設 計 自 動 化 83－7
1997. 2. 14)

レイアウトを考慮したディレイ削減テクノロジマッピングシステム

石岡 尚　室伏 真佐子
(株) 東芝 半導体事業本部 半導体設計・評価技術センター 設計自動化開発部

本論文は、配置後の回路データに対して局所的な回路変更を行ないクリティカルパスの遅延を減少させるシステム LDR (Layout Driven Re-Synthesis) に関するものである。このシステムは、クリティカルパスの周辺の回路を選択して、論理的に等価かつ性能の高い回路に変換する。このシステムを配置処理後の 400k ゲートクラスのゲートアレイに対して実行した結果、回路の遅延が 10% 程度改善されることを確認した。

Layout Driven Delay Optimization with Logic Re-Synthesis

Takashi Ishioka　Masako Murofushi
Semiconductor DA & Test Engineering Center, TOSHIBA, Kawasaki, Japan.

In this paper, we propose a method to improve the delay of critical paths by re-synthesis on after-layout circuits. We build the system on the Sparc work station, apply this system on the Gate-Arrays of the 400k-gates class, and obtain improvements of the worst path delay on the order of 10% .

1　はじめに

論理回路は、論理設計及びレイアウト設計という 大きな 2 つの段階を経て設計される。論理設計段階では論理記述を対象とし、2 段論理最適化, 多段論理最適化, テクノロジマッピング[1-3] の大きく三つの処理を経て最適化する。この場合、各々の素子が未配置かつ配線も行われていないため、様々な方法で素子の配置位置及び配線経路を見積もり (仮想配線) ながら最適化をする。しかしながら近年のチップの微細化と大規模化に伴い、回路全体の遅延に対する配線遅延の影響が相対的に増大してきている。このような状況下では、仮想配線による性能評価では精度が不十分であり、論理設計段階で精度良く遅延を見積もることができず最適化が不十分になる場合があった。

一方のレイアウト設計では、論理設計で生成されたネットリストを用いて各セルの配置位置、各ネットの配線経路を確定する。この段階では、セルの配置位置を操作することで配線容量による遅延の最適化を行っている[4-7]。レイアウト設計段階ではネットリストは固定化されているため、論理設計段階で最適化が不十分なネットが入力された場合でもそのまま設計するしかないので、最適な設計ができない場合があった。

これらの問題に対処するために、自動論理合成にレイアウト結果を取り込むアプローチや、レイアウト設計時に回路変更をするアプローチがある。前者については、レイアウト時のパフォーマンスデータを論理合成にフィードバックする手法や、論理合成時の仮配線長による制約をレイアウトに付加する手法があるが処理時間が増大する問題や仮想配線による遅延の精度が十分でないという問題があった。

また後者については、レイアウト途中 (特に配置後) にバッファ挿入やセルの駆動力変更のためにセルを置換する方法[8,9] がある。しかしながら、これらはセルのサイズ変更やバッファを入れるだけでより進んだ回路の改良をするものではなかった。

この論文で紹介する LDR では、配置設計後に回路の一部を抽出して等価な回路で置き換えることで論理の最適化を行う。遅延について最適化する場合、クリティカルパス (遅延が問題となるパス) 周辺の部分回路を選択し、その回

路に対して遅延が減る等価な回路で交換する。交換する回路は、元の回路からパスの段数削減, 複合セルの分解, 複数セルの複合セル化等により遅延が減少するものを再マッピング処理で得て使用している.

論文の残りでは第 2 節で LDR で使用している遅延及び制約情報について、第 3 節で LDR での処理について、実験結果と結論を第 4,5節で示す。

2 LDR の遅延及び制約情報

LDR では、遅延最適化のために回路中のセルやネットの接続状態を変更する。このとき、クリティカルパスとして単純にパスを構成するセルとネットの連鎖を覚えておくだけでは、論理接続を変化させる LDR の処理には向かなかった。そこで、LDR では回路の接続を変更した場合でもクリティカルパスについての情報が保持できるようなデータ構造を採用している。 本節では、この遅延情報と制約について説明する。ただし、あるフリップフロップから別のフリップフロップ迄のセルとネットの連鎖のことをパスと呼び、クリティカルパスとは、 各パスに与えられた遅延の制約を満たしていないか、もしくは満たしていても 制約と近い遅延を持つものものとする。

