設計自動化 55－11
(1990.12.13)

論理を考慮した遅延時間最適化手法

松永　裕介†　　藤田　昌宏†
高橋　登††　　佐藤　妃登美††

†（株）富士通研究所　††富士通株式会社

あらまし　本稿では，組み合わせ論理回路の遅延時間改善処理の手法について述べる．本手法は，許容関数を用いることで回路内部のドントケア条件を考慮した大局的な回路変換を行なえるところに大きな特徴を持ち，また，テクノロジに依存したレベルで回路を扱えるように，仮想ゲートネットワークと呼ばれる回路モデルを用いている．ベンチマーク回路による実験では，面積の増加を１０％以内に抑えたうえで最大遅延時間を１０％～５０％削減できることが示されており，本手法の有効性を裏付けるものとなっている．

A Boolean Delay Optimizer

Yusuke Matsunaga†, Masahiro Fujita†,
Noboru Takahashi††, and Hitomi Sato††

†FUJITSU LABORATORIES LTD., ††FUJITSU LTD.
1015, Kamikodanaka Nakahara-ku, Kawasaki 211, JAPAN

Abstract　This paper introduces a delay optimization method which uses permissible functions. Since our method is able to re-configure global circuit structure with taking don't care conditions into consideration, we can optimize maximum delay time significantly without increasing gate area. Moreover, to calculate exact delay time, we introduce a new circuit model called "virtual gate network", which is an intermediate model between Boolean network and real gate network. From experiments with benchmark circuits we achieved up to 50 percent delay reduction within less than 10 percent area increase.

1　はじめに

論理合成処理では，誤りなく高品質（ゲート面積小・最大遅延時間小）の回路を生成することが目的となっている．この内，面積を最小化する処理については，ESPRESSO-II[1]等の二段論理簡単化手法やweak-division[2]による多段化手法，MIS[3], BOLD[4], Transduction[5]等のドントケアを用いた多段論理簡単化手法と，様々な手法が提案されており，その実験結果からそれらの手法が十分有効であることが示されている．一方，遅延時間を改善する処理に関してもいくつかの手法が提案されているが，大幅な改善が期待できない・面積が大きく増える等の問題があり，面積最小化処理のように確立された手法があるとは言いがたい．

本稿では，Transduction[5]で提案されている許容関数の概念を用いた遅延時間改善手法について述べる（遅延時間最適化という用語を用いる場合もあるが，明らかなように絶対的な最適性の保証はない）．本手法は他の多くの手法と異なり，回路各部の論理を考慮した大局的な回路変換を行なうことが可能となっている．さらに，仮想ゲートネットワークと呼ばれる回路モデルで回路を表現することにより，テクノロジに密着した遅延時間の解析を行なっている．ベンチマーク回路を用いた実験では，面積の増加を１０％以内に抑えながら遅延時間を１０％～５０％程度短縮できることが確認されている．

以下，２章では本手法で用いられる遅延時間のモデルを，３章では従来手法の簡単な紹介を行なう．そして，４章で許容関数に基づく遅延時間最適化手法の詳細について述べ，５章にベンチマーク回路を用いた実験結果を示す．

2　遅延時間モデル

LSI回路の遅延時間モデルとしては様々なものが考えられるが，ここではfalse path（実際にはいかなる入力系列においても活性化されない経路）の特定を行なわない静的なモデルを扱うものとする．このモデルによれば，ネットワークの構造のみから最大遅延時間が求められるため計算処理は単純となるが，最大遅延時間を与える経路がfalse pathの場合，真の最大遅延時間よりも大きな値が得られることになるという問題もある．後述のように，本手法では回路各部の論理を計算しているのでfalse pathを求めることも可能であるが，false pathには本当に活性化されないものとタイミングによってはハザードが駆け抜けて一瞬だけ活性化されてしまう偽のfalse path(true path?)の二種類があり，込み入った議論が必要となるため，ここでは簡単のため前述のモデルを用いる．

