論　文

トランスダクション法による Wired-Logic を併用した論理回路の最適化

山下　茂[†*]　上林　彌彦[†]　室賀　三郎[††]

Desigin of Logic Circcuits with Wired-Logic Utilizing Transduction Method

Shigeru YAMASHITA[†*], Yahiko KAMBAYASHI[†], and Saburo MUROGA[††]

あらまし　Wired-Logic はファンイン制限のある回路の素子数や段数を減らすのに特に有用と言える．しかし，Wired-Logic を用いた論理設計については筆者らのものを除いてほとんど知られていない．
本論文では，Wired-Logic を使用して論理段数を減らす設計手法について述べる．この手法は，論理回路最適化手法であるトランスダクション法を用いて結線の形状を変更することにより，効率良く Wired-Logic を使用してファンイン制限を行う手法である．そして，特に重要な NOR 素子と Wired-OR の組合せに対して，ファンイン制限を行うための Wired-OR の利用アルゴリズムを示している．また，効率の良い直列分割である一般化直列分割を用いた手法と Wired-OR を用いた手法をそれぞれ 3 段 NOR の初期回路のファンイン制限に適用し，比較を行っている．この結果，Wire-OR の利用によりほとんどの回路で段数が減少することがわかり，Wired-Logic を回路合成に併用することの有用性が示された．コストとしては単なる段数だけでなく，LSI 上での線長が重要であるが，本論文の結果はそのようなコストを用いた場合に拡張することも可能である．

キーワード　論理設計，トランスダクション法，Wired-Logic，ファンイン制限

1. まえがき

近年，半導体技術の進歩により，ワンチップの中に実現される回路は大規模となってきている．そのため，計算機による論理回路設計支援がますます重要となり，初期回路の生成や設計改良によるコストの低減等の論理回路の自動合成について，さまざまな角度から研究が行われている．そのような論理回路の自動合成システムの中で，トランスダクション法が，様々な分野で回路の最適化に強力な性能を発揮することが確認され，注目されるようになってきた [4] [6] [8]．トランスダクション法は，イリノイ大学で筆者らが 1971 年から 1973 年にかけて開発した設計改良による論理回路最適化手法で，許容関数（Permissible Function）という概念に基づき，回路の冗長部分の削除，回路の形状変換を実現する広域的な論理回路合成法である [2] [5]．

高速の VLSI 回路の設計のためにはファンイン制限を考えた上で，段数と素子数をできる限り少なくすることが重要である．ファンイン制限に対しては，否定素子を 2 段入れる直列分割 [1] が知られており，直列分割の結果にトランスダクション法を適用すると結果は大幅に改善される．更に一般化直列分割を用いると直列分割より段数が減るので，段数削減に有効であった [7] [9]．本論文では，更に Wired-Logic [3] を併用してファンイン制限を行う手法について述べる．

TTL や ECL などの代表的な論理素子では，出力を結合させるだけで論理関数の実現できる Wired-Logic を用いることが可能である．Wired-Logic は遅延が他の素子に比べてかなり小さいという性質があるため，回路の素子数や実効的な段数を減少させるために有用であると考えられる．しかし，Wired-Logic を通常の素子と同じように扱うためには，結線の形状にある種の制約があるので，どんな場合にも Wired-Logic を用いて素子の出力をまとめられるとは限らない [3]．

[†] 京都大学工学部，京都市
Faculty of Engineering, Kyoto University, Kyoto Unversity, Kyoto-shi, 606-01 Japan
[††] イリノイ大学計算機科学科，イリノイ州アーバナ市
Department of Computer Science, University of Illinois, Urbana, Illinois
[*] NTT コミュニケーション科学研究所，京都府

そこで，本論文ではトランスダクション法における接続可能 / 切断可能の概念を利用することによって，結線の形状のみから判断するよりもより多くの場合に Wired-Logic を使用することができる条件を提案し，Wired-OR を併用した回路変換について述べている．実際に，本論文で述べるような手法を実現し，評価を行ったところ，Wired-Logic を回路変換に使用することの有用性が示された．

2. 基本的事項

本章では，基本的事項として [5] で述べられているトランスダクション法，および [3] で述べられている Wired-Logic の性質，および [9] で述べられている一般化直列分割についてその概要を述べる．

