設 計 自 動 化 81－7
(1996. 10. 18)

レイアウト設計を考慮した低消費電力化テクノロジーマッピング

室伏 真佐子　石岡 尚　村方 正美
株式会社 東芝 半導体設計評価技術センター
〒210 川崎市幸区堀川町 580-1
E-mail: murofusi@dad.eec.toshiba.co.jp

LSIの設計において、高集積化、高速化と共に、低消費電力化が大きな課題になってきている。本稿では、低消費電力化をターゲットとする新しいテクノロジーマッピングの手法について述べる。自動配置後にテクノロジーマピングを部分的に適用することで、論理最適化をはかる。配置情報を有効に利用することで、適切に配線経路と配線長を見積り、配線容量による消費電力を精度良く算出する。テクノロジー再マッピング時に、算出した配線容量と共に、スイッチング確率に基づいて消費電力を算出し、DAGマッチングを行う。プログラムを試作し、数万セル規模の大規模データに適用したところ、組合せ回路部の消費電力について、10-20% 削減することを確認した。

論理合成、テクノロジーマッピング、レイアウト、自動配置、CAD、LSI、ASIC

A Layout Driven Resynthesis Method

Masako Murofushi　Tadashi Ishioka　Masami Murakata
Semiconductor DA & Test Engineering Center, Toshiba Corporation
580-1, Horikawa-cho, Saiwai-ku, Kawasaki 210, Japan
E-mail: murofusi@dad.eec.toshiba.co.jp

A new technology re-mapping method which estimates wire capacitance arcurately with placement infomation is proposed. Experimental results show that this method reduces the power consumption of the circuits, which gate size are over 10,000 cells, 10 % further than the original circuits.

logic synthesis, placemenet, technology mapping, ASIC, LSI, wire capasitance

1 はじめに

LSIの設計において、高集積化、高速化ともに、低消費電力が大きな課題になってきている。本稿では、低消費電力化をターゲットとする、新しいテクノロジーマッピングの手法について述べる。

テクノロジーマッピングの方法としては、DAGマッチング手法[1]が広く知られている。これは、あらかじめ使用できる回路素子の論理機能をNAND2(ND2)とINVERTER(IV)のDAG(Directed Acyclic Graph)として登録しておき、おなじく、ND2とIVのDAGとして表現されている論理回路のどの部分にどの回路素子を割り当てたら良いか、DAG同士のマッチングを行い、評価関数に従って、最適なマッチングを決定していく手法である。

従来の低消費電力化のテクノロジーマッピングとしては、DAGマッチングの評価関数に、消費電力の項を盛り込む方法[2]、DAGマッチングを行う前に、論理回路のND2とIVで表現されたDAGそのものを消費電力の評価関数に基づいて、変更する方法[3]、消費電力低減の可能性のある回路部分を抽出し、その部分を再論理合成する手法[14]等、多くの方法が知られている。しかしながら、これらの方法は、レイアウトを行う前の回路に適用されるため、組合せ回路部のの30-60%を占める配線容量の充放電による消費電力の影響を正確に見積もることが出来ず、十分な最適化をおこなうことが出来ない。

一方、テクノロジマッピング時(一般的に論理合成時においても)における配線の影響の重要性も、認識されており、論理合成と、自動配置を同時におこなう方法[4, 5, 6, 13]、配置後に、配線ディレイを改善するために回路素子の駆動力を変更する方法、ファンアウト容量をバランスさせるためにバッファを挿入する方法などが提案されている[7, 8, 9]しかしながら、論理合成と自動配置を同時に行う方法では、複合セルなどを用いないプリミティブな回路を自動配置が直接扱うため、自動配置の扱うセル数が膨大になり、自動配置自体の性能が十分にでない[4, 13]、FPGAなどの、チップ内の素子数の少ない特殊なレイアウト構造のための処理である、などの問題点があった。大規模で、一般的なASICの設計フローには向いていなかった[5, 6]。また、低消費電力化を狙ったものではなかった。配置後に回路の駆動力を変更したり、バッファを挿入する方法を、パワー削減に応用する場合は、低消費電力ライブラリが準備されていない場合には、十分な低消費電力化が行えない。

