システムＬＳＩ設計技術　99－13
(2001. 1. 12)

VoyagerによるPCクラスタ上の階層型並列分散配置処理について

綿貫　哲久，白石　洋一
群馬大学　工学部　情報工学科
〒376-8515　桐生市天神町１－５－１
0277-30-1854, 1855
{norihisa, siraisi}@keim.cs.guma-u.ac.jp

論理VLSIチップのレイアウト設計問題におけるセル配置問題を対象として，ランダマイズドアルゴリズムに基づく階層型並列分散処理を開発した．これをJava言語を用いて実装し，特に並列分散処理を実現するためにVoyagerを使用してその実用性を検討した．本稿では，まず1台のワークステーションによりセル配置問題自体がどの程度高速化されるかをシミュレーションによって求めた結果について述べる．続いて，実際のPCクラスタ上にVoyagerによる階層型並列分散処理を実装して，その機能実現を確認した．未だ高速化を評価できる段階ではないが，ベンチマークデータを対象とした実際の処理において通信時間がどの程度になるかを定量的に評価すると同時に，通信における問題点を明らかにした．

キーワード：VLSI，レイアウト，配置問題，ランダマイズドアルゴリズム，並列分散，Voyager，Java

A Hierarchical Parallel and Distributed Placer on a PC Cluster by using Voyager

Norihisa WATANUKI, Yoichi SHIRAISHI
Department of Computer Science, Faculty of Engineering, Gunma University
1-5-1 Tenjin-cho Kiryu, Gunma, 376-8515, Japan
Tel.: 0277-30-1854,1855
{norihisa, siraisi}@keim.cs.guma-u.ac.jp

A hierarchical parallel and distributed placer based on the randomized algorithm is developed. The target of this process is the cell placement problem in the layout design for a logic VLSI chip. This process is implemented by using Java and the core of the parallel and distributed process is realized by using Voyager. In this report, the simulation on one EWS first reveals the acceleration of the cell placement process itself. Then, the actual hierarchical parallel and distributed placement is implemented on a PC cluster by using Voyager and its feasibility is checked. The acceleration itself can not yet be evaluated but the communication overhead is evaluated against some benchmark data and the problems occurred in the communication overhead is revealed.

Keywords: VLSI, layout, placement problem, randomized algorithm, parallel and distributed process, Voyager, Java

1. はじめに

自動レイアウトツールが解決しようとしている問題は、超大規模組合せ問題として定式化できる。現在のコンピュータを用いて、次世代の半導体を設計しなければならないため、大きなブレークスルーがない限り問題の難しさは増大しても減少することは無い。自動レイアウト問題はその初期から、組合せ最適化アルゴリズム研究の非常に良い対象となっている。システム・オン・チップの時代に対して、急激に増大する問題の複雑度に対応するためと、懸案となっている設計コストの劇的な改善を行なわなければならない。また、現在、ワークステーション(EWS：Engineering WorkStation)とパソコン(PC：Personal Computer)からなる、この分散環境を有効利用したツールも切望されている[1]。実際に設計データ量の増大から、1台のワークステーションに仕様ギリギリのメモリを搭載しても対処できなくなりつつある。処理時間も1台のコンピュータによる処理では限界が見えてきている。これらの点から、本稿では、自動レイアウトツールをPCをネットワーク結合したプラットフォーム上で並列に処理する手法について検討する[2]、[3]。

電子系レイアウトDAにおける部品配置問題は、組合せ最適化問題として定式化できる。しかし、大規模配置問題に対しては、実用時間内にその最適解を得ることは不可能である。したがって、従来より様々な発見的手法が用いられてきた。局所解に陥らずにいかにして大域的最適解に到達するかが問題となる。現在まで行われてきた発見的手法による方法は、解の構成に人間の経験をおり込み、望ましくない解を捨て、高速に解を得る効果がある。しかし、各部品処理では、最良な解を採用しても、他の部品処理において、初期解の決定が悪影響を与えることがあった。そこで、望ましくない解を採用する確率を0とせず、初期解が後の処理に悪影響を与えないように分割していく手法としてランダマイズドクラスタリング手法を開発した。