2.1 トランスダクション法

2.1.1 基本的事項

本論文では，帰還ループのない組合せ論理回路を扱う．回路中の素子としては，NOR 素子と Wired-OR のみを扱うが，他の素子への拡張は容易である．

以下で扱う回路は n 個の入力変数をもつとものとする．また，IP(v) で素子 v の入力がつながっている素子の集合を，IS(v) で素子 v の出力がつながっている素子の集合を表すものとする．

$f(c)$ で回路中の要素 c（素子または結線）で実現される論理関数を表すものとする．$f(c)$ は 2^n 次元ベクトルで，$f(c) = (f(c)^{(1)}, f(c)^{(2)}, \cdots, f(c)^{(2^n)})$ と表現される．ここで $f(c)^{(j)}$ は，$f(c)$ を表す真理値表の j 番めの値で，0 か 1 である．

2.1.2 許容関数

［定義 1］ 入力端子，素子または結線の実現する論理関数を，論理関数 f で置き換えても，回路のすべての出力の定義されている部分に変化がなければ，f は，入力端子，素子または結線の**許容関数（Permissible Function）**であると言う．

許容関数は 一つしかないとは限らないので，一般に入力端子，素子，結線に対して許容関数の集合が考えられる．また許容関数の集合の中で，同時に置き換え可能なものからなる部分集合を CSPF （Compatible Set of Permissible Functions）と呼ぶ．本論文では，この CSPF を用いて回路変換を行う．以下では，回路中の要素 c（素子または結線）の CSPF を $G(c)$ で表すことにする．$G(c)$ は 0, 1, *(don't care) の 3 値をとる 2^n 次元ベクトルで表現でき，ベクトルの要素に * を含む場合，* の代わりに 0 と 1 とをすべての可能な組合せで代入して生成されるすべてのベクトル（論理関数）の集合が c の CSPF であることを表している．例えば c の CSPF が {(1,0,0,0),(1,0,0,1),(1,0,1,0),(1,0,1,1)} である場合，$G(c) = (1, 0, *, *)$ と表現される． CSPF の計算方法については，文献 [2]， [5] を見られたい．

2.1.3 許容関数集合による論理回路の形状変換

トランスダクション法の利点として，許容関数を用いた冗長部分の除去の他に，回路の結線のつなぎ換え，素子の置き換えなどが行えることがあげられる．以下では，本論文で用いる許容関数集合による回路の結線のつなぎ換えの手法について述べる．

［定理 1］ 次の条件が成り立てば，入力端子あるいは素子 v_i の出力を素子 v_j の入力へ接続可能である [2].

（1） v_i を v_j へ接続後の v_j が実現する関数を $f'(v_j)$ とする．このとき，

$$f'(v_j) \in G(v_j)$$

を満足する．

（2） v_j から v_i へ至るパスが存在しない．

［定理 2］ 次の条件が成り立てば，入力端子あるいは素子 v_i の出力を素子 v_j の入力から切断可能である [2].

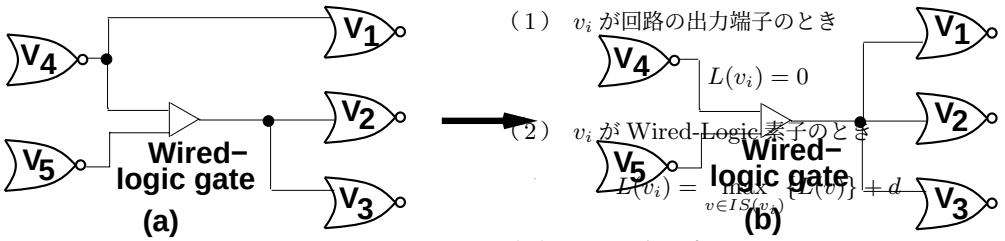
> v_i を v_j から切断後の v_j が実現する関数を $f'(v_j)$ とする．このとき，
>
> $$f'(v_j) \in G(v_j)$$
>
> を満足する．

定理 1により接続可能な結線をつなぐことによって冗長度を増やし，定理 2により切断可能なものを切り，回路の形状を変えながら設計改良を行う手法を C/DC(Connectable/Disconnectable) と呼ぶ [2].

2.2 Wired-Logic の基本的性質

Wired-Logic は，二つ以上の素子の出力を接続してまとめることにより， AND や OR の論理を実現するものである． Wired-Logic は， NOR 素子や NAND 素子等を含む回路で通常使用することが可能で，特に LSI 素子で用いると有用である． OR 素子や NOR 素子への入力線は， Wired-OR で， AND 素子や NAND 素子への入力線は， Wired-AND でまとめると素子への入力線数を減らすことが可能である [3].