本手法であるLDR(Layout Driven Resynthesis)は、自動配置後にチップ内の部分回路を再テクノロジーマッピングすることで、自動配置そのものの性能を損なわず、配置結果の情報を用いて高精度に配線容量を見積り、消費電力を削減する。DAGマッチング内部でも、配置情報を適切に用いるため、DAGノードの仮配置を行う。その結果から、マッチング後の素子の、配置位置を決めていく(図1参照)。

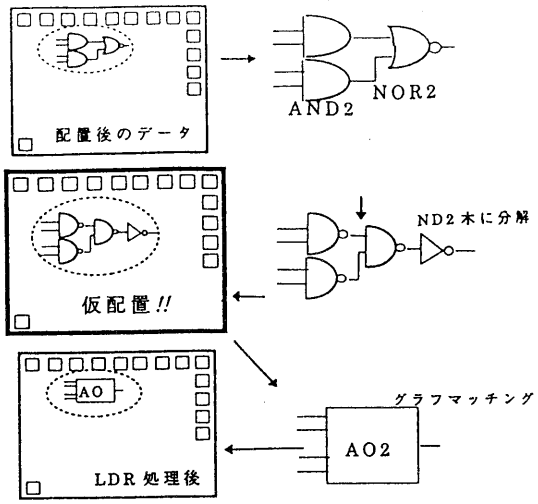

図1: LDRの処理概要

大規模回路に適用したところ、組合せ回路部で15%前後の消費電力の改善を行うことが出来た。

以下では、2章で、LSIの設計フローの中でのLDRの位置付け、3章でLDRの詳細について(特に配置情報との組み合わせ方について)、4章で計算機実験結果について、5章で、結論と今後の課題について述べる。

2 設計フローの中でのLDRの位置付け

レイアウト設計と、論理設計で起こりがちなことは、レイアウト前の論理設計では、仮想配線長をもとにしたパフォーマンスの予測、面積の予測を行っているので、実際のレイアウト後での評価とずれが生じ、設計を何度もやり直すことである。レイアウト設計と、論理設計の間に大きなフィードバックループが存在する(図2(a))。

LDRは、自動配置後、回路の一部に対してテクノロジーマッピングをやり直し、接続情報の最適化をおこなう。最適化の指標としては、ディレイ、消費電力、配線混雑度、面積など、パフォーマンス、面積に関するあらゆるものを対象とすることが出来る。つまり、消費電力その他を最適化するための大きなフィード

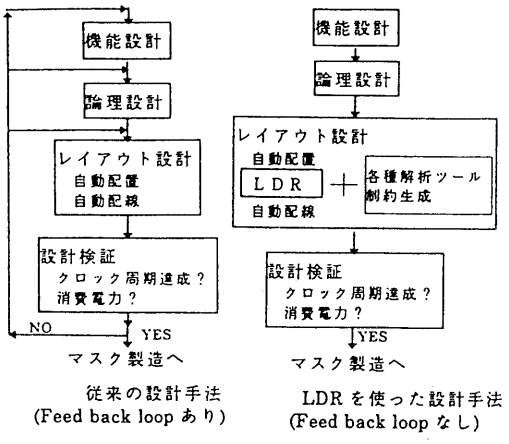

図 2: 従来の設計手法例と LDR を使った設計手法例

バックループを解消し、設計期間を短縮することが可能である。また。配置結果を考慮しているので、高精度で各評価値を得て、高精度な最適化が可能となる (図 2(b))。

