計算機アーキテクチャ 115－27
設 計 自 動 化 78－27
（1995. 12. 15）

非線形計画法に基づく並列タイミングドリブンスタンダードセル配置手法

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
〒739 広島県東広島市鏡山一丁目4番1号
*E-mail: axe@ecs.hiroshima-u.ac.jp

‡広島市立大学 情報科学部
〒731-31 広島市安佐南区沼田町大塚 151-5

本稿では，VLSIチップのレイアウト設計において，与えられたタイミング制約を考慮し，マルチプロセッサワークステーション上で動作する並列スタンダードセル配置手法を提案する．並列処理の導入により大規模な回路に対しても実用的な計算時間でタイミング制約を満たす配置を求めることができる．提案配置手法は，まずタイミング制約を考慮した階層的回路分割に基づく高速な手法で初期配置を行い，次に非線形計画法に基づく手法で配置を改良する．最後に，タイミングを考慮した列割り当て手法でセルを列状に配置する．提案手法を4CPUマルチプロセッサワークステーション上に実現し実験を行った結果，提案手法の有効性を確認した．

A Parallel Timing Driven Standard Cell Placement Method with Nonlinear Programming

Mitsuhiro ONO*† Tetsushi KOIDE† Yutaka Nishimaru†
Shin'ichi WAKABAYASHI† and Noriyoshi YOSHIDA‡

†Faculty of Engineering, Hiroshima University
4-1, Kagamiyama 1 chome, Higashi-Hiroshima 739, JAPAN
*E-mail: axe@ecs.hiroshima-u.ac.jp

‡Faculty of Information Sciences, Hiroshima City University
151-5, Ozuka, Numata-Cho, Asa-Minami-Ku, Hiroshima 731-31, JAPAN

In this paper, we present a parallel performance driven placement method running on a multi-processor workstation for large scale standard cell layout. Adopting parallel processing allows us to obtain results for large scale circuits in a short computation time. In the proposed placement method, first, an initial placement is obtained with a fast performance driven hierarchical partitioning method. Next, an iterative improvement method by nonlinear programming improves the placement. Finally, row assignment considering timing constraint is performed. We have implemented the proposed method on a 4CPU multi-processor workstation and showed the effectiveness of the method from experimental results.

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 [3, 7, 11, 12, 14, 15]，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．並列に配置を行う手法としては [1, 2, 4, 9, 13] 等が提案されているが，プロセッサ数に比例する高速化を実現し，かつ従来の逐次アルゴリズムと同等の解を得ることは困難であり，配線遅延についても考慮されていない．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 [8] の並列化について述べる．提案並列配置手法をマルチプロセッサワークステーション上に実現し，POPINS2.0[8], RITUAL[12] と比較実験を行った結果，提案手法の有効性を確認した．

2 準備

2.1 レイアウトモデルと配線遅延モデル

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．配線の等価回路は RC 集中定数回路を仮定し，配線遅延モデルとしては Elmore Delay[5] を用いる．多端子ネット n_i をスタイナ木で表した場合，文献 [10] において Kuh らは Elmore Delay の上限値 $d_i(W, L_{0j}) = (cW + \sum_k C_{lk})(R_0 + rL_{0j})$ を導出している．ここで，W はスタイナ木の配線長，L_{0j} はソースからロード j までの配線長，c と r は単位長あたりの容量と抵抗，R_0 はソースの出力抵抗，$\sum_k C_{lk}$ はロード容量の総和である．本稿ではネット 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 1 に示すように見積もるため，ネット n_i のソース 0 から ロード j までの遅延時間は $d_{ij}(w_i, h_i, l1_{ij}, l2_{ij}) = (c_1 w_i + c_2 h_i + \sum_k C_{lk})(R_0 + r_1 l1_{ij} + r_2 l2_{ij})$ で表される．ただし，w_i, h_i はネット n_i の第 1, 2 層の配線長，$l1_{ij}, l2_{ij}$ はネット n_i のソース 0 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．

