ICOT Technical Report: TR-648

TR-648

# 並列オブジェクトモデルに基づく ＬＳＩ配線プログラム

伊達　博、大嶽　能久、瀧　和男

May, 1991



ICOT

Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan

(03)3456-3191～5
Telex ICOT J32964

Institute for New Generation Computer Technology

# 並列オブジェクトモデルに基づくLSI配線プログラム *

伊達 博　　大嶽 能久　　瀧 和男 †
(財) 新世代コンピュータ技術開発機構 ‡

E-mail: date@icot.or.jp

概要. LSI配線は、LSI CADの処理の中では、多大な計算時間を必要とする問題として知られており、並列処理による速度向上は、設計期間の短縮に効果が大きいと期待されている。本論文では、分散メモリ型の大規模な並列マシンに適用できることを目標に、並列オブジェクト指向モデルに基づいた新しい並列配線手法を提案する。この手法は、配線領域における線分のすべてをオブジェクトに対応させ、それらがメッセージを交換しながら与えられた端子間の経路を探索していくものであり、高い並列性を内在しうる。探索の基本アルゴリズムとしては、予測線分探索法を並列化して用い、それをKL1言語を用いて分散メモリ型並列マシンのMulti-PSI上に実装した。LSIの実データを用いて性能評価を開始しており、その初期結果を報告する。

## 1　はじめに

LSIのチップ表面に配置された多数の素子間を結線するために、その配線経路を決めることを、「LSI配線設計」と呼ぶ。LSI配線設計は、LSI設計の各工程の中でも、計算機による自動化の進んでいるものの一つである。また、論理シミュレーションやテスト生成と並んで、多量の計算を必要とする応用問題としてもよく知られている。

LSIの集積度は、およそ2年で2倍の割合で、着実に増加をつづけている。その一方で、配線設計に要する計算時間は、素子数に比例するのではなく、多くの場合、その2乗、あるいはそれ以上の割合で増加する。このことは、LSIの集積度の向上にともなって、その設計時間が急激に増大することを意味しており、高速度の自動配線設計システムの実現が強く望まれる理由となっている。

また素子数の増大により、自動設計で未結線が出た場合の人手介入が、しだいに難しくなっており、未結線を出さない高品質の自動配線設計への要求も高まっている。

我々は、並列処理を用いてこれらの問題を解決すべく研究を続けているが、本稿では、配線設計の高速化を目的とした、新しい並列配線方式を提案するとともに、それを分散メモリ型並列マシン、Multi-PSI上に実装し、初期評価を行った結果を報告する。

以下では、過去の並列配線の事例と本方式の特徴、基本アルゴリズム、並列オブジェクトモデルとKL1言語によるプログラミング、プログラム設計、負荷の割り付け、測定と初期評価、考察、の順に述べる。

## 2　過去の並列配線の事例と本方式の特徴

配線設計の並列処理では、特定のアルゴリズムを実行する特別のハードウェアを設計・試作した例[4, 5, 8]と、汎用的に使用可能な並列マシンの上に、ソフトウェアで並列配線プログラムを実現した例[7, 1, 6, 12, 11]などがある。前者は、処理性能がきわめて高いかわりに、処理の柔軟性に乏しく、後者は、アルゴリズムの変更に対処しやすい反面、処理性能は前者に劣るという傾向がある。今後高い処理性能が実現できたあとは、配線品質の向上に向かいたいことから、今回は、処理の柔軟性を重視し、後者のアプローチをとった。

また後者の中では、SIMD型の並列マシンを用いた例[11]と、MIMD型を用いた例[7, 1, 6, 12]がある。今回の提案は、(1) MIMD型並列マシンは、SIMD型よりも適用領域が広く、将来の汎用並列マシンに成り得ると考え、将来のMIMD型汎用大規模並列マシンに適用できるような並列配線処理方式の実現を目指したこと、(2) 大規模なMIMD型並列マシンに適用できるためには、問題が、大きな並列性を内在する必要があるため、計算の粒度の小さい、並列オブジェクトモデルに基づく新しい配線問題のモデル化を試みたこと、(3) その上で、逐次アルゴリズムの中でも、処理効率と配線品質の両方に優れたアルゴリズムを選んで、それを分散アルゴリズムに設計しなおしたこと、などの特徴を有する。

