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††

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 KL 1 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．本手法の性能を LSI の実データを用いて評価した結果，64 台のプロセッサを用いて，16 倍の台数効果を得た．また，汎用計算機と処理速度を比較した結果，ほぼ同程度であることがわかった．

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 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，素子数に比例もるのではなく，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．

この要求を満たすために，並列処理を用いた高速な LSI 配線設計法の研究がなされている．特定のアルゴリズムを実行する特別なハードウェアを設計・試作した例[3),5),8)]，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[1),6),7),9),12),13)]などが報告されている．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．われわれは，高い処理性能が実現できたあとは，配線品質の向上を図りたいことから，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[12)]と MIMD 型を用いた例[1),6),7),9),13)]がある．われわれは，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を目標に置いた．

一般に大規模な MIMD 型並列マシンを用いて効率良く問題を解かせる場合，その問題が，大きな並列性を内在する必要がある．そのため本稿では，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[10)] 上に実装した結果を報告する．また，初期評価として，問題の規模と処理効率，並列性と配線率の関係，汎用マシンとの性能比較に関する計測結果を報告する．

以下では，最初に今回使用した並列プログラミング手法について説明し，つづいて並列配線プログラム，並列化に伴う問題点と対処法，計測評価と考察の順に述べる．

2. 並列プログラミングの手法

並列オブジェクトモデル (Concurrent Objects Model) に基づく問題の定式化は，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．ここでは，問題の定式化から，並列実行に至るまでの並列プログラム設計法について，今回使用した方法の概要と KL 1 言語による記述方法について述べる．

2.1 並列オブジェクトモデルに基づく並列プログラムの設計法

図 1 に，今回使用した並列プログラムの設計方法の概念図を示す．最初に問題が与えられると，それを複数の能動的に動作するオブジェクトがメッセージを交換しながら問題を解決するモデル（並列オブジェクト

† A Parallel LSI Router Based on the Concurrent Objects Model by HIROSHI DATE, YOSHIHISA OHTAKE and KAZUO TAKI (The Seventh Laboratory, Institute for New Generation Computer Technology).

†† (財)新世代コンピュータ技術開発機構第 7 研究室

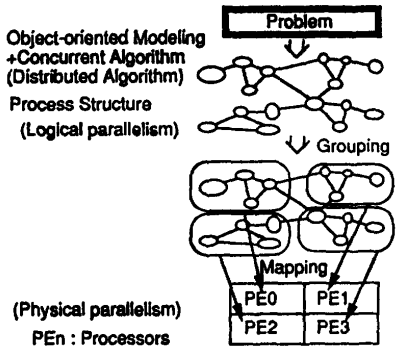

図 1 並列オブジェクトモデルに基づくプログラム設計法
Fig. 1 Program design paradigm based on the concurrent objects model.

モデル）に合わせて定式化する．同時に問題解決の並列アルゴリズムを設計する．これは，メッセージに対するオブジェクトの動作を設計することであり，分散アルゴリズムの設計とも言える．つぎに，オブジェクトをプロセスとして記述することにより，プロセス構造とプロセスの動作が決まる．この時点でプログラムの論理的並列性の設計が完了する．この方法により，プロセッサ台数に比べ，十分大きな並列性を内在させるようにプログラムする．

次にこれらのプロセスの中で頻繁にメッセージを交換するもの同士をグループ分けし，グループを単位としてプロセッサに割り付ける．このことにより実行時の物理的な並列性が抽出される．この段階では，プロセッサ間の負荷の均等化とプロセッサ間通信の減少の両方を実現するように，グループ分けと割り付けを決定する．処理効率の改善が必要な場合は，この部分を調整する．なお，本設計法の詳細については，別稿にまとめる予定である．

2.2 KL 1 による並列オブジェクトの記述

KL 1 言語[2),11)]では，並列オブジェクトモデルに基づいて設計された並列プログラムを極めて自然に記述できる．すなわち，オブジェクトを表現するプロセスは，KL 1 言語の自己再帰呼出しを用いて実現し，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．

図 2 に KL 1 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一連の節によって定義される．その各節は，それが受け取るメッセージに対応している．また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しており，リストのユニフィケーションによってメッセージを受け取る．そして各節のボディー部にメッセージを受け取った場合の処理が記述されている．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表す変数に，「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．

3. 並列配線プログラム

従来提案されている配線アルゴリズムの代表的なものとしてチャネル配線法，線分探索法，迷路法などがある．基本的なアルゴリズムの選択に当たっては，並列実行で高速配線処理を実現することを目指し，

a. 計算の効率が良いこと
b. 配線率が高いこと
c. 適用範囲が広いこと
d. 並列化に適すること

の 4 点を考慮した．チャネル配線法は，c. に，迷路法は，a. に問題があるが，線分探索法を改良したものは，ほぼすべてを満足する．特に，2. で述べた並列オブジェクトモデルと親和性が良いのは線分探索法である．すなわち，線分探索法は線分を処理単位とするので，各線分をオブジェクトに対応させると，自然に並列オブジェクトモデルとして定式化でき，高い並列性を内在させ得る．よって基本アルゴリズムとして線分探索法に改良を加えた予測線分探索法[4)]を選び，それを分散アルゴリズムとして設計し直した．

```
Concurrent_Object ( [ Message_1 | Rest ], Interior state variables, Stream variable ) :-
    True |
        Process corespond to Message_1,
        Renew of Interior state variables,
        Stream variable = [Message | New Stream variable],
        Concurrent_Object ( Rest , New Interior state variables, New Stream variable ).

Concurrent_Object ( [ Message_2 | Rest ], Interior state variables, Stream variable ) :-
        ⋮
```

Output of Message to other objects

図 2 KL 1 による並列オブジェクトの実現例
Fig. 2 Implementation of concurrent object in KL 1.

3.1 基本アルゴリズムの概要

基本アルゴリズムとして採用した予測線分探索法[4]とは，線分探索に“先読み”を加えた逐次配線手法で，縦方向・横方向で配線層を使い分ける2層配線を対象としている．

また，予測線分探索法は，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ている．さらに配線の通過禁止，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．

ここで，“先読み”について簡単に説明する．今，始点から目標点に向かって経路探索を行う場合，折れ曲がり一回で到達可能な点の内で，目標点に一番近い点を探す．これを“先読み”と呼ぶ．次に始点からその折れ曲がり点までを結線する．この一連の処理を繰り返して，目標点までの配線経路を決定する．

予測線分探索法は，一回の先読みで得られた評価値の，もっともよい方角へ進路をとってゆく，一種の山登り法ということができる．しかし袋小路に入ると探索が終了してしまう．すなわち“局所最適点”で停止してしまうので，そこから脱出するために，探索領域を制限する方法として，IEP フラグを導入している．これにより，最短経路に比べると遠回りになることが多いが，局所最適点から脱出できる．さらに，IEP フラグを用いても期待位置が求まらないことがあるが，その場合は真の行きどまりなので，バックトラック処理を行い，一つ前の探索線に戻って，別の経路を探す．それでも良い経路がみつからなかったら，バックトラックを繰り返す．スタート点に戻ってしまったら，経路が存在しなかったことを意味する．このようにして経路が存在すれば必ず発見できることを保証している．

3.2 並列化とプロセス構造の設計

3.2.1 基本プロセス構造

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．

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現した．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．

図3に示すように配線格子上の各線分に対応するプロセスをライン・プロセス，配線格子の縦または横1本ずつに対応するプロセスをマスタライン・プロセスと呼ぶ．そしてこれらのライン・プロセスは，互いに通信しながら，それぞれの状態に応じた処理を行うことによって，探索・配線処理が進められる．このときマスタライン・プロセスは，対応する格子上のライン・プロセス群を管理するとともに，直交するライン・プロセスから自らの管理下のライン・プロセスへ送られるメッセージを仲介する．

処理途中の配線状況は各ライン・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．

3.2.2 2通りの並列性

本プロセスでは，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．

3.2.3 先読みの並列実行例

以下に1ネット探索での先読み処理の並列化の様子

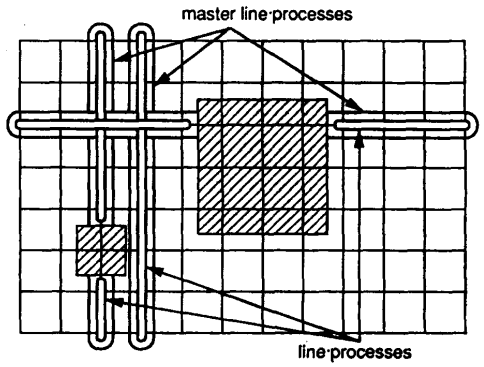

図3 マスタライン・プロセスとライン・プロセス
Fig. 3 Master line-processes and line-processes.

を示す．S を始点，T を目標点とする．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，各領域中で目標点に最も近い位置，すなわ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・プロセスは回答がすべて集まった時点で，その中から最も目標点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，先の最も目標点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．目標点＝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．

以上の処理において，目標点 T に到達する前に経路探索が終了してしまう場合は，逐次版の予測線分探索法と同様に IEP フラグを用いて迂回路を探索する．それでも期待位置が求まらない場合は，バックトラック処理を行い，一つ前の探索線に戻って経路探索を続ける．バックトラック処理の結果，始点 S にもどってしまったら経路が存在しなかったとする．

3.2.4 排他制御について

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．

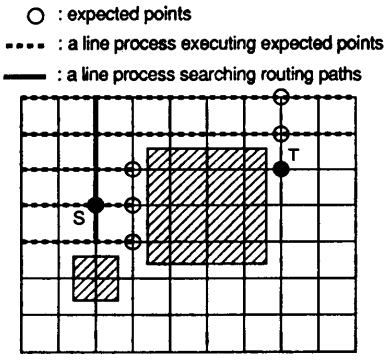

図 4 期待位置の並列計算
Fig. 4 Parallel execution of expected points.

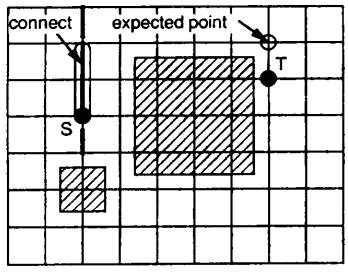

図 5 経路接続
Fig. 5 Connection of routing path.

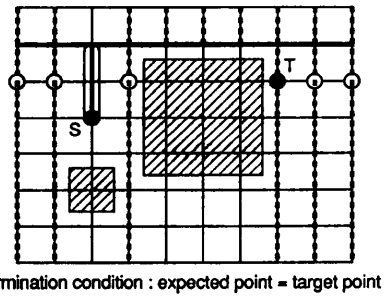

図 6 並列期待位置計算の終了条件
Fig. 6 Termination condition of parallel execution of expected points.

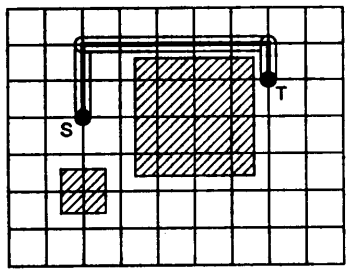

図 7 配線完了図
Fig. 7 Completion of routing.

3.3 マッピング設計

KL1 プログラムを現在のターゲットマシンである Multi-PSI[10] 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，Multi-PSI 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．

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っており，通信の局所化については，マスタライン・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．今後，プロセッサ間通信の抑制につながるプロセッサマッピング方式について検討していく．

4. 並列化に伴う問題点と対処法

並列に予測線分探索法を行おうとすると二つの問題点が生じた．一つは，デッドロックの問題，もう一つは，複数ネット間の競合の問題である．

4.1 デッドロックの回避

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った場合，先読み処理の並列化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すべて返って来るのを待って，その結果を集計する．

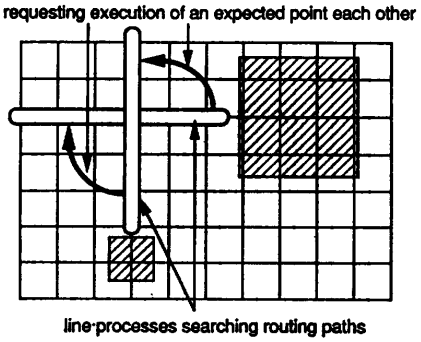

図 8 デッドロックの発生例
Fig. 8 Example of deadlock.

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うことができない．したが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．

4.2 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，別の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必要となるが，処理の並列性を制限することになり，今後の検討課題と考えている．

このような相互干渉がどの程度配線率を低下させるのかの一例を実験によって調べたので次章で報告する．

5. 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 LSI 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．DATA 1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，DATA 2は，局所的に端子が集中しているデータである．

5.1 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，台数効果も大きくなると考えられる．そこで，われわれ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，DATA 1 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，マスタライン・プロセスは，コピーした全領域で共通しているので，マスタライン上にあるライン・プロセスとの通信量が増える．また，コピーして領域が広がったことにより領域の境界付近にあるネットの探索面積が増える．その測定結果を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 のデータに対して，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 16 倍の速度向上が得られることがわかった．今後，どの程度の規模のデータまで台数効果が向上するのかを調べる予定である．

表 1 データの仕様
Table 1 Testing data.

データ名	DATA 1	DATA 2
格子規模	262×106	322×389
ネット数	136	71
IO 端子	ネットなし	ネットあり
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

5.2 配線率と並列度

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，DATA 1 と DATA 2 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 との関係について測定した結果を**図 10** にまとめた．この図において，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，端子位置が分散しているデータでは，配線率をあまり落とさずに処理速度も向上するが，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，DATA 2 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，5.7 倍であることがわかる．

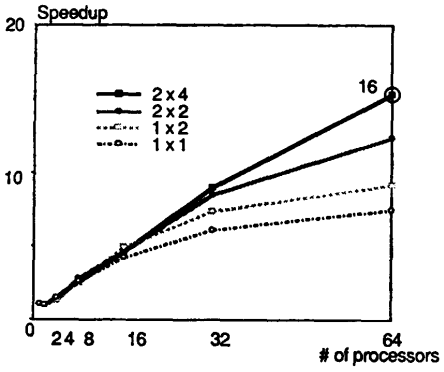

図 9 データ規模と相対性能
Fig. 9 Data size vs. speedup.

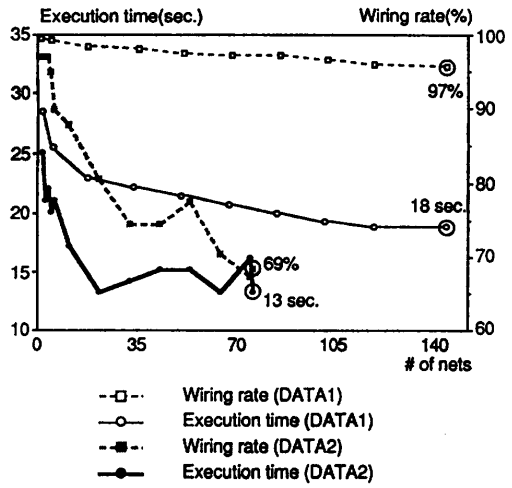

図 10 配線率と並列度
Fig. 10 Wiring rate vs. parallelism.

5.3 汎用計算機と性能比較

NTT LSI研究所の北沢氏の協力により，IBM 3090/400 (15 MIPS) 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，DATA 2 を配線させた結果，7.45 秒で 100% の配線率であった．われわれ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 3.4 倍，われわれのプログラムの方が遅い*．しかしながら，このデータに関する台数効果が 10 倍から5倍であることから，台数効果が，16 倍の (並列処理効率の高い) DATA 1 では，処理性能は，ほぼ同程度になるものと考える．

Multi-PSI の単一プロセッサ性能は，マイクロ命令の処理速度で 5 MIPS，汎用命令に換算すると 2〜3 MIPS 相当である．それを 16 倍すると 32〜48 MIPS 出るべきところである．IBM 3090/400 の 15 MIPS と同等の実行性能しかないとすると，2分の1から3分の1の効率といえる．これは，KL 1 言語の実装オーバヘッドとオブジェクト指向プログラムの実行オーバヘッドによるものと考えられるが，オブジェクト指向プログラムのオーバヘッドとしては，とりたてて大きな値とはいえない．

6. おわりに

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．そして LSI のデータを用いて性能の初期評価を行った．

速度向上に関する実験では，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 16 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，さらに高い効率が期待される．

並列度と配線率に関する実験では，端子の位置が分散しているデータでは，台数効果およ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として 64 台のプロセッサを用いて約5倍処理速度が向上するが，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．この問題に関しては，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，汎用計算機 (IBM 3090/400) 上の配線プログラムと Multi-PSI 上の並列配線プログラムを処理速度に関して比較した結果，ほぼ同程度であることがわかった．

今後，さらに大規模な LSI 実データへの適用と詳細評価を行うとともに，高速化のためのマッピング方式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いただ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，有益な助言をいただいた NTT LSI 研究所の北沢仁志氏，また，活発にご討論いただいた ICOT PIC ワーキンググループの諸氏に深謝します．

* われわれの配線プログラムは，単一プロセッサの実行でも，97% の配線率しか得られないが，これは，ネット投入順序他のノウハウが未熟なことによる．未配線がある場合，バックトラックをともなうため，計算量は増加傾向となる．

参考文献

1) Brouwer, R. J. and Banerjee, P.: PHIGURE: A Parallel Hierarchical Global Router, *Proc. 27th Design Automation Conf.*, pp. 650-653 (1990).
2) Chikayama, T., Sato, H. and Miyazaki, T.: Overview of the Parallel Inference Machine Operating System (PIMOS), *Proceedings of International Conference on Fifth Generation Computer Systems 1988*, pp. 230-251 (1988).
3) Kawamura, K., Shindo, T., Miwatari, H. and Ohki, Y.: Touch and Cross Router, *Proc. IEEE ICCAD 90*, pp. 56-59 (1990).
4) 北沢仁志：高配線率線分探索の一手法，情報処理，Vol. 26, No. 11, pp. 1366-1375 (1985).
5) Nair, R., Hong, S. J., Liles, S. and Villani, R.: Global Wiring on a Wire Routing Machine, *Proc. 19th Design Automation Conf.*, pp. 224-

231 (1982).
6) Olukotun, O. A. and Mudge, T. N.: A Preliminary Investigation into Parallel Routing on a Hypercube Computer, *Proc. 24th Design Automation Conf.*, pp. 814-820 (1987).
7) Rose, J.: Locusroute: A Parallel Global Router for Standard Cells, *Proc. 25th Design Automation Conf.*, pp. 189-195 (1988).
8) Suzuki, K., Matsunaga, Y., Tachibana, M. and Ohtsuki, T.: A Hardware Maze Router with Application to Interactive Rip-up and Reroute, *IEEE Trans. CAD*, Vol. CAD-5, No. 4, pp. 466-476 (1986).
9) 高橋義造, 佐々木茂高: 競合プロセッサ群による配線問題の並列処理, 情報処理学会計算機アーキテクチャ研究会報告, 90-ARC-82, No. 82-7 (1990. 4).
10) Taki, K.: The Parallel Software Research and Development Tool: Multi-PSI System, *Programming of Future Generation Computers*, pp. 411-426, North-Holland (1988).
11) Ueda, K. and Chikayama, T.: Design of the Kernel Language for the Parallel Inference Machine, *Comput. J.*, Vol. 33, No. 6, pp. 494-500 (1990).
12) Watanabe, T., Kitazawa, H. and Sugiyama, Y.: A Parallel Adaptable Routing Algorithm and Its Implementation on a Two-Dimensional Array Processor, *IEEE Trans. CAD*, Vol. CAD-6, No. 2, pp. 241-250 (1987).
13) Won, Y., Sahni, S. and El-Ziq, Y.: A Hardware Accelerator for Maze Routing, *Proc. 24th Design Automation Conf.*, pp. 800-806 (1987).

付録 配線出力結果

われわれの配線プログラムを DATA1 と DATA2 に適用した出力結果を図 11 と図 12 に各々示す.

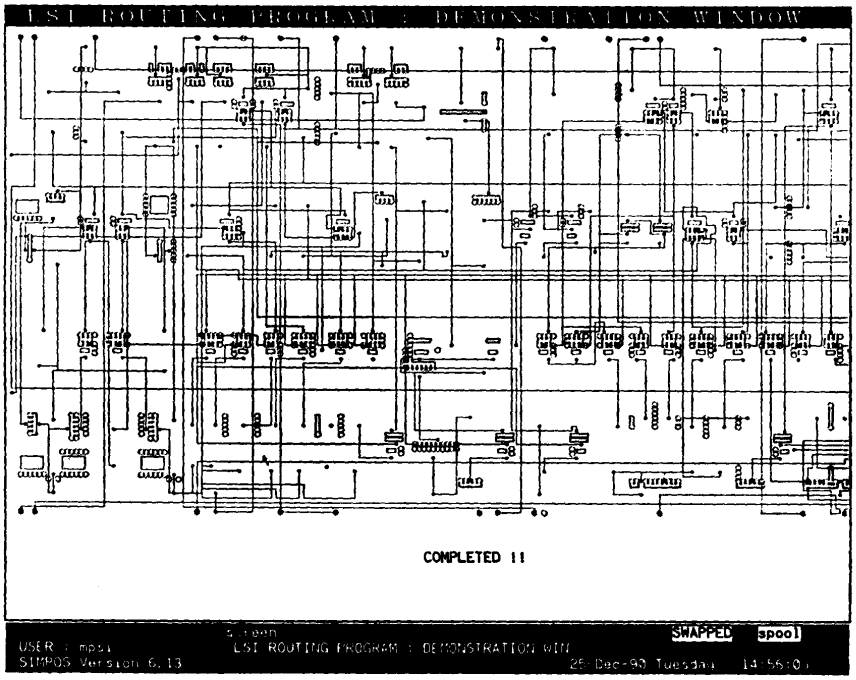

図 11 配線結果の一部 (DATA 1)
Fig. 11 A part of routing result (DATA 1).

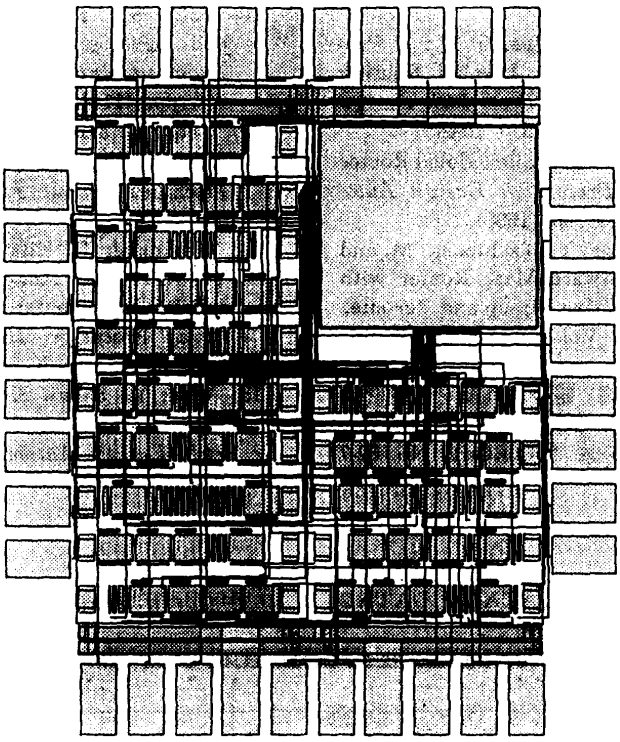

図 12 配線結果 (DATA 2)
Fig. 12 Routing result (DATA 2).

(平成3年 8 月1日受付)
(平成3年12月9日採録)

伊達 博 (正会員)

昭和36年生. 昭和62年九州大学大学院理学研究科数学専攻修士課程修了. 同年, (株)日立製作所入社. 現在, (株)日立製作所・日立研究所所属. 平成2年 10 月より, (財)新世代コンピュータ技術開発機構に出向中. 電子回路の設計自動化, 並列処理に関する研究に従事. 電子情報通信学会, IEEE 各会員.

大嶽 能久 (正会員)

昭和33年生. 昭和59年京都大学大学院工学研究科精密工学専攻修士課程修了. 同年, 東京芝浦電気(株)入社. 現在(株)東芝・総合研究所情報システム研究所所属. 平成元年4月より(財)新世代コンピュータ技術開発機構に出向中. 並列プログラミングの研究に従事. 人工知能学会会員.

瀧 和男 (正会員)

昭和27年生. 昭和51年神戸大学工学部電子工学科卒業. 昭和54年同大学院修士課程システム工学修了. 工学博士. 同年(株)日立製作所入社. 同社大みか工場にて制御用計算機システムの設計に従事. 昭和57年(財)新世代コンピュータ技術開発機構に出向. 以来逐次型および並列型推論マシンと並列応用プログラムの研究開発に従事. 現在第1研究室室長. 並列マシンのアーキテクチャ, 並列プログラミング, LSI CAD などに興味を持つ. 電子情報通信学会, IEEE, ソフトウェア科学会各会員.