2.2 タイミング制約

タイミング制約とは，回路が満足すべき信号伝搬遅延の制約であり，図 2 に示すように，与えられた回路の任意の端子対 (s_τ, e_τ) とそれらの間の部分回路に与えられる最大許容遅延時間 D_{req_τ} によって与えられる．すなわち，端子対 (s_τ, e_τ) 間の全ての信号経路（パス）の遅延時間が D_{req_τ} 以下であればそのタイミング制約は満たされたことになる．タイミング制約集合は $\mathcal{T}$ で表される．

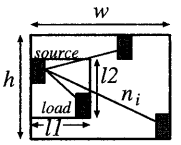

図 1 ネット n_i の配線長

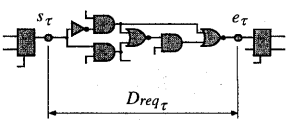

図 2 タイミング制約

2.3 パフォーマンスドリブン配置問題

論理回路を $\mathcal{L} = (\mathcal{M}, \mathcal{N})$ する．ここで，$\mathcal{M} = \{m_1, m_2, \cdots, m_M\}$ はセル集合，$\mathcal{N} = \{n_1, n_2, \cdots, n_N\}$ はネット集合を表す．セル m_i に接続するネットの集合を $\mathcal{N}_i$とし，ネット n_j に接続するセルの集合を $\mathcal{M}_j$とする．任意のタイミング制約 $t_\tau \in \mathcal{T}$ に対し，始点が s_τ 終点が e_τ であるような任意のパス $p_\pi = (\mathcal{M}_\pi, \mathcal{N}_\pi)$ を被制約パスと呼ぶ．ここで，$\mathcal{M}_\pi$は p_π に含まれるセルの集合，$\mathcal{N}_\pi$は p_π に含まれるネットの集合である．被制約パス集合を $\mathcal{P}$とし，$\mathcal{P}_\tau$を $t_\tau \in \mathcal{T}$ に対する被制約パス集合とする．また，D_{req_π} をパス $p_\pi \in \mathcal{P}$ の最大許容遅延時間，D_{act_π} をパス $p_\pi \in \mathcal{P}$ の実伝搬遅延時間とする．

任意のタイミング制約 $t_\tau \in \mathcal{T}$ について，$\mathcal{L}$ の部分回路 $\mathcal{L}_\tau = (\mathcal{M}_\tau, \mathcal{N}_\tau)$ を被制約回路と定義する．ただし，$\mathcal{M}_\tau = \bigcup_{\forall p_\pi \in \mathcal{P}_\tau} \mathcal{M}_\pi$ は $\mathcal{L}_\tau$ のセルの集合，$\mathcal{N}_\tau = \bigcup_{\forall p_\pi \in \mathcal{P}_\tau} \mathcal{N}_\pi$ は $\mathcal{L}_\tau$ のネットの集合である．また，被制約回路集合を $\mathcal{C}$とし，回路 $\mathcal{L}_\tau$ の実伝搬遅延時間を $D_{act_\tau} = \max_{\forall p_\pi \in \mathcal{P}_\tau} D_{act_\pi}$ とする．この時，本稿で取り扱うパフォーマンスドリブン配置問題は入力として論理回路 $\mathcal{L} = (\mathcal{M}, \mathcal{N})$ とタイミング制約集合 $\mathcal{T}$ が与えられた時にタイミング制約の下でチップ面積を最小化する $\mathcal{M}$ の配置を求めることである．

3 逐次配置手法[8]

3.1 手法の概要

提案配置手法はこれまでに我々が開発したタイミング制約を考慮したスタンダードセル配置手法 POPINS [8] を基にして，アルゴリズムの並列化を行ったものである．本節ではこの逐次配置手法について述べる．

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．以下では各フェーズについて説明する．

3.2 フェーズ 1 ：階層的分割による初期配置

タイミング制約を全て満足するような配置を求めることはフェーズ 2 で行うため，フェーズ 1 の目的は，タイミング制約を大体満足し，総配線長が最小となるようにセルを配置領域上に均一に分散させることである．

