数 値 解 析 42－5
(1992. 8. 20)

プロセッサ競合方式による並列自動配線
～ランダム引き剥し法～

佐野 雅彦　岡 圭司　高橋義造
徳島大学工学部知能情報工学科
E-mail: {sano, oka , taka}@n30.is.tokushima-u.ac.jp

我々の研究室では各種アプリケーション側から見た最適かつ計算機アーキティクチャへの依存性の低い並列処理アルゴリズムの研究・開発を行っている．その一例に配線問題の並列処理がある．これまでの研究結果からマスタ／スレーブモデルによるプロセッサ競合方式とネット間の並列性を用いたネット割り当て法により，速度面に関して十分な性能が得られた．しかし配線品質に関しては不十分であった．本稿では配線品質の改善方法の一つとしてランダム引き剥し法による並列配線アルゴリズムを提案する．

Parallel Automated Wire Routing with Competing Processors:
- A Random Rip Up Method -

Masahiko Sano　Keiji Oka　Yoshizo Takahashi

Department of Information Science and Intelligent Systems,
Faculty of Engineering, Tokushima University
Minami-Josanjima-Cho, Tokushima 770, Japan
E-mail: {sano, oka , taka}@n30.is.tokushima-u.ac.jp

From our research on parallel automated wire routing problems, an effective parallel processing algorithm which processes problems with the competing processors and net assignment strategies has been developed by the authors in the past. The proposed algorithm featured high parallelism and processing speed, but the resulting wired routes were not satisfactory. In this paper, a new parallel algorithm using the random rip up method is proposed in order to improve routing results.

1.はじめに

配線問題の並列処理方法には２つの方法がある．ひとつは専用ハードウエアを用いて配線処理の最も時間のかかる経路探索を高速に行うことが出来るものである．もうひとつは汎用並列計算機上でソフトウエアによって並列に行うものである．前者の方法はハードウエア化による高速化のために計算機方式に強く依存している．一方ソフトウエアによる方法では，計算機方式に対する依存性は前者に比べて低いものの，高い並列性を発揮できない限り前者を凌ぐことは出来ない．

我々は計算機方式に対する依存性の低い並列アルゴリズムの開発を目的とし，その研究対象の一つに配線問題を取り上げて並列アルゴリズムの研究を行ってきた．これまでの結果からマスタ／スレーブモデルを用いたプロセッサ競合方式とネット間の並列性を用いたネット割り当て法による非同期的な並列処理方式が，処理速度の点で有効であることを確認している[1-2]．しかし配線品質の点で問題があり改良の余地が残されている．本稿ではこの問題を解決する方法のひとつとしてランダム引き剥し法を用いた並列配線アルゴリズムを提案する．

2. 並列配線問題

並列性　配線問題には２つの並列性が存在する．ひとつはネットの並列性であり，一本のネットは複数のプロセッサを用いて配線処理できることを意味する．ふたつめはネット間の並列性である．これは互いに関係のないネットは並列処理が可能であることを意味する．ネットの並列性に注目した並列ルータには幾つか例がある．この方式の問題点には，経路探索に関係のないプロセッサは動作しないので利用率が余り高くならないことが挙げられるが，実装方法が比較的簡単なのでハードウエア化に適している．しかし，より高並列に処理するためにはもうひとつの並列性であるネット間の並列性を用いる必要がある．これにより空いているプロセッサを少なくすることが可能になり，利用率が向上する．我々の研究結果ではネット間の並列性のみを用いた並列配線アルゴリズムにおいて，良好な並列性を得ている[1-2]．

配線順序　並列配線処理には各々のネットを逐次的に配線処理する方法と並列に処理する方法がある．逐次的な場合はネットの並列性のみが利用可能であり，その配線順序は逐次ルータと全く同じにすることが出来る．また，ネット間が完全に独立なときには同時配線が可能である．配線順序を予め決定しておく方式も，各ネットの配線順序から見ると逐次的であると言える．これらの処理方式の殆どは全体の同期化を行っている．