3 レイアウトドリブンリシンセシス

3.1 概要

LDR は、DAG マッチングをもとにした、テクノロジーマッピングである。低消費電力化を狙っている。

まずチップ内回路のうち、LDR での回路変換対象を抽出する。その後、テクノロジマッピングを行い、ディレイ最適化、消費電力最小化を施した新しい回路を生成する。最後にその結果を再配置する。テクノロジマッピング処理内では、選ばれた対象回路を ND2 と IV の DAG に分解する。その DAG をチップ内で仮配置する。こうすることで、DAG マッチング内部でも、配置情報を更新しながら、正確に配線容量を見積もることが可能となり、ひいては、正確に消費電力を見積もることが出来、質の良い、最適化が可能となる。最後に、このようにして新規生成された回路をもともとの回路にはめ込む。この処理を、チップ内から LDR 対象回路を選ぶことが出来なくなるまで、繰り返す。

具体的なアルゴリズムとしては、消費電力の高いネットの順に、

Step 1 Selection

Step 2 DAG Matching with Placement

Step 2.1 DAG Decomposition

Step 2.2 DAG Placement

Step 2.3 DAG Matching

Step 3 Replace Circuit

を繰り返す。

以下では、3.2節で消費電力の計算方法について、3.3節で LDR の詳細 (上記 Step 1,2 の部分) について述べる。3.3.1では、低消費電力化指向の対象回路の選び方、3.3.2では、DAG マッチングについて述べる。特に、DAG マッチング内部での DAG の配置、配置情報の更新の仕方、配置情報の使い方について、詳しく述べる。回路を再配置する部分 (Step 3) に関しては、配置の詳細に関するので、本稿では省略する。

3.2 消費電力の計算方法

LDR では、以下の式で表される消費電力計算方法を用いている。

$$P = C \times V^2 \times f_{clk} \times S_w \qquad (1)$$

ただし、P: 消費電力、C: 容量、V: 動作電圧、f_{clk}: クロック周波数 S_w: その容量のスイッチング確率

スイッチング確率は、静的な確率計算による計算方法により求めている。この方法では、信号の空間的な依存関係はもとより、時間的な依存関係も考慮して、スイッチング確率を求めることが出来る [10, 11]。

3.3 LDR の処理フロー

3.3.1 対象回路の選択

再テクノロジマッピングすることで消費電力の改善が得られるように、消費電力の高いネットを中心に、これを取り囲んで、logic cone を切り出すようにした。具体的には、ネットを、消費電力の大きい順にソートする。ネット毎に、当該ネットから、m 段出力側に信号をたどり、たどり着いたところのセルをルートとして、ルートから、n 段入力側にある回路を LDR の対象回路とする (図 3参照)。こうすることで、消費電力の高いところが優先的に大きな logic cone を切り出せるようになり、つまり、消費電力が大きいところがより良く改善されるようになる。

また、このアルゴリズムからわかるように、対象回路を選択する場合は、物理的な位置の近さによる制限は設けていない。LDR 内で位置の最適化機能も備えているからである。このことにより、論理接続情報の最適化をよりはかることが出来る。しかしながら、もし、フロアプランなどの制約を守りたい時には、そのようにすることも可能である。

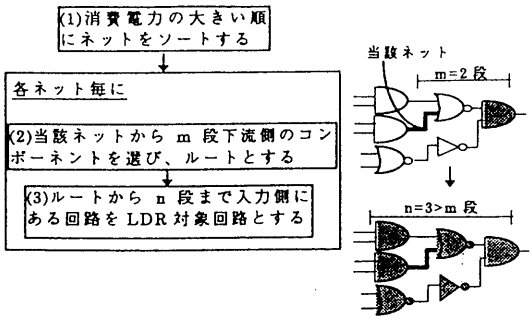

図 3: LDR 対象回路の選び方

3.3.2 配置情報を考慮した DAG マッチング

LDR で使用している DAG マッチングの疑似コードを表 3.3.2に示す。さらに、図 4に DAG の分解と再合成の例を示す。図 4についての詳細は、DAG への分解と最適マッチングの項で述べる。ここでは、表 3.3.2について説明する。

