タイミング制約を考慮した並列スタンダードセル配置手法

5P－5　小野 光博† 小出 哲士† 西丸 由貴† 若林 真一† 吉田 典可‡
†広島大学 工学部 ‡広島市立大学 情報科学部

1 まえがき

VLSIの設計は大変複雑であるため，幾つかの工程に分けて行なわれる．回路中の素子をチップ上に配置する工程を配置設計という．近年では，素子間の配線遅延がチップの動作速度を左右する要因のひとつになっているが，配線長は配置設計に依存するため，高いパフォーマンスを得るためには配置設計において配線遅延を考慮することが重要である．タイミング制約を考慮した配置問題に対するアルゴリズムはこれまでにも幾つか提案されているが[2, 3]，大規模な回路に対しても実用的な計算時間で良い解を得ることは困難であった．計算時間の短縮化をはかるアプローチの一つとしてアルゴリズムの並列化を行うことが挙げられる．本稿では，著者らが提案した非線形計画法に基づくタイミング制約を考慮した配置手法[1]の並列化について述べる．

2 準備

本稿では，スタンダードセル，ゲートアレイ，及びSOG方式などのセルが列状に配置されるレイアウト方式を仮定する．配線層は2層配線とし，第1層を水平配線に，第2層を垂直配線に使用するものと仮定する．タイミング制約 $t_\tau \in \mathcal{T}$ は，図1に示すように，回路中の任意の端子対 (s_τ, e_τ) とそれらの間の部分回路に許される最大許容遅延時間 D_τ で与えられる．すなわち，s_τ, e_τ 間のすべての信号経路（クリティカルパス）に対して配線による遅延時間とゲートの遅延時間の総和が D_τ 以下にならなければならない．

配線遅延の評価には等価回路としてRC集中定数回路を仮定し，ネット n_i のソース0からロード j までの遅延時間を $d_{ij}(w_1, w_2, l_1, l_2) = (c_1 w_1 + c_2 w_2 + \sum_k C_{lk})(R_0 + r_1 l_1 + r_2 l_2)$ で表す．ただし，w_1, w_2 はネット n_i の第1, 2層の配線長，l_1, l_2 はネット n_i のソースからロード j までの第1, 2層の配線長，c_1, c_2 は第1, 2層の単位長あたりの配線容量，r_1, r_2 は第1, 2層の単位長あたりの配線抵抗，$\sum_k C_{lk}$ はネット n_i のロード容量の総和，R_0 はネット n_i に入力するセルの出力等価抵抗を表す．ここで，ネット n_i の配線長，ネット n_i のソースからロード j までの配線長は図2に示すように見積もる．本稿で取り扱うパフォーマンスドリブン配置問題は以下のように定式化される．

[パフォーマンスドリブン配置問題]
入力：　1) 論理回路 $\mathcal{L} = (\mathcal{M}, \mathcal{N})$
　　2) タイミング制約集合 $\mathcal{T}$
　　3) 配線遅延を求めるためのパラメータ

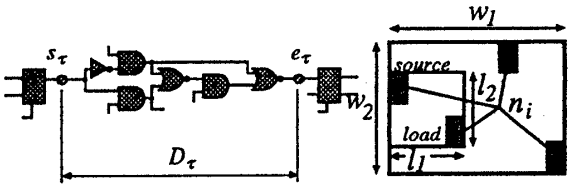

図1 タイミング制約

図2 ネット n_i の配線長

出力：　$\mathcal{M}$ の配置
目的関数：チップ面積 → 最小化
制約条件：1) レイアウトモデルに従う．
　　2) 全てのタイミング制約を満たす．　□

3 提案並列配置手法

3.1 提案並列配置手法の概要

提案配置手法はこれまでに我々が開発したスタンダードセル配置手法 POPINS [1] を基にして，アルゴリズムの並列化を行ったものである．仮定する並列計算機モデルは共有メモリモデルとし，プログラム構造としてはマスタ / スレーブモデルを採用する．