ネット間の並列性を最大限に活用するのなら，全体の同期化は使用しない方が良い．これは各々のネットの配線処理に必要な処理時間が異なり，粒度が揃っていないからである．すると，配線順序を一意することは困難になってくる．従って，同時に配線する方法では，前者と異なる並列アルゴリズムが必要である．その様な方式ではネット間が完全に独立していなくても配線処理が可能である．その代りに各プロセッサ間の配線結果に矛盾が生じるのでこれを解決する方法が必要である．我々は[1-2]において，早い者順による競合解決方式によって発生する矛盾を再配線処理を用いることで解決を試みたが，配線可能領域が減少するに従い再配線による経路確保が困難になる問題が生じた．これは局所的な配線率は配線順序の影響が強いため再配線処理だけでは充分に解決出来なかったからである．

ネットの引き剥し　これまでに開発されてきた並列ルータでは引き剥し処理を行っていないのもが多い．それらの多くは（経路探索を除くと）逐次ルータと同様のアルゴリズムであり，引き剥しを極力行わない方針である．そのために予め配線順序を決定しておくことで配線品質の低下を防いでいる．一方，並行して配線処理されるルータでは配線順序が決っていないので引き剥し法などを用

いて繰り返し処理を行う必要がある．しかし，引き剥すべきネットを選択するには計算時間が必要であるが，我々が用いているマスタ／スレーブ型の処理モデルではマスタの処理時間の長さは全体の性能に影響するため，極力短い方が良い．そこで，逐次型アプローチと異なる方法が必要である．

3. 従来の並列配線処理方式

本章ではこれまでの研究結果の概要について述べる．プロセッサ競合方式による並列配線は先に述べたネット間の並列性を最大限に活用するための処理方式のひとつであり，図1に示すマスタ／スレーブモデルを用いている．スレーブはマスタと通信を行うので各種の計算機網に対する依存性が低く，並列処理モデルを構成し易い特徴がある．

このモデルはマスタ，スレーブ，データベース，及びネットワークから構成されている．通常，マスタ，スレーブにはプロセッサを割り当てる．マスタはデータベースの管理とスレーブ間の競合の調停を主な役割とし，スレーブはマスタから割り当てられた処理を他のスレーブとは独立して非同期に処理を行う．データベースには配線処理に必要なネットデータや配線結果等があり，スレーブはマスタを通じて配線処理に必要なデータを得る．

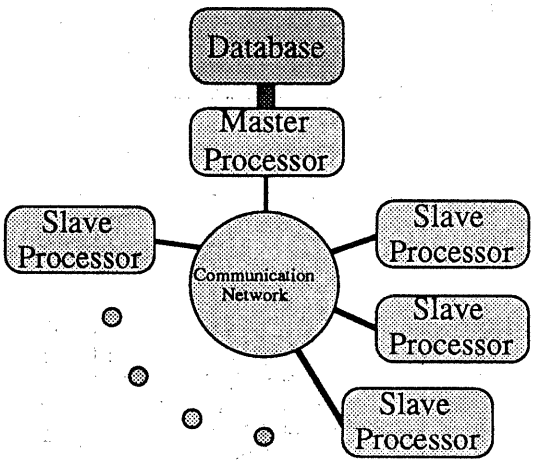


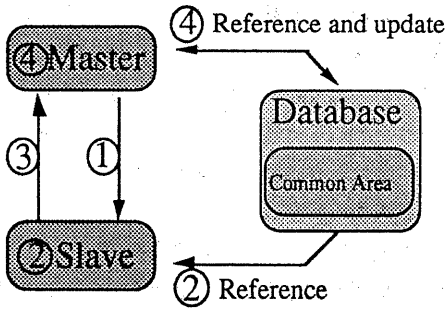
Fig.1 Processing model of competing processors

並列配線アルゴリズム　プロセッサ競合方式による並列配線処理はマスタとスレーブの二つに分けられる．全体のアルゴリズムの概要を示す．

マスタは全スレーブに各々異なるネットを割り当てる．ネットを割り当てられたスレーブはマスタが持つ配線領域を参照しながら他のスレーブとは独立して配線処理を行う．配線処理を終えたスレーブは配線結果をマスタに送る．結果を受け取ったマスタはその配線結果を検証する．検証の結果，正しければ配線領域に登録する．そうでなければ再配線処理を指示する．スレーブ間の競合は早い者順に解決される．スレーブのルータには線分探索法を[3]を用いている．

マスタ：　マスタはスレーブへのネット割り当て，配線領域・配線結果の管理，配線結果の検証，スレーブ間の競合の裁定，及び，競合に負けたスレーブに対する再配線処理を行う．

スレーブ：　スレーブはマスタから割り当てられるネットを他のスレーブとは独立して配線処理する．通信はマスタとの間行われる．



(1)Master assigns a net to slave.
(2)Slave executes wire routing by referencing the database.
(3)Slave transmits a result to master.
(4)Master verifies the result then updates the database.