次に，個々のゲート（セル）の伝播遅延をどのように扱うか，に関してもいくつかの段階がある．ゲートgの伝播遅延時間をd_gとすると，次のようなモデルが考えられる．

A.　$d_g=1$：

つまり全てのゲートが同一の遅延を持つ．テクノロジに依存しない多段論理式の論理段数と一致する．

B.　$d_g=\alpha_g$：

各々のゲートはそれぞれ固有の遅延α_gを持つ．

C.　$d_g=\alpha_g+\beta_g\cdot\Sigma\gamma_i$：

ゲート固有の遅延α_gの他にファンアウト先の負荷γ_iの総和に比例した遅延を持つ．β_gはその比例定数である．

D.　$d_g=\alpha_g+\beta_g(\Sigma\gamma_i)$：

ファンアウト先の負荷$\Sigma\gamma_i$に対する遅延時間が非線形関数$\beta_g()$で表されるもの．具体的には，

$$\beta_g(\Sigma\gamma_i)=\beta 1_g\cdot\Sigma\gamma_i \quad (0\leqq\Sigma\gamma_i<c1)$$
$$=\beta 2_g\cdot\Sigma\gamma_i \quad (c1\leqq\Sigma\gamma_i<c2)$$
$$\cdots$$

という形で，分割された区間を線形関数で近似するものがある（図１）．

通常はC．のモデルが用いられているようである（ＭＣＮＣのベンチマークとして提供されているライブラリもこの形式である）が，ファンアウト先の負荷が極端に大きい場合にはそのような近似は適当ではないのでD．のモデルが必要となる．本手法ではC．，D．どちらのモデルを用いることも可能である．

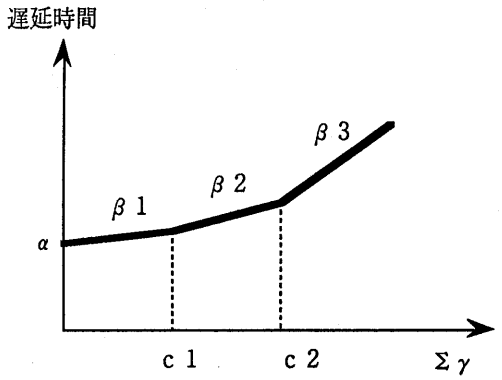


図1：負荷に対して非線形な遅延時間モデル

3 従来の手法

ここでは，遅延時間改善手法に関する従来の手法について簡単に紹介する．

3．1 部分回路を二段論理式に戻して再合成する手法[6]

最大遅延時間を与える経路（ここでは最大遅延経路と呼ぶ）上の部分回路を一旦，二段論理式に戻して（何らかの制約を設けて）再度，多段回路を合成する手法．多段化処理では実際のゲート遅延を見積れないので論理段数を目安にしておいて，細かな遅延時間は次の局所変換を用いた手法で行なう，という組み合わせもしばしば用いられる．問題点は，論理段数がそれほど良い目安にならない場合が多い（特に浅い多段論理ではその傾向が顕著である）ことと，再合成する部分から他のゲートへファンアウトがある場合には，同一の部分回路の複製を作る必要があり，大幅な面積の増加を招くことである．

3．2 ルールベースによる局所変換を用いた手法[7]

あらかじめ用意された変換パターンに従って局所変換を繰り返すことにより，遅延時間を改善するもの．同様の仕組みで面積を小さくすることもできる．基本的に着目したゲートのすぐ周りしか見ないで変換を行なう（peephole transformation）ため，初期回路の構造を大きく変えてしまうような変換は行なえないし，大局的な見地で回路変換をコントロールすることも難しい．

3．3 バッファを挿入する手法[8]

多数のファンアウトを持つゲートの出力にバッファを挿入することでドライブ能力を高め，ファンアウト先の負荷に起因する遅延時間を短くするもの．初期回路の構造を全く変更しないため，ファンアウト数が多くないかぎり，大きな改善は期待できない．

3．4 遅延時間を考慮したテクノロジ・マッピングを行なう手法[9]

DAGON[9]のようなtree matchingに基づくテクノロジ・マッパを用いるもので，tree状の回路に対して，前述のA．やB．の遅延時間モデルを仮定すれば回路規模に比例した時間で最適解を求めることができる．ただし，ここで言う最適解とはtree matchingの入力となる2入力NANDのみで構成された回路－subject graphと呼ぶ－に対するものであって，初期回路からどのようなsubject graphを作るのかによって得られる回路は異なってくる．そこで，遅延時間に関する要請でsubject graphを作り直す手法（restructuringと呼ぶ）[10]も提案されている．この変換も初期回路の構造を保ち続けるので，最大遅延時間は初期回路の構造に大きく依存する．

