ICOT Technical Memorandom: TM-1250

TM-1250

タイムワープによる並列無格子配線システム

松本　幸則、瀧　和男（神戸大学）

March, 1993

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191～5

Institute for New Generation Computer Technology

タイムワープによる並列無格子配線システム

松本 幸則　　瀧 和男

1　はじめに

配線設計はLSI設計工程の一つであり、多大な処理時間を必要とする。このため、従来から計算機を用いた自動配線システム(ルータ)が利用されてきた。しかし、LSIの大規模化が進んだ現在、ルータに対する高速化要求が極めて強くなっている。また、半導体製造技術の進歩に追従するための柔軟性も同時に要求される。

ルータの高速化方法としては、ハードウェアエンジンの利用と並列計算機の利用という二つの選択枝がある。前者は、高速ルータを実現する有効な方法であるが、柔軟性の面で問題がある。これに対し、後者は高速性・柔軟性の両面で有望であると考えられる。我々は、大規模並列計算機上で動作する高速かつ柔軟なルータの実現を目指し、分散メモリ並列計算機上にルータを試作し、その評価を行なった。

並列ルータには、逐次ルータと同等の高い配線品質を維持しつつ高速に配線処理することが要求される。高い配線品質を維持するためには、予め注意深く決定されている配線順序を守るべきと考えられる。なぜなら、配線処理では、一つのネット(配線されるべき端子の組)の配線経路は、すでに引かれた他のネットの経路を避けなければならないことから、配線結果の品質は配線順序に大きく依存するためである。一方、高速処理を行なうためには高い並列性を抽出する必要がある。このためには、複数のネットの経路探索を同時に実行することが望ましい。

効率的な並列処理を行ないつつ、高品質な解を得るルータ実現のためには、上記の相反する二つの要求を満たす必要がある。我々はこれらを共に実現する方法としてタイムワープ機構[1]に着目し、この機構を組み込んだ並列ルータを試作した。タイムワープ機構は、対象問題の持つ逐次性(処理順序に関する制約)を守りつつ、積極的に見込み計算を行うことによって高い並列性を抽出する機構である。既に、並列イベントシミュレーションの分野で用いられ、効率良い並列性抽出ができることが報告されている[8]。しかしながら、未だ配線処理に適用された例はない。なお、本ルータでは、基本配線アルゴリズムとして矩形分割に基づく無格子配線アルゴリズム[2]を用いている。このアルゴリズムは最適経路を得るとともに、自由な配線幅を扱える手法であり、1)端子・経路の位置を柔軟に設定できるため、より小さいチップ面積を実現できる 2)デジタル回路とアナログ回路の混在にも対応できるなどの特徴をもつ。タイムワープ機構は、他の配線アルゴリズムにも適用可能であるが、無格子配線アルゴリズムは将来的にも極めて有望な配線手法と考えられるため、これを採用した。

我々は、並列推論マシンPIM上において、上記ルータを用いた配線処理を実行し、台数効果などの性能評価を行なった。

以下、第2章では無格子配線の基本アルゴリズム、および並行オブジェクトを用いたモデル化方法に付いて記す。第3章ではタイムワープ機構の配線への適用方法に付いて述べ、第4章では実装上の工夫などについて述べる。第5章では性能評価結果に付いて報告し、第6章で本稿をまとめる。

2　並行オブジェクトモデルと無格子配線アルゴリズム

我々は、配線領域を並行オブジェクトモデルを用いて表現し、矩形分割に基づく無格子配線アルゴリズムに従って配線処理を行なうプログラムを設計した。以下、並行オブジェクトを用いた配線領域のモデル化の方法と、矩形分割に基づく無格子配線アルゴリズムの処理の流れを記す。なお、簡単のため2層配線を仮定する。各層ではそれぞれ縦配線、横配線のみを行い、スルーホールを設定することで配線経路が折れ曲がる。配線可能領域、配線経路、および障害物は矩形で表され、経路幅や、スルーホールの大きさは、予め、設計規則により与えられる。また、2端子間のみの配線を仮定し、始点、終点の区別がなされているとする。

2.1　配線領域のモデル化

並行オブジェクトモデルは、問題から高い並列性を抽出するのに適した方法である[7]。本ルータでは、各層の配線可能領域を複数の矩形領域に分割し、それぞれを能動的かつ並行に動作し得るオブジェクトとして表現する。図1に矩形分割によって生成された矩形の例を示す。例えば、縦配線層の矩形分割では、障害物の全頂点において配線可能領域を縦方向に分割する。