DAG マッチング内部では、まず回路を DAG に分解する (decompose_circuit()) 。その後、DAG の各ノードを、外部接続しているセルの位置を考慮して配置する (place_DAG())。DAG マッチングを利用したテクノロジーマッピングでは、DAG 内の各 tree をトレースしながら (Preorder_traversal() , Postorder_traversal())、ダイナミックプログラミングの手法を用いる [1]。

LDR では、消費電力削減を効果的に行うために、配置情報を使って精度良く配線容量を見積りながら、DAG マッチングを行う。そのために、DAG マッチング内部で、必要な配置情報を適宜生成する。実際に特定のノードにマッチする最適なセルを得る部分 (Get_best_match()) では、得られたセルの配置位置を仮に決めている (get_position())。その後、生成した情報を、配線の容量を見積もる (get_loadcapa_for_output() , get_loadcapa_for_input()) ところで利用している。以上、下線を引いた 4 つの関数が LDR で特徴的な処理となっている。

```
DAG_Matching_with_placement(circuit){
  dag = decompose_circuit(circuit);
  place_DAG(dag);
  while ( tree in dag ){
        Preorder_travelsal(tree);
        Postorder_travelsal(tree);}
}
Preorder_travelsal(tree){
  order1 = oder_node_from_inputs_to_output(tree);
  while ( node in order1 ){
        Get_best_match(node);}
}
Postorder_travelsal(tree){
  order2 = oder_node_from_output_to_inputs(tree);
  while ( node in order2 ){
        create_best_match_cell(node);
        connect_best_match_cell(node);}
}
Get_best_match(node){
  pre_value = (MAXFLOAT);
  best_match = NULL;
  while ( inputs = get_new_inputs(node)! = NULL ){
        cell = get_cell(node, inputs);
        get_position(cell);
        get_loadcapa_for_output(cell);
        get_loadcapa_for_inputs(cell);
        value = evaluate_cell(cell);
        if (pre_value > value ) {
          best_match = cell;
          pre_value = value;}}
}
```

表 3.3.2: LDR の DAG マッチングの疑似コード

DAG への分解 まず、LDR の対象となる回路を、ND2/IV のノードからなる木に分解する (decompose_circuit_to_DAG())。(図 4 (a) と (b) 参照) そのとき、ファンアウトが 2 以上のネットと DAG の外部の端子につながるネットについては、特別にネットノードとよぶノードを発生する。図 4(b) 内で、○で表しているノードである。DAG マッチングは、ファンアウトが 2 以上のノードが DAG 内部にあると適用できないためである。また、LDR の対象回路に接続しており、対象回路外にあるセルの位置情報を適切に扱うためでもある。

ネットノードの位置は、そのネットが接続する LDR 対象回路外にあるセルの位置の重心とする。LDR 対象回路外にあるセルの位置は、すでに決まっているので、ネットノードの位置の算出は容易にできる。もし、ネットノードが LDR 対象回路のみに接続しているのであれば、そのネットノードの位置は、以下で述べる ND2/IV ノードの位置決めと同様の方法で決定する。

DAG の配置 得られた ND2/IV 木を、Force Directed Method を使って、配置する (place_DAG())。厳密に配置位置を決めるためには、たとえば、ゲートアレイで