本論文では， Wired-Logic により結線をまとめたものを仮想の素子として， Wired-Logic 素子と呼び，これを通常の素子と同等に扱って回路変換を行う．そのた



6.7cm

図 1　Wired-Logic 素子の制約
Fig. 1　Restriction of Wired-Logic Gate.

め，Wired-Logic 素子の使用には以下のような制限を行う．

［定義 2］　Wired-Logic 素子が，通常の素子と同じように扱われる条件は，以下の 2 つである．

（1）　Wired-Logic 素子に接続されているすべての素子が他の素子や Wired-Logic 素子，あるいは出力端子に接続されていないこと．

（2）　Wired-Logic 素子が，他の Wired-Logic 素子や回路の入力端子や出力端子に接続されていないこと．

例えば，図 1（a）のように，v_4 が v_1 と Wired-Logic 素子に接続されている場合，この回路は，図 1（b）に示される回路と同等である．それは Wired-Logic 素子が，v_4 と v_5 の出力を電気的に v_1 の入力に接続することにより実現されているからである．Wired-Logic 素子を通常の素子と同じように扱えば，このような混乱がおこる．そのため本論文では，Wired-Logic 素子に接続されている素子（この例では v_4 と v_5）が Wired-Logic 素子以外へはつながっていないような場合（この例では図 1（b）の場合）に統一することによって上述したような混乱を避ける．これが条件の一つ目に相当する．また，Wired-Logic 素子は素子の出力を結合することによって実現されているだけなので，Wired-Logic 素子が他の Wired-Logic 素子や回路の入出力端子につながっていれば，それらの本来の論理を変更するおそれがある．条件の二つめはそのようなことがないようにするためである．

Wired-Logic 素子を含んだ回路の段数の計算方法は以下のとおりである．

［定義 3］　一般の素子の遅延時間を 1，Wired-Logic 素子の遅延時間を d(d << 1) とし，素子 v_i の回路の出力側から計算した段数を $L(v_i)$ とすると，$L(v_i)$ は以下のように再帰的に定義される．

（1）　v_i が回路の出力端子のとき

$$L(v_i) = 0$$

（2）　v_i が Wired-Logic 素子のとき

$$L(v_i) = \max_{v \in IS(v_i)} \{L(v)\} + d$$

（3）　v_i が一般の素子のとき

$$L(v_i) = \max_{v \in IS(v_i)} \{L(v)\} + 1$$

そして，回路の段数は

$$\max_{v \in V} \{L(v)\}$$

で与えられる．

ここで V は回路に含まれる素子の集合とする．実際には d の値は Wired-Logic 素子への入力数と出力数の和によって異なる．本論文では簡単のため Wired-Logic にもファンイン制限をつけて，d を d << 1 として 0 とみなす．

Wired-Logic 素子を含んだ回路の結線数の計算方法には，以下の定義を用いる．

［定義 4］　入力線数が k_1，出力線数が k_2 の Wired-Logic 素子は，$(k_1 + k_2 - 1)$ 本の結線で実現できる．これを Wired-Logic 素子のコストとする．

2.3　一般化直列分割

NOR 回路のファンインを減らす方法として直列分割（Serial Duplication）がある [1]．これは，ある素子 v に対して，$IP(v)$ を幾つかのグループに分割し，各グループをそれぞれ直列につながれた NOR 素子対の入力に割当て，その各 NOR 素子対の出力を v の入力とする手法である．この直列分割をより効率良くしたファンイン制限手法として以下に述べる一般化直列分割がある [9]．

直列分割によって $IP(v_e)$ を分割するときあるグループの素子が全て共通の入力 v をもつならば，v をそのグループに割り当てられた NOR 素子対のうち出力側の素子の入力に加えても v_e の出力は変化しない．またそのとき，そのグループに含まれる素子で出力線数の数がただ一つである素子の入力から v を削除しても出力に変化はない．この性質をうまく使うことにより，直列分割に伴う段数や回路のサイズの増加をある程度抑える手法が一般化直列分割である．

例えば図 2（a）のように，直列分割の際のあるグループの素子が回路 A よりすべて共通の入力 x をもってい

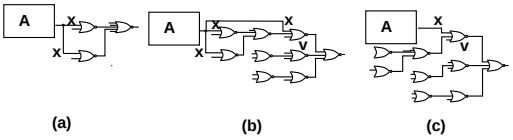


図 2　一般化直列分割の例
Fig. 2　Example of Generalized Serial Duplication.