提案配置手法は3つのフェーズで構成されている．フェーズ1ではタイミング制約を考慮しながら階層的に分割手法を適用し，カット数と総配線長を最小化し，高速にセルの初期配置を求める．フェーズ2ではフェーズ1で得られたセルの配置を改良する．このフェーズは反復改良法であり，1回の試行では，制約の違反が大きいクリティカルパスを含む部分回路を選択し，この部分回路に対してタイミング制約のもとで配線長を最小化する問題を非線形計画問題に定式化し配置の改良を行なう．最後にフェーズ3ではフェーズ2の配置結果に基づき，タイミング制約を考慮してセルを列状に並べ換える．以下では紙面の都合上，各フェーズの並列化の概要について説明する．

3.2 フェーズ1の並列化

フェーズ1では，著者らが開発したタイミング制約を考慮した手法[3]を基に改良した手法を適用している．この手法は階層的4分割手法に基づいている．並列化の方針としてはマスタが各分割領域を複数のスレーブに割り当てることにより並列化する．また，計算時間が最も多くかかると予想される最上位レベルでの分割については，次のような並列分割手法を提案する．図3に示すように，まず分割領域内のセルをプロセッサ数と同じ数の集合になるまでクラスタリングする．次に各クラスタに対する2分割を複数のプロセッサでそれぞれ独立に行い，それらの結果を合わせてその分割領域全体の2分割とする．この操作を解の改善が見られなくなるまで繰り返し，最終的な2分割を得る．

3.3 フェーズ2の並列化

フェーズ2ではフェーズ1の配置結果を初期配置として反復改良を行う．フェーズ2の目的は全てのタイ

"A Parallel Standard Cell Placement Method Considering Timing Constraint," by Mitsuhiro ONO†, Tetsushi KOIDE†, Yutaka NISHIMARU†, Shin'ichi WAKABAYASHI†, and Noriyoshi YOSHIDA‡, †Faculty of Engineering, Hiroshima University. ‡Faculty of Information Sciences, Hiroshima City University. e-mail: axe@ecs.hiroshima-u.ac.jp

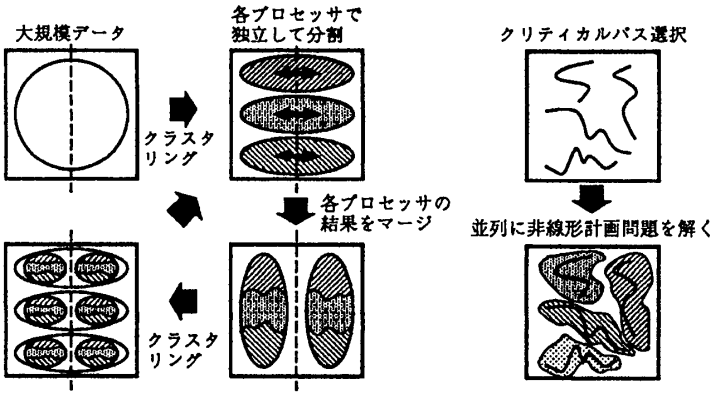

図 3 再上位レベルの分割の並列化

図 4 フェーズ 2 の並列化

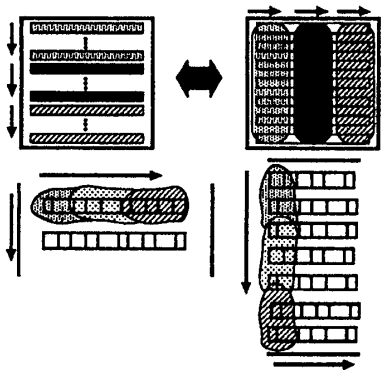

図 5 列割当ての並列化

ミング制約を満足し，総配線長を最小化することである．このフェーズは反復改良法であり，1 回の試行では，制約の違反が大きいクリティカルパスを含む部分回路に対して非線形計画法を適用する．フェーズ 2 における並列化の方針としては図 4 に示すように部分回路に対する非線形計画法に基づく配置改良を，プロセッサの数だけ並列に行う．この時，複数の部分回路にまたがるパスの扱いをどのようにするかが問題となるが，部分回路の大きさは限られているため大規模な回路では複数の部分回路にまたがるパスは少ないと予測される．したがって特に考慮しなくても比較的良い解が得られると思われる．