以上のように，従来手法の多くは可能な解空間のごく一部を探索しているに過ぎず，初期回路の構造を大きく変更することはない．

4 許容関数に基づく遅延時間改善手法

本章では，回路構造を大幅に変更しうる回路変換を伴った遅延時間改善手法について述べる．本手法の特徴は，

1： 許容関数に基づく回路変換を行なうことにより，回路構造を大幅に変更することができる．さらに，weak-divisionなどの代数的な方法では扱うことのできない回路内部のドントケア条件を考慮することができる．

2： 回路を仮想ゲートネットワーク（Virtual Gate Network）と呼ぶモデルで扱うことにより，テクノロジに依存した形で遅延時間の評価が行なえる．

の2点である．

4．1　許容関数

許容関数（permissible function）はTransduction法[5]で用いられている概念で，多段回路内部の構造から生ずるドントケア条件を表している．論理関数および，許容関数の定義は次のようになる．

定義１：

回路ネットワークηの外部入力をx_i (i = 1, ..., n)，外部出力をy_j (j = 1, ..., m)，ゲートをv_k (k = 1, ..., l) とする．この時，外部出力y_jおよび，ゲートv_kの値を外部入力の関数として表現したものを論理関数と呼び，Fy_j，およびFv_kと表す．

定義２：

回路ネットワークηにおいて，ゲートvの論理関数を他の論理関数Fvに変更した結果，ηのいかなる外部出力y_jの論理関数Fy_jにも変化が現われない時，そのような論理関数F'vをゲートvに対する許容関数と呼ぶ．

例えば，図２の例で，ゲートcの論理関数は$\bar{b}$となっており，a＝0, b＝0の時と，a＝1, b＝0の時に値が１になっている．しかし，a＝0の時はゲートeの出力が（ゲートcの値にかかわらず）０となるため，ゲートcの値が１である必要はない．そこで，例えばゲートcの論理関数を$\bar{a}+\bar{b}$に変えてみても回路の出力は変化しない．つまり，$\bar{a}+\bar{b}$はゲートcの許容関数である．また，$\bar{a}b$もゲートcの許容関数となっている．

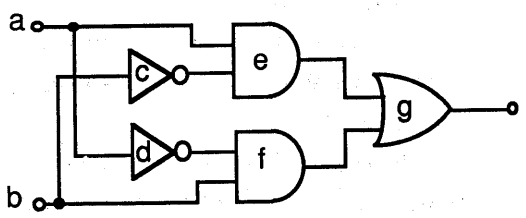
通常，１つのゲートに対する許容関数は複数存在するが，そのような許容関数の集合を１つの不特定記述関数(imcompletely specified function）として表現することが可能である[5]（任意の論理関数の集合は必ずしも１つの不特定記述関数で表現できないが，許容関数の集合の場合には可能である）．不特定記述関数で，出力が'1'となるような入力値の集合をON-set，出力が'0'となるような入力値の集合をOFF-set，残りをDC-setと呼ぶ．例えば，先のゲートcの例では，（ON-set, OFF-set）という表記で，$(\bar{a}\bar{b}, ab)$と表される．回路変換の処理などでは，個々の許容関数よりも許容関数の集合を主に扱うので，以後記述の簡略化のために，不特定記述関数として表現された許容関数の集合を許容関数と書くことにする．

許容関数に基づく回路変換の手法としては様々なものが提案されているがその基本となる考えは極めて単純なものである．つまり，ある部分回路に対して回路変換を施した結果が外部出力の論理を変更しないかの判定を，その部分回路の出力部分の論理関数がその許容関数に含まれているかどうかで行なうというものである．遅延時間改善処理のための回路変換については４．３節で述べる．

4．2　仮想ゲートネットワーク

回路の遅延時間を考える場合，テクノロジに密着したレベルで回路を扱えることが望ましいが，その半面，あまりテクノロジに依存してしまっては自由な回路変換が行なえなくなるという問題もある．これらの点を考慮して，仮想ゲートネットワーク（Virtual Gate Network）と呼ばれる回路モデルの導入を行なう．後述するように，仮想ゲートネットワークにはCMOS用のVGN-CとECL用のVGN-Eの２種類がある．

