設計自動化 55－10
(1990.12.13)

許容関数集合に基づく組合せ論理回路の遅延最適化

藤本徹哉、神戸尚志
シャープ株式会社

あらまし　近年、論理合成技術の実用化にともない、遅延時間に関する仕様を満足した回路を合成できる論理合成システムが要請されている。本論文では、遅延時間を大域的に最適化することを目的として、許容関数集合の概念に基づいて論理回路の段数を最小化する二つのブール的手法を提案する。これは、面積コストの最適化に関して強力なアルゴリズムとされてきたトランスダクション法の一拡張である。本手法を用いて論理回路を大域的に変換することにより、論理回路の段数を大きく削減できること、および面積最適化とテクノロジマッピングを適用した後の遅延時間も、さらに改善されることが実験により示された。

Delay Time Optimization for Combinational Circuits based on the Permissible Functions

T. Fujimoto and T. Kambe

SHARP Corporation
632, Nara, JAPAN

Abstract　Logic synthesis technology is nowadays getting practical in designing commercial chips, consequently logic synthesis system is required to be capable for satisfying specifications in terms of circuit delay. In this paper, in order of global timing optimization, we present two boolean algorithms for logic-depth minimization based on the concept of the Set of Permissible Functions. Our approarch is an extention from the Transduction Method which is a fairly effective algorithm for area minimization. By the use of the presented method, it is shown that logic depth of given circuits is reduced effectively and also circuit delay even after area minimization and technology mapping.

1 はじめに

多段論理回路の合成技術として、Weak Division [3] などによる多段化とブール代数的な最適化手法を併用したものがいくつか知られている [2, 4, 10]。また、今日では、それが論理合成システムとしての一般的な形態になりつつある。この分野の研究の中心は、最近の傾向では (1) テスタビリティの高い回路の合成、および本論文の主題である (2) 遅延の小さい回路の合成、の主として二方向に移りつつある。また、遅延に関しても従来多段化およびテクノロジマッピングの工程における局所的な最適化が主流であったが、最近になって大域的な最適化手法の中で遅延が考慮されはじめた [5]。

論理最適化手法として知られているいくつかの手法の中で、許容関数集合の概念に基づくトランスダクション法 [8, 10] は、1970年代前半という早い時期に開発されたものであるが、ブール代数的な手法の中でも極めて強力なものであることが知られている [6, 12]。また、許容関数集合に基づいて面積コストと同時に遅延時間を最適化する手法も開発され、よい結果が得られている [7, 13]。これらの手法が、面積コストの最小化と同時に遅延の最適化を行なうもので、かつ前者を優先して行なっているのに対し、本論文では面積コストには特に注意を払わず、遅延時間のみの最小化を目指した手法について述べる。

このような試みは、現実の論理設計では遅延時間に関する仕様が面積のそれとは性質を異にすること、すなわち、一定以上小さくする必要はないがある絶対的な基準値を越えることは許されない、ということから十分意味があると考えられる。実際、例えば50MHz動作のチップの設計では、合成された回路の遅延時間がクロック周期20nsに対して十分であってはじめて面積コストが意味を持つ。

また、トランスダクション法は許容関数の範囲内で論理回路の構造を変換することによって最適化を進めるものであり、そのため、初期回路に存在する冗長が多い方が最適化の可能性が広いと考えられる。従って、面積最適化によって冗長の多くが取り除かれる前に、遅延の最適化のみを行なえば、より遅延の小さい回路を合成できることが期待できる。

本論文の手法は、テクノロジマッピングに先だつ最適化工程に用いるもので、与えられた組合せ論理回路に対し遅延時間の代わりに段数（論理深度）を必要なだけ小さくすることを目標とする。取り扱う回路は、基本論理ゲートからなる組合せ回路である。3節で、回路中の最長パスを切断することにより段数を削減する手法を二つ示し、4節で、それらによる実験結果および知られている結果との比較を示す。

なお、本論文で述べる手法を用いた論理合成システムでは、段数削減後に、段数を悪化させない面積最適化とテクノロジマッピングを適用することを想定している。従って、論理段数とテクノロジマッピング後の実遅延の間の相関性を強めるため、本手法および面積最適化ではファンインの制約を守った最適化を行なっている。