るような場合，図 2（b）のように，x をそのグループの先の素子の入力に加えても全体の出力は変わらない．またそのとき，図 2（c）のように，そのグループに含まれる素子で，その接続先がただ一つであるものから x を削除しても出力に変化はない．

3. Wired-Logic を併用した回路変換

本章では，トランスダクション法の結線変換を採り入れての Wired-Logic によるファンインの統合可能条件および，Wired-OR 素子によるファンインの統合手法について述べる．

3.1 Wired-Logic によるファンインの統合可能条件

まず以下のことを定義する．

［定義 5］ **un-com-fanout**
$IS(v_i)$ には含まれているが，$IS(v_j)$ には含まれていないような素子の集合を un-com-fanout(v_i, v_j) と呼ぶ．

［定義 6］ **com-fan-out**
$IS(v_i)$ と $IS(v_j)$ の両方に含まれている ような素子の集合を com-fan-out(v_i, v_j) と呼ぶ．

定義 2，定義 5より以下のことが言える．

［定理 3］ v_i と v_j の出力を Wired-Logic 素子でまとめられるとき，v_i と v_j の出力は統合可能であると言い，その条件は以下のとおりである．

> un-com-fanout(v_i, v_j) の全ての要素が v_i と切断可能か v_j と接続可能であり，かつ un-com-fanout(v_j, v_i) の全ての要素が v_j と切断可能か v_i と接続可能である．

例えば図 3（a）のような回路の形状の場合，定義 2より v_i と v_j の出力を Wired-OR 素子でまとめることはできない．しかし，un-com-fanout(v_i, v_j) = $\{v_1\}$, un-com-fanout(v_j, v_i) = $\{v_4\}$ であるので，定理 3より以下の場合は統合可能である．

（1） v_j と v_1 が接続可能，v_i と v_4 が接続可能．
（2） v_i と v_1 が切断可能，v_j と v_4 が切断可能．
（3） v_j と v_1 が接続可能，v_j と v_4 が切断可能．

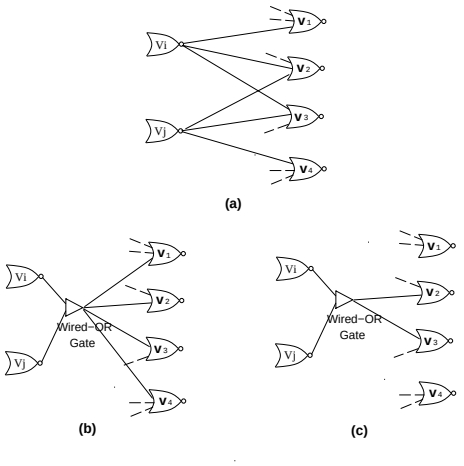


図 3　統合可能である例
Fig. 3　Example of Assemblable.

（4） v_i と v_1 が切断可能，v_i と v_4 が接続可能．

例えば，1. の場合図 3（b）のように，2. の場合図 3（c）のように v_i と v_j の出力を Wired-OR 素子でまとめることができる．

3.2 Wired-OR 素子によるファンインの統合手法

v_i と v_j の出力を Wired-OR 素子でまとめる手続き assemble の大略を以下に示す．この方法は，定義 2の条件を満たすようになっている．ここで，v_i, v_j は，NOR 素子または，Wired-OR 素子でそれらの出力が統合可能であるとする．v_i, v_j の出力のうちで切断可能な結線はあらかじめ削除しておくものとする．

手続き assemble

v_i と v_j がともに NOR 素子の場合
step1　新しく Wired-OR 素子を作り，それを w とする．
step2　w の出力端子を com-fanout(v_i, v_j) と un-com-fanout(v_i, v_j) と un-com-fanout(v_j, v_i) に含まれる素子につなぐ．
step3　v_i, v_j の出力端子を w に接続する．
step4　v_i, v_j の出力のうちで，w の入力となっているもの以外を切断して終了．

v_i と v_j のどちらかが Wired-OR 素子の場合
step1　Wired-OR 素子の方を w，そうでない方を v とする．
step2　un-com-fanout(v, w) に含まれる素子に w の出力端子を接続する．

