5L−5

ランダム引き剥し法を用いた並列配線処理

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

1.はじめに

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．本論文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.並列配線における問題点

並列性　効果的な並列処理の可能性を探る重要な要素に並列性がある．この並列性が高いほど並列処理の効果が期待できる．そこで，配線問題の持つ並列性について検討する．

配線問題には2つの並列性が存在する．一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．二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．この方式の長所は，実装方法が比較的簡単なのでハードウエア化に適するが，経路探索に関係のないプロセッサは動作しないので十分な利用率が得られない短所がある．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も併せて用いる必要がある．

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，良好な並列性が得られるている[1-2]．

これはFig.1に示すマスタ／スレーブモデルを用いており，各スレーブプロセッサにネットを割り当てることで並列処理を行っている．スレーブはマスタのみ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長がある．

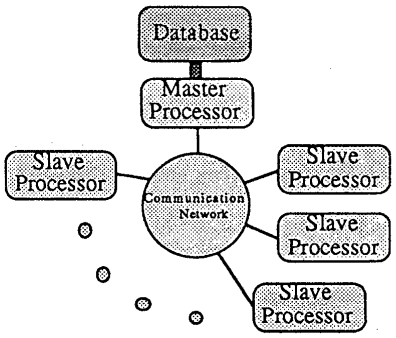

Fig.1 Processing model of competing processors

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ネット間の並列性による同時配線が可能である．このような処理方式の殆どは全体の同期化を行っている場合が多い．しかし，ネット間の並列性を最大限に活用するのなら，全体の同期化をしない方が良い．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るからである．すると，配線順序を一意的に決定することは困難になる．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．

我々は早い者順による競合解決による再配線処理方式を用いて矛盾の解決を試みたが，並列配線処理の進行による配線可能領域の減少のために，再配線による経路の発見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いためである[1-2]．

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しないものが多い．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処理を用いて繰り返し処理を行う必要がある．しかし，引き剥すネットの選択には，ネットの相互関係を調べるために計算時間を必要とするが，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になる．また，配線品質の向上のためには効果的な矛盾解決を行う必要がある．しかし，矛盾解決を直ちに行わず，引き剥し処理を行うことで矛盾を解消することができる．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．

3. ランダム引き剥し法

引き剥すネットを選択する方法は単純なほうが良い．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，大規模な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し，収束を早めるための工夫が必要である．

また，無暗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．つまり逐次処理ではより高い配線品質が得られる場合に引き剥し処理を行うが，ランダム引き剥がし処理では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活用できる経路探索アルゴリズムが重要になる．

Parallel Wire Routing using Random Rip up Method
Masahiko SANO, Keiji OKA, Yoshizo TAKAHASHI
Department of Information Science and Intelligent Systems,
University of Tokushima

ルータアルゴリズム　ランダム引き剥し法におけるルータアルゴリズムに要求されることは，衝突のない配線経路が得られない時にある程度の衝突を許容した，準最適な経路を求めることが出来ることである．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．

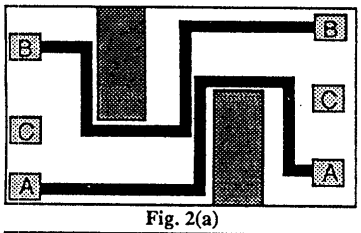

Fig. 2(a)

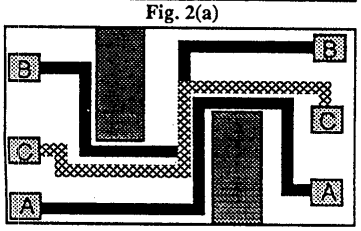

Fig. 2(b)

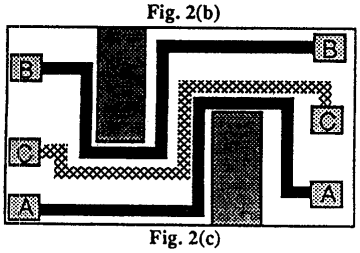

Fig. 2(c)

(Fig. 2 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ものとする．ここでCのネットを配線しようとすると，配線不能である．この状態で準最適な配線経路は(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を引き剥して再配線することにより(c)の様に全て配線できる．)

この様に，配線できない場合でも準最適な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．

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た．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．しかし，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保証は無く，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．

4. ランダム引き剥し法による並列配線処理

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Fig. 1 に示したマスタ／スレーブモデルを用いる．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．このアルゴリズムをFig. 3 に示す．

5. おわりに

本論文ではプロセッサ競合方式による並列配線問題において，配線品質を改善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提案した．このアルゴリズムでは，配線経路が存在しない場合でもある程度の衝突を許容した準最適な経路を求めることが出来るスレーブのルータアルゴリズムが重要である．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によってプロセッサ間の配線結果の矛盾解決が可能になる．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た．現在，並列計算機Coral 68K [4]を使用して，アルゴリズムの評価中である．

```
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. 3 Master and slave processor algorithms

参考文献

[1] Y. Takahashi, 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," in T.Otsuki ed.," LAYOUT DESIGN AND VERIFICATION," 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, vol. 30 No.1, 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).