---

*A Parallel Router based on a Concurrent Object-oriented Model

†Hiroshi DATE, Yoshihisa OHTAKE, Kazuo TAKI

‡Institute for New Generation Computer Technology

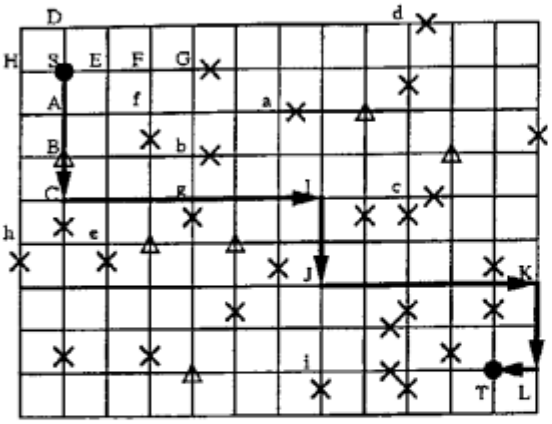


図 1: 先読み線分探索法

具体的には、高い並列性を実現するため、未配線、既配線を含めたすべての配線格子 (LSI 配線では、通常あらかじめ定めた格子上でのみ配線が許される) 上の線分を互いに能動的に動作しうるオブジェクトとしてモデル化した。この場合、配線アルゴリズムは、必然的にオブジェクトが互いにメッセージ交換しながら良い配線経路を決定してゆく分散アルゴリズムとなる。

また線分をオブジェクトにすることとの親和性から、基本アルゴリズムは、計算効率の高い線分探索法がよいと考えた。そしてその中でも迷路法のように配線経路が存在するときの結線可能性を保証している予測線分探索法 [3] を選んだ。予測線分探索法は、逐次アルゴリズムであるため、これを、互いに交差する線分オブジェクトが通信しながら短い配線経路を決定してゆく分散アルゴリズムに設計しなおした。

## 3　基本アルゴリズム

基本アルゴリズムとして採用した予測線分探索法 [3] について簡単に説明しておく。

この予測線分探索法とは、線分探索に”先読み”を加えた手法を基本として、二重探索を避けるフラグとバックトラックなどを導入し、経路が存在すれば必ず発見できることを保証した手法である。本手法では 2 層配線を対象とし、縦方向・横方向で配線層を使い分け、配線は全てあらかじめ固定された配線格子上を通るものとする。更に配線の通過禁止、スルーホール禁止（線の折り曲げの禁止）などの制約も扱う。ただしこれは逐次アルゴリズムである。図 1を例としてその基本処理を示す。Sから探索を開始してTへ接続する場合、Sから下方に向かった線分がA点で折り曲げられたとすると、到達できるTに最も近い点はaである。同様にして、C, Dで折り曲げられたとき到達できる点はc，dとなる。これらの点（a，c，d）をSからの“期待位置”と呼ぶ。B点はスルーホール禁止であり、ここで折り曲げることはできないため、b点は期待位置としない。水平方向に進んだ場合も同様にして期待位置（e，f，g，h）が得られる。これら期待位置（a，c，d，e，f，g，h）の中で最もTに近い点はcである。これより、S－C間を接続する。次にCから水平方向に探索を行ない、Tに最も近い期待位置を求めるとiが得られる。これより、C－I間を接続する。なお、この場合は、I点は前回の期待位置cとは異なっている。同様に探索を続けることによりS－T間を接続する一つの経路S－C－I－J－K－L－Tが求められる。

## 4　並列オブジェクトモデルと KL1

基本アルゴリズムを並列化するにあたって、高い並列性を実現すること、およびプログラミング言語との親和性から、問題を互いに通信しあいながら並列に処理を進めて行く基本要素の組み合わせとしてモデル化した。これはすなわち、並列オブジェクトに基づくモデル化であり、問題から並列性を引き出そうとする時に有効な方法の一つである。

KL1[2, 10] では、このようにモデル化された問題を極めて自然に記述できる。即ち、オブジェクトをプロセスと呼ばれる処理の基本要素とし、それらのプロセスがメッセージストリームを用いて通信し合うように実現することが容易にできる。

例えば図 2に KL1 による並列オブジェクトの実現イメージの一例を示す。並列オブジェクトはそれに対応する述語を、それが受け取るメッセージに対応する一連の節によって定義する。メッセージ列はストリームと呼ばれるリストによって表現される。リストの car 部分が最初のメッセージ、cdr 部分がその後に届くメッセージ列を表す。逆にメッセージを送出するには、それを送り付けたいオブジェクトのメッセージストリームを指す変数に、「car 部分がこれから送るメッセージ、cdr 部分が後に送るメッセージ列を入れるための変数」からなるリストをユニファイすればよい。

## 5　並列プログラミング

線分をオブジェクトとみなす実行モデルに基づき、このような線分のオブジェクトをプロセスとして実

```
concurrent_object ( [ Message_1 | Rest ], 内部状態変数, ストリーム変数 ) : -
  true |
    Message_1に対応する処理,
    内部状態変数の更新,
    ストリーム変数 = [ メッセージ | 新しいストリーム変数 ],          %他のオブジェクトへのメッセージ送出
    concurrent_object ( Rest , 新しい内部状態変数, 新しいストリーム変数 ).

concurrent_object ( [ Message_2 | Rest ], 内部状態変数, ストリーム変数 ) : -
                              ⋮
```