2 用語と定義

n 個のプライマリ入力を持つ論理回路を考える。各々の論理ゲートおよびプライマリ入力を記号 g で表し、プライマリ入力の集合 PI、プライマリ出力の集合を PO とする。g_i の出力から g_j の入力への接続をコネクション e_{ij} と表す。g_i から g_j へのコネクションがあるとき、g_i を g_j のファンイン、g_j を g_i のファンアウトと呼ぶ。g_j のファンイン、ファンアウトの集合をそれぞれ $FI(g_j)$、$FO(g_j)$ と表す。

ゲート g_j をプライマリ入力側からみたときの段数を L_j^{in}、プライマリ出力側からみたときのそれを L_j^{out} と表す。これは次の式で帰納的に与えられる。

$$L_j^{in} = \begin{cases} 0 & \text{if } g_j \in PI \\ \max\limits_{g_i \in FI(g_j)} L_i^{in} + 1 & \text{otherwise} \end{cases}$$

$$L_j^{out} = \begin{cases} 1 & \text{if } g_j \in PO \\ \max\limits_{g_k \in FO(g_j)} L_k^{out} + 1 & \text{otherwise} \end{cases}$$

従って、回路全体の段数 L^{max} は $L^{max} = max(L_j^{in} + L_j^{out})$ となり、また、最長パスとは $L_j^{in} + L_j^{out} = L^{max}$ となるゲート g_j をたどるものである。

各 g_j に対し、n 次元ブール空間 $\boldsymbol{B}^n$ の上、換言すればプライマリ入力に関して、論理関数

$$f_j(x) : \boldsymbol{B}^n \longmapsto \boldsymbol{B}$$

が実現されている。但し、単に f_j と表記した場合には、$f_j(x) = 1$ の $\boldsymbol{B}^n$ への像、すなわち $f_j^{-1}(1) \subset \boldsymbol{B}^n$ を意味する。

各ゲート g_j およびコネクション e_{ij} の上に、許容関数集合 G_j 或は G_{ij} を、以下のようにして定義する。

許容関数集合 G を、$\boldsymbol{B}^n$ を完全に分割する三つの論理関数の組 $(G^{on}(x), G^{DC}(x), G^{off}(x))$ で表す。すなわち

$$\begin{cases} G^{on} \cap G^{DC} = G^{DC} \cap G^{off} = G^{off} \cap G^{on} = \phi \\ G^{on} \cup G^{DC} \cup G^{off} = \boldsymbol{B}^n \end{cases}$$

である。ここで、G^{DC} は内部ドントケア [1] の表現の一種である。ある g_j の上で、$f_j(x)$ を別の関数 ${}^*f_j(x)$ に変えてもプライマリ出力に影響がないとき、${}^*f_j(x)$ を許容関数という。一般的に許容関数は一つ ($f_j(x)$ それ自身のみ) とは限らない。そして

$$\begin{cases} {}^*f_j \supseteq G_j^{on} \\ \overline{{}^*f_j} \supseteq G_j^{off} \end{cases}$$

を満たす関数が全て許容関数になっているとき、G_j を g_j の許容関数集合という。$g_j \in PO$ に関しては、外

部から与えられたドントケア集合を $X_j \subset \boldsymbol{B}^n$ とすると $G_j^{on} = f_j - X_j$, $G_j^{off} = \overline{f}_j - X_j$, $G_j^{DC} = X_j$ である。コネクション上の許容関数集合も同様にして定義される。許容関数集合の計算方法については説明を省略するが、文献 [10] に詳しい。

許容関数集合のとりかたには二つあり、最大許容関数集合 (MSPF)、互換許容関数集合 (CSPF) とそれぞれ呼ばれる。前者には最大かつ一意であるという特徴が、後者には許容関数の範囲内での論理回路の変換に対して不変であるという特徴がそれぞれある [10]。許容関数集合の計算コストは大きく、かつ、論理の変更回数は極めて多いので、本論文では CSPF を採用している。

