解　説

MIMD 型計算機の並列アルゴリズム

——電気系 CAD への応用を中心として†——

中田　登志之††

1. はじめに

近年 MIMD (Multiple Instruction Multiple Datastream) 型並列計算機の研究が進み，一部商用化も開始されている．MIMD 型並列計算機の(SIMD 型並列計算機と比較した場合の) 最大の特徴は MIMD 型並列計算機を構成する各要素プロセッサ (PE) が独自のプログラムカウンタをもち，他の PE とは異なった命令を実行できることにある[1]．この特徴のために，MIMD 型並列計算機ではより複雑な処理を行うことが可能である．一方複雑な処理が可能であるゆえに，効率良くプログラムを実行するためには，以下のような並列処理の隘路を克服する必要がある．

- 逐次部分のオーバヘッド
- データ転送のオーバヘッド
- 負荷の不均衡に基づく同期のオーバヘッド

本稿では，主に並列処理のニーズが高く，かつ応用に内包される並列度が高い電気系 CAD 用の応用における MIMD 型計算機用のアルゴリズムをあげ，上記の問題を克服し，かつ高い並列性を得るためにどのような工夫がなされているかについて述べる*．なお，スペースの都合上 CAD 分野における全ての研究を網羅することは不可能である．あらかじめご了承いただきたい．

一口に MIMD 型並列計算機といっても，バス結合に基づく共有メモリマシンから，メッセージパシングに基づく分散メモリマシンまで，種々のアーキテクチャを有する MIMD 型並列計算機が存在する．本解説では，可能なかぎりアルゴリズムレベルの議論に焦点を絞ることにする．

† Parallel Algroithms for MIMD Machines Focusing on CAD Algorithms by Toshiyuki NAKATA (NEC Corporation, C&C Systems Research Laboratories).

†† 日本電気(株) C & C システム研究所

* もう一つの有望な分野は科学技術計算であるが，スペースの都合上今回は電気系 CAD に焦点を絞る．

2. MIMD 型並列計算機での並列化のパターン

MIMD 型並列計算機での並列化のパターンとしては以下の3種類のものがあげられる．

機能分散的　プログラムを論理的な複数のユニットに分割し，それらの機能間をデータの流れで結合する．(ベルトコンベヤを思い出していただきたい)

データ分割的　処理されるデータを分割し，おのおのの部分をプロセッサに割り付ける．

Branch and bound 的　ある問題のサブ問題ができた時点で，それを動的にプロセッサに割り当てる．(探索問題などがあげられる．)

多数のプロセッサを用いようとした場合，やはりデータ分割に基づいて並列処理を行っているアルゴリズムが多い．特に CAD 分野では対照となるデータの量が膨大である場合が多いため，自然とこのようになる傾向がある．一部の応用では機能分散とデータ分割を併用する．

3. 論理シミュレーション

3.1 論理シミュレーションのアルゴリズム

論理シミュレーションは設計した論理回路が正しく動作するか否かを検証するために，テストパターンを用いて，回路の動作を論理レベル (0, 1, X (未定)-Z (ハイインピーダンス))* で模擬するものである (図-1 参照)．このシミュレーションのアルゴリズムとしては，

- タイミング情報を問題としない零／ユニット遅延アルゴリズム
- 素子の標準遅延を考慮する標準遅延アルゴリズム
- 論理の 0→1，1→0 の変化に対応して遅延

* 最近は信号の強弱も考慮に入れるものが多い．

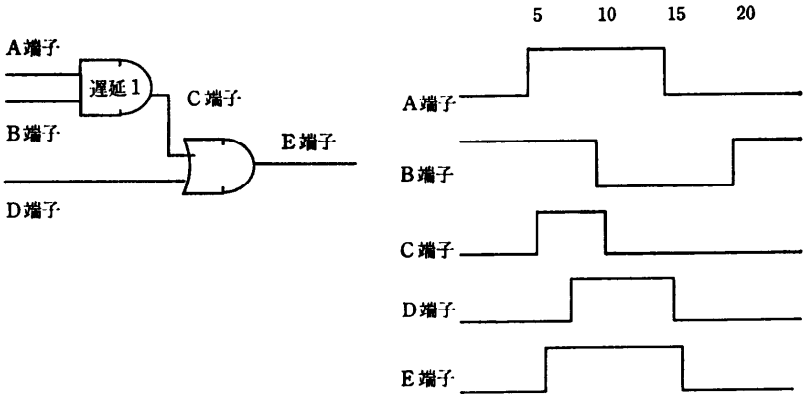


図-1 論理シミュレーションの概念

時間を変えることが可能な rise-fall 遅延アルゴリズム

などが存在する．また，一部のマシンを除いては処理の高速化を図るために，入力ピンに変化が生じた素子だけ新たに評価を行うイベント駆動アルゴリズムが用いられる．

YSE/EVE[3]，HAL[4]，SP[5]などの装置（大型計算機など）のシミュレーションを行うハードウェアシミュレータ（これらは通常複数ユニットで構成される．）では，タイミング情報を問題としない零／ユニット遅延アルゴリズムを用いるものが多い．たとえば HAL の場合は零遅延アルゴリズムに図-2 に示すように入力段からの論理回路の段

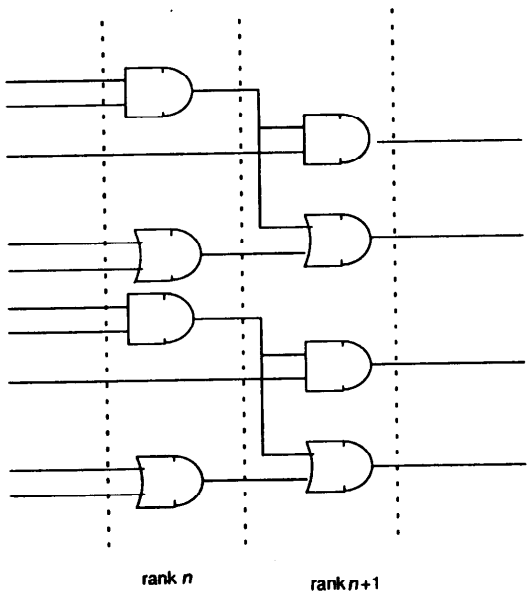


図-2 ランクオーダの概念

数が同じであるものを同時に行うランクオーダアルゴリズムを用いて高速化を図っている．この場合同一ランクに存在する素子のシミュレーションは同時に実行できる．したがって得られる並列性としては各ランクの平均素子数だけ存在することになる．

一方 LSI のシミュレーションで用いられる標準遅延アルゴリズムや rise-fall アルゴリズムでは，シミュレーションの時間を細かい時間（たとえば 0.1 nsec 単位）（これを 1 タイムステップと呼ぶ）で離散化する．タイムホイールという時間を保持する構造体を用いて，離散化した時間きざみにシミュレーションを行う（この方法をイベントホイール法と呼ぶ）．

ZYCAD 社の LE のような標準遅延アルゴリズムを実現するハードウェアシミュレータでは，全システムで 1 個のタイムホイールをもち，各プロセッサが同期をとりながら処理を進めていく（この方法をグローバルタイムホイール法と呼ぶ）．この場合，容易に想