一般に，テクノロジ非依存のレベルの回路モデルとしてはBoolean network[3]が広く用いられている．Boolean networkは非循環有向グラフ（DAG）で，各々の節点が



入力		論理関数					許容関数				
a	b	c	d	e	f	g	g	e	f	c	d
0	0	1	1	0	0	0	0	0	0	*	*
0	1	0	1	0	1	1	1	*	1	*	1
1	0	1	0	1	0	1	1	1	*	1	*
1	1	0	0	0	0	0	0	0	0	0	0

図２：許容関数の例

論理式を持ち（ここではそのような論理式を節点の機能と呼ぶ），直接に依存関係のある節点間に有向枝が接続されているというものである．また，Boolean networkの節点の機能を任意の論理式とせずに，特定のライブラリのセルの論理式のみにするという制約を与えれば，ライブラリ依存のゲートネットワークを表すことができる．ここで提案している仮想ゲートネットワークはこの両者の中間にあたるもので，ファンイン数の制限を持たない単一ゲートのみが許されたBoolean Networkである．

A． VGN-C

一般のCMOSセルライブラリはNAND/NORゲートを基本としているので，CMOS用の仮想ゲートネットワークではｎ－入力NAND/NORゲートをプリミティブとしている．もちろん，VGNでは複合ゲートやNAND/ORゲートを単一のノードとして表すことはできないので，遅延時間の計算は実際のセルのネットワークに基づいて行ない，回路変換はこのVGN上で行なう，というアプローチをとる．

B． VGN-E

ECLセルライブラリはOR/NORゲートを基本としているが，CMOSの場合と異なり，ゲートの出力が肯定／否定のどちらも利用することができるという特徴を持つ．

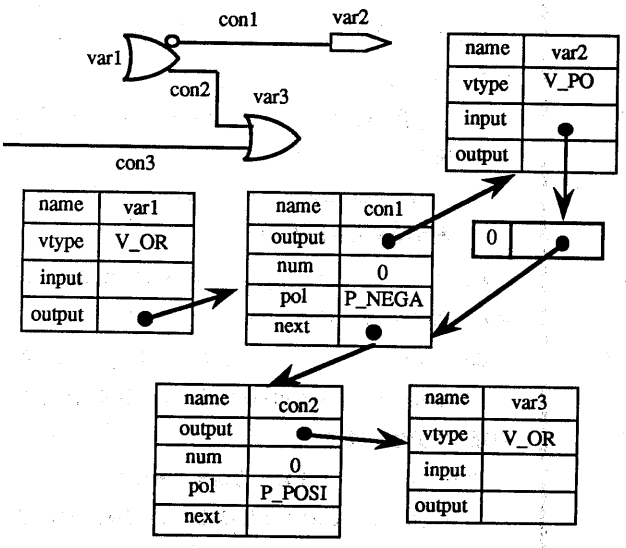


図３：VGN-Eの例

そこで，VGN-Eは一般のBoolean networkと異なりノード間の枝が極性を持った構造となっている．また，ノードはｎ－入力ORゲートをプリミティブとしている．このVGN-Eの例を図３に示す．

4．3 Procedure DELAY_OPT

この遅延時間改善手法はテクノロジ依存の論理合成システムの一部として実装されている．このシステムは入力としてテクノロジ非依存のBoolean networkを受け取り，与えられたセルライブラリを用いたテクノロジ依存のゲートネットワークを生成する．処理を図４に示す．

まず最初にBoolean networkから仮想ゲートネットワークの生成を行ない，仮想ゲートネットワーク上で面積最小化処理を行なう．具体的にはTransduction[5]の手法やBoolean division[11]を用いている．次に，仮想ゲートネットワークに対してセル割り当てを行なう．ここでは主に複合ゲートへの割り当てと多入力ゲートの分解等の処理をルールベースで行なう．複合ゲートは仮想ゲートネットワーク中の複数のノードに対応させておき，面積や遅延時間の計算時にはそのような複数のノードを一かたまりで扱う．しかし，遅延時間改善処理の回路変換時には別個のノードとして扱っている．