このフェーズには，我々が開発したタイミング制約を考慮した配置手法 [15] を更に改良したものを用いている．この手法はパスベースの重み付け手法であり，アルゴリズムはオーソドックスな階層的 4 分割手法に基づいており，分割には Fiduccia らのいわゆる FM 法 [6] に基づいた手法を適用しているが，提案手法ではカット数の減少数を表すカットゲイン g_{cut_i} だけでなく，タイミング制約を考慮したスラックゲイン g_{slack_i}，セル m_i に接続しているセルの位置を考慮したゲイン g_{term_i}，配線長の減少量を考慮した g_{wire_i} の 4 つのゲインの線形和をゲインとし，2 つのセル集合のセルの面積の和がバランスするよう分割する．これによりカット数と総配線長を同時に考慮することができ，配線によるチップの混雑の度合が均一でかつ総配線長の最小な配置を求めることができる．

3.3 フェーズ 2 ：非線形計画法に基づく配置改良

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ミング制約を満足し，総配線長を最小化することである．提案手法ではタイミング制約の違反が大きい被制約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．この非線形計画法に基づく配置改良をタイミング制約の違反がなくなるか，反復回数が与えられた上限に達するまで反復する．

部分回路に対する配置問題は次のような数理計画問題に定式化される．

目的関数：

$$\sum_{\forall(\text{部分回路に含まれるネット } n_i)} \{w_i^2 + h_i^2\} \rightarrow Minimize$$

制約条件：

1) 部分回路に含まれるすべてのネット n_i に対し
   (n_iの任意の 2 端子の x 方向の距離) $\leq w_i$
   (n_iの任意の 2 端子の y 方向の距離) $\leq h_i$
   (n_iのソースから任意のロード jまでの x 方向の距離) $\leq l1_{ij}$
   (n_iのソースから任意のロード jまでの y 方向の距離) $\leq l2_{ij}$

2) 部分回路を通過するすべてのクリティカルパスに対し
   (パスの伝搬遅延時間) $\leq$ (パスの要求遅延時間) □

ただし，$w_i, h_i, l1_{ij}, l2_{ij}$は図 1に示すようにネット n_i の配線長を表す変数である．この問題は目的関数と制約条件 2) が 2 次式であるため非線形計画問題になる．

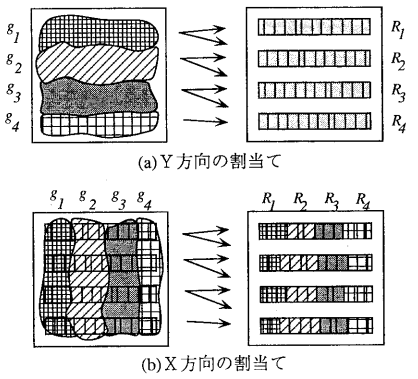

図 3 列割当て

3.4 フェーズ 3 ：列割当て

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 の列割当て手法は図 3に示すように，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．

線形割当ての際のセル m_j をスロット s_k に割り当てた時のコスト c_{jk} は次の式で表される．

$$c_{jk} = \sum_{n_i \in \mathcal{N}_j} \Delta l_i(m_j, s_k) f(\omega_i)$$

ただし，$\Delta l_i(m_j, s_k)$ はセル m_j をスロット s_k に割り当てた時のネット n_i の配線長の変化を表す．また，$f(\omega_i)$ はネット n_i のクリティカリティ ω_i に関する関数であり，ω_i が 1 を越えると急激に増加する．ω_i はタイミング制約 $t_\tau \in \mathcal{T}$ について，D_{req_τ} を最大許容遅延時間，D_{act_τ} を実伝搬遅延時間とするとき，次の式で表される．

$$\omega_i = \max_{\forall \mathcal{L}_\tau \in \{(\mathcal{M}_\tau, \mathcal{N}_\tau) | n_i \in \mathcal{N}_\tau\}} \frac{D_{act_\tau}}{D_{req_\tau}}$$

つまり，ネット n_i を含む全ての被制約回路 $\mathcal{L}_\tau$ の違反率 $D_{act_\tau}/D_{req_\tau}$ の最大値となる．