図 2: KL1 による並列オブジェクトの実現例

現する。以下では、オブジェクト＝プロセスとして、プログラムの説明を行う。

配線格子の各格子に対応するプロセスをマスターライン・プロセス、線分に対応するプロセスをライン・プロセスと呼ぶ。これらのライン・プロセスが互いに通信しながら、それぞれの状態に応じた処理を行なうことによって、探索・配線処理が進められる。

本プログラムでは、2通りの並列性を実現している。第1は、1対の端子間を配線するアルゴリズムの中で、予測線分探索法の先読み部分を並列化している。第2は、複数の配線処理を同時並行に実行することによる並列性である。個々の線分プロセスは並列に動作することができるので、探索・配線処理を複数ネットについて、同時に並行して進めることができる。

処理途中の配線状況は各線分プロセスの内部状態として表現される。配線過程では未配線領域が既配線領域と残りの未配線領域に分割されたり、バックトラックによって既配線領域が未配線領域に戻されたりする。それに対応してライン・プロセスは動的に分裂／結合する。すなわち新たな配線の際には、一つの未配線領域のライン・プロセスが、既配線領域と残りの未配線領域のライン・プロセスにそれぞれ分割される。一方バックトラックを行なう場合には、既配線領域を再び未配線の状態に戻さなければならないので、配線の際分割されたライン・プロセスは一つの未配線領域のライン・プロセスとして再結合される。このようにライン・プロセスは探索過程で動的に分裂／結合が行なわれる。

先述の基本アルゴリズムの説明から分かるように、期待位置を求めるためにはライン・プロセスは自分と直交する線分のライン・プロセスと通信しなければならない。図3に示すようにマスターライン・プロセスは互いに直交する線分のライン・プロセス間の通信を仲介し、ライン・プロセスが動的に変化しても、正しく通信が行なわれるようにメッセージの送

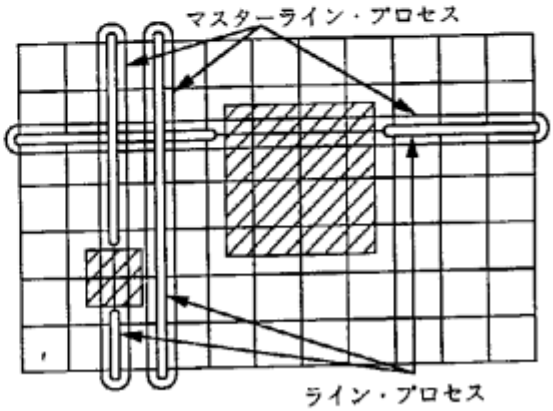


図 3: マスターライン・プロセスとライン・プロセス

り先を管理する。

以下に1ネット探索での先読み処理の並列化の様子を示す。Sをスタート、Tをターゲットとする。Sから垂直方向に期待位置を求める場合、図4に示すようにSを含む未配線領域のライン・プロセスは、自分と直交する同じく未配線領域のライン・プロセスに対して、各領域中でターゲットに最も近い位置、即ち期待位置の計算を依頼する。そして依頼元のライン・プロセスは、すべての計算結果が返されるまで待ちにはいる。

依頼されたライン・プロセスは各自の期待位置の計算結果を依頼元のライン・プロセスに返す。依頼元のライン・プロセスは回答が全て集まった時点で、その中から最もターゲットに近い期待位置を回答してきたライン・プロセスを選び、自分との交点とSとの間を新たな既配線領域とする。すなわち図5に示すように探索中のライン・プロセスが、新たな既配線領域のライン・プロセスと残りの未配線領域のライン・プロセスとに分裂する。