3.4 フェーズ 3 の並列化

フェーズ 3 では配置領域上に不規則に分散したセルを列状に配置する．フェーズ 3 の列割当て手法はまず，セルの Y 座標に基づいてセルのクラスタを構成し，次に各クラスタごとに配線長とタイミング制約を考慮した線形割当て問題に定式化し，列にセルを割り当てていく (Y 方向の割当て)．その後，セルの X 座標に基づいてセルのクラスタを構成し，Y 座標に基づいた場合と同様の操作によりセルの列への再割当てを行なう (X 方向の割当て)．以上の操作を解が改善されなくなるまで繰り返す．

フェーズ 3 の並列化の概要は以下の通りである．図 5 に示すように逐次アルゴリズムでは Y 方向 (X 方向) の割当てについて，それぞれ上 (左) から順に列割り当てを行っていたのを，提案手法では複数の領域に分けて，それぞれの集合に対して上 (左) から順に線形割当てを並列に行う．こうすることで各領域ごとに独立に線形割当てを行うことができうまく並列化できる．

また，さらに高速に線形割当てを行うために，各プロセッサは図 5 に示すように各セル集合を更に Y 座標 (X座標) により複数の集合に分割し，左 (上) から順に割り当てを行っていく．もともと左 (上) 端の方にあったセルが，線形割り当てにより右 (下) 端に割り当てられるということはほとんどないと考えられるため，得られる解は従来の手法 [1] とあまり変わらないと予想される．

4 実験結果

提案手法 PAR-POPINS を 4 個の CPU を搭載する Sun SPARCserver1000 上で C 言語を用いて実現し，POPINS[1], RITUAL[2] と比較実験を行った．実験に用いたテストデータを表 1 に示す．表中の制約数 (#cons) はタイミング制約の数を表す．なお，提案手法のフェーズ 1, 2 は現在実現中であるため POPINS のフェーズ 2 の出力を PAR-POPINS のフェーズ 3 の入力として実験を行った．実験結果を表 2 に示す．表中の time はフェーズ 3 の CPU 時間を表す．表中の delay とは各タイミング制約に対して実伝搬遅延時間を最大許容遅延時間で割った値を表しており，この値が 1 以下なら制約を満足していることになる．表にはこの値の最大値 (Max.) と平均値 (Ave.) を載せた．なお，RITUAL では s15850, s38584 についてはエラーのため結果が得られなかった．結果より提案手法は全てのデータに対してタイミング制約を満足する解を得ることができたが，POPINS と比較すると最大違反率，総配線長共に若干悪くなっているが，RITUAL よりも良い解を出力していることが分かる．また，CPU 時間については POPINS よりもかなり高速になっており，並列化の有効性が確認できた．今後フェーズ 1, フェーズ 2 を計算機上で実現し，シミュレーション実験を行なう予定である．

表 1 実験データ

data	#cells	#nets	#I/O	#rows	#cons
s15850	3228	3332	227	22	1105
s38417	7572	7734	134	37	2619
s38584	8964	9344	342	46	2729
s35932	11838	12228	355	45	3488

表 2 実験結果

data	method	delay		length	time
		Max.	Ave.	(λ)	(sec)
s15850	p-popins	0.69	0.17	2171417	126
	popins	0.69	0.16	1986242	390
s38417	p-popins	0.70	0.26	4902314	703
	popins	0.67	0.23	4468116	2297
	ritual	0.58	0.24	6930625	—
s38584	p-popins	0.62	0.14	8188792	825
	popins	0.60	0.14	7365154	3316
s35932	p-popins	0.72	0.32	10331906	2448
	popins	0.72	0.31	9731642	7104
	ritual	0.75	0.35	11630953	—

参考文献

[1] T. Koide, M. Ono, S. Wakabayashi, Y. Nishimaru and N. Yoshida: "A new performance driven placement method with the Elmore delay model for row based VLSIs," Proc. ASP-DAC, to appear (1995).

[2] A. Srinivasan, K. Chaudhary and E. S. Kuh: "RITUAL: A performance driven placement algorithm for small cell ICs," Proc. ICCAD, pp. 48–51 (1991).

[3] S. Wakabayashi, H. Kusumoto, H. Mishima, T. Koide and N. Yoshida: "Gate array placement based on min-cut partitioning with path delay constraints," Proc. IS-CAS, pp. 2059–2062 (1993).