3 最長パス解消アルゴリズム

最長パスの解消を行なう二つの手法を示す。いずれも、最長パス上のコネクションを切断し、許容関数集合を用いてプライマリ出力を不変に保つよう新たなコネクションの追加を行なうものである。これらを用いて、段数削減アルゴリズムの全体は以下のようなステップからなる。

1. 回路全体の段数を計算する。目標段数まで削減が進んでいれば終了（成功）。さらに削減を要する時、最長パス上の全ゲートについて許容関数の計算と最長パス解消の順序づけを行なってステップ 2 へ。
2. 最長パスの解消を試みるゲートを一つ選択する。ゲートが残っていないときに終了する（解消に失敗した）。
3. 本節で述べる二つのパス切断アルゴリズムのどちらかを適用する。切断に成功した場合ステップ 1 へ、失敗した場合ステップ 2 へそれぞれ戻る。

なお、本節では簡単化のため全て NOR ゲートを用いて説明しているが、他の基本論理ゲートへの拡張は容易である。

3.1 コネクションの置き換えと再合成

これはコネクションの置き換え (R) と再合成 (S) の二部分からなるアルゴリズムである（以下 RS アルゴリズムと呼ぶ）。すなわち、R アルゴリズムでは、パス切断に伴って必要となる回路の変換を、最長パス解消の対象として注目するゲートのファンインに対してのみ行なう。これが成功した場合、RS アルゴリズムは R アルゴリズムの適用のみで終了する。

S アルゴリズムでは、R アルゴリズムでの変換が完全には成功しなかった場合、プライマリ出力を不変に保つために、必要な出力を持つ新たなゲートを合成して注目するゲートのファンインに追加する。S アルゴリズムは再帰的なもので、正しいプライマリ出力が得られるまで繰り返し適用される。すなわち、新たに合成されたゲートのファンインに、さらに新たに合成したゲートを追加する。

R アルゴリズムは、面積最小化を行なうトランスダクション法の一つ、connectable / disconnectable[10] に基づいており、既存のアルゴリズム [7, 13] に近いものである。

g_j が最長パス上にあると仮定する。このとき、最長パスは g_j へのファンインのうちのいくつかを通っている。また、一般的に最長パスは複数本あり得る。e_{ij} が最長パスを構成しているとき、ファンイン側のゲート g_i の集合を D_j とする。すなわち、

$$D_j = \left\{ g_k \,\middle|\, g_k \in FI(g_j) \vee L_k^{in} + L_j^{out} = L^{max} \right\}$$

g_j を通る最長パスを全て解消するためには、D_j から g_j へのコネクションを全て切断することが必要である。その際、プライマリ出力を不変に保つためには、G_j^{off} のうち、f_k $(g_k \in D_j)$ によってカバーされていた部分を、f_k に代わってカバーするようなコネクションを追加しなければならない。そこで、g_j に対する接続可能条件すなわち

- $\overline{f_i} \supseteq G_j^{on}$
- コネクション e_{ij} は閉ループを形成しない

および、新たに最長パスを作らないための条件

- $L_i^{in} + L_j^{out} < L^{max}$

を同時に満足するようなゲート g_i の集合 C_j を考える。このとき、C_j の適当な部分集合 $\tilde{C}_j$ が、D_j によってカバーされていた部分の完全なカバーとなる、すなわち

$$\bigcup_{g_i \in \tilde{C}_j} f_i \supseteq \left[G_j^{off} - \bigcup_{g_k \in D_j} f_k \right]$$

を満たすことができれば、$\tilde{C}_j$ による D_j の置き換えが可能となる。

この結果、L_j^{in} は 1 またはそれ以上小さくなるので、g_j を通る最長パスはすべて解消されたといえる。

次に、S アルゴリズムについて述べる。R アルゴリズムでは、適当なカバー $\tilde{C}_j$ の存在が必要であった。しかし、このような部分集合が必ず存在するとはいえない。そこで、S アルゴリズムの目的は、$\tilde{C}_j$ によってカバーできなかった残りの部分をカバーする論理関数を持つ新たなゲートを合成することである。この合成は、ブール的手法であり、置き換え可能な関数の論理式表現を求める手法より広い解空間を持つ。