4 提案並列配置手法

本節では共有メモリを持つ MIMD 並列計算機上で動作する提案並列配置手法について述べる．提案配置手法は前節で述べた逐次配置手法を基にアルゴリズムの並列化を行ったものであり，処理の流れは前節に述べたように 3 つのフェーズからなっている．以降では，各フェーズの並列化について述べる．

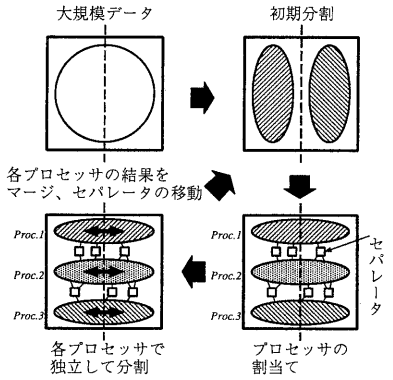

図 4 上位階層の分割の並列化

4.1 フェーズ 1 の並列化

4.1.1 フェーズ 1 の並列化の概要

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 [15] 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．フェーズ 1 における並列化の方針は以下の 2 点である．まず，上位の階層における大規模な回路の分割の際は後に述べる並列回路 2 分割手法により，回路の分割を行う．下位の階層では小規模な回路の分割を多数行う必要があるため，各分割領域をそれぞれ各プロセッサに割当てて，同時に複数の分割領域に対する分割を独立に実行することにより並列化する．

4.1.2 上位階層における並列回路 2 分割手法

上位階層における大規模な回路の分割は以下に述べる並列回路 2 分割手法を適用する．図 4 に示すようにまず，初期分割を基に分割領域内のセルを各プロセッサに割当てる．次に，各プロセッサ P_i はそれぞれ P_i に割当てられたセル集合の 2 分割を同時に独立して行い，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．そして，得られた 2 分割を基にして再度セルをプロセッサに割り当てる．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．

しかし，互いに接続のあるセルが異なるプロセッサに割当てられると正確にゲインの計算を行うことができない．そこで，提案手法ではセパレータとなるセルを設定し，セパレータとなったセルはどのプロセッサにも割り当てず，各プロセッサが分割を行っている間はその位置を固定しておく．そして各プロセッサが分割を行った後で，セパレータとなったセルの集合を分割する．こうすることにより，カットゲイン g_{cut_i} については正確に求めることができる．しかし，他の $g_{slack_i}, g_{term_i}, g_{wire_i}$ についてはセルの座標を基に計算するため，セパレータを導入しても正確には計算できない．そこで，分割の際には主にカットゲインを考慮して分割を行い，他のゲインはその補助として用いるようにする．

セルをプロセッサに割り当てる際にはセパレータになるセルを少なくするために，接続のあるセル同士はできるだけ同じプロセッサに割当てたほうがよい．また，各プロセッサにおける分割の際にかかる計算時間の差をできるだけなくするために，各プロセッサに割り当てられるセルの数をほぼ等しくする必要がある．さらに与えられた初期分割，あるいは前回のループにより得られた分割を基にしてセルの割当てを行うため，割り当てられた各セル集合内でセルの面積は左右でほぼバランスしていなければならない．以上のことを考慮して，以下のように各セルをプロセッサに割り当てる．

[プロセッサへのセル割当てアルゴリズム]

Step 1 : $S = \emptyset$
Step 2 : for all processors in parallel
Step 2.1 : $C_i = \emptyset, Ca_i = \emptyset, Cb_i = \emptyset, Ua_i = Ua, Ub_i = Ub$
Step 2.2 : seed となるセル $m_j \in Ua, m_k \in Ub, N_j \cap \mathcal{N}_k \neq \emptyset$ を選択．
Step 2.3 : $C_i = \{m_j, m_k\}$, $Ca_i = \{m_j\}$, $Cb_i = \{m_k\}$, $Ua_i = Ua_i - \{m_j\}$, $Ub_i = Ub_i - \{m_k\}$.
Step 2.4 : $m_j \in Ua_i \cup Ub_i$ なる全てのセルについて C_i との接続度を求める．
Step 2.5 : もし，$a(Ca_i) \leq a(Cb_i)$ ならば C_i との接続度の最も大きいセル $m_j \in Ua$ を選択，$Ua_i = Ua_i - \{m_j\}$，そうでなければ C_i との接続度の最も大きいセル $m_j \in Ub$ を選択，$Ub_i = Ub_i - \{m_j\}$.
Step 2.6 : もし，$m_j \in C_k (k \neq i)$ ならば，$S = S \cup \{m_j\}$，そうでなければ $C_i = C_i \cup \{m_j\}, Ca_i(Cb_i) = Ca_i(Cb_i) \cup \{m_j\}$
Step 2.7 : $a(C_i) >$ (プロセッサへ割り当てるセルの面積の総和の上限) または選択できるセルがなくなれば終了．
Step 2.8 : 接続度を更新し，**Step 2.5** へ．