2.1 LDR の遅延情報

タイミング制約を満たすためにはクリティカルパスに着目して最適化することが重要である。LDR では遅延に関する情報を保持する為に、各素子の端子と端子間の接続である端子間接続とでネットワークを構成し、それら端子に

- パスの始点からその端子までの最大積算遅延 (a)
- 各端子からパスの終点までの最大積算遅延 (b)
- 各端子に隣接する最大遅延パスに属する端子間接続 (c)

の情報を付加している (図 2.1参照)。これにより各端子は$(a)+(b)$により自分を通る最大パス遅延を持つパスについての遅延を、 (c) によりそのパス上の隣接端子を知ることができる。これにより, 各端子を含むパス全てをすぐに知ることはできないものの、隣接する端子をトレースすることで少なくとも各端子の周辺を通るクリティカルなパスについての情報を知ることができる。

図 2.1の各パス始点 (A ～ E) からパス終点までのパスとその信号伝搬遅延は図 2.2 のようになる。ただし、C を始点とするパスは 2 つあるのでそれぞれ C_1, C_2 と表している。また、図中で Offset とあるのは、回路の前置き遅延等による分を示す。これについては 2.1.2 節で詳しく示す。

LDR ではこのようなデータ構造をとることで、パスの一部が変わった場合でも端子の接続関係と、積算遅延情報の更新を容易にしている。さらに、LDR では処理の順序をパスの始点から終点へとたどる方向に限定することで積算遅延情報の更新が必要な範囲を限定し、よりいっそうの高速化をはかっている。これについては 2.1.1節で説明する。

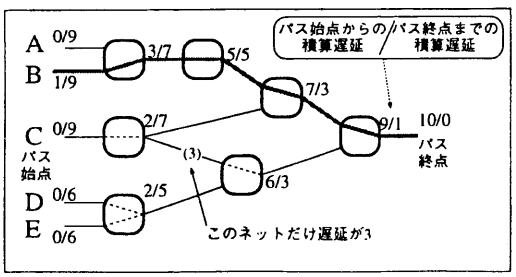

図 2.1: LDR のもつ遅延情報

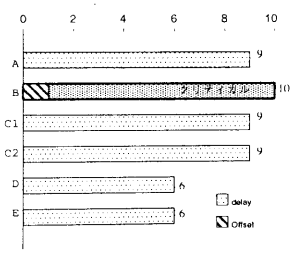

図 2.2: パス遅延の様子

2.1.1 処理順序及び遅延情報の更新

この節では、LDR の処理及び遅延情報の更新について述べる。LDR では、回路中のパスの始点側から順に回路変更を行い、更に処理済みの部分を変更対象としないという方法をとっている。この方法は、

- 処理対象となった回路の入力までの積算遅延に変更がない。
- また出力側からパスの終点までの積算遅延にも変更がない。

ので、積算遅延情報を変更する範囲を少なくすることができるメリットがある。

処理順序と遅延情報更新の例として、図 2.1 の回路に対して LDR を実行する場合を考える。ただし問題を簡単にするために、回路は変更せず遅延だけが減ったとした場合の処理を示す。

最初は全ての回路が未処理状態であるので、最大の遅延をもつパスの始点である B に着目する。始点 B から 終点方向へとパスをたどり、端子 P までの部分を回路変更の処理対象としたとし、始点 B から端子 P 迄の間の遅延が 2 から 0.5 に減ったとする。これにより実際には図 2.4に示すように各パスの遅延が変わる。しかしながら、LDR では 図 2.3のように処理した部分▭は 2 度以上は参照されないので遅延情報は更新せず、▨の部分のみ更新する。

これにより、C から始まる パス C_1, C_2 が新たなるクリティカルパスとなる。このため、新しい遅延削減の候補として、始点 C から始まるパスの周辺回路を次の候補として処理を続ける (図 2.5参照)。前述の様に、一旦処理した部分については再度処理することはないので、図 2.3 において選択された回路部分は対象とされない。