step3　v の出力端子を w に接続する．
step4　v の出力のうちで，w の入力となっているもの以外を切断して終了．
v_i と v_j がともに Wired-OR 素子の場合
step1　v_j の出力端子を un-com-fanout(v_i, v_j) に含まれる素子の入力端子に接続する．
step2　v_i の入力となっている素子で v_j の入力となっていない素子の出力端子を v_j に接続する．
step3　v_i を削除して終了．
v_i と v_j がともに NOR 素子の場合，ファンイン数が 2 の Wired-OR 素子ができるが，この Wired-OR 素子と他の Wired-OR 素子や NOR 素子の出力を統合することによって，使用する Wired-OR 素子のファンイン数は多くなる．ただし，Wired-OR 素子の入力線数と出力線数の和が多くなり過ぎると Wired-OR 素子の遅延が大きくなるので，上記の手続きは Wired-OR 素子の入力線数と出力線数の和がある一定値を越えない様に制限しておこなう必要がある．

次に，Wired-OR 素子を用いて素子 v の入力線数を可能な限りファンイン制限を満たすように減らしていく手続き fanin-rest の大略を以下に示す．

手続き fanin-rest

step1　v がファンイン制限を満たしていれば，終了．そうでなければ，step2 へ．
step2　v の入力のうちで，統合可能な素子の出力を手続き assemble で統合して，step1 へ．統合可能な素子が存在しなければ，終了．
step2 の手続き assemble において統合可能な素子が複数ある場合には，その中から v_i と v_j の選び方によって最終的な結果が変わってくると考えられる．ただ，本稿ではこの選び方については特に考慮に入れずに変換を行っている．

3.3　統合可能でない場合の Wired-OR 素子での統合

ある素子の入力となっている素子同士が統合可能でないため，前述した手続き fanin-rest では，効率良くファンイン制限が行えない場合がある．しかし，素子 v_i と v_j の出力が統合可能でない場合でも，素子 v_i と v_j が入力端子でも出力端子でもなければ，素子を付け加えることによって，Wired-OR 素子により v_i と v_j の出力を統合できる．この場合，余分な素子を付け加えることになるが，段数を増やすことなくファンイン制限を行うことができる．

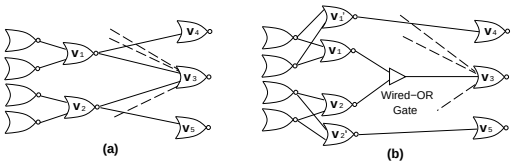


図 4　統合可能でない場合の Wired-OR での統合例
Fig. 4　Example of Assemble with Wired-OR in case of Unassemblable.

例えば，図 4（a）において，v_3 がファンイン制限を越しているが v_1 と v_2 の出力が統合可能でないときには，入力が v_1 と同じである v_1'，v_2 と同じである v_2' をつくることにより図 4（b）のように Wired-OR 素子で v_3 の入力をまとめることが可能となる．このような変換が有効な条件は，v_3 がファンイン制限を越していて，かつファンイン制限を施すと最長パスとなるような場合である．

v_i と v_j の出力が統合可能でない場合でも，素子を付け加えることによって統合する手続き assemble-with-copy の大略を以下に示す．

手続き assemble-with-copy

step1　un-com-fanout(v_i, v_j) の要素の中で，v_i と切断可能でなく，v_j と接続可能でない素子を obstacle(v_i) の要素とする．
step2　un-com-fanout(v_j, v_i) の要素の中で，v_j と切断可能でなく，v_i と接続可能でない素子を obstacle(v_j) の要素とする．
step3　obstacle(v_i) が空集合でなければ，v_i と入力が共通な素子 v_i' をつくり，v_i' の出力端子を obstacle(v_i) の要素につなぎ，v_i の出力のうちで，obstacle(v_i) の要素の入力となっているものを切る．
step4　obstacle(v_j) が空集合でなければ，v_j と入力が共通な素子 v_j' をつくり，v_j' の出力端子を obstacle(v_j) の要素につなぎ，v_j の出力のうちで，obstacle(v_j) の要素の入力となっているものを切る．
step5　v_i と v_j に対して，手続き assemble を適用する．
ここで，obstacle(v_i) は，v_i の出力がつながれている素子の中で，v_i と v_j の出力を Wired-OR 素子でまとめるための条件（定理 3）を満たさない原因となっている素子の集合を表す．obstacle(v_j) も同様である．