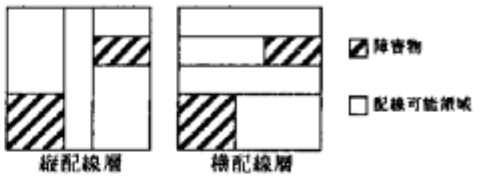

図1: 矩形オブジェクト

さらに、端子(始点、終点とも)もオブジェクトとして表現し、同じ層で接する矩形オブジェクトと端子オブジェクト間、および、別の層で接する矩形オブジェクト間には接続関係があるとみなす。配線処理は接続関係にあるオブジェクト間のメッセージ通信によって進む。また、オブジェクトはその状態として、オブジェクト内の配線可能領域情報をもつ。初期値はオブジェクト全体の領域である。配線処理が進むに連れ経路領域が決定してゆくが、これらは新たな障害物となるため、配線可能領域は次第に複数の矩形に分割されていく(図2)。

2.2　配線処理の流れ

一つのネットの配線処理は探索処理と詳細経路決定処理の二つに分けられる。以下、それぞれの処理の流れを記す。

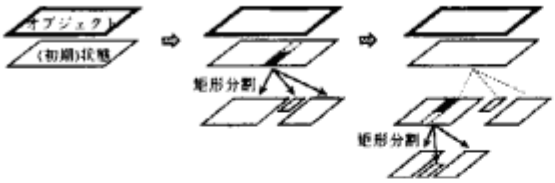

図 2: 矩形オブジェクトの状態変化 (矩形分割の進行)

2.2.1 探索処理

探索処理は、ネットの始点端子オブジェクトから、それに接する矩形オブジェクトに search メッセージを送ることにより開始される。

search メッセージは、送信オブジェクトの状態の中で経路の候補となる矩形の情報を持つ。これを受信した矩形オブジェクトは、経路候補情報を保存する (詳細経路決定処理で用いる)。また、自分の状態 (配線可能領域) を調べ、さらに経路候補矩形があれば探索続行可能と判断して、接続関係にある矩形オブジェクト、および (もし存在すれば) 終点端子オブジェクトに search メッセージを送る[1]。

なお、search メッセージには、以下の式で与えられるコスト評価値 f が付加されている。

$$f = g + h$$

ここで、g はスタート点 S から中間点 M までのコスト (確定値) であり、M までの距離と、折り曲げペナルティの和で表す。また、h は M からゴール点 G までのコストの見積もり値である。h は M から G までのマンハッタン距離と、最小折り曲げ回数に対応したペナルティの和で表す。h は、実際のコスト以下であることが保証される。なお、M は矩形オブジェクト間の交差領域で設定可能なスルーホール中、最小の確定コスト g を与える点とする (図 3)。ここで、search メッセージが必ず f の小さい順に評価されるとすると、探索は A* アルゴリズムに基づいて行なわれる。

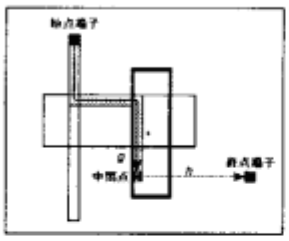

図 3: 中間点

終点端子オブジェクトは search メッセージを受信することにより、経路の存在を知る。このとき、終点端子オブジェクトは terminate メッセージを放送する。terminate メッセージは発見経路のコスト値 g_t[2] をもつ。terminate メッセージを受信した矩形オブジェクトは、以後、コスト評価値が g_t 以上の search メッセージを受信しても、探索処理を継続しない。全オブジェクトおよび通信路上に search メッセージが存在しなくなった時点で、探索処理が終了する。この時までに終点端子オブジェクトに到着した search メッセージのうち、最小コスト値 $g_{t_{min}}$ を持つものが最適経路を与えることになる[3]。

2.2.2 詳細経路決定処理

$g_{t_{min}}$ を与えた search メッセージ送信元の矩形オブジェクトに対し、終点端子オブジェクトから traceback メッセージを送ることで詳細経路決定処理が開始する。

traceback メッセージを受信したオブジェクトは、その状態情報 (配線可能領域) の中で経路にあたる矩形領域を特定し、その領域内に配線経路 (新たな障害物) を設けて矩形分割する。また、保存されていた経路候補の情報に基づいて、search メッセージ送信側の矩形オブジェクトに対し、traceback メッセージを送る。

traceback メッセージは最適経路を与える矩形列を辿り、最終的に始点端子オブジェクトに到達する。この時点で詳細経路決定処理が終了する。

3 タイムワープ機構を用いた並列配線

上述した配線処理のうち、探索処理が厳密な A* アルゴリズムに従わなければ、異なる f のメッセージが同時に処理可能である。しかし、最適経路を知る必要から、探索処理終了時に最小 g_t を与えた search メッセージを求めなければならない。このとき大域的な同期処理が必要になる[4]。