Fig.2 Processing flow

取り扱った配線問題は縦横256x256，ピン間0本の２層配線問題であり，平均マンハッタン距離が10,20の3000本のランダムネットデータ(図3)を用いて評価した結果を図4に示す．(a)はプロセッサ利用率と速度向上比，(b)は配線率と再配線率，(c)

には500本毎の配線率と再配線率を示す．

この評価には我々の研究室で開発されたCoral 68Kを用いた．CPUにMC68000を用いており，63台のプロセッサを二進木状に結合した分散メモリ型マルチプロセッサである．演算能力は全体で40MIPS, 0.5MFLOPSである[3]．

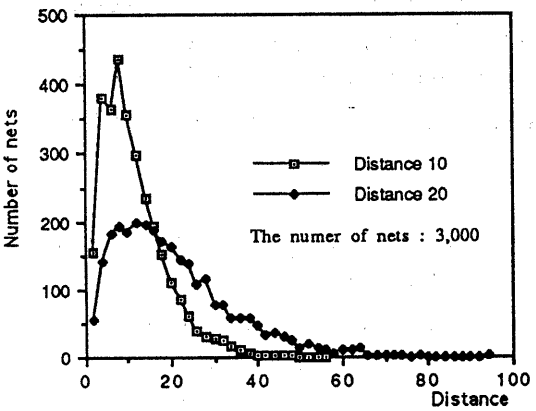


Fig.3 Distribution of Manhattan distance of random net

この結果から，配線処理が進行しまってくると再配線が出来なくなり配線率が低下することが解る．これは，競合に負けたプロセッサが再配線処理を行うと配線不可能になるからである．更に，この評価では引き剥し処理は行われていなかったことも原因である[2]．

配線品質の向上のためには再配線処理の改良と引き剥し処理を取り入れる必要がある．本来，再配線処理は同時に処理により発生する矛盾の解決の為に行うのであるが，配線領域が埋まっている場合では引き剥しを行う以外には配線不可能である．このために引き剥しを行う必要があるが，全ての配線結果に矛盾を許すことで両者を一つの問題にできる．それは，引き剥して再配線を繰り返すことで矛盾を解消することである．これにより配線結果が得られるまでに複数の繰り返し処理が必要になるが，アルゴリズム的には単純になる．

4. ランダム引き剥し法

2章で述べたように引き剥すネットを選択する方法は単純な方が良い．そこでランダムに選択する方法を考える．ここで配線問題を探索問題として見たとき，配線結果を得ることは何等かの評価関数による満足すべき解（最適解である必要はない）を得ることである．ランダム探索法は理論上最適解を発見することで知られており，局所解からの脱出についての有効性も高い．これと同様の