それ以降の探索は図6に示すように、先の最もターゲットに近い期待位置を回答してきたライン・プロセスに制御が移され、同様に進められる。ターゲッ


図 4: 期待値の並列計算

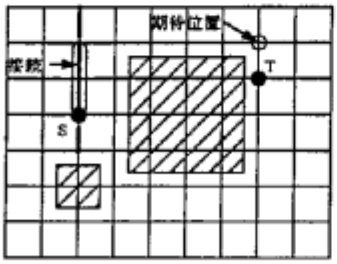


図 5: 経路接続

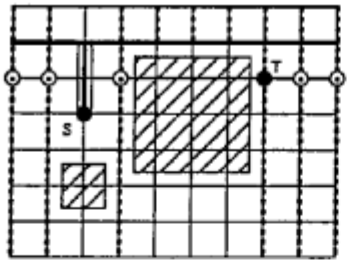


図 6: 並列期待値計算の終了条件

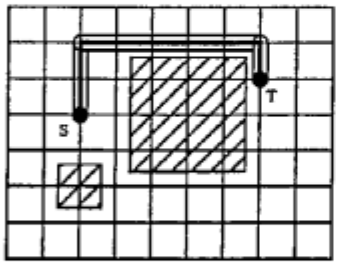


図 7: 配線完了図

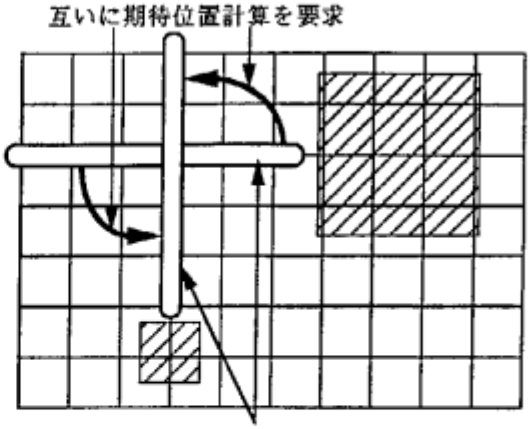


図 8: デッドロックの発生例

ト＝期待位置となるライン・プロセスまで制御が移れば図 7に示すように 1 ネットの配線が終了する。

複数のネットの配線を同時に行おうとする場合、複数のネットが同じ配線領域を取り合うような状況が発生し得る。このような場合、従来行われていた手法として、配線領域をビットマップで表現し、共有データとして刻々の配線状況を管理するものがある。この方法を用いる場合、上記のような競合時にデータの矛盾を発生させないようにするためには、データをある処理単位の間ロックする必要が生じる。しかしながら、共有データのロックは、しばしば並列性を阻害する要因となる。本手法では、すべての配線領域 (線分) を独立のオブジェクトとし、個々のオブジェクトはプロセスによって実現していることから、基本的に一時に 1 個のメッセージしか処理しない。またメッセージは、プロセスへの入力ストリーム上で直列化 (serialize) される。これによりオブジェクトに対する排他制御は、プログラムレベルでの何ら特別な記述を必要とせず、きわめて自然にかつ美しく実現できる。

## 6 デッドロックの回避

上記のような 1 ネットについての探索を、複数のネットについて同時に進めることは、全く独立に実行可能なわけではなく、ある場合にはデッドロックを起こし得る。

例えば図 8のように直交する線分のライン・プロセスがほぼ同時期に探索を行なった場合、先の並列化アルゴリズムの説明にもあるように、双方のライン・プロセスは、直交する線分のライン・プロセスからの期待位置計算結果が全て返って来るのを待っ

て、その結果を集計する。

しかし、プロセスは基本的には一つのメッセージの処理が完了するまでは、それ以降に届いたメッセージに対する処理を行なうことが出来ない。従って現在処理中の探索処理の実行が終わるまでは直交するライン・プロセスからの期待位置計算要求メッセージに対する処理を行なうことはできない。このようにして直交するライン・プロセスが、互いに相手からの期待位置計算結果を待ち合って、デッドロックに陥る場合がある。

一方、一つのメッセージに対する処理を必ず一定時間内に終了するとすることが保証されるならば、このようなデッドロックは起こらない。そこで、オブジェクト内でのメッセージ処理方式をこの条件を満たすように修正する。

メッセージを二つのグループA、Bに分ける。Aは、その処理中に他のオブジェクトに新たなメッセージ送出を行い、その応答を待たないと終了できない種類のメッセージとする。またBは、メッセージ処理が始まると、無条件に一定時間内にその処理を終了し応答を返す種類のメッセージとする。そしてAの処理に入ると、並行してオブジェクトの入口に後から到着してくるメッセージをすべて監視し、その中にBのタイプのメッセージを発見すると、Aの処理中でもBに対する処理を行い応答を返すことにする。これにより上記デッドロック回避の条件を満足する。

これを実現するために、探索処理を行なっている間に届くメッセージを監視し、直交するライン・プロセスからの期待位置計算要求メッセージが届いたならば、それに対して探索処理の開始直前の状態に基づいた仮の値を計算して返すという処理を行なっている。また監視中に届いた新たな探索要求メッセージはバッファにためておく。

## 7 複数ネット間の競合

複数ネットを同時配線すると、異なるネット間で同じ配線領域を取り合う場合が生じることがある。このような状況では、一つのオブジェクトに対するメッセージの順序化により、早いもの勝ちで配線が行われる。後から到着した配線要求は、達成されず、バックトラックを発生する。（普通は、期待位置計算時に既配線を除外するので、はじめから別経路をとるが、競合状態のときにだけバックトラックとなる。）ところが、競合に勝って先に配線領域を確保したネットが、後になって結局バックトラックせざるを得ないことがある。この場合、別ネットが競合した配線領域は、未使用に戻されるが、先に競合に負けたネットは、二度と同領域を探索しない。すなわち同領域は、使われずに残る場合がある。それが配線品質の低下（遠回り）や配線率の低下を招く場合がある。