本稿では、ランダマイズドクラスタリング手法をJavaにより実装し、階層型並列配置処理の評価をシミュレーションにより行う。また、Javaベース ORB(Object Request Broker)である Voyager を用いて実並列環境を PC クラスタ上に実装し、動作実験を行う。

2. レイアウト設計と配置問題

VLSI設計のフェーズにおけて、本稿で扱う配置処理を含むレイアウト設計について説明する。

2.1. レイアウト設計の流れ

レイアウト設計のフェーズは、大きく分けて2つのフェーズに分けることができる。部品の配置により配線経路が変わり、さらに配線するのに十分な配線領域が必要なことなどから、配置と配線は本来1つの問題であるが、問題が複雑になりすぎることから2つのフェーズに分けて処理するのが一般的である。

レイアウト設計では、部品の情報と部品間の接続関係の情報を入力とし、各部品をチップ上に並べるという配置処理を含む。

次に配置フェーズで配置した部品間の全配線を可能にするべく概略配線を行い、ついで詳細配線で実際の詳細配線径路を決定する。さらに詳細配線で未配線が出ると回路が動作しないので、未配線追加一括配線をする。

最後にレイアウト結果が要求仕様を満たす場合にはそれを出力とし、要求を満たさない場合には制約を変更するなどして配置フェーズ、あるいは配線フェーズの要求仕様を満たすまで反復的に繰り返す[4]。

レイアウト設計は、最適解を求める手間が問題の規模増大に伴って、指数関数的に増加する NP 困難や NP 完全に属する問題であることが実証されており、その解法にはヒューリスティックな手法を用いることが多い。

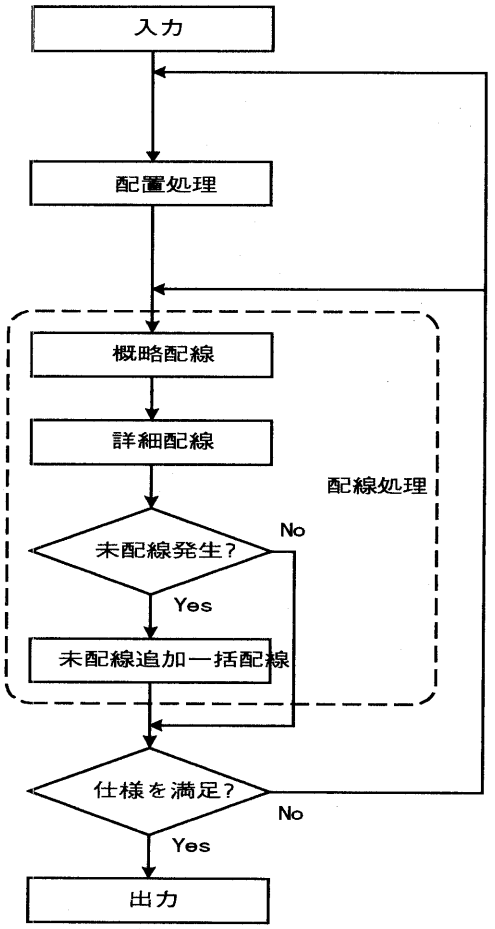

図 1：レイアウト設計の流れ図

2.2. 並列配置問題

VLSI レイアウト設計における配置問題とは、部品の集合を配置領域内に配置し、配置領域原点を基準とした絶対座標を求める問題であり、ある制約条件の下で目的関数を最適化する組合せ最適化問題として定式化できる。

配置問題を解くには、ある配置に対してその必要配線領域面積を求める必要がある。しかし、配線処理後でないと必要配線領域面積を求めることは不可能であり、また 1 配置ごとに配線処理まで実行することは事実上不可能である。配線処理だけでも実用レベルの数万ネットで数時間以上の処理時間がかかるからである。配置の良さを推定する代表的なものとして以下の項目が挙げられる。