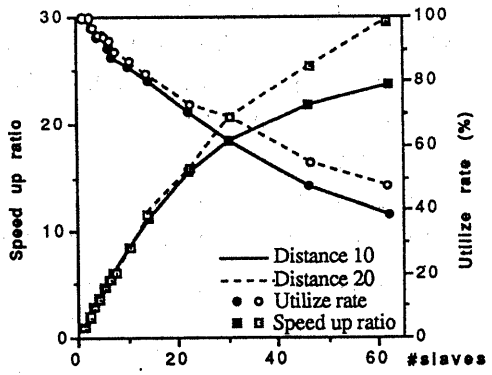


Fig.4(a) Utlization and speed up ratio

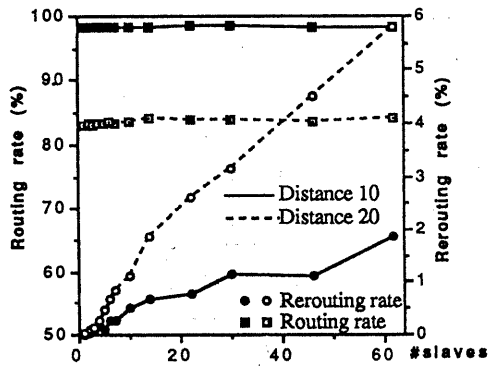


Fig.4(b) Routing and rerouting rate

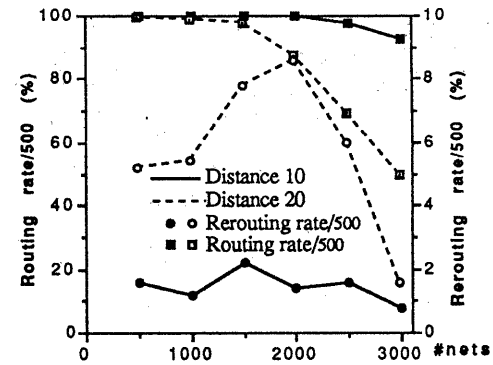


Fig. 4(c) Routing and rerouting rate by the 500 nets

ことが配線問題についても言えるだろう．ただし，ランダム引き剥し法は発見的要素が大きく，規模の大きな配線問題では解を得るまでに非常に多くの繰り返しを必要とするので，収束を早めるための工夫が必要である．(最近では遺伝的アルゴリズムによる探索方法が脚光を浴びつつある．これはランダム法の欠点である収束の遅さを補うものであり，配線問題への応用も試みられている．)

我々はランダムリップ引き剥し法の有効性を確認するために次のような実験を行った．

I. 引き剥し処理をせずに配線処理
II. 配線不可能な場合に引き剥し処理を加える．
III. II.に加えてランダムに引き剥す．配線不可能な場合でもランダム性を用いる．

引き剥がしの選択方法にはMinimal Separaters[3]を用いておこなった．評価にはワークステーションを用い，そのアルゴリズムは逐次的なものを用いた．問題規模は縦横32x64グリッドの単層配線問題であり，配線するネット数は61本のランダムデータである．図5(a)の実験Iの結果では3087回再配線を行って10本の未配線が残っている．図5(b)の実験IIでは，2329回の引き剥し再配線を行い，未配線は4本に減少している．図5(c)の実験IIIでは855回行って未配線は0本である．

IIの実験によって引き剥し処理は有効であるが，配線結果の品質向上には不十分であることがわかる．これは，局所的な最適解が全体的な最適化に結び付くとは限らないからである．また，引き剥し・再配線の結果が局所解に陥り脱出に消費する繰り返し数が多い．これに対してIIIの実験によってランダムに引き剥した場合の方がIIの場合より良い結果になっている．これは，IIの場合では陥り易かった局所解の状態からの脱出が容易であることが理由であると思われる．そのため，繰り返し回数も減少している．但し，常に短い回数で収束する保証はない．