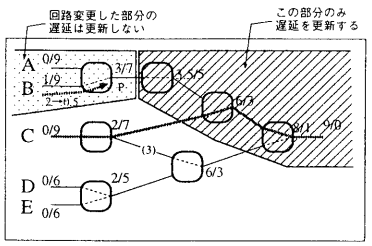

図 2.3: 遅延情報の更新

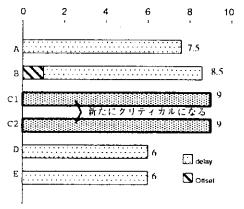

図 2.4: 変更後のパス遅延の様子

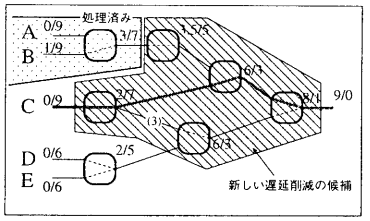

図 2.5: 新規対象回路の選択

2.1.2 制約の生成

LDR では、パスの始点から各端子の目標遅延を制約情報として持つ。ある端子の目標遅延は、その端子の実遅延とパスの終点の目標遅延と実遅延との比率とから求める。ただし、パス遅延の中にはメガセル内部の遅延やチップの入力側についている前置き遅延等の LDR では改良できないものが含まれている場合がある。LDR ではこのような遅延のことを **offset**と呼び、比率の計算から取り除いている。

ある端子 t_{from} の制約値は、その端子点の出力側の制約値から以下の方法で再帰的に求める。

$$t_{from}\text{の制約値} \quad = \quad \textbf{offset} + (t_{from}\text{の遅延} - \textbf{offset}) \times \underset{(\forall t_{to} \in T_{to})}{\textbf{MAX}} \left(\frac{t_{to}\text{の制約値} - \textbf{offset}}{t_{to}\text{の遅延} - \textbf{offset}} \right)$$

ただし、t_{from} の出力側には端子 t_{to} が隣接しているものとし、t_{from} を通るパスの始点の前置き遅延のうち最大のものを**offset** と表すこととする。出力側に複数の端子が接続する端子の制約は、出力側の端子の制約値が全て決った後でもっとも制約がきつい (制約値の小さい) 方を採用する。

図 2.1.2 を用いて一つ例を示す。例では、遅延 5 である端子 a について既に制約 3 が付加された状態であり、この時に 遅延 3 である端子 b の制約を求める方法について示す。

まず、端子 a 及び b は、双方 offset が 1 である。端子 a についている制約は、遅延から offset を引いた分 4 を 2 にするというものなので、端子 b に関しても同じ比率で縮小したものを制約とする。このため 端子 b の制約は

$$\begin{aligned} \text{端子}b\text{の制約} &= 1 + (3-1) \times \frac{(3-1)}{(5-1)} \\ &= 2 \end{aligned}$$

となる. 実際の LDR の処理では、 このような処理をパスの終点側から、パスの始点まで繰り返して順に制約値を付加していく。

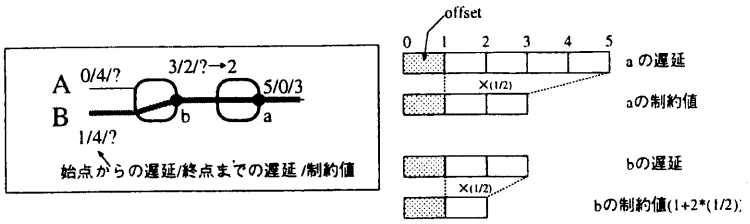

図 2.6: 制約付加の例

3 LDR の処理

クリティカルパスとは、 各パスに与えられた遅延の制約を満たしていないか、もしくは満たしていても 制約と近い遅延を持つものものとする。クリティカルパスが回路の性能を決めるので、これらの遅延を減らすことが重要である。

LDR を使った設計フローは図 3.1のようになり、 LDR はレイアウト設計のうち配置設計の終った段階に実行される。LDR 自体は図 3.2のような順序で処理をする。

以降では、 回路の選択処理及び再マッピング処理について説明する.