1. 仮想配線長合計

   配置フェーズへの入力として部品の接続関係が与えられる。接続関係は同電位の配線要求を持つ端子集合ごとに設定され、それをネットという。

   配線長は、ネットごとに計算する。全ネットの配線長の合計を配線長合計という。配線長同様、仮想配線長もネットごとに求める。実際の配線は配線格子と呼ばれるDA格子よりもやや粗い格子上に径路を定める。仮想配線長は配線格子やDA格子を単位とする。

   仮想配線長の計算法については、本稿では最も高速である、図2に示す最小外接矩形の半周の長さとした。

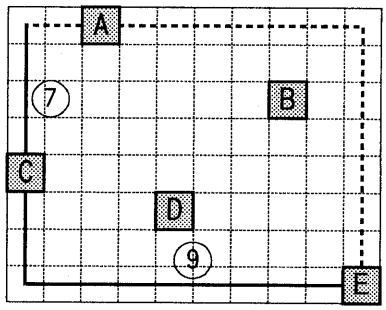

図 2：最小矩形の半周長

2. カット数

   チップ上の垂直または水平のカットライン(cut line)を横切る配線本数。カットラインとは、配置を決めるときに仮設定する、配置領域を分割する分割線を意味する。

以上の項目を用いて配置問題を定義する。

入力　　：　部品の全体集合、ネットリスト
出力　　：　部品の配置絶対座標
目的関数：　仮想配線長合計最小化

本稿では、1の仮想配線長合計最小化および2のカット数最小化を目標とする。

配置問題は NP 完全問題である。この問題を解くためのヒューリスティックなアルゴリズムが既に数多く提案されている[5]、[6]。

ディレイ、消費電力などの電気的特性も最良化し

なければならない。本稿では、並列処理手法に注目しているため、まずは目的関数を上記としている。

3. アルゴリズム

本稿で用いた配置処理の流れについて述べる。

3.1. クラスタ配置

図3に配置処理の流れを示す。初めに部品の全体集合と配置領域の入力が行われる。次に、配置領域をスロットと呼ぶ部分領域に分割するスロット生成処理を行う。そして、次に部品の全体集合から結合度の高い部品同士をまとめてクラスタと呼ぶ部分集合を生成するクラスタリング処理を行う。そして、各クラスタの座標をスロット生成処理で生成されたスロットの対角線の中心にあると近似してクラスタに接続するネットの評価値を求める。この評価値の良いものをそのスロットに配置するクラスタとする。1つの階層レベルでの配置を終えると、スロットを細分化して次のレベルで同様の処理を行う。このようにして、レベルごとの処理を進め、最終的にチップ全体の部品配置座標を決定し、配置結果として出力する。次に本稿で用いているアルゴリズムであるランダマイズドアルゴリズムについて述べる。

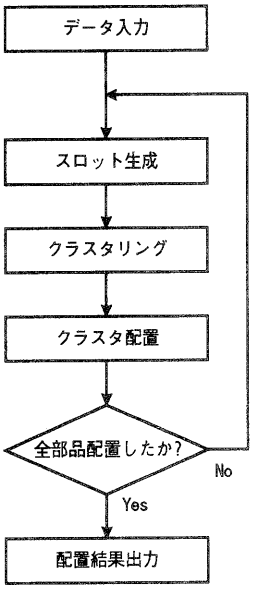

図 3：配置処理の流れ

3.2. ランダマイズドアルゴリズムの概要

ランダマイズドアルゴリズム(Randomized Algorithm)とは、乱数を利用したアルゴリズムである。ランダマイズドアルゴリズムはソーティングやハッシュなどのリスト処理、組合せ問題、グラフ問題、線形・整数計画法、計算幾何学、素数判定等の数論アルゴリズム、分散並列処理など、ほとんどの計算分野で使われている。一般に、ランダマイズドアルゴリズムは乱数を使わない場合に比べて単純で、プログラムするのに簡単であり、計算効率が良い(すなわち速い)ため、実用的である。近年は計算理論の主要学会でも非常に多く取り上げられ、多くの興味深い理論が展開されている。ランダムサンプリングが統計やデータ解析を必要とする様々な分野で活躍している理由と同じく、手軽さ、簡単さがランダマイズドアルゴリズムの最大の優位性である[7]。