また，無闇に引き剥しても効果は得られず，むしろ悪影響を及ぼす場合が少なからず存在する．

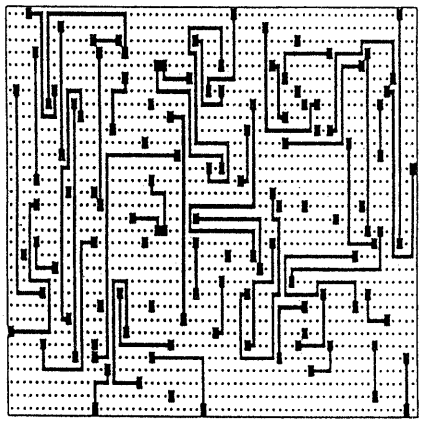


Fig.5(a) No rip up

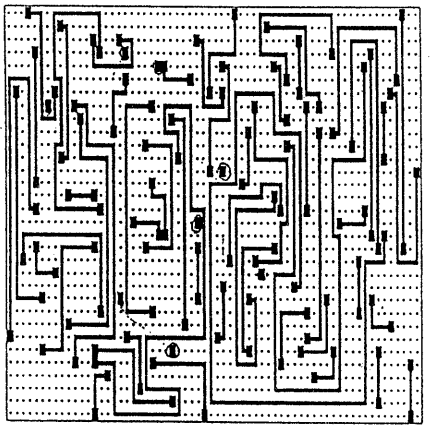


Fig.5(b) Only rip up

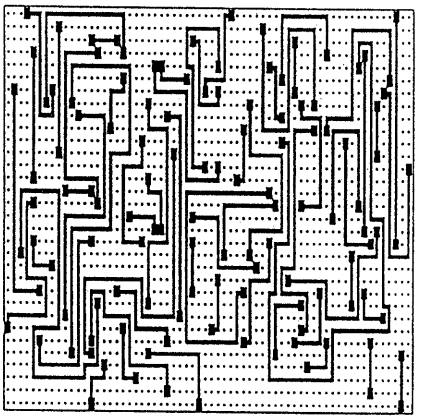


Fig.5(c) Rip up and random rip up

つまり逐次処理における引き剥し処理ではその対象になるネットが存在しないと仮定し，より高い配線品質が得られる場合に行うのに対し，ラン

ダムな引き剥がしではそのような仮定は出来ないので，引き剥した結果が必ず良くなる保証は無いからである．多くの配線状態を作り出すことが出来る特長を持つが，旨く配線結果に反映出来なければランダムに行うことは無意味である．すなわちランダム引き剥し処理は配線状態を変化させるに過ぎないため，そこから良い配線結果を得るには状態の変化を効果的に反映できる経路探索アルゴリズムが重要になる．また，実験に用いたアルゴリズムは逐次アルゴリズムで行ったものであって並列処理向きではない．そこで並列処理向きの引き剥しアルゴリズムを考える必要がある．

同時配線　これはネット間の並列性を有効活用するものであるが，プロセッサ競合方式上で互いに独立でないネットで同時配線を行う場合，それぞれの配線結果に矛盾が生じる．従来はこの矛盾を再配線処理を用いることで解決していた．これは，マスタが管理する配線領域は正しいものであると仮定していたからである．しかし，繰り返し処理を必要とする配線処理方式では配線結果の矛盾に拘る必要はない．つまり，繰り返し処理の間に矛盾を解決すればよく，一度に解決する必要はないのである．この様にすることで同時配線のアルゴリズムは次のように簡単になる．マスタはランダムに引き剥されたネットをスレーブに割り当てる．マスタからネットを受け取ったスレーブはマスタが持つ配線領域を参照しながら，矛盾が減少するように配線する．これらを全てのネットが配線され，矛盾が無くなるまで繰り返す．

ルータアルゴリズム　同時配線を可能にするアルゴリズムに要求されることは，求める配線経路が（衝突無しに）存在しないときでも望ましい経路を求めることが出来ることである．多くの探索アルゴリズムでは図6(a)の状態になると配線不可能になってしまう．仮にネットBが選択されて引き剥されても，今度はネットCによって配線不可能になる(b)．つまり，通常のアルゴリズムでは配線不可能になる場合でも既配線結果上を乗り越えて配線経路を求めることが出来なければ無限に繰り返しが行われ配線は不可能である．図7の例ではA,B,Cのネットがあり，A,Bは既に配線されている．ここでCのネットを配線するが，既に配線不可能な状態である．この状態で望ましい配線経路は(b)に示す通りである．この様な配線経路を求めることが出来れば，ネットBが引き剥されることにより(c)の様に全て配線できる．

このように，配線できない場合でも望ましい配線経路を得ることが出来れば，配線結果中の矛盾は自然に解消できる．多数のスレーブによって同時配線を行っても各々のスレーブが繰り返し処理の間に解決することが可能になる．これを実現するアルゴリズムには幾つか考えることが出来る．例えば，逐次処理で用いられる方法でも良い．これはマスタとは異なって処理に時間がかかっても良いからである．我々は配線コストを取り入れたアルゴリズムを用いることで解決を試みる．配線コストに基づいたルータアルゴリズムでは配線に付随する条件を配線コストとして表現し[3,5]，経路探索を行うときに最小コストの経路を求める．探索時に既配線結果の上を探索可能にしておくことにより図7(b)に示すような経路を求めることが出来る．このアルゴリズムでは最小コストの経路が衝突していない経路である保証は無い．また，矛盾の無い配線結果を得るには繰り返し処理を必要とする．