ただし，Ua, Ub は分割領域内のセル集合の初期分割，C_i はプロセッサ P_i に割り当てられるセルの集合，S はセパレータになるセルの集合を表す．また，α をセルの集合とした時，$a(\alpha)$ はセル集合に含まれるセルの面積の総和を表す．

4.1.3 下位階層における並列化

下位の階層では，多数の分割領域で小規模な回路の分割を実行しなければならない．ここでは，各分割領域をそれぞれ各プロセッサに割当てる．各プロセッサはそれぞれ前節で述べた逐次分割手法を実行し，複数の分割領域に対する分割を同時に独立して行うことにより並列化する．図 5 は 4 分割された回路から 16 分割を得るまでの処理の流れを示している．ここでは図の (e) ～ (h) に示すように一度分割を行った領域に対しても更に分割領域をずらして分割を行うことによりより良い分割が得られるようにしている．

4.2 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 における並列化の方針と

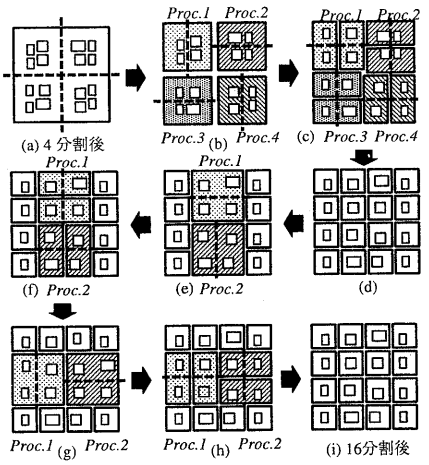

図 5 フェーズ 1 の分割手法の適用方法

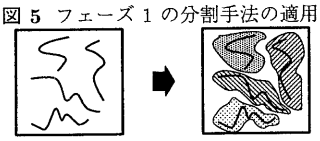

図 6 フェーズ 2 の並列化

しては図 6 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．しかし，同時に配置改良を行う部分回路が互いに重なっているとタイミング制約，配線長等が正確に考慮できなくなってしまうため，同時に配置改良を行う部分回路間で共有するセルがないように部分回路を構成する．配置改良を行う部分回路は次の様にして選択する．

まず，違反率の大きい被制約回路の中から任意に 1 つ選択し，その中で最も伝搬遅延時間の大きいクリティカルパスを部分回路とする．ここで違反率とは，被制約回路の実伝搬遅延時間を制約で与えられた要求遅延時間で割った値であり，全ての被制約回路の中の違反率の大きい方から 10 ～ 20 % 程度の被制約回路を選択される候補とする．このとき，少ない改良回数で効率良く配置を改良するため，過去 3 回以内に選ばれたことのある被制約回路は選ばないようにしている．次に，選択された部分回路に接続度の大きいセルから順にセルを 1 個ずつ部分回路に加えていくが，現在，他のプロセッサで配置改良を行っている部分回路に含まれるセルは加えない．また，以前と同じ部分回路が選択されないようにするため，ランダム性を導入してそのセルを選択するかしないかを決定している．ここでセルの接続度とは，セルと配置改良の対象としている部分回路とを接続するネットの数である．セルを部分回路に加えたら，そのセルに接続している全てのネットを部分回路に加える．以上の操作を繰り返し，部分回路をあらかじめ定められた大きさとなるまで大きくしていく．