3.4 Wired-OR 素子を併用したトランスダクション法

トランスダクション法は初期回路として与えられた回路をもとに最適化を行なう手法であるため，初期回路の性質によって最適化後の結果が大きく左右され，局所解に落ち込んでしまうことも稀ではない．特に，テクノロジマッピングを考慮し，ファンイン制限を行った上でのトランスダクション法の適用は，更に回路変換の可能性を狭め，最適化結果が局所解に陥る要因となっていると考えられる．そこで [10] では，ファンイン制限をしないでトランスダクション法を施すことによって回路変形の自由度を変化させながら局所解からなるべく抜け出せるようにし，最終的に一般化直列分割を用いてファンイン制限を行う手法を提案している．この手法では，平均して 65% の回路では通常のトランスダクション法よりも良い結果が得られている．この手法の概要を以下に，手続きファンイン制限付トランスダクション法として述べる．

手続き ファンイン制限付トランスダクション法

step1　この手続きの 2 回目以降のループの場合，前回のループの実行の際よりコストの改善が見られないときは終了する．そうでないなら， step2 へ.
step2　トランスダクション法の C/DC をファンイン制限を行わずに実行する.
step3　ファンイン制限を満たしていない素子に対してファンイン制限を施し， step1 へ.

[10] では， step3 のファンイン制限の際に一般化直列分割を用いているため，ファンイン制限を一般化直列分割で行う際にコストが増加することの方がファンイン制限をなくしてトランスダクション法を行うことによりコストが減少することより大きくなるような場合には逆に結果が悪くなっている．そこで，本論文ではこの [10] で提案された手法においてファンイン制限を行う際に Wired-Logic を用いる手法を提案し，その実験結果を 4 章で述べる.

4. 実験結果および考察

4.1 各種の回路変換手法

3 章で述べた手法を用いて，以下のような五つの回路変換の手法を C 言語を用いて実現し結果の比較を行った．なお，プログラム中での関数表現には，京都大学工学部矢島研究室で開発された SBDD パッケージを使用している．なお， NOR 素子の入力線数は 4 に制限して実験を行った.

手法 1　一般化直列分割のみを行う.
手法 2　ファンイン制限を満たしていない素子に対して手続き fanin-rest を行った後に一般化直列分割を行う．Wired-OR 素子の入力線数も 4 に制限し， Wired-OR 素子の遅延 $d = 0$ とした.
手法 3　手法 2 において，手続き fainin-rest の中で，手続き assemble の代わりに手続き assemble-with-copy を行う.
手法 4　ファンイン制限付トランスダクション法においてファンイン制限を手法 1 で行う.
手法 5　ファンイン制限付トランスダクション法においてファンイン制限手法を手法 2 で行う.

表 2　手法 4, 5 の結果（素子数 / 結線数 / 段数 /W-OR 数）
Table 2　Results of the method 4 and 5 (gates/connections/levels/W-ORs).

回路名	手法 4	手法 5
5xp1	98/269/9	80/252/7/3
Z9s	140/338/15	77/275/17/6
adr4	123/363/9	86/289/8/8
alu2	106/254/9	64/211/9/2
alu3	93/241/9	69/218/8/4
apla	118/288/9	69/239/7/3
con1	17/43/5	15/41/4/1
dc2	102/263/10	58/217/9/5
f51m	133/359/13	102/328/14/7
misex1	32/88/8	30/84/8/1
mlp4	414/1017/19	175/777/18/22
radd	105/282/9	76/270/12/8
risk	91/222/6	59/190/5/4
root	121/304/12	67/270/9/7
sao2	176/444/27	104/386/29/10
sex	49/125/5	43/119/5/2
sqn	113/291/12	73/250/10/5
sqr6	104/253/9	68/217/8/8
z4	95/269/7	62/215/8/8

4.2 実験結果

文献 [9] の手法により生成した MCNC ベンチマーク回路に対して， 4.1で述べた手法による回路変換を行った．手法 1，手法 2，手法 3 の比較を表 1に示す．表 1では，手法 1 の欄には，左から素子数，結線数，段数を，手法 2，手法 3 の欄には，左から素子数，結線数，段数， Wired-OR 素子数を示している．また，段数改良の欄は手法 1 に対して，手法 2 または手法 3 でどれだけ段数が減っているかの比較を示している．また，手法 4 と手法 5 の比較を表 2に示す．手法 4 の欄には，左から素子数，結線数，段数を，手法 5 の欄には，左から素子

表 1　手法 1，2，3 の結果（素子数 / 結線数 / 段数 /W-OR 数）
Table 1　Results of the method 1, 2 and 3(gates/connections/levels/W-ORs).