このような相互干渉が起こりにくくするためには、ネットの投入順や、同時投入の本数を制御することによって、配線率を確保するなどの対処方法が考えられるが、処理の並列性を落とす可能性がある。

## 8 マッピング

KL1 プログラムを現在のターゲットマシンである Multi-PSI[9] 上で効率良く実行させるためには、（1）アルゴリズムの並列性（2）負荷の均等化（3）通信の局所化の三つについて十分に検討しなければならない。この負荷の均等化と通信の局所化を決定付けるのがマッピング方式である。

マッピングとは、各プロセッサへの処理の分配のことである。マッピングでは各プロセッサの負荷の均等をまず考えるが、本試作プログラムのターゲットマシンである Multi-PSI や PIM のような分散メモリ型並列計算機では、プロセッサ間の通信コストが比較的大きいため、プロセッサ間通信を極力抑えることも考慮しなければならない。

先述の並列化アルゴリズムの説明からも分かるように、本アルゴリズムは、配線・探索に必要なメッセージ通信量が多く、それらはプロセッサ間通信となる可能性がある。現在は初期評価のために、負荷の均等化を主に考えたマッピングを行なっており、通信の局所化については、マスターライブン・プロセスとその上のライン・プロセスを同一プロセッサに配置するだけに留めている。そしてそれらのプロセスをプロセッサにランダムに割り当てている。従って本プログラムは、プロセッサ間通信の抑制についてはまだ改良の余地が残されている。

## 9 実験および評価

ここでは、（1）問題の規模と台数効果との関係（2）並列度と配線率との関係（3）汎用計算機との性能比較の3点を明らかにするために二種類の実データを用いて実験を行った。それらのデータの仕様を表1に示す。ＤＡＴＡ１は、接続すべき端子が比較的全体に分散しているデータである。また、ＤＡＴＡ２は、局所的に端子が集中しているデータである。

### 9.1 問題の規模と台数効果

一般的に問題の規模が大きくなれば、並列に動作するプロセスの数が増えるので、台数効果も大きく

表 1: データの仕様

| データ名 | DATA1 | DATA2 |
|---|---|---|
| 格子規模 | 262×106 | 322×389 |
| ネット数 | 136 | 71 |
| IO端子 | ネット無し | ネット有り |
| 提供元 | 日立製作所 | NTT |
| 特徴 | 端子が全体に分散 | 端子が局所的に集中 |

表 2: 実行時間と配線性能 (DATA1)

| データ規模 | PE数 | T(秒) | S | W(%) |
|---|---|---|---|---|
| 1 × 1 | 1 | 123 | 1.1 | 99 |
| | 2 | 135 | 1.0 | 100 |
| | 4 | 92 | 1.5 | 96 |
| | 8 | 51 | 2.6 | 97 |
| | 16 | 32 | 4.2 | 97 |
| | 32 | 22 | 6.1 | 97 |
| | 64 | 18 | 7.5 | 94 |
| 1 × 2 | 1 | 289 | 1.1 | 99 |
| | 2 | 304 | 1.0 | 100 |
| | 4 | 243 | 1.3 | 98 |
| | 8 | 121 | 2.5 | 96 |
| | 16 | 69 | 4.4 | 96 |
| | 32 | 41 | 7.4 | 96 |
| | 64 | 33 | 9.2 | 96 |
| 2 × 2 | 1 | N.A. | N.A. | N.A. |
| | 2 | 829 | 1.0 | 100 |
| | 4 | 616 | 1.3 | 96 |
| | 8 | 301 | 2.8 | 96 |
| | 16 | 181 | 4.6 | 96 |
| | 32 | 98 | 8.5 | 96 |
| | 64 | 67 | 12.4 | 95 |
| 2 × 4 | 1 | N.A. | N.A. | N.A. |
| | 2 | 1769 | 1.0 | 99 |
| | 4 | N.A. | N.A. | N.A. |
| | 8 | N.A. | N.A. | N.A. |
| | 16 | 380 | 4.6 | 96 |
| | 32 | 197 | 9.0 | 96 |
| | 64 | 115 | 15.4 | 97 |