R アルゴリズムが失敗して、$\tilde{C}_j$ による G_j^{off} のカバーが不完全であるとしよう。すなわち、

$$\left[G_j^{off} - \bigcup_{g_k \in D_j} f_k \right] \not\subset \bigcup_{g_i \in \tilde{C}_j} f_i$$

である。すると、合成すべき新たなゲートとは、このG_j^{off}の欠落部分すなわちD_jによってカバーされていたが$\tilde{C}_j$によってはカバーできなかった部分をそのG^{on}とするようなゲートである。この仮想的ゲート${}^+g_j$の許容関数集合を${}^+G_j$とすると、

$$\begin{cases} {}^+G_j^{on} = G_j^{off} - \bigcup_{g_k \in D_j} f_k - \bigcup_{g_i \in \tilde{C}_j} f_i \\ {}^+G_j^{DC} = G_j^{DC} + \bigcup_{g_k \in D_j} f_k + \bigcup_{g_i \in \tilde{C}_j} f_i \\ {}^+G_j^{off} = G_j^{on} \end{cases}$$

である。また、${}^+g_j$からg_jへのコネクションが再び最長パスにならないため

$${}^+L_j^{in} = L_j^{in} - 2$$

としよう。この${}^+g_j$に対し、再び接続可能ゲートの集合${}^+C_j$を求め、その部分集合${}^+\tilde{C}_j$による${}^+G_j^{off} \equiv G_j^{on}$のカバーを試みる。

Sアルゴリズムを一度適用して得られたカバーが不完全であった場合、${}^+g_j$をg_jとみて、Sアルゴリズムを再帰的に適用することを繰り返す。

3.2 エラー補償法による最長パスの解消

エラー補償法を提案した原論文[8]では、この方法がゲート数の最小化に用いられたが、特別な考慮なしにこれを最長パス上のコネクション切断のために転用することができる。以下ではこれをEアルゴリズムと呼ぶ。エラー補償法自身については、さらに議論を要するものと思われるが、本論文では、前記論文のアルゴリズムに対し、(1) ファンイン制約を満たす、(2) 段数の増大を防ぐ、の二点の変更のみを行なって使用した。

以下でエラー補償法の概要を示す。もとの論理回路から、プライマリ入出力以外のゲートを一つまたはいくつか取り除いた回路を考える。ある選択基準に従ってゲートを除去すると、新たな回路の出力をもとの回路のそれに近いものにできる。すなわち、もとの回路でプライマリ出力f_jに対し、内部ゲートを取り除いた後の回路のそれを${}^*f_j$とする。そのとき二つの回路間の矛盾

$$\begin{cases} G_j^{Eon} \equiv G_j^{on} \cap \overline{{}^*f_j} \\ G_j^{Eoff} \equiv G_j^{off} \cap {}^*f_j \end{cases}$$

が小さい場合、コネクションの追加、削除を行なってもとどおりの出力を得られる可能性がある。そこで、g_jに対し接続可能条件を満たすゲートを用いて、

1. G_j^{Eoff}をカバーするコネクションの追加
2. G_j^{Eon}をカバーしているコネクションの置き換え

を試みる。これは、プライマリ出力に対応するゲートに対して常に成功するとは限らない。その場合、エラー補償後に残ったエラー部分をファンインへ向けて伝搬させ、それらを内部ゲート上で補償することを試みる。この処理はエラー部分が消滅する、すなわちエラー補償が成功するか、またはエラー部分がプライマリ入力へ達して失敗した時点で終了する。

3.3 切断箇所の特定

これらのアルゴリズムによって、g_jに関して最長パス上のコネクションe_{ij}を全て切断することができれば、g_jを通る最長パスが全て解消された、ということになる。したがって、このような切断が成功すると仮定すると、除去できる最長パスの本数に関して効果の高いゲートに注目してアルゴリズムを適用するのが有効である。