回路名	手法 1	手法 2	手法 3	段数改良
5xp1	126/332/7	93/338/4/16	94/341/4/13	7→4
Z9s	145/387/13	284/787/4/30	259/752/4/2	13→4
adr4	158/466/9	146/519/7/23	110/403/4/9	9→4
alu2	199/518/9	234/602/7/16	234/602/7/13	9→7
alu3	173/456/9	223/568/7/15	223/568/7/13	9→7
apla	244/568/9	237/573/7/7	233/566/7/9	9→7
con1	24/67/7	22/65/5/1	22/65/5/1	7→5
dc2	151/398/9	158/447/7/19	154/457/7/16	9→7
f51m	155/440/9	120/426/7/20	122/438/7/12	9→7
misex1	70/189/9	63/193/7/4	65/199/7/5	9→7
mlp4	536/1539/9	616/1782/9/78	507/1650/7/37	9→7
radd	138/411/9	124/419/7/18	110/403/4/9	9→4
risk	135/328/7	129/326/7/2	127/335/7/2	7→7
root	212/538/11	239/645/9/27	204/593/7/13	11→7
sao2	457/1179/11	483/1237/9/41	545/1394/7/22	11→7
sex	82/202/7	84/207/7/1	88/221/7/2	7→7
sqn	181/471/9	159/502/9/24	165/520/7/15	9→7
sqr6	130/339/7	112 /381/5/21	121/402/5/15	7→5
z4	105/314/7	81/296/5/14	69/253/4/7	7→4

数，結線数，段数，Wired-OR 素子数を示している．なお，手法 1 と手法 4 の欄は Wired-Or 素子数は 0 なので省いている．

4.3　考　　察

Wired-OR 素子を併用しない手法 1 と Wired-OR 素子を併用する手法 2 と手法 3 を比較してみるとほとんどの場合において，手法 2 と手法 3 は特に段数について良い結果を出していることがわかる．しかし，sex などでは，Wired-OR 素子を併用しない手法の方が素子数や結線数などが少なくなっているものがある．これは，Wired-OR 素子を使用することによってその後の回路変換の自由度が減るため，一般化直列分割を効率良く行うことができなくなり，その結果素子数や結線数が増加する場合があるからだと考えられる．しかし，Wired-OR 素子を用いれば，たとえ段数が減らない場合でも，NOR 素子の平均入力数を減らすことができているのは明らかである．この結果より，特に段数について，回路合成に Wired-Logic を併用することが有効であることがわかる．

次に手法 2 と手法 3 の比較であるが，手法 3 は手法 2 に比べてほとんどの場合で特に段数については結果が良くなっている．しかし，手法 3 は余分な素子を付け加える手法のため，素子数や結線数などが手法 2 に比べて若干多くなっている場合がある．

また，手法 3 を行えば，ほとんど段数を増やすことなくファンイン制限を行えるはずであるが，中にはそれほど段数が減っていない例もある．この原因は初期回路が 3 段のため，回路の入力端子とつながっている入力線が多く，そのような入力線をまとめる際には assemble-with-copy が行えないからだと考えられる．しかし，入力線が回路の入力端子につながっていない素子ならこの手法で段数を増やさずにファンイン制限ができるため，この手法も有効である場合が多いと思われる．

最後に手法 4 と手法 5 の比較であるが，この場合でもおおむね，Wired-OR 素子を併用している手法 5 の方が段数について結果が良いことが示されている．ただ回路によっては手法 4 の結果の方が少ない段数になっている場合もある．これは手法 1 と手法 2 の比較の場合と同じように，Wired-OR 素子を使用することによってその後の回路変換の自由度が減り，最終結果が悪くなる場合があるのだと考えられる．どのような場合に Wired-OR 素子を使用したら，結果が悪くなるかを考察することは今後の課題である．

5.　む　す　び

本論文では，トランスダクション法における接続可能 / 切断可能の概念を利用することによって結線の形状のみから判断するよりもより多くの場合に Wired-Logic を使用することができる条件を提案し，Wired-OR 素子を併用した回路変換について述べた．Wired-Logic は，通常の素子に比べて遅延が非常に小さいという性質があるため，これを回路変換に用いることは有用である

と考えられる．本論文では，素子として， NOR 素子と Wired-OR 素子のみを扱ったが，他の素子への拡張は容易である．実際に， Wired-OR 素子を併用した回路変換手法を実現し，評価を行ったところほとんどの回路において，特に段数については， Wired-OR 素子の有用性が示された．