3.3. ランダマイズドアルゴリズムによるクラスタ配置処理

3.3.1. 処理の流れについて

図4にランダマイズドアルゴリズムを用いたクラスタ配置の流れを示す。

4分割された配置領域1スロットに、$n(n=1,2,3,4)$個の中から配置すべきクラスタを決定する時、まず、n個のクラスタをそのスロットに仮配置し、それぞれの仮想配線長を計算する。次に、計算された仮想配線長に基づき、そのスロットに対する各クラスタの採択確率を求める。そして、0、1閉区間の乱数を発生させ、採択すべき(配置すべき)クラスタ(以下、採択クラスタと呼ぶ)を決定する。この操作を全スロットに対して行い配置結果を得る。

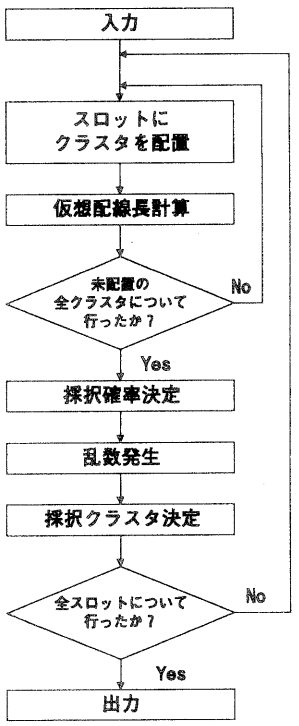

図 4：ランダマイズドクラスタリング配置処理の流れ

3.3.2. 採択確率の計算方法

採択確率は以下の計算式により求められる。

$n(n=1,2,3,4)$個のクラスタの中から、4分割された配置領域の1スロットに配置するクラスタを決定する場合、$k(\leq n)$番目のクラスタのそのスロットに対する採択確率P_kは次式で表される。

$$P_k = (\sum_{i=1}^{n} W_i \Big/ W_k) \Big/ (\sum_{j=1}^{n} (\sum_{i=1}^{n} W_i \Big/ W_j)) \quad [\sum_{i=1}^{n} P_i = 1]$$

ここで、W_kは、k番目のクラスタをスロットに配置したときの仮想配線長を表し、$\sum_{i=1}^{n} W_i$はn個のクラスタをスロットに配置したときの仮想配線長の総和である。この式は、仮想配線長が短いものほど、そのクラスタが採択される確率が高くなるようにするという考え方に基づいて構成した。

仮想配線長最小の配置の採択確率を1、他の配置の採択確率を0とすると、従来の発見的アルゴリズムになる。この意味でこのランダマイズドアルゴリズムは従来の発見的手法の一般化になっている[8]。

4. 階層型並列配置処理

階層型並列配置処理とは、3章で述べたクラスタ配置を階層的に繰り返して配置結果を求める手法である。本章では階層型並列配置手法と並列処理スケジューリングについて述べる。

4.1. 階層型並列配置

階層型並列配置処理の流れは、まず指定された分割数ごとの第一階層の初期配置をサーバ上で行い、その処理により生成された部分問題を保存した配列をクライアントにVoyagerを用いて転送する。クライアントでは非同期にサーバの問題を受理した後、スロット生成、クラスタリング処理を実行し、未配置部品がなくなるまで、同様の処理を続け、処理が終了すると各クライアントごとに結果をサーバ側の配列に保存する。最後にサーバ側では、全ての部品の配置処理を待ち、結果を出力する。