一方で、通常このようなゲートは論理回路の出力に対して大きな影響をもつ。従って、同時にG^{DC}が小さくなる傾向を持つ。G^{DC}の大きさは、変換の自由度と密接な関係があると考えられるので、前述の選択基準は切断の失敗に至る可能性を高めることになる。したがって、切断成功の可能性を高めるためには、G^{DC}の大きなゲートに注目した方がよいとも考えられる。

現実的に最も有効な切断箇所の特定方法は、両極端のこれら二案のトレードオフを考慮した中間的なものと思われるが、本論文では、最初の案によるインプリメントを行ない、実験結果を示した。

4 実験結果

アルゴリズムRSおよびEをインプリメントし、実験を行なった。本手法は、段数を目標値まで削減するためのものであるが、実験に際しては目標値を設定せず、それ以上の削減が不可能になるまで適用を行なった。最適化の対象とする初期回路はWeak Divisionによって多段化した。実験に用いたプログラムはNORゲートのみを取り扱う。ファンインの制約は、テクノロジマッピングを考慮して4とした。用いた計算機はDS5000であり、cpu時間の単位は秒である。

表1は、段数削減に関して行なった実験の結果である。 表の最初のカラムが初期回路とその段数を示し、以降のカラムが、段数最小化を適用した結果である。カラム(1)は、Rアルゴリズムのみによるものである。すなわち、Rアルゴリズムが失敗した場合、そのゲートでの最長パスの解消を断念する。これに対し、カラム(2)がSアルゴリズムを併用した結果である。35個のベンチマーク回路のうち、2/3にあたる24個で(1)に比べて削減が進み、唯一の例外としてaplaで2段増加した。平均では(1)よりさらに2.2段の削減が進み、ゲートとコネクション数の増大は4.9%、また計算時間の増加は10–20%であった。すなわち、SアルゴリズムはRアルゴリズムにほとんど悪影響を及ぼすことなく、解の品質を向上させているといえる。カラム(3)は、Eアルゴリズムによる。この

回路 名前 (段数)	(1) 段数	(2) 段数	(2) cpu	(3) 段数	(3) cpu	(4) 段数
5xp1(32)	12	10	1.1	14	59.8	9
9sym(17)	15	15	1.8	13	544	12
add6(28)	14	12	22.3	27	2696	10
adr4(20)	11	10	0.7	12	2.4	9
alu1(5)	5	5	2.2	5	13.3	5
alu2(18)	10	9	4.5	9	322	8
alu3(16)	12	9	4.7	9	221	9
apla(15)	10	12	3.2	14	64.3	10
bw(40)	16	6	4.6	12	77.1	7
clip(27)	13	10	4.7	11	469	10
co14(17)	17	17	15.4	12	249	11
con1(7)	7	7	0.2	7	0.5	7
dc1(19)	7	7	0.2	10	1.9	5
dc2(40)	14	9	2.8	9	175	7
dist(33)	18	14	5.3	19	7092	11
dk17(14)	10	8	4.6	10	59.9	8
dk27(6)	6	6	0.2	5	1.3	5
f2(10)	7	4	0.1	8	0.2	4
f51m(38)	18	10	2.3	27	46.3	14
misex1(19)	8	8	1.0	11	4.7	7
mlp4(42)	30	29	6.2	20	6107	16
radd(20)	11	10	0.9	12	22.1	10
rd53(18)	11	11	0.2	17	1.0	11
rd73(24)	13	12	0.8	19	79.7	12
rd84(22)	15	13	2.3	14	367	10
risc(16)	7	5	4.0	9	50.8	5
root(29)	15	12	2.3	14	543	13
rot8(28)	16	12	1.9	13	255	11
sao1(25)	18	12	3.0	23	507	14
sao2(23)	14	10	5.4	18	192	10
sex(10)	8	8	0.8	7	7.0	6
sqn(25)	19	11	1.5	14	76.6	10
sqr6(25)	9	8	1.3	20	32.1	8
wim(10)	8	6	0.1	10	0.1	6
z4(17)	10	10	0.2	10	4.8	9
平均 (21.6)	12.4	10.2	-	13.3	-	9.1