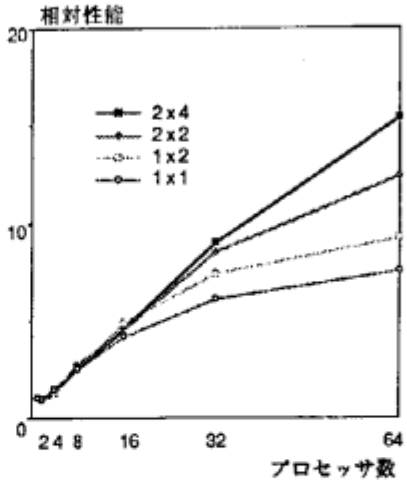


図 9: データ規模と相対性能

なると考えられる。そこで、我々の並列配線プログラムに関して、データの規模と台数効果との関係を測定した。用いたデータは、DATA1を縦方向にn個、横方向にm個コピーしたもので、ここでは、四つの場合（1 × 1, 1 × 2, 2 × 2, 2 × 4）に対して測定した。コピーしたデータに配線プログラムを適用する場合、その配線可能空間が増えるので、計算量も増えることとなる。その測定結果を表2に示す。表中、Sは相対性能を示しており、プロセッサ2台を用いた時の実行時間と比較して何倍速度が速くなったかを表している。またTは、実行時間、Wは、配線率である。これらの結果に対して、それぞれのデータに対する相対性能を一つのグラフとしてまとめたのが、図9である。このグラフからデータの規模が大きくなるに従って相対性能が向上することがわかる。2 × 4データに対して、2台のプロセッサを使用した場合と比較して、64台のプロセッサを用いると約15倍の速度向上が得られることがわかった。

## 9.2 配線率と並列度

前節の実験では、比較的端子が分散しているデータに対して配線率を余り落とさずにかなりの台数効果が得られることがわかった。しかし端子が局所的に集中するデータに対しては、多数のネットを同時に配線すると、配線ネット間の競合が起こり配線率が落ちる可能性がある。そこで、局所的に端子が集中しているデータであるDATA2に関して、64台のプロセッサを用い、同時に配線を実行するネット数Nと配線率Wとの関係を測定したのが表3である。そしてこれらの結果を図10にまとめた。この図にお

表 3: 同時投入ネット数と配線率（ＤＡＴＡ２）

| N | W (%) | T (秒) |
|---|---|---|
| 1 | 97 | 25 |
| 2 | 97 | 21 |
| 3 | 97 | 22 |
| 4 | 95 | 20 |
| 5 | 90 | 21 |
| 10 | 88 | 18 |
| 20 | 81 | 13 |
| 30 | 75 | 14 |
| 40 | 75 | 15 |
| 50 | 78 | 14 |
| 60 | 71 | 13 |
| 71 | 69 | 13 |

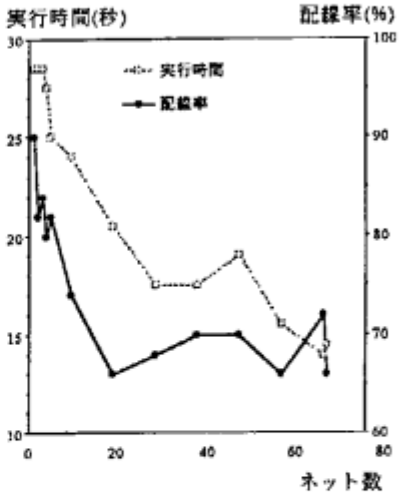


図 10: 配線率と並列度

いて、二つの縦軸は、それぞれ実行時間と配線率である。また、横軸は、同時に配線を実行するネットの数を示している。すなわち、同時に実行するネット数が１ということは、並列処理の効果としては、期待位置の並列計算のみである。この図から、局所的に端子が集中しているようなデータに関しては、複数ネットを同時配線すると配線率が低下してしまうことがわかった。また、このデータに関して、１台のプロセッサを用いたときの実行時間は、１４２秒であった。このことから、期待位置計算部の並列処理による効果は、約５倍であることがわかる。

## 9.3 汎用計算機との性能比較

ＮＴＴＬＳＩ研究所の北沢氏の協力により、IBM 3090/400 (15MIPS) 上に実装された予測線分探索法による配線プログラムを用いて、ＤＡＴＡ２を配線させた結果、7.45秒で１００％の配線率であった。我々の配線プログラムと比較すると、現バージョンでは、約1.7倍から３倍、我々のプログラムの方が遅いことがわかった。配線率の違いは、ＮＴＴのプログラムがアルゴリズムの細部に渡って、配線率向上のための努力をしていることによる。

## 10 まとめ

並列オブジェクト指向モデルに基づく新しい並列配線手法を提案するとともに、それを分散メモリ型マシンの Multi-PSI 上に実現した。LSI の実データを用いて性能の初期評価を行った。その結果、速度向上としては、２台のプロセッサを使用した場合と比較して、64台のプロセッサを用いると約15倍の速度向上を実現した。データ規模の増大につれて速度向上も大きくなる傾向を示しており、大規模データでは、更に高い効率が期待される。また、性質の異なる二種類のデータで評価した結果、端子の位置が分散しているデータでは、台数効果及び配線率とも良好な結果を得た。しかし、端子の位置が局所的に集中しているデータに対しては、期待位置の並列計算の効果はあるが、複数ネットを同時配線すると、ネット間の競合により配線率が低下することが確認された。