4.2. Voyagerについて

階層型並列処理に必要とされる機能として、以下の4つが挙げられる。

1. データ分配
2. 問題データ転送
3. サーバにおけるデータ管理
4. 処理完了後、データ転送

これらの機能を効率的に実現する為のプログラミング言語について、NOW(Network Of Workstation)方式を前提としたVoyager[1]を用いた。NOW方式とは、EWS、PCをネットワーク結合したクラスタを一つの計算機システムとみなし、その上で従来のプログラミングパラダイムに基づき、プログラムを開発する方式である。Voyagerを用いれば、マシンの種類に依存することなく、他のホスト上に存在するプログラムとの通信もオブジェクト間通信によって、統一的にプログラミング可能である。さまざまなマシン上で動くというJavaの利点を継承しながらも、ホスト間にまたがるオブジェクト間通信を比較的簡単に実現できる言語がVoyagerである[9]。

5. 実験、評価

5.1. 実験の目的

階層型並列手法を用いたクラスタ配置問題に対し、下記の項目に関して実験することを目的とする。

(1) 並列処理の逐次シミュレーションにより、サーバ1台とクライアント4台の処理環境における予想並列処理時間を導出する

(2) Voyagerを用いた階層型並列配置手法を実現する

5.2. 実験環境

5.2.1. 実験データについて

実験には、MCNCベンチマークデータ[2]のindustry1、industry3、avq.small、avq.largeという5種類のデータを使用した。本実験で用いた各ベンチマークデ

---

[1] 米Object Space社の分散プログラミング言語

[2] 米国においては、MCNC(Microelectronics Center of North Carolina)が中心となり、ベンチマーク回路の収集、作成、配布、ワークショップの開催などを行っている。

ータを使用した。本実験で用いた各ベンチマークデータの諸元を表1に示す。

表 1：ベンチマークデータ諸元

データ名	industry1	industry3	avq.small	avq.large
部品数	2271	15059	21854	25114
ネット数	2479	21966	30038	33298

この3種類のデータはそれぞれ特徴があり、ネットが複雑に絡み合っていて、データ読み込み処理やクラスタリング処理に時間を要するデータもある。

5.2.2. EWSについて

予想並列処理時間をシミュレーションする際の実験には、以下の表2に示す性能のマシンを使用した。

表 2：EWSの性能

CPU	Clock(MHz)	メモリ(Mbyte)
UltraSPARCx2	168x2	640

5.2.3. ＰＣクラスタについて

以下の表3に示す性能のマシンを図5に示すように１７台接続して、PCクラスタとよばれる実並列処理環境を使用している。これらのマシンの OS は、Solaris 7 である。

表 3：ＰＣクラスタ性能表

CPU	Clock(MHz)	メモリ (Mbyte)	台数(台)
Pentium Ⅲ	500×2	1000	1(サーバ)
Pentium Ⅲ	450	128	4
Celeron	500	128	5
Celeron	466	128	5
Celeron	433	128	2

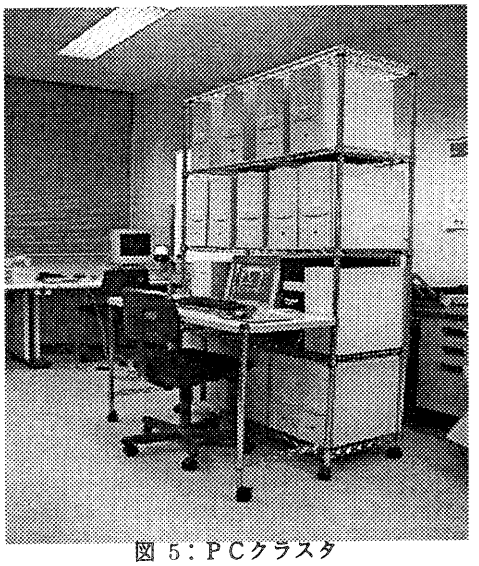

図 5：ＰＣクラスタ

5.3. 実験と評価

5.3.1. 逐次処理と並列処理シミュレーション

[目的]

1台のEWS上において分割問題数4の場合、クライアント台数4台として、総処理時間と仮想配線長総和を求め、逐次処理と並列処理での処理時間の高速化を評価する。

[実験方法]