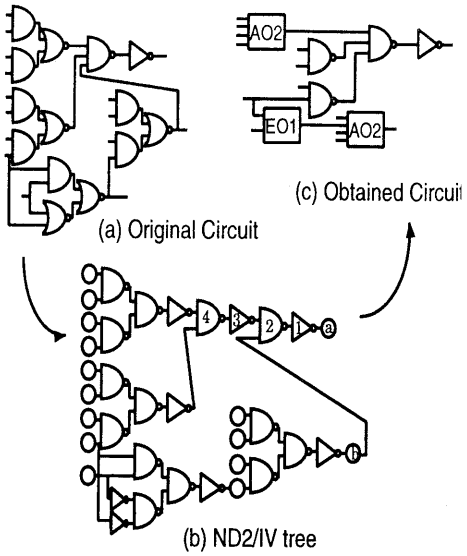

図4: テクノロジー再マッピングの例

は、下地のトランジスタを考慮して、決める必要がある。スタンダードセルでは、セル列に沿って配置する必要がある。また、LDRの対象回路以外の回路部分はチップ内にすでに配置されているので、それらに重ならないように配置する必要がある。ここでは、配置位置を手がかりにして、配線容量を見積もることが目的なので、そのような厳密な配置位置を得ることはせず、以下のような計算式により、仮の配置位置を算出する。厳密な配置位置は、DAGマッチングの後で、ライブラリ内の回路素子が割り付けられた後でおこなう。位置の決まっていない各ノードvのX座標について、

$$\sum_{v_c \in V} (X_v - X_{v_c}) = 0 \qquad (2)$$

という式を立てる。(Y座標についても同様の式を立てることができ、X座標とは、独立に解くことができるので、ここでは省略する。)ただし、Vは位置を決めるべきノードの集合、ノードv_cとvの間には枝が存在する(いいかえるとノードv_cとvは接続している)。

座標位置の決まっていないものを式の右辺に、座標位置の決まっているものを式の左辺にまとめると、

$$A \cdot X = B \qquad (3)$$

というようにマトリクスで表現することが出来る。位置を決めるべきノードの数をnとすると、Aは$n \times n$次元の2次元配列、BとXはn次元の列ベクトルである。Aは、DAGの接続行列PとP^tの積で表される($A = P \times P^t$)。$P \times P^t$の行列式はDAG内の木の数なので[15]、必ず0より大きい。よって、Aの行列式も0より大きく、必ず逆行列が存在する。よって、Xは、

$$X = A^{-1} \cdot B \qquad (4)$$

で、求めることが出来る。

例1: 図4のノード1,2,3について式を立てると、

$$\begin{aligned} x_1 - x_a + x_1 - x_2 &= 0 \\ x_2 - x_1 + x_2 - x_3 + x_2 - x_b &= 0 \\ x_3 - x_2 + x_3 - x_4 &= 0 \end{aligned}$$

変形して、

$$\begin{aligned} 2\ x_1 - x_2 &= x_a \\ -\ x_1 + 3\ x_2 - x_3 &= x_b \\ -\ x_2 + 2\ x_3 - x_4 &= 0 \end{aligned}$$

さらに行列表現すると、

$$\begin{bmatrix} 2 & -1 & 0 & 0 \\ -1 & 3 & -1 & 0 \\ 0 & -1 & 2 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix} = \begin{bmatrix} x_a \\ x_b \\ 0 \end{bmatrix}$$

と表現することが出来る。

ノードの数(Xの次元)は、せいぜい数十(20−30)なので、高速に、メモリ量を気にすることなくXを求めることが出来る。

最適マッチングを求める 最適マッチングは、入力側から出力側へ向かってのトレース(Postorder_traversal())する部分で行われる。各ノードの処理順序を決め(order_node_from_inputs_to_output())、その後、各ノードのための最適マッチングを求める(Get_best_match())。この処理は、図4内の(b)から(c)への矢印で表されている。図4(c)の$EO1$はExclusive ORゲート($A \oplus B$)を表し、$AO2$は$\overline{A \cdot B + C \cdot D}$を表している。

最適マッチングを求めるために、当該ノードをルートとする木を次々と変化させ、各々の木にマッチするcellを得る(get_cell())。配置情報を考慮した精度良い消費電力の評価値を得るために、得られたセルの仮の配置位置を決定する(get_position())。この処理は、LDRに特徴的な処理である。セルの仮配置位置を、そのセルを構成するND2/IVの各ノードの位置の重心として求める。(図5参照)

得られたセルの仮配置位置から、そのセルが駆動するネットの容量を求める(get_loadcapa_of_output())。そのセルの出力が接続しているのがND2/IVノードの場合には、セルの位置と接続しているND2/IVの位置のマンハッタン距離として配線長を求め、得られた配線