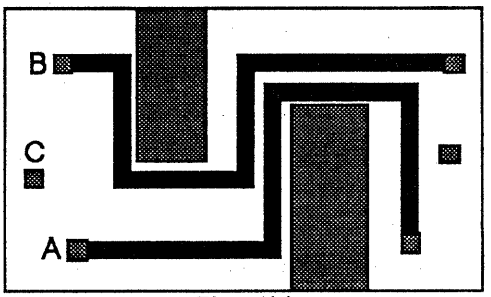


Fig. 6(a)

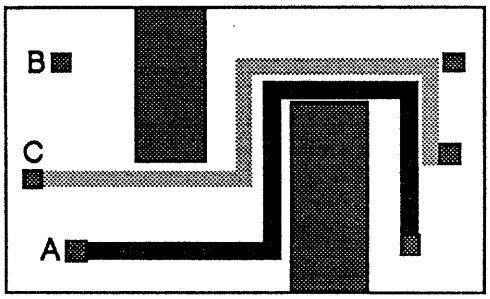


Fig.6(b)

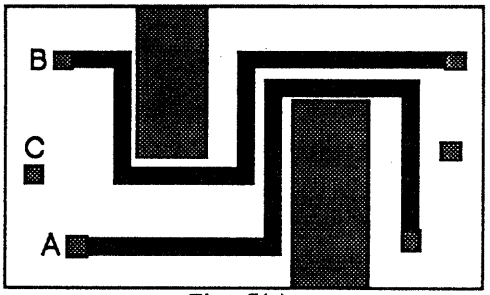


Fig. 7(a)

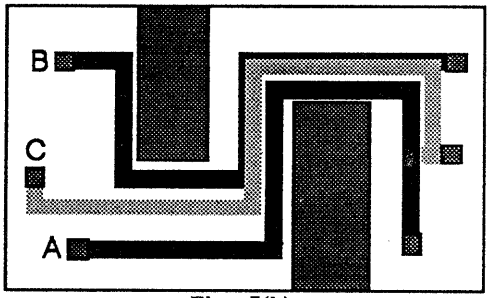


Fig. 7(b)

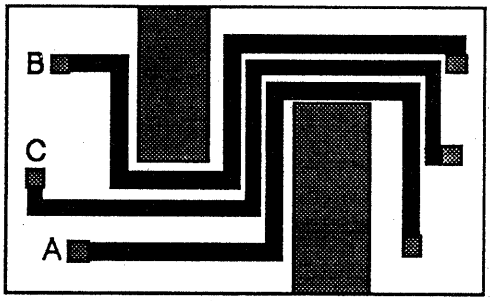


Fig. 7(c)

配線コスト　我々が設定した配線コストを示す．経路探索ではこれらの条件に触れる毎にコストを加算することで配線経路を評価する．

交差数(Cross)　　：既配線結果との交差数
接触距離(Touch)　：既配線結果との接触距離
配線距離(Length)　：そのネットの配線距離

各層毎に縦方向と横方向のコストを独立して設定できる．これを用いることによりXYルールを実現する．

ビア数(Via)　　：配線結果が持つビアの総数
折れ曲がり数(Bend)：そのネットの折れ曲がり数

ルータの評価　ルータの評価にはワークステーションSun Sparc Station 2 (28MIPS)を用いて行った．処理した配線問題は64x64グリッドの２層配線問題である．ネットデータはランダムに作成した100本のネットを使用し，配線コストを次のように設定した．Cross = 50, Touch = 10, Bend = 5, Length (表層 縦=1,横=5，裏層 縦=5,横=1)　更に，繰り返し処理の最中にコストを変化させることによって収束を早めた．引き剥しアルゴリズムにはランダム引き剥しを用いた．同時に１本だけを処理したので，５章で述べる処理方式のスレーブ台数が１台の場合に相当する．図８に逐次引き剥がしとランダム引き剥がしの比較結果を示す．ランダム引き剥がしの場合は最短の例である．

（一回の繰り返しは逐次処理の場合ネットを始めから最後まで処理しその後，最後から始めのネットに向って逆順で処理するまで，ランダムの場合は同数回の処理を行っている．）