- クライアント台数：4台
- 分割問題数：4分割
- データ：industry3、avq.small、avq.large
- 並列処理を逐次処理したシミュレーション
- 並列処理時間は、サーバとクライアントの処理時間から、以下のように導出(通信時間を除く)

  並列処理時間＝

  サーバ処理時間＋最遅クライアント処理時間

[実験結果]

図6より、並列処理時間は逐次処理時間の約15～17%高速化することが分かった。しかし、この実験では、サーバとクライアント間に生じる通信時間を考慮していないため、実際の並列環境における処理時間の高速化は、今回の実験よりも減少すると推測される。

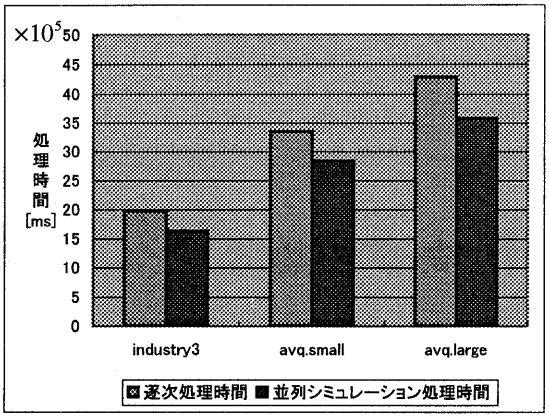

図 6：並列シミュレーション

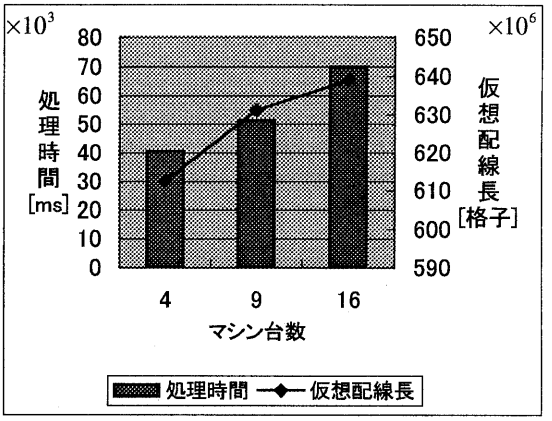

図 7：処理時間と仮想配線長

5.3.2. industry1 における実並列処理の評価

[目的]

PC クラスタを使用し、実並列環境における処理時間と仮想配線長総和の変化を確認する。

[実験方法]

- クライアント台数：4、9、16 台
- 分割問題数：クライアント台数に等しく問題を分割
- データ：industry1
- クライアント処理時間＝最大マシン処理時間とする

[実験結果]

図 7 に示すように、クライアント台数の増加に伴い、処理時間と仮想配線長が増加している。分割問題数の増加に伴い、仮想配線長も増加することは予想していた。しかし、処理時間については、本来ならばクライアント台数の増加に伴って減少するが、今回の実験では増加している。

そこで、総処理時間に占めるサーバおよびクライアントの処理時間を評価し、結果を図 8 に示す。この図より、サーバにおける処理時間はほぼ一定であるのに対し、本来減少していなければならない、クライアントにおける処理時間の増加が全体の処理時間増加に影響を与えていることが分かる。

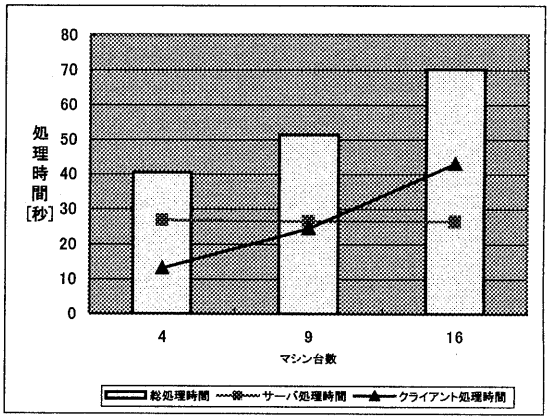

図 8：サーバとクライアント処理時間

クライアントにおける処理の内訳を図 9 に示す。この図に示すように、マシン台数の増加に伴い、分割処理時間および配置処理時間は減少している。これにより、初期問題の分割には成功している。しかし、クライアント処理における通信時間が、マシン台数の増加に伴って増大してしまっている。