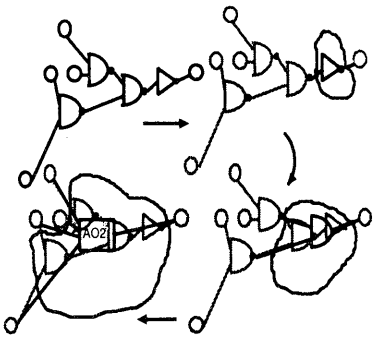

図 5: マッチしたセルの仮位置を得る

長から、配線容量を得る。そのセルの出力が接続しているのがネットノードの場合には、当該セルと、そのネットノードの接続している対象回路より外部のセルの位置、対象回路内の ND2/IV ノードの位置から、階層的にシュタイナーツリーを構成し [12]、配線長を求め、配線容量を得る。階層的にシュタイナーツリーを構成する方法では、実配線長に比べ、5-8% 以内の誤差で、精度良く見積もることが可能である。

最適マッチングを求めるためには、さらに、得られたセルの仮配置位置から、得られたセルの入力ネットの容量を算出し直し、入力側の評価値を更新する必要がある。得られたセルを駆動しているセルの位置は、すでに決定しているので、ネットの配線長を、マンハッタン距離か階層的なシュタイナーツリーを構成する方法で求め、配線容量を得る。

消費電力の評価値　消費電力の評価値は、入力側から消費電力を当該ノードのマッチしたセルまで足したものとして、求める。

$$
\begin{aligned}
value &= evaluate\_cell(cell) \\
&= \sum_{cell\text{の入力ピン}} \{\text{接続する入力側セルの} best\_value \\
&\quad +\text{入力側ネットの消費電力}\} \\
&\quad +\text{当該} cell\text{の消費電力} + \text{出力側ネットの消費電力}
\end{aligned}
$$

4 計算機実験結果

iscas ベンチマークデータと、数万セル規模のデータを使って、LDR の試作評価を行った。当社の 0.5μm デザインルールのゲートアレイのライブラリを用いて、自動配置、および、LDR(再テクノロジーマッピング) を行った。消費電力の評価も、このライブラリのパフォーマンスデータによって測定した。

iscas ベンチマークデータは、組合せ回路のみで構成されている論理合成用のベンチマークデータである。

表 1: iscas データでの実験結果

		消費電流 (mA)		
C880	セル数	総消費電流	IO 部	組合せ回路部
配置後	589	5.22	2.39	0.905
LDR 後	453	4.96(-5%)	2.39	0.703(-22%)

		消費電流 (mA)		
C1908	セル数	総消費電流	IO 部	組合せ回路部
配置後	1204	6.74	2.39	2.20
LDR 後	850	6.22(-7.7%)	2.39	1.78(-19%)

表 2: 大規模データでの実験結果

		消費電流 (mA)		
DATA1	セル数	総消費電流	IO 部	組合せ回路部
配置後	12454	90.5	3.53	22.33
LDR 後	12553	88.4(-2.3%)	3.53	20.29(-9.1%)

		消費電流 (mA)		
DATA2	セル数	総消費電流	IO 部	組合せ回路部
配置後	32657	34.70	0.61	5.73
LDR 後	31879	32.63(-6.7%)	0.61	4.97(-13.3%)

本実験では、iscas データのうち、C880 と C1908 について評価した (表 1参照)。消費電力 (ここでは消費電流) の評価には、静的な確率計算による消費電力解析手法を用いた [10, 11]。LDR を用いることで、組合せ回路内の消費電力は、約 20% 削減された。また、IO バッファを含めたチップ全体でも、5 − 8% 電力が削減されている。さらに、1 − 2 万セル規模のゲートアレイデータ (順序回路データ) を使って LDR を評価した (表 2参照)。LDR を使うことによって、チップ内の組合せ回路部で 10% 前後消費電力が低減した。LDR では、スイッチング確率を考慮して、高いスイッチング確率の部分をセル内部に閉じ込める機能がうまく働き、論理最適化が行われているためと考えられる。