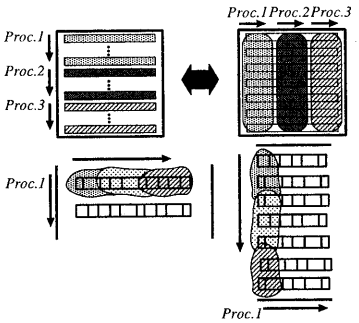

図 7 列割当ての並列化

フェーズ 2 のアルゴリズムは以下に示すように 1 個のマスタと複数のスレーブから構成されており，マスタはタイミング解析，部分回路の構成を行いスレーブを起動する．スレーブは部分回路の配置改良を非線形計画問題に定式化し，その問題を解く．

[フェーズ 2 : 非線形計画法に基づく配置改良]

マスタ:

Step 1 : 全ての被制約回路についてタイミング解析を行う．

Step 2 : もし，最大違反率が指定許容違反率以下か，$LoopNumber$ がある定数以上であれば全てのスレーブが終了するのを待ち，結果を受けとって終了する．

Step 3 : 配置改良を行う部分回路を構成し，スレーブを起動．

Step 4 : 終了したスレーブがあれば，結果を受けとり変化があった部分に対してタイミング解析を行う．これ以上スレーブを増やせない場合は任意のスレーブが終了するのを待つ．

Step 5 : $LoopNumber = LoopNumber + 1$, **Step 2** に戻る．

スレーブ:

Step 1 : 部分回路を通過する全てのクリティカルパスを選択する．

Step 2 : 非線形計画問題に定式化し，問題を解く．

Step 3 : 結果をマスタに送る．

4.3 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 の並列化の概要は以下の通りである．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，提案手法では図 7 に示すように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 7 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，得られる解は逐次手法 [8] とあまり変わらないことが期待される．

5 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，逐次手法である POPINS2.0[8], RITUAL[12] と比較実験を行った．提案手法のフェーズ 1 は現在実現中であるため POPINS のフェーズ 1 の出力を PAR-POPINS のフェーズ 2 の入力として実験を行った．実験に用いたテストデータは表 1 に示す ISCAS ベンチマークデータである．表中の制約数 (#cons) はタイミング制約の数を表す．

表 2 に逐次手法である POPINS2.0(pop)[8] と今回提案した並列配置手法 PAR-POPINS(p-pop) の実験結果を示す．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の最大値を表しており，この値が 1 以下なら制約を満足していることになる．CPU 時間についてはフェーズ 2, 3 の実行にかかった時間の合計を表した．また，表中の括弧で表した数字は POPINS2.0 の結果を 1.00 とした時の比を表している．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最大 3.3 倍，平均 2.8 倍高速に得ることができ，並列化の有効性が確認できた．総配線長については平均 2.4% 増加しているが，POPINS2.0 よりも良い結果が得られている場合もありほぼ同等な解が得られているといえる．

また，表 3 は RITUAL[12] との比較実験の結果である．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，総配線長で最大 29.2%，平均 16.3% 良い解を最大 29.0 倍，平均 5.3 倍高速に得ることができた．

6 あとがき

本稿では，タイミング制約を考慮した非線形計画法に基づく並列スタンダードセル配置手法を提案した．今後の課題として，(1) フェーズ 1 の実現，(2) フェーズ 2 における部分回路構成手法の改良，(3) フェーズ 3 の改良等が挙げられる．

表 1 実験データ

data	#cells	#nets	#I/O	#rows	#cons
C5	1081	1560	301	13	334
C6	1037	1516	301	14	334
C7	2150	2678	315	18	405
s9234	917	1028	75	12	412
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 POPINS との比較 (フェーズ 2, 3)