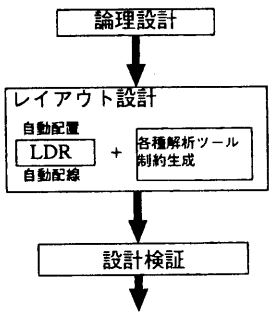

図 3.1: LDR を使った設計フロー

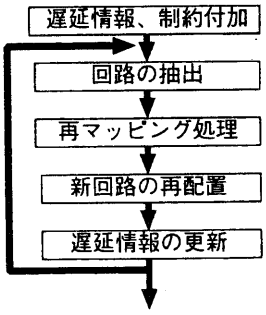

図 3.2: LDR の処理フロー

3.1 部分回路選択処理

変更すべき部分回路を最小限にして遅延の最適化をするため、LDR では、遅延が問題となるパス (クリティカルパス) を抽出し、その周辺部分のみを処理対象としている。最もクリティカルなパスは、パス上の端子が

"パス遅延 = (パス始点からの積算遅延) + (パス終点までの積算遅延)"

の最大値を持つ。処理対象となる回路を抽出するには、

1. まず、最大パス遅延を持つ端子のうち最も始点側の端子を選択する。
2. 次に選択した端子を使い次のように回路を抽出する。

(a) 選択した端子 x から終点方向に向かって n 段分パスをたどった端子 z を取得する。経験的に n は 4 ～ 6 程度である。ここでいう "段" とはセルとネットの組のことを指す。

(b) 逆に端子 z からパスの始点方向に向かって、接続するセルの集合を n 段だけ選び出す。選んだセル集合を選択回路とする。このとき既に処理済みのものは選ばないようにする。

3. 選んだ回路に対して変更処理を加えた後、遅延等の情報を更新する。

という処理を繰り返す。このような処理では、端子 z を 1 つの出力とする 木状の回路を抽出することができる.

図 3.3 の回路を例として、端子 a から $n = 2$ として回路を選択する場合を示す。まず始めに着目した端子 a から 2 段分パスの終点方向にたどった端子は b である。この端子 b から逆にパスの始点方向へ 2 段分だけ木状に接続するセルを選ぶ (図 3.4参照)。ただし、このとき セル C は処理済みなのでこの選択処理では選ばれない。

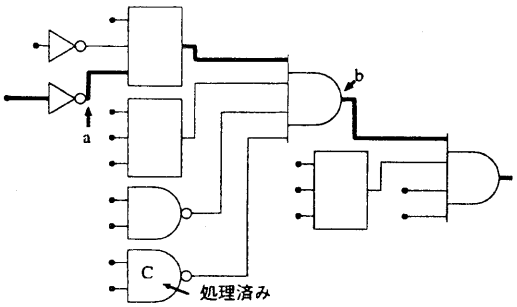

図 3.3: 回路選択の例題回路

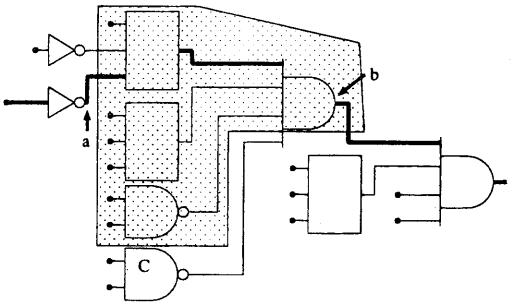

図 3.4: 回路選択の例

3.2 LDR のマッピング処理

前節の選択処理で選択された木状の回路に対して次の手順でマッピング処理を行う。

1. 選択回路の各セルを NAND2 と IV の組合せで表現した形に分解する。

2. 分解した回路の各 NAND2,IV に対して、元のセルの配置位置を考慮した仮想的な配置位置を与える。

3. 逆に NAND2,IV をセルにまとめる。NAND2,IV をまとめたセルの配置位置を決める。

本節ではこれらの処理ステップ各々について説明する。

3.2.1 セルの NAND2,IV への分解処理と位置設定

回路が選択されたのち、選択された回路を NAND2,IV の組合わせのみで表現した物で置き換える。このために LDR はセルと NAND2,IV の組合せによる表現との対応表を持っている。この表を参照して、与えられた回路中の個々のセルに対応する NAND2,IV 表現を得る。これら NAND2,IV による表現に展開した後、個々の NAND2,IV に対して配置位置を与えてマッピング時に遅延を計算するのに使用する。位置の与え方を説明する前に、まず選択した回路中の端子集合 $P(p_1, p_2, \ldots, p_n)$ とその回路外の端子集合 $A(a_1, a_2, \ldots, a_m)$ とする (図 3.5参照)。これら端子集合 A に属する端子の配置位置は固定しているものと考え、端子集合 P に属する端子の位置は 未定であるとする。このとき、P に属する端子は隣接する端子