表 1: 段数最小化の効果

回路	[RS] 面積	[RS] 遅延	[E+RS] 面積	[E+RS] 遅延	SDLR[7] 面積	SDLR[7] 遅延
5xp1	178	9.1	172	9.0	204	15.5
9sym	301	11.9	290	14.6	165	14.3
add6	162	12.3	168	12.2	-	-
adr4	106	8.0	111	9.9	-	-
alu1	64	2.7	64	2.7	-	-
alu2	163	8.9	185	9.2	-	-
alu3	164	8.2	175	9.6	-	-
apla	231	13.5	255	12.5	-	-
bw	399	13.2	362	13.3	342	12.2
clip	187	8.8	194	11.7	232	14.9
co14	112	11.2	120	10.3	-	-
con1	40	3.8	73	5.0	46	6.4
dc1	88	7.3	94	8.0	-	-
dc2	218	8.9	233	10.7	-	-
dist	455	17.3	486	16.7	-	-
dk17	160	7.8	181	9.6	-	-
dk27	75	4.2	79	4.4	55	2.7
f2	47	3.0	46	3.2	39	5.8
f51m	176	10.2	165	11.1	142	17.9
misex1	136	8.1	129	11.0	113	11.9
mlp4	456	21.8	494	18.9	-	-
radd	116	7.3	101	9.4	106	11.3
rd53	67	9.3	77	8.7	56	10.0
rd73	149	12.0	154	10.5	86	16.3
rd84	177	11.7	232	12.1	206	19.0
risc	249	9.2	253	8.9	-	-
root	239	12.8	251	11.5	-	-
rot8	233	12.5	253	14.4	209	24.9
sao1	231	12.3	228	13.1	312	26.3
sao2	231	12.7	206	11.5	234	17.9
sex	123	6.6	118	6.3	129	8.6
sqn	181	10.7	204	11.1	217	14.0
sqr6	231	9.7	235	10.2	-	-
wim	55	4.1	55	4.1	-	-
z4	99	9.7	104	8.7	77	10.2

表 2: 面積最適化との併用

手法による結果は、概してRSアルゴリズムの結果より悪い。しかし、Eアルゴリズムの結果に対してRSアルゴリズムを適用した結果(4)を見ると、大部分の回路でRSアルゴリスム単独と同等またはそれ以上に小さくなっており、平均では(2)からさらに1.1段の削減が行なわれている。また、RSアルゴリズムで結果がかんばしくなかった回路（9sym、co14、dist、mlp4、rd84等）では、いずれもよい結果が得られている。これらのことから、二つのアルゴリズムが相補的な性格をもっていることが予想されるが、回路構造とアルゴリズムの関係などはさらに検討を必要とする。

表1に示したRSおよびE+RSアルゴリズムの結果に対して、面積最適化の後テクノロジマッピングを行なった結果を表2に示した。面積最適化手法としては、トランスダクション法を用いている。すなわち、connectable / disconnectable[10]に、ファンイン制約を守り段数を増大させないよう変更を加えたものである。テクノロジマッピングでは、ルールベースで局所的改善のみを行ない、遅延に関する最適化は考慮していない。用いたテクノロジはベンチマークデータMCNC2[9]のLibrary2で、面積については基本セルの大きさ(464)で正規化して示した。

同時に、本論文と同様、許容関数に基づく段数の最小化を行なっているSYLON-DREAM[7]の結果を示した。本論文の手法をこれと比較すると、遅延に関しては大部分の回路で優れている。しかし、面積に関しては二倍近くになっているものがある。

三つの結果を総合的に見て、面積と遅延時間の間に一貫した傾向は見られず、実験に用いた回路の性質を次の三つのタイプに分類できる。

1. 面積と遅延との間に妥当なトレードオフがあると見られる回路。f51m、misex1、rd53、rd73、z4など。

2. 面積的によく最適化されているものが遅延に関してもよい結果になっている回路。5xp1、bw、clip、sao2、sqnなど。

3. 特に傾向が見い出せない回路。9symなど。

このように、結果はデータに強く依存している。

論理合成における遅延の最適化が、一番目のグループに見られるトレードオフに対して最適点を見い出すものであるとすると、今後の重要な研究課題として(1) 最適化能力の強化、および (2) 回路構造と最適化手法の関係を論じる必要がある。しかし、実用的な論理合成システムを構築するうえでは、遅延時間の仕様を満たす回路を合成するためにこれらの手法は有効と考えられる。とくに、RSアルゴリズムのcpu時間はほとんどの回路で数秒程度であり、十分実用性のあるものといえる。しかし、Eアルゴリズムについては、結果は良好とはいえcpu時間は膨大であり、RSアルゴリズムの数倍からときに1000倍以上に及んでいる。これを実用化するためにはエラー補償法自身の改良と高速化が必要となる。

5 おわりに

論理合成に際して、遅延時間の最適化に有効な手法について述べた。提案された二つの手法を用い、遅延時間の仕様にあった回路を合成できる可能性を一層高めることができると考えられる。今後、処理時間の短縮、大規模回路の取り扱いを可能にし、実用的な論理合成システムの確立を目指す。

謝辞

本研究の一部は、著者の一人（藤本）がイリノイ大学に滞在中に行なわれたものである。御指導と有益な助言をいただいた室賀三郎教授に深謝します。

参考文献

[1] K. A. Bartlett, R. K. Brayton, G. D. Hachtel, R. M. Jacoby, C. R. Morrison, R. L. Rudell, A. Sangiovanni-Vincentelli, and A. R. Wang, "Multilevel Logic Minimization Using Implicit Don't Cares", *IEEE Trans. on CAD,* vol. CAD-7, pp. 723-740, June 1988.

[2] D. Bostick, G. D. Hachtel, R. Jacoby, M. R. Lightner, P. Moceyunas, C. R. Morrison, and D. Ravenscroft, "The Boulder Optimal Logic Design System", *Proc. IEEE ICCAD,* pp. 62-65, 1987.

[3] R. K. Brayton and C. McMullen, "The Decomposition and Factorization of Boolean Expressions", *Proc. IEEE ISCAS,* pp. 49-54, 1982.

[4] R. K. Brayton, R. Rudell, A. Sangiovanni - Vincentelli, and A. R. Wang, "MIS: Multi-level Interactive Logic Optimization System", *IEEE Trans. on CAD,* vol. CAD-6, pp. 1062-1081, November 1987.

[5] R. K. Brayton, A. Sangiovanni-Vincentelli, and G. D. Hachtel, "Multi-Level Logic Synthesis", *Proc. of the IEEE,* vol. 78, pp. 264-300, February 1990.

[6] K-C. Chen and S. Muroga, "SYLON-DREAM: A Multi-Level Network Synthesizer", *Proc. IEEE ICCAD,* pp. 552-555, 1989.

[7] K-C. Chen and S. Muroga, "Timing Optimization for Multi-Level Combinational Networks", *Proc. 27th ACM/IEEE DAC.,* 1990.

[8] Y. Kambayashi, H.C. Lai, J. N. Culliney, and S. Muroga, "NOR Network Transduction based on Error Compensation (Principles of NOR Network Transduction Programs NETTRA-E1, NETTRA-E2, NETTRA-E3)", UIUCDCS-R-75-737, Dep. Comput. Sci., Univ. of Illinois, June 1975.

[9] R. Lisanke, "Logic Synthesis and Optimization Benchmarks User Guide Ver. 2.0", *Technical Report, Microelectronics Center of North Carolina,* December 1988.

[10] S. Muroga, Y. Kambayashi, H. C. Lai, and J. N. Culliney, "The Transduction Method - Design of Logic Networks based on Permissible Functions", *IEEE Trans. Comput.,* vol. C-38, pp. 1404-1424, October 1989.

[11] 高橋 登, 松永 裕介, "論理合成システムにおける遅延時間最適化手法", 第41回情報処理全国大会 4N-12, 1990.

[12] 藤田 昌宏, "トランスダクション法に基づく多段論理簡単化機能を持つ論理合成システムとその評価", 情報処理学会論文誌, Vol. 30, No. 5, pp. 613-623, 1989.

[13] 藤田 昌宏, 角田 多苗子, "遅延時間の増加を押さえたトランスダクション法について", 第39回情報処理全国大会 4X-5, 1989.