今後、更に高速化するために、プロセッサ間通信を抑制するためのプロセスのプロセッサへのマッピング方法を検討して行く予定である。また、端子が局所的に集中しているデータに対して、同時に複数のネットを配線する時に生じる配線率の低下に対しては、効果的な、ネットの同時投入量の制御、未結線ネットの自動再試行などを検討していく。また、大規模な LSI の実データに適用できるようプログラムの改良を行う予定である。

## 謝辞

本研究を行うにあたり、LSI のデータを提供して頂いた日立製作所中央研究所の白石洋一氏、データの提供と共に、有益な助言を頂いたＮＴＴ ＬＳＩ研究所の北沢仁志氏、また、活発に御討論いただいたＰＩＣワーキンググループの諸氏に深謝します。

## A 配線出力結果

我々の配線プログラムをＤＡＴＡ１及びＤＡＴＡ２に適用した出力結果を次に示す。

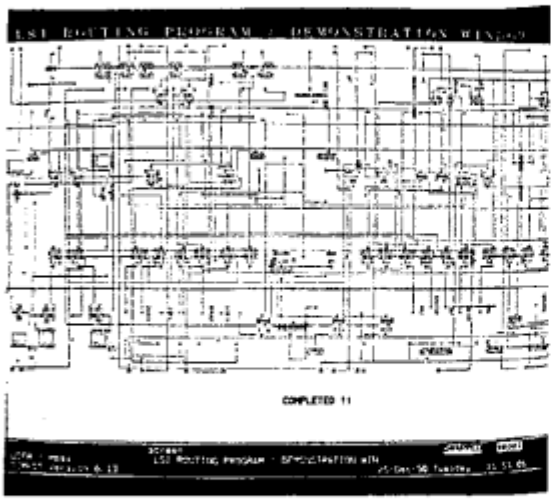

図 11: DATA 1の配線結果（一部分）

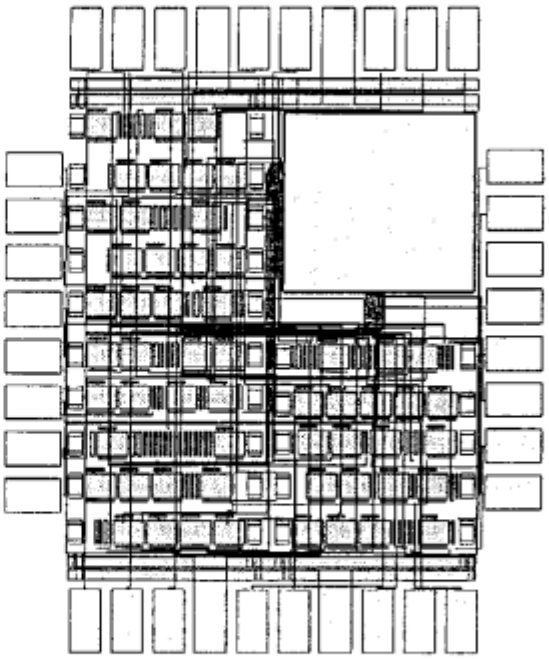

図 12: DATA 2の配線結果

## 参考文献


[1] R.J. Brouwer and P. Banerjee, "PHIGURE:A Parallel Hierarchical Global Router", *Proc. 27th Design Automation Conf.* , pp.650–653, 1990.

[2] T. Chikayama, H. Sato and T. Miyazaki, "Overview of the parallel inference machine operating system (PIMOS)" , *Proceedings of International Conference on Fifth Generation Computer Systems 1988* , pp.230–251, 1988.

[3] 北沢 仁志, "高配線率線分探索の一手法" , 情報処理 , 第26巻第11号,pp.1366–1375, 1985.

[4] K. Kawamura, T. Shindo, H. Miwatari and Y. Ohki, "Touch and Cross Router," *Proc. IEEE ICCAD90* , pp.56–59, 1990.

[5] R. Nair, S. J. Hong, S. Liles and R. Villani, "Global Wiring on a Wire Routing Machine" , *Proc. 19th Design Automation Conf.* , pp.224–231, 1982.

[6] O.A. Olukotun and T.N. Mudge, "A Preliminary Investigation into Parallel Routing on a Hypercube Computer" , *Proc. 24th Design Automation Conf.* , pp.814–820, 1987.

[7] J. Rose, "Locusroute:A Parallel Global Router for Standard Cells", *Proc. 25th Design Automation Conf.* , pp.189–195, 1988.

[8] K. Suzuki, Y. Matsunaga, M. Tachibana and T. Ohtsuki, "A Hardware Maze Router with Application to Interactive Rip-up and Reroute" , *IEEE Trans. on CAD* , vol.CAD-5, no.4, pp.466–476, 1986.

[9] K. Taki, "The parallel software research and development tool: Multi-PSI system", *Programming of Future Generaiton Computers*, North-Holland, pp. 411–426, 1988.

[10] K. Ueda, "Guarded Horn Clauses: A Parallel Logic Programming Language with the Concept of a Guard" , Technical Report TR-208, ICOT, 1986.

[11] T. Watanabe, H. Kitazawa, Y. Sugiyama, "A Parallel Adaptable Routing Algorithm and its Implementation on a Two-Dimensional Array Processor" , *IEEE Trans. on CAD* , vol.CAD-6, no.2, pp.241–250, 1987.

[12] Y. Won, S. Sahni and Y. El-Ziq, "A Hardware Accelerator for Maze Routing" , *Proc. 24th Design Automation Conf.* , pp.800–806, 1987.

# ＬＳＩ配線設計のシナリオ(改訂版)

１９９１．２．２６伊達作成

## 1　背景と目的

【画面】　ＬＳＩ設計のフロー

計算機を構成している部品であるＬＳＩは、とても複雑であり、その設計にはたくさんの工程があります。（Ｒ１）これは、その設計フローを示したものです。ＩＣＯＴでは、このフローの中で、特に多大の処理時間がかかるとされているＬＳＩの配置配線処理と論理シミュレーションに対して並列処理による高速化の研究を進めています。最初にＬＳＩの配線処理への並列処理応用について紹介しましょう。

【画面】　ＬＳＩチップの拡大写真

それでは実際にＬＳＩの中身を見てみましょう。このようにＬＳＩ内では、トランジスタ等の素子が配置され、それらの端子間が配線されています。ＬＳＩの配線処理とは、このような複雑な配線を実現することです。年々、ＬＳＩは高密度，高集積化の傾向にあり、高速に上質な配線が得られる手法が望まれています。

## 2　問題の説明

【画面】　配線格子の図

この配線処理というのは（Ｒ２）ＩＣの表面にある部品同士の端子をつなげるのにあらかじめきめられた格子上（Ｒ３）を通って接続する（Ｒ４，Ｒ５）問題です。ここでいう格子は、二つの配線層をもっており、一層目は、縦方向の配線（Ｒ６）、もう一つは、横方向の配線（Ｒ７）に使用されます。

【画面】　システム画面の右上にパフォーマンスメータ 16 台版実行

それでは次に実際に並列配線プログラムが動作している様子をみながら我々の並列処理のアイデアについて説明しましょう。

## 3　技術上のアピールポイントと解法

これは、ＬＳＩの実データに対して 16 台のプロセッサを用いてＬＳＩの配線処理を実行している様子です。このシステムで用いている基本的な配線手法は、配線速度，配線品質共に良好な予測線分探索法と呼ばれるものです。我々は、この方法を新しい並列処理のモデルを用いて高速化しました。

【画面】　格子上の線分の図

それではここで、そのモデルと実現方法について説明しましょう。

（Ｒ８）線分探索法の処理単位は、このような線分です。

【画面】　格子上の線分に対するオブジェクトの図

よって、格子上で使えるあらゆる線分をオブジェクト（Ｒ９）と呼ばれる能動的に活動できる処理単位とします。

【画面】　オブジェクト間通信の図

そして、複数のオブジェクトが接続すべき端子間をメッセージを交換しながら配線するモデルにしました。

配線処理が終了したようです。実行時間は５２秒です。

またより良い経路を探すために一段づつの先読みを行っています。

例えば（Ｒ１０）これらの端子間を接続する場合、最初に端子を含んでいる線分オブジェクト

（Ｒ１１）は交差する線分オブジェクトとメッセージを交換します。（Ｒ１２）そしてターゲット点に一番近い点（Ｒ１３）に到達できる線分オブジェクトを探します。（Ｒ１４）これを先読み処理（Ｒ１５）と呼びます。そしてこれらのオブジェクトの交点まで配線します。（Ｒ１６）この一連の処理を繰り返してターゲットまでの配線経路が決定されるわけです。（Ｒ１７）

【画面】　配線の拡大画面

このモデルでは、一つ一つの線分オブジェクトがメッセージを交換しあいながら互いに独立に動作するので、高い並列性が実現できます。

## 4　プログラムの実行の様子

【画面】　システム画面の右上にパフォーマンスメータ６４台版実行

それでは、同じデータに対して６４台のプロセッサを用いたものを見てみましょう。

右上画面のパフォーマンスメータを見てみると６４台のプロセッサが、効率良く稼働している様子がわかります。

このシステムでは、複数の配線を同時に行ない、高速配線を実現しています。

実行時間は、１６秒です。

先ほどのプロセッサを１６台使用した場合と比較して３倍の高速化となっています。

## 5　結果のコメント

【画面】　性能向上のグラフ

（Ｒ１８）現在、実験機であるマルチＰＳＩ上で動作する並列配線プログラムの処理速度は、汎用大型計算機と同程度です。 今後、並列推論マシンＰＩＭ上では、数十倍性能が向上し、更に、順調に技術が進歩すれば、飛躍的なＬＳＩの設計工数の短縮につながると、我々は期待しています。