遅延時間を改善させる回路変換を１つ行なうごとにセル割り当て処理を再度行ない正確な遅延時間を計算じなおす．遅延時間を減少させるような回路変換が行なえなくなるまで（あるいは，最大遅延時間が与えられた制約値より小さくなるまで），このような処理が繰り返し行なわれる．セル割り当て処理は繰り返しループ中で毎回行なわれるが，あらかじめ回路が仮想ゲートネットワークで表現されているため，比較的高速に処理することができる．

遅延時間改善処理を図５に示す．ここでは２つのストラテジが用いられている．１つは最大遅延経路のカット，もう１つはファンアウトの削減である．

```
テクノロジ依存の最適化処理(η){
／＊　η：Boolean network　＊／
／＊　ν：virtual gate network　＊／
      ν = CREATE_VGN(η);
      AREA_OPT(ν);
      do {
            CELL_ASSIGN(ν);
            status = DELAY_OPT(ν);
      } while (status == improved);
      return ν;
}
```

図4：テクノロジ依存の最適化処理

まず最初に試される回路変換は，最大遅延経路上の接続を切ってそのかわりに信号伝播時間の早いゲートの出力をつなぐ，というものである．このような回路変換が行なえないときには，最大遅延経路上のゲートのファンアウト数の削減を試みる．この回路変換も前者と同様に，ファンアウト先のゲートから接続を切ってそのかわりに他のゲートの出力をつなぐもので，どのゲートからどの接続を切って，新たにどのゲートをつなぐか，という候補が異なっている．

ゲートから接続を削除できるかどうか／ゲートに新たな接続をつなぐことができるか，という判断は前述の許容関数を用いて行なう．削除の判断は極めて簡単で，ある接続の許容関数のON-setまたはOFF-setのどちらかがϕの時にそのような接続を切ることができる．もちろん，通常はそのように単独で削除できる接続はほとんど存在しないので，その接続を冗長にするように新たに他の接続を追加する．接続の追加条件は少し複雑である．例えばNANDゲートgに他のゲートhから新たに接続を追加することを考える．ゲートgの許容関数をPFg，ゲートhの論理関数をFhとすると，ゲートhからの接続を追加することによって，Fhに含まれる部分が1となる．そこで，PFgのオフセットがFhと共通部分を持てば，0でなければならない入力値に対して実際には1となってしまうことになり，追加が認められないことになる．このようにして，接続の削除／追加の判断を行なっている．詳細は文献[5]を参照されたい．

```
DELAY_OPT(ν){
／＊　ν：virtual gate network　＊／
    for each critical node v {
        try to connect faster nodes to v;
        try to delete slower inputs from v;
        if success {
            return improved;
        }
    }
    for each critical node v {
        for each fanout node u of v {
            try to connect nodes to u;
            try to delete v from u;
            try to delete other inputs from u;
            if success {
                return improved;
            }
        }
    }
    return failed;
}
```

図5：遅延時間改善処理

5　実験結果

本手法をUNIX上のC言語を用いて実装し，ベンチマーク回路に対して適用した結果を示す．論理関数・許容関数を表現するために文献[11]と同様に2分決定グラフ[12]を用いている．入力としてMCNC'89 Logic Synthesis Workshopの2段論理のベンチマークを用いた．処理の手順としては，

1．　ESPRESSO-IIアルゴリズムによる2段論理式簡単化

2．　weak-divisionおよびBoolean division[11]による多段論理生成／簡単化

を行なって，テクノロジ非依存の多段回路を生成した後に，CMOS，ECLのセルライブラリを用いてテクノロジ依存のゲートネットワークの生成を行なった．その際に，以下のような2種類の処理を行ない，遅延時間改善処理の有効性を調べた．

3－（a）．　面積最小化処理のみ行なう．

3－（b）．　面積最小化処理の後に遅延時間改善処理を行なう．

CMOS回路の結果を表1に，ECL回路の結果を表2に

示す．使用計算機はSUN4/330，セルライブラリは富士通のセル敷詰め型CMOSゲートアレイAUシリーズ，およびECLゲートアレイETHシリーズを用いている．遅延時間モデルはファンアウトの負荷が増えるに従って負荷容量遅延の係数が段階的に大きくなっていくものを用いている．面積の単位はベーシックセル数，遅延時間はナノ秒である．