data	meth.	delay	length (μm)	time (sec)
C5	pop	0.66	1018940(1.00)	692(1.00)
	p-pop	0.68	1025636(1.01)	244(0.35)
C6	pop	0.64	958758(1.00)	516(1.00)
	p-pop	0.70	1008423(1.05)	155(0.30)
C7	pop	0.80	1427672(1.00)	3901(1.00)
	p-pop	0.79	1483594(1.04)	1352(0.34)
s9234	pop	0.71	518440(1.00)	1415(1.00)
	p-pop	0.70	539080(1.03)	479(0.34)
s15850	pop	0.71	2128367(1.00)	9805(1.00)
	p-pop	0.68	2104955(0.99)	5648(0.57)
s38417	pop	0.72	4767848(1.00)	3111(1.00)
	p-pop	0.70	4909707(1.03)	1066(0.34)
s38584	pop	0.55	7298534(1.00)	9173(1.00)
	p-pop	0.58	7660870(1.05)	2980(0.32)
s35932	pop	0.72	10442594(1.00)	12762(1.00)
	p-pop	0.72	10276918(0.98)	4567(0.36)

表 3 RITUAL との比較

data	meth.	delay	length (μm)	time (sec)
C5	ritual	0.63	1286249(1.00)	243(1.00)
	p-pop	0.68	1025636(0.80)	257(1.05)
C6	ritual	0.65	1186017(1.00)	243(1.00)
	p-pop	0.70	1008423(0.85)	168(0.27)
C7	ritual	0.72	1727330(1.00)	436(1.00)
	p-pop	0.79	1483594(0.86)	1389(0.34)
s9234	ritual	0.67	583559(1.00)	250(1.00)
	p-pop	0.70	539080(0.92)	491(1.96)
s38417	ritual	0.58	6930625(1.00)	10172(1.00)
	p-pop	0.70	4909707(0.71)	1582(0.16)
s35932	ritual	0.75	11630953(1.00)	132542(1.00)
	p-pop	0.72	10276918(0.88)	6581(0.05)

参考文献

[1] R. J. Brouwer, et al.: "PARAGRAPH: a parallel algorithms for simultaneous placement and routing hierarchy," Proc. of European Conference on Design Automation, pp. 328–332 (1992).

[2] R. J. Brouwer: "Parallel Algorithms for Placement and Routing in VLSI Design," Ph.D. thesis, Graduate College of the University of Illinois (1991).

[3] M. Burstein, et al.: "Timing influenced layout design," Proc. 22nd DAC, pp. 124–130 (1985).

[4] A. Casotto, et al.: "A parallel simulated annealing algorithm for the placement of macro-cells," IEEE Trans. CAD, Vol. 6, No. 5, pp. 838–847 (1987).

[5] W. C. Elmore: "The transient response of damped linear networks with particular regard to wideband amplifiers," J. Appl. Phys., Vol.19, pp. 55–63 (1948).

[6] C. M. Fiduccia, et al.: "A linear-time heuristic for improving network partitions," Proc. 19th DAC, pp. 175–181 (1982).

[7] T. Gao, et al.: "A new performance driven placement algorithm," Proc. ICCAD, pp. 44–47 (1991).

[8] T. Koide, et al.: "A new performance driven placement method with the Elmore delay model for row based VLSIs," Proc. ASPDAC, pp. 405–412 (1995).

[9] S. Kravitz et al.: "Placement by simulated annealing on a multiprocessor," IEEE Trans. CAD, Vol. 6, No. 4, pp. 534–549 (1987).

[10] E. S. Kuh et al.: "Recent advances in timing-driven physical design," Proc. APCCAS, pp. 23–28 (1992).

[11] A. Mathur et al.: "A new approach to performance driven placement for regular architectures," Proc. ICCAD, pp. 130–136 (1994).

[12] A. Srinivasan, et al.: "RITUAL: A performance-driven placement algorithm," IEEE Trans. CAS II, Vol. 39, No. 11, pp. 825–839 (1992).

[13] W.-J. Sun et al.: "A loosely coupled parallel algorithm for standard cell placement," Proc. ICCAD, pp. 137–144 (1994).

[14] S. Sutanthavibul et al.: "An adaptive timing-driven placement for high performance VLSI's," IEEE Trans. CAD, Vol. 12, No. 10, pp. 1488–1498 (1993).

[15] S. Wakabayashi, et al.: "Gate array placement based on mincut partitioning with path delay constraints," Proc. ISCAS, pp. 2059–2062 (1993).