5 終りに

レイアウト結果をもとに消費電力を削減する再テクノロジマッピングツール LDR を開発した。配置情報を利用して、配線経路を見積り正確なパフォーマンス情報をもとに、精度良く論理最適化を行う。

LDRにより、パフォーマンス改善に非常に時間のかかっていた、論理設計とレイアウト設計の繰り返しを解消することが可能である。これにより、短期間でハイパフォーマンスなLSIの設計が可能となる。計算機実験結果による評価では、一般セルによる消費電力を10－20%程度削減することを確認した。

今後ディレイ制約下で消費電力を最小にする手法を確立し、さらに多数のチップ設計に適用していく予定である。

参考文献

[1] Kurt Keutzer : "DAGON: Technology binding and local optimization by DAG matching",*Proc. ACM/IEEE Design Automation Conf.*, pp. 341–347 (1987).

[2] Bill Lin and Hugo De Man : "Low-power driven technology mapping under timing constraints", *Proc. IEEE Intl. Conf. on Computer Design: VLSI in Computers & Processors*, pp. 421–427 (1991).

[3] Chi-Ying Tsui, Massoud Pedram and Alvin M. Despain : "Technology decomposition and mapping targeting low power dessipation", *Proc. ACM/IEEE Design Automation Conf.*, pp. 68–73 (1993).

[4] Massoud Pedram and Narasimha Bhat : "Layout driven technology mapping", *Proc. ACM/IEEE Design Automation Conf.*, pp. 99–105 (1991).

[5] Shih-Cieh Chang, Kwang-Ting Cheng, Nam-Sung Woo and Malgorzata Marek-Sadowska: "Layout driven logic synthesis for FPGAs", *Proc. ACM/IEEE Design Automation Conf.*, pp. 308–313 (1994).

[6] Jingshegn Cong et al. : "Combinibg technology mapping and placement for delay-minimization in FPGA Design", *IEEE Trans. on CAD* 14(9), pp. 1076–1084 (1995).

[7] Lalgudi N. Kannan, Peter R. Suaris and Hong-Gee Fang : "A methodology and algorithms for post-placement delay optimization", *Proc. ACM/IEEE Design Automation Conf.*, pp. 327–332 (1994).

[8] Arnold Ginetti and Daniel Brasen : "Modifying the netlist after placement for performance improvement", *Proc. IEEE Custom Integrated Circuits Conf.*, pp. 9.2.1–9.2.4 (1993).

[9] Aoki,T., Murakta,M., Mitsuhashi,T. and Goto,N. : "Fanout-Tree Restructuring Algorithm for Post-placement Timing Optimization", ASP-DAC , pp. 417–420 (1995).

[10] Taku Uchino, Fumihiro Minami, Takashi Mitsuhashi and Nobuyuki Goto : "Switching activity using boolean approximation method", *Proc. ACM/IEEE Design Automation Conf.*, pp. 20–25 (1995).

[11] Taku Uchino, Fumihiro Minami and Takashi Mitsuhashi : "Switching activity analysis for sequential circuits using boolean approximation method", *Proc. ACM/IEEE Intl. Symp. on Low Power Electronics and Design*, pp. 79–84 (1996).

[12] Mutsunori Igarashi, Masako Murofushi and Masami Murakata, "Timing Diven Placement with an RC wire Delay Model for Sub-Micron CMOS Gate-Arrays", SASIMI '93. (1993).

[13] 田宮 豊："レイアウトを考慮した論理合成", DAシンポジウム, pp. 17–20 (1992).

[14] 上田祐彰, 樹下行三 : "論理最適化手法を用いた消費電力の低減化手法", 信学技報, **VLD95,** No.10, pp. 47–53 (1995).

[15] 伊理正夫ほか："演習グラフ理論", コロナ社, pp. 51–53 (1983).