一方、複数のネットの配線処理において、それらの最適経路を与える矩形列に共通部分がなければ、同時に処理できる (図 4)。しかし、探索処理の前にこの矩形列を知ることは極めて難しい[5]。矩形列に共通部分がないことが保証できない限り、各ネットの配線処理を順に行なわなければならないため、一つのネットの配線処理が終了する毎に同期を取る必要がある。

このように、上述した配線処理においてはいくつかの同期処理が必要になる。これは、逐次性の存在を意味する。

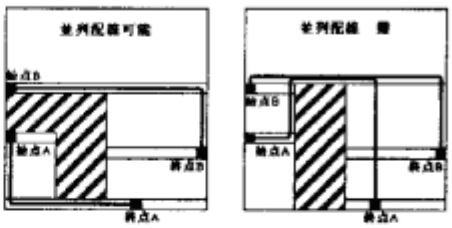

図 4: 並列配線可能なネット

タイムワープ機構は、見込み計算を積極的に行うことで、実行時に高い並列性を抽出しつつ、誤っておこなった見込み計算についてはロールバック処理によって修正することで、結果的には厳密に逐次性 (処理順序に関する制約) が守られる機構である。我々は、配線処理にタイムワープ機構

[1] 但し、既に search メッセージを送ったオブジェクトに対しては、重複してメッセージ送信しない (探索の枝刈り)。

[2] この時、$h = 0$ であるため、$f = g_t$ である。

[3] 探索処理が厳密に A* アルゴリズムによって進むとすれば、最初に到着した search メッセージが $g_{t_{min}}$ を与える。

[4] 厳密な A* アルゴリズムに従えば、並列処理可能な箇所は、最小のコスト評価値 f をもつメッセージの処理のみである。この方法では、最小 f を持つメッセージの処理が終わる度に、新たな最小 f を求めるため大域的な同期が頻繁に必要となる。

[5] この矩形列は探索処理によって初めて求まるものである。但し、予め探索範囲を限定し、探索範囲の重ならないネットに付いて並列処理することは可能である [6]。しかし、この方法では、経路が存在しても発見されない場合が有り、また最適経路であることも保証されない。

を組み合わせることで上記の同期処理が不要になり、その結果高い並列性を引きだせると考えた。以下、まず、タイムワープ機構の概要について述べ、次に、配線処理への適用方法について説明する。

3.1 タイムワープ機構の概要

いま複数のオブジェクトがメッセージ通信を行うことによって、次々と状態を変えていくモデルを考える。ここで、メッセージはタイムスタンプを持ち、各オブジェクトはタイムスタンプ順にメッセージを処理しなければならないという制約があるとする。

タイムワープ機構では、各オブジェクトは、メッセージはタイムスタンプ順に到着するという楽観的な仮定をおく。メッセージがタイムスタンプ順に到着している限り、その処理は正しいものとして処理を進める(見込み計算)。

しかし、実際には誤った順序でメッセージが到着する場合が存在する。このような状況に備え、オブジェクトは履歴を保存しておく。そして、メッセージの到着順序の矛盾を発見したところで履歴を巻き戻し(ロールバック)、処理のやりなおしをする。さらに、その時点で誤って送信したことが判明したメッセージに対しては、メッセージを取り消す役割を持つアンチメッセージを送信する。

3.2 配線処理への適用

タイムワープ機構を配線処理に適用するにあたり、以下のような楽観的な仮定をおき、見込み計算をおこなう。

- 探索処理はA* アルゴリズムにしたがって進み、最初に終点端子オブジェクトに到着したsearchメッセージが最小の $g_{t_{min}}$ を与えると仮定する。詳細経路決定処理は、探索処理中および終了時の同期を取ることなく開始させる。
- 複数のネットについて、最適経路の矩形列が重ならないと仮定する。各ネットの配線処理終了時の同期を取らず、同時に配線処理を行う。

上記の仮定がもし正しければ、極めて高い並列性が抽出できる。また、仮定が誤っていた場合でも、ロールバックにより修正されるため、最適経路の発見、および配線順遵守は保証される。