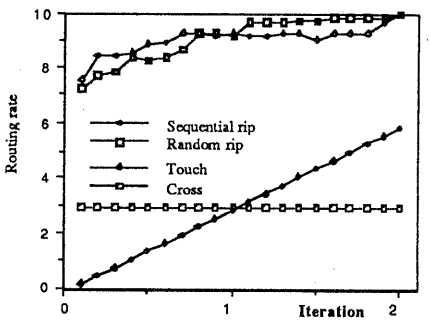


Fig.8 Comparison of Sequential and Random rip up

この結果では逐次処理の場合とランダム処理の場合で同じ結果になっている．多くの平均をとるとランダムの方がより多くの処理回数を必要とす

る．また，このルータでは配線コストの与え方が収束速度に影響する．現在のところこの値は経験的に変化させている．

5. ランダム引き剥し法による並列配線

前述のランダム引き剥し法と，ルータアルゴリズム，及びプロセッサ競合方式を組み合わせた並列配線アルゴリズムを提案する．この方法によりこれまで問題であった配線品質を改善することが可能になる．その処理モデルは図1に示したマスタ／スレーブモデルを用いる．マスタとスレーブのアルゴリズムは前回の場合と異なる．その概略を示す．

マスタは全スレーブに異なるネットを割り当てる．ネットを割り当てられたスレーブはマスタが持つ配線領域を参照しながら配線処理をする．このとき，スレーブは前述のルータアルゴリズムを用いて経路探索を行う．経路が求まればマスタに送る．配線結果を受け取ったマスタは配線領域に結果を書き込む．その後，ランダムに1本のネットを引き剥し，スレーブに割り当てる．以降，全ての配線結果が正しく配線されるまで繰り返す．処理の基本的な流れは図2に示すものと同じである．このアルゴリズムを図9に示す．

6. おわりに

本稿ではプロセッサ競合方式による並列配線問題において，以前よりも配線品質を良くするためのアルゴリズムとしてランダム引き剥し法による並列アルゴリズムについて説明した．このアルゴリズムでは，スレーブのルータアルゴリズムにおいて，配線経路が存在しない場合でも望ましい経路を求めることが出来るアルゴリズムが重要であることを説明した．このアルゴリズムとランダム引き剥し法による繰り返し処理よってプロセッサ間の配線結果の矛盾が解決されて行くことも示した．我々はこのアルゴリズムに配線コストを用いることで問題解決を試みる．現在，並列配線アルゴリズムをCoral 68K上において実装中である．

```
Master()
{
    Initialize( grid, netdata[] ) ;
    activeslave = 0 ;
    do {
      switch( Recv( resbuf, pnum ) ) {
        case JOB_REQUEST:
            JobAssigne( grid, netdata, pnum ) ;
            activeslave = activeslave + 1 ;
          break ;
        case OK:
          WriteResult( grid ) ;
          if( Verification( grid ) == COMPLETE ) {
              Send( COMPLETE, pnum ) ;
               activeslave = activeslave - 1 ;
          } else {
              JobAssigne( grid, netdata, pnum ) ;
          }
          break ;
      }
    } while( activeslave > 0 ) ;
}
```

```
Slave()
{
     Initialize( local-grid );
    Send( JOB_REQUEST ) ;
    while( Recv( local-grid, netdata ) != COMPLETE ) {
        Wire-Route( local-grid,netdata ) ;
        SendResult( netdata, OK ) ;
    }
}
```

Fig.9 Master and slave processor algorithms

参考文献

[1] Y. Takahasi, S. Sasaki,"Parallel Automated Wire Routing With a Number of Competing Processors," Proc. ACM International Conf. on Supercomputing, pp 310-317(1990).

[2] 佐野雅彦,高橋義造:分散メモリ型と共有メモリ型マルチプロセッサによる並列配線処理の性能評価,情報処理学会論文誌,vol.33,No.3(1992).

[3] T. Otsuki, "Maze-running and Line-search Algorithms on LAYOUT DESIGN AND VERIFICATION," ed T. Otsuki, Elsevier Science Publishers B.V. pp 99-131 (1986).

[4] 高橋義造,遠藤俊雄,松尾賢二,槌谷　一:二進木結合並列計算機Coral68Kの開発とその評価,情報処理学会論文誌,30巻,pp.46-57(1989).

[5] 河村薫,進藤達也,澁谷利行,土肥実久:超並列配線マシンMAPLE-RP,並列シンポジウム'９１, pp.373-379(1991).