まず，CMOSの例では，面積をほとんど増加させることなく（平均－１%，最大７%），平均で２０%，最大で４０%程度最大遅延時間を短縮している．また，ECLでも面積の増加は平均で４%，最大でも１５%にとどまり，最大遅延時間は平均で３０%，最大で５０%短縮されている．従来の遅延時間改善手法では面積と遅延時間のトレードオフが論じられてきているが，本手法の実験結果を見るかぎりそのような傾向は認められない．原因として考えられることは，まず第一に初期回路が真の面積最小の回路ではなくより小さな回路が存在する可能性があるということ，そして第二に，本手法のように論理を考慮して大局的に回路変換を行なうことでそのような解空間を探索することが可能になるということである．３章で示したような従来手法では，面積を増加させずに遅延時間を改善できるとは考えにくい．

この実験結果を見るかぎり，遅延時間改善処理そのものは非常に強力で有効であるといえるが，問題はその計算時間であろう．１００ゲート程度の回路は数分以内に処理しているが，３００ゲートを越えると１時間～数時間を要している．試行する回路変換の候補を絞るなどのヒューリスティックが必要と思われる．

6　終わりに

本稿では，組み合わせ論理回路の遅延時間改善処理の手法について述べた．本手法は他の多くの手法と異なり，論理を考慮した大局的な回路変換を行なえるところに大きな特徴を持ち，そのための手段として許容関数に基づく回路変換を用いている．また，テクノロジに密着したレベルで回路を扱うために，仮想ゲートネットワークと呼ばれる回路モデルを用いている．ベンチマーク回路による実験結果では，（面積最小化後の回路に対して）面積の増加を１０%以内に抑えたうえで，最大遅延時間を１０%～５０%削減できることが示されており，本手法の有効性を裏付けるものとなっている．

今後の課題の１つに，より複雑な遅延モデル－例えばfalse pathの特定を行なう動的な遅延時間モデル－への対応が挙げられる．前述のように，回路各部の論理関数はすでに計算されているのでそのような遅延時間モデル上での解析も比較的容易に行なえるものと思われる．

もう１つの課題は，レイアウトの影響を何らかの形で考慮した論理合成手法であろう．高集積化が進みゲート１個あたりの遅延時間が短くなった分，配線の遅延が占

name	遅延改善なし			遅延改善処理あり		
	area	delay	CPU	area	delay	CPU
5xp1	66	20.21	10	67 (1.02)	13.29 (0.66)	46
9sym	121	19.29	11	119 (0.98)	15.29 (0.79)	90
alu4	100	15.09	10	98 (0.98)	10.97 (0.73)	77
bw	116	19.67	13	124 (1.07)	14.51 (0.74)	198
clip	83	15.01	15	81 (0.98)	9.51 (0.63)	73
duke2	280	37.78	178	278 (0.99)	27.56 (0.73)	2059
misex1	41	9.9	7	43 (1.05)	6.96 (0.70)	16
misex2	96	7.3	10	97 (1.01)	6.03 (0.83)	22
misex3	413	43.09	1122	400 (0.97)	41.67 (0.97)	13762
misex3c	353	43.19	448	348 (0.99)	33.94 (0.79)	5287
rd53	30	8.59	9	31 (1.03)	8.19 (0.95)	13
rd73	63	12.73	7	57 (0.90)	10.16 (0.80)	33
rd84	135	20.63	21	128 (0.95)	17.48 (0.85)	56
sao2	106	18.23	11	102 (0.96)	13.41 (0.74)	88
vg2	82	9.18	9	82 (1.00)	7.14 (0.78)	47
total	2085	299.89	1881	2055 (0.99)	236.1 (0.79)	21867