タイムワープ機構では、タイムスタンプによって処理順序の正しさを判断し、見込み計算が正しいかどうかを知る。A* アルゴリズムに従った配線処理では、同一ネットの経路探索におけるコスト評価値の大小が、また、ネットの経路探索における配線順序の大小が処理順序を表すことになる。したがって、本ルータでは、配線順序 O とコスト評価値 f の値の組 (O, f) をタイムスタンプに対応付け、各メッセージに付加する。任意の2つのメッセージの持つタイムスタンプの大小は、以下の規則により比較する。

$$
\begin{array}{lll}
\text{IF} & O_1 < O_2 & \text{THEN } TS_1 < TS_2 \\
\text{ELSE IF} & O_1 = O_2 \text{ and } f_1 < f_2 & \text{THEN } TS_1 < TS_2 \\
\text{ELSE IF} & O_1 = O_2 \text{ and } f_1 = f_2 & \text{THEN } TS_1 = TS_2 \\
\text{ELSE} & & TS_1 > TS_2
\end{array}
$$

本ルータでは、オブジェクトが並行にメッセージ処理を進めていく。しかし処理結果においては、上記の対応付けにより表された処理順序は、タイムワープ機構により完全に守られる。すなわち、同一ネットの探索では、メッセージはコスト評価値の小さい順に処理されたように見え、その結果、得られた経路も最適経路であることが保証される。また、各ネットは正しい配線順にしたがって処理されたように見える。

4 実装

4.1 実行環境

本システムは、並行論理型言語KL1[5]で記述され、PIM/m[34]上に実装されたシステムである。PIM/mはMIMD型計算機で、要素プロセッサが2次元メッシュ状のネットワークで結合されている。最大で、要素プロセッサ256台による構成が可能である。メモリは全て分散管理されており、他の要素プロセッサへのデータアクセスコストすなわちプロセッサ間通信コストは高いが、台数拡張性に富む。

4.2 スケジューラ

ロールバック処理はコストが高いと考えられるため、その発生頻度をできるだけ小さく抑えることが重要である。通常オブジェクト数はプロセッサ数に比べはるかに多いため、各プロセッサ内でメッセージのスケジューリングを行うことは、ロールバック頻度低減に有効である。

この目的で、本ルータではプロセッサ毎にスケジューラを置く。オブジェクトが送信したメッセージは、目的地オブジェクトの属するプロセッサのスケジューラに一旦登録される。スケジューラは、登録されているメッセージのうち、最小タイムスタンプのものから処理を行う(最小時刻優先戦略)[6]。

4.3 局所的な時刻検定

タイムワープ機構では、オブジェクト毎にメッセージの到着順序の矛盾を発見したところでロールバック処理が行われる。しかし、メッセージに影響を受ける状態情報(矩形)が異なれば、メッセージの到着順序が矛盾していても、ロールバックする必要がない。このため、本ルータでは、履歴保存および時刻の検定をオブジェクト単位ではなく、分割された矩形単位で行う。同一の矩形に対して影響を与えるメッセージの到着順序が矛盾した場合のみ、ロールバック処理が行われる。これにより、ロールバック頻度を小さくするとともに、その影響範囲が局所的になり、オーバヘッドを低減できる。

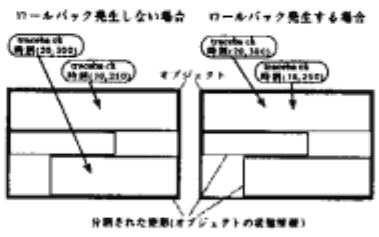

図5: ロールバックの発生

[6] この戦略にしたがえば、1プロセッサのみを用いる場合、探索処理は厳密なA* アルゴリズムになり、また、ロールバックは発生しない。

4.4 負荷分散

負荷分散はラウンドロビンの方法を取った。すなわち、各オブジェクトのID番号について総プロセッサ数の剰余を求め、これに相当するプロセッサに分配した。この方法は、負荷の均一化および、並列性の抽出という観点からは極めて良い方法である。しかしながら、プロセッサ間通信の低減の面からは悪いと言える。なお、オブジェクトはプロセッサに割り当てられた後、別のプロセッサには移動しない(静的負荷分散)。

5 評価

本配線システムをPIM/m 64プロセッサを用いて実行し、性能を評価した。使用したデータは、136ネットを配線する実際のLSIデータで、各層とも528個の障害物が存在する。我々は、以下に示す2種類の実験を行った。

(a) タイムワープ機構を用い、複数ネットの配線処理を並列に行う。

(b) 一つのネットの探索処理のみ並列に行う。そのネットの経路が求まった後、同期を取って、次の配線順を持つネットの探索を開始する。

(a)、(b)共に、最適経路発見および配線順の遵守が保証される。なお、(b)は、タイムワープ機構を用いない従来のルータにおいて、上記2点を保証する方法に相当する。

