用することによって比較的大規模な回路を扱うことが可能であることも示した。

謝辞

本研究を進めるにあたり、多くの御助言を頂いた東京大学 工学系研究科 藤田昌宏教授に深く感謝致します。

参考文献

[1] H. Yamaoka, M. Ikeda and K. Asada, “A High-Speed PLA Using Array Logic Circuits with Latch Sense Amplifiers and a Charge Sharing Scheme,” *Asia South Pacific Design Automation Conference(ASP-DAC)*, Jan. 2001.

[2] 山岡 寛明, “センスアンプを用いた配列型 CMOS 論理回路の研究,” 修士論文, 東京大学大学院電子工学専攻, 2001 年.

[3] 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, Nov. 1987.

[4] 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.