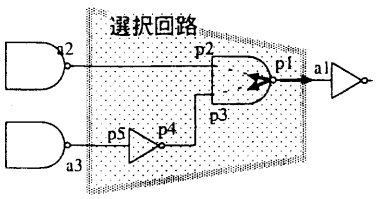

図 3.5: NAND2,IV の位置決め

から端子間の距離に比例する力で引かれているとする。各端子 $p_1, p_2, \ldots, p_n$ はこの力がバランスする点に配置されるとする。例えば、端子 x と y 間の距離を d(x,y) で表すとするとして、p_1 に関してバランスのための式を示すと

$$d(a_1, p_1) = d(p_1, p_2) + d(p_1, p_3)$$

となる。これと同様の式を P の各端子について立て、n 元方程式を解くことで各端子の位置を求める。

3.2.2 セルへのマッピング処理

前節で選択した NAND2,IV の組合せによる回路に対して再マッピングをし、新しい回路を生成する．選択された回路は木状になっている (出力が 1 つ) ので、その部分回路もまた木状になっている．これを NAND2,IV への変換の時にも使用したテーブルのエントリとグラフマッチングさせることで部分木をセルに対応させることができる。LDR では、セルとしてマッピングする境界を順に変えつつクリティカルパスの遅延が制約を満たす候補を採用している．このとき、遅延を計算する際に新しくできたセルの配置位置が必要になるが、これは、一つのセルとしてまとめた NAND2, IV の位置の重心位置としている．

3.2.3 実行例

LDR で実行される回路変更について例を示す (図 3.6参照)。簡単のために全ての配線は遅延が 1 であり、標準サイズの NAND2,IV は遅延 1、NOR は 2 の遅延を持つとし、これ以外の遅延を持つ素子については図中で示すこととする。

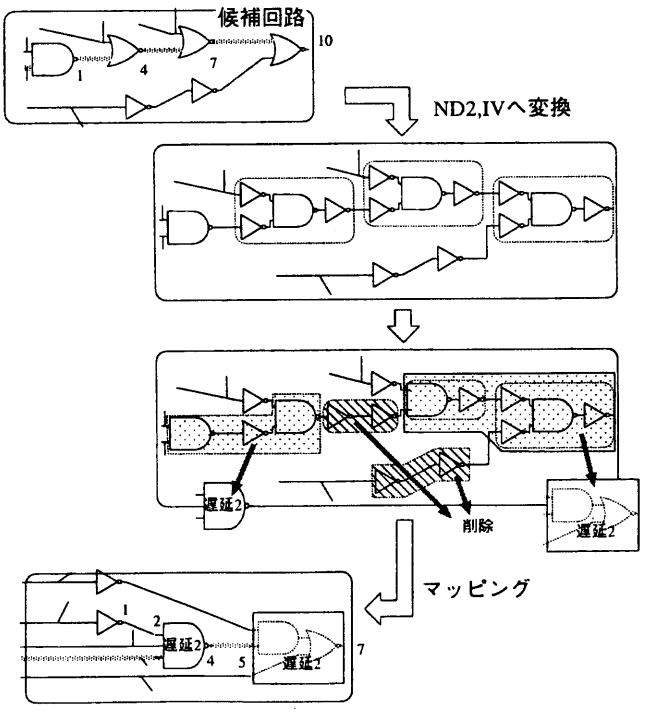

図 3.6: 回路変更例: 遅延減少の場合

LDR では、前節で示した回路選択処理で順に回路を選びながら、選択された回路に対して上記のような回路変更処理をして回路の遅延を減少させている。

4 実験結果

ISCAS のベンチマークデータのうち小規模の回路 C880 と C1908 および、1 万から 2 万セル規模のゲートアレイデータ (以後 Data1 とする) を用いて LDR の評価を行った。C880 と C1908 については遅延制約の元で初期的な配置のみを行ったデータに対して、Data1 については、配置改良後のデータについて、LDR の効果を調べた。これは、現状の LDR は 階層の切れ目については処理をしないようになっているため Data1 での選択される回路のサイズが小さく十分に最適化が行われないことを考慮したためである。表 1 には各データにおけるセル数及び遅延の変化を、表 2 には各データにおける選択した回路の数及び、選択回路毎のセル数の増減を載せてある。