また，ファンイン制限付トランスダクション法と Wired-Logic を用いた回路変換を組み合わせることによって，効率が良い回路変換が行えることも示された．本論文では，トランスダクション法の C/DC(Connectable/Disconnectable) と呼ばれる手法と併用した手法について実験を行ったが，トランスダクション法の他の手法との組合せは今後の課題である．

また，今回用いた回路は 3 段回路ばかりだったため，一般的でない点があるかもしれない．とくに，素子に対する入力が回路の入力端子からのものが多かったため，Wired-Logic が使用できないところが多かったことが回路変換の結果が思ったほど良くなっていない一つの要因であると考えられる．より一般的な段数の多い回路に対する性能評価は，今後の課題である．

謝辞　有益な御助言を下さった九州大学工学部助手澤田直氏，三菱電機の高田秀志氏に深謝致します． SBDD パッケージの使用を快諾して頂いた京都大学工学部矢島脩三教授ならびに矢島研究室の皆様に深謝致します．なお，本研究は文部省の科研国際共同研究による援助を受けています．

文　　献

[1] H.Lee and E.S.Davidson, “A transform for NAND network design,” *IEEE Trans. Comput.,* Vol.c-21, No.1, pp.12-20 (1972).

[2] 上林弥彦, 室賀三郎, “Permissible Function による論理回路の簡単化,” 電子通信学会研究報告, AL73-13 (1973).

[3] Y.Kambayashi and S.Muroga, “Properties of wired logic,” *IEEE Trans. Comput.,* Vol.c-35, No 6,pp.550-563 (1986).

[4] Y.Matsunaga and M.Fujita, “Multi-level logic optimization using binary decision diagrams,” *Proceedings of International Conference of Computer Aided Design*, pp.556-559 (1989).

[5] 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.38, No.10, pp.1404-1423 (1989).

[6] H.Sato, Y.Yasue, Y.Matsunaga and M.Fujita, “Boolean resubstitution with permissible functions and binary decision diagrams,” *Proceedings of 27th Design Automatation Conference*, pp.284-289 (1990).

[7] Y.Kambayashi, H.C.Lai and S.Muroga, ”Pattern-oriented transformations of NOR networks,” UIUCDCS-R-90-1573, Dep. Comput. Sci., Univ.of Illinois (1990).

[8] T.Fujimoto, H.Honjo and T.Kambe, “Large network optimization using maximal CSPF,” *Report of Technical Group on Design Automation IPS Japan*, 91-DA-60 ,pp.123-130 (1991).

[9] S.Sawada, Y.Kambayashi and S.Muroga, “Generation of fan-in restricted initial networks for transduction method, *Proc. the Synthesis and Simulation Meeting and International Interchange, SASIMI '92*, pp. 36-45 (1992).

[10] 高田秀志, 石垣博康, 上林弥彦, “効率の良い直列分解多段化に基づくファンイン制限付トランスダクション法,” 情報処理学会第４６回全国大会講演論文集, 8M-4 (1993).

（平成１年１月１日受付，７年７月 11 日再受付）

山下　　茂

平５京大・工・情報卒．平７同大大学院修士課程修了．同年 NTT コミュニケーション科学研究所入所．論理回路の設計の研究に従事．情報処理学会会員．

上林　彌彦　（正員）

昭 40 京大・工・電子卒．昭 45 同大大学院博士課程了．工博．京大助手，イリノイ大リサーチアソシエイト，京大助教授，九大教授を経て平２より京大工学部教授．論理回路，オートマトン，データベースの研究に従事．カナダマッギル大，クウェート大，中国武漢大の客員教授．「Database-A Bibliography」(Computer Science Press)，「データベース」（昭晃堂）など．昭 50 本会米沢賞，昭 58 丹羽賞，昭 62 本会著述賞，平 7ACM SIGMOD 貢献賞．情報処理学会，日本ソフトウェア科学会， ACM 各会員， IEEE シニア会員．

室賀　三郎　（正員）

昭 22 東大・工・電気卒．日本国有鉄道研究所，電波庁電波管理管理局，日本電信電話公社電気通信研究所， IBM T. J. WATSON 研究所を経て，昭 39 よりイリノイ大コンピュータ・サイエンス科教授（電気工学科兼任）．音声認識，多重化通信方式，情報理論，閾値論理，論理回路の自動合成の研究に従事． Threshold Logic and Its Applications, Logic Desigin and Switching Theory, VLSI System Design など． IEEE Fellow, ACM など各会員．