この原因として、以下に示す可能性が推測される。

- 問題参照時の競合
  複数のクライアントが同時に一つの配列を参照しているため、クライアントマシン台数が増加すると、参照の際に待ち時間が生じると考えられる。

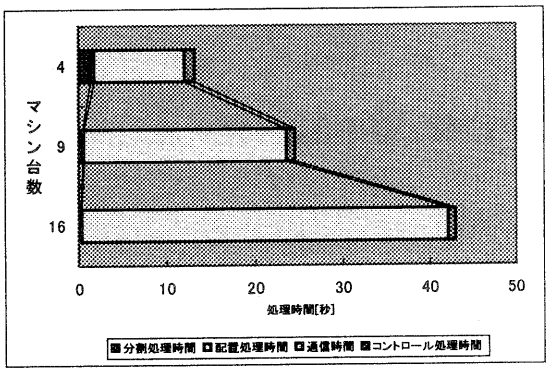

図 9：クライアント処理時間の内訳

6. まとめと今後の課題

本稿では、論理 VLSI 自動レイアウト設計配置問題について、階層型並列配置手法を実装した。そして、MCNC ベンチマークを用いて、階層型並列配置処理を逐次処理したシミュレーション実験を行い、並列処理による高速化を予想した。また、実並列処理環境である PC クラスタを用いて、実並列処理実験を行った。

実並列処理実験では、分散処理機能を確認したが、クライアントによる部分問題参照時の競合が通信時間増大の原因となることを推測した。今後の課題は、クライアントの増加による処理時間増大の原因をさらに厳密に調査し、並列分散処理の効果を確認することである。

謝辞

本アルゴリズムの実装に協力していただいた群馬大学白石研究室出身の芝聡子氏、松井美幸氏、そして同研究室の田中貴明氏を始めとする皆様に感謝いたします。

参考文献

[1] The National Technology Roadmap for Semiconductors 1994 Edition,Semiconductor Industry Association, 1994.
http ://www.sematech.org/pulic/roadmap/1994rdmp.htm

[2] T.Gao, P.M.Vaidya and C.L.Liu, "A New Performance Driven Placement Algorithm," The Digest Papers of the International Conference of Computer-Aided Design, pp.44-47, 1991.

[3] A.Srinivasan, K.Chaudhary and E.S.Kho, "RITUAL: A Performance Driven Placement Algorithm for Small Cell Ics,"The Digest Papers of the International Conference of Computer-Aided Design, pp.48-51, 1991.

[4] 渡辺 誠、浅田 邦博、可児 賢二、大附 辰夫、「VLSI の設計(回路とレイアウト)」、岩波講座マイクロエレクトロニクス、岩波書店、1985.

[5] Goto,S., et al."Partition Assignment and Placement, Layout Design and Verification", ed. By Ohtsuki, T., pp.55-97. Elsevier Science Publishers、1996.

[6] Shing, M.T., et al. "Computational Complexity of Layout Problems. Layout Design and Verification", ed. By Ohtsuki, T., pp.267-294, Elsevier Science Publishers、1986.

[7] 徳山 豪、「ランダマイズドアルゴリズムの話題から」、電子情報通信学会誌、Vol.77、No.9、9 月、1994.

[8] 芝 聡子、松井 美幸、田中 貴明、綿貫 哲久、白石 洋一:「論理 VLSI チップ自動設計における階層型並列部品配置処理手法の検討」.実装 CAE 研究会公開研究会報告 書,
CAE99-6,p.137-144(2000).

[9] ObjectSpace Home of Voyager Application Server and ORB,
http://www.objectspace.com/