C880	セル数	クリティカルパスの遅延	実行時間
初期配置のみ	589	51.24(ns)	-
LDR 実行後	601(+2.0%)	46.67(-8.9%)	1m57s
C1908	セル数	クリティカルパスの遅延	実行時間
初期配置のみ	1204	36.67(ns)	-
LDR 実行後	1201(-0.2%)	35.41(-3.4%)	2m 43s
Data1	セル数	クリティカルパスの遅延	実行時間
配置改良のみ	32657	27.51(ns)	-
LDR 実行後	32894(+0.7%)	24.84(-9.6%)	12m 27s

表 1: 各データでの遅延の改善

	選択回路数	セル数の変化した回路数			処理対象のセル数	
		不変	増加	減少	入力	出力
C880	43	14	20	9	184	196
C1908	62	54	2	6	222	219
Data1	19	0	17	2	514	751

表 2: 各データでの LDR 実行対象の回路数 及びセル数の変化

すべてのデータにおいて、遅延は減っているものの同時にセル数がかなり増大している。これは、LDR が一旦セルを NAND と IV の組合せに分解したのち、まとめ直すという手順を取っているために、マッピングの仕方ではセルを分解したままにしてしまうことと、今回使用したライブラリでは複合セルが多くないためであると考えられる。

5 結論

LDR はレイアウトの状況を考慮して論理接続を改善するツールである。レイアウト情報をもとに配線 経路や配線遅延の予測をし、これをもとに論理を最適化する。現状の LDR の課題点は次のようになる。

- 論理の等価性の検証
  回路全体に渡っての論理等価性の検証が必要となるが、現状ではそのチェックをしていない。現在のバージョンはできうる限り人手で検証を行ってはいるが大規模な回路全てを検証し尽くしてはいない。今後何らかの検証自動化が必要である。

- 消費電力削減と遅延削減処理のトレードオフの設定
  現在のLDRは、消費電力の最適化もしくは遅延の最適化を単独にしか行えない。これは今後の改善課題である。これには遅延制約つきの消費電力最小化処理が含まれる。

参考文献

[1] Kurt Keutzer. Dagon: Technology binding and local optimization by dag mapping. In *ACM/IEEE DAC'87*, pages 341–346, 1987.

[2] Kamal Chaudhary and Massoud Pedram. A near optimal alghorithm for technology minimizing under timing constraints. In *ACM/IEEE DAC'92*, pages 492–497, 1992.

[3] Eric Lehman, Yoshinori Watanabe, Joel Grodstein, and Heather Harkness. Logic decomposition during technology mapping. In *IEEE/ACM ICCAD95*, pages 264–271, 1995.

[4] S. Lin, M.Marek-Sadowska, and E.E. Kuh. Delay and area optimization in standard-cell design. In *IEEE DAC'90*, pages 349–352, 1990.

[5] Peter S Hauge et al. Circuit placement for predictable performance. In *IEEE ICCAD'87*, pages 88–91, 1987.

[6] Habib Youssef and Eugene Shragowitz. Timing constarints for correct performance. In *IEEE ICCAD'90*, pages 24–27, 1990.

[7] Mutsunori IGARASHI, Masako MUROFUSHI, Masami MURAKATA, and Takashi MITSUHASSHI. Timing driven placement with an rc wire delay model for sub-micron cmos gate arrays. In *SASIMI Workshop '93*, pages 235–244, 1993.

[8] T.Aoki, M.Murakata, T.Mitsuhashi, and N.Goto. Fanout-tree restructuring algorithm for post placement timing optimization. In *ASP-DAC'95*, pages 417–422. ASP-DAC, 1995.

[9] Guangqiu Chen, Hidetoshi Onodera, and Keikichi Tamura. An iterative gate sizing approach with accurate delay evalution. In *IEEE/ACM ICCAD95*, pages 422–427, 1995.