※面積はベーシックセル数、遅延はナノ秒、CPU時間は秒（SUN4/330）

表１：CMOS回路の実験結果

name	遅延改善なし			遅延改善処理あり		
	area	delay	CPU	area	delay	CPU
5xp1	43	14.08	42	46 (1.07)	12.92 (0.92)	62
9sym	79	14.19	248	83 (1.05)	12 (0.85)	291
alu4	60	9.31	138	61 (1.02)	7.8 (0.84)	155
bw	83	17.63	67	84 (1.01)	11.07 (0.63)	261
clip	55	13.29	47	56 (1.02)	9.21 (0.69)	73
duke2	159	20.14	1357	170 (1.07)	14.33 (0.71)	2021
misex1	27	7.11	19	30 (1.11)	6.07 (0.85)	31
misex2	53	5.38	25	61 (1.15)	3.94 (0.73)	51
misex3	234	30.95	2286	242 (1.03)	17.16 (0.55)	11878
misex3c	195	32.53	4885	207 (1.06)	15.82 (0.49)	7235
rd53	21	7.12	7	23 (1.10)	5.77 (0.81)	19
rd73	39	8.7	26	39 (1.00)	7.68 (0.88)	37
rd84	73	13.98	113	75 (1.03)	13.19 (0.94)	156
sao2	65	12.86	109	58 (0.89)	7.54 (0.59)	174
vg2	59	6.27	42	65 (1.10)	6.11 (0.97)	73
total	1245	213.54	9411	1300 (1.04)	150.6 (0.71)	22517

※面積はベーシックセル数、遅延はナノ秒、CPU時間は秒（SUN4/330）

表２：ECL回路の実験結果

める割合が大きくなりつつあるため，見かけの最大遅延時間が改善されても，レイアウトしてみると配線長が長くなり実際には最大遅延時間が増加してしまう場合も考えられる．同様のことは面積についても言える．そこで，論理合成処理の段階で，結果の回路の配線容易性や配線長を近似する手段が必要になるものと思われる．

参考文献

[１] R. K. Brayton, G. D. Hachtel, C. McMullen, and A. Sangiovanni-Vincentelli, "Logic Minimization Algorithms for VLSI Synthesis", Kluwer Academic Publishers, 1984.

[２] R. K. Brayton, C. McMullen, "The Decomposition and Factorization of Boolean Expressions", Proceedings of the International Symposium on Circuits and Systems, Rome, 1982, pp. 49-54.

[３] R. Brayton, R. Rudell, A. Sangiovanni-Vincentelli, A. Wang, "MIS: Multiple-Level Interactive Logic Optimization System", IEEE Trans. on CAD. Vol. CAD-6, no.6, pp. 1062 - 1081, Nov. 1987.

[４] D. Bostick, G. D. Hachtel, R. Jacoby, M. R. Lightner, P. Moceyunas, C. R. Morrison, D. Ravenscroft, "The Boulder Optimal Logic Design System", IEEE International Conference on Computer Aided Design(ICCAD-87), pp. 62-65, Nov. 1987

[５] S. Muroga, Y. Kambayashi, H. C. Lai, J. N. Culliney, "The Transduction Method - Design of Logic Networks based on Permissible Functions", IEEE Trans. on Computers, Vol. 38, No. 10, pp.1404 - 1424, Oct. 1989.

[６] 三浦 順子，竹田 信弘，神戸 尚志，" 多段論理合成における時間最適化の一手法"，情処研報 Vol. 89, No. 108, 89-DA-50-11, pp. 79-84, Dec. 1989.

[７] D. Gregory, K. Bartlett, A. de Geus, and G. Hachtel, "Socrates: A System for Automatically Synthesizing and Optimizing Combinational Logic", 23th Design Automation Conference, pp. 79 - 85, June 1986.

[８] K. J. Singh and A. Sangiovanni-Vincentelli, "A Heuristic Algorithm for the Fanout Problem", 27th Design Automation Conference, pp. 357- 360, June 1990.

[９] K. Keutzer, "DAGON: Technology Binding and Local Optimization by DAG Matching", 24th Design Automation Conference, pp. 341-347, June 1987.

[１０] K. Singh, A. Wang, R. Brayton, and A. Sangiovanni-Vincentelli, "Timing Optimization of Combinational Logic", IEEE International Conference on Computer Aided Design(ICCAD-88), pp. 282-285, Nov. 1988.

[１１] 松永 裕介、藤田 昌宏、" トランスダクション法の評価と改良"、情処研報 Vol. 89, No. 108, 89-DA-50-8, pp. 55-62, Dec. 1989.

[１２] R. E. Bryant, "Graph-Based Algorithms for Boolean Function Manipulation", IEEE Trans. on Computers, August, 1986, pp.677-691.