図6に台数効果(1プロセッサ使用時の実行速度に対する、複数プロセッサ使用時の相対速度)、および配線処理中に発生したsearchメッセージ数をを示す。実線は(a)に対応し、破線が(b)に対応する。図から(a)ではプロセッサ64台を用いて約14倍の台数効果を得たのに対し、(b)では、プロセッサ32台で最高6倍の台数効果に止まり、プロセッサ64台使用時には却って処理速度が低下したことが分かる。これは、大規模分散メモリマシンでは、大域的な同期処理のオーバヘッドが大きいためと考えられる。

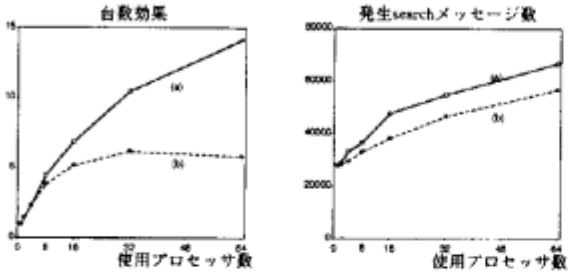

図 6: 台数効果

また、タイムワープを用いたルータにおけるsearchメッセージ発生数については、1プロセッサ使用時に比べ、64プロセッサを使用時では143%の増加が計測された。しかしながら、タイムワープを用いない場合でも105%の増加が計測されていることを考えると、この値は深刻なものでないといえる。この差が小さい理由は、ラウンドロビンによる負荷分散により、負荷が均一になっており、各プロセッサでの時刻の進み具合がほぼ等しいためと考えられる。

なお、配線結果に付いては、100%の配線率が達成された。各ネットの配線経路については最適経路であることを確認した。

以上の結果から、本並列ルータは最適経路発見・配線順の遵守を保証しつつ、タイムワープ機構を用いて複数ネットを並列に配線することで、配線処理を高速化することが確認できた。

6 おわりに

本稿では、タイムワープ機構による無格子配線システムについて述べた。基本アルゴリズムは矩形分割に基づく無格子配線アルゴリズムであり、配線領域を並列オブジェクトモデルを用いて表現することで、高い並列性抽出を図った。さらに、最適経路と配線順遵守を保証しつつ、効率的な並列処理を実現するため、タイムワープ機構を組み込んだ。その結果、並列計算機上でも、逐次配線システムと同等の高品質な解を得ることが保証されるようになる。

上記のシステムをMIMD型分散メモリマシンであるPIM/m上に構築し、性能評価を行った。その結果、64プロセッサを用いて14倍という台数効果を得た。また、得られた配線品質に付いては、100%の配線率を達成し、各ネットに付いては、各最適経路を得ていることを確認した。

今後は、更に大きい回路データを用いて実験、評価する予定である。また、予め与えられた配線順が不適切な場合にも対処するため、実行時のインクリメンタルな配線順変更機能を検討中である。これは、タイムワープ機構におけるロールバックを、自動的な引き剥がし、再配線処理に利用するものである。さらに、適切な負荷分散方法の検討も残された課題の一つである。

参考文献

[1] D. R. Jefferson, "Virtual Time," *ACM Transactions on Programming Languages and Systems*, Vol.7, No.3, pp. 404–425, 1985.

[2] A. Margarino *et al.*, "A Tile-Expansion Router," *IEEE Trans. on CAD*, Vol. CAD-6, No.4, pp. 507–517, 1987.

[3] H. Nakashima em et al., "Archtecture and Implementation of PIM/m," *Proc. Int. Symp. on Fifth Generation Comp. Sys.*, pp. 425–435, 1992.

[4] K. Taki, "Parallel Inference Machine PIM," *Proc. Int. Symp. on Fifth Generarion Comp. Sys.*, pp. 50–72, 1992.

[5] K. Ueda and T. Chikayama, "Design of the Kernel Language for the Parallel Inference Machine," *The Computer Journal*, Vol.33, No.6, pp. 494–500, 1990.

[6] T. Yamauchi *et al.*, "PROTON: A Parallel Detailed Router on an MIMD Parallel Machine", *Proc. IC-CAD*, pp. 340–343, 1991.

[7] 伊達, 瀧：並列オブジェクトモデルに基づくLSI配線プログラム, 情報処理学会論文誌, Vol. 33, No. 3, pp. 378–394, 1992.

[8] 松本, 瀧：バーチャルタイムによる並列論理シミュレーション, 情報処理学会論文誌, Vol. 33, No. 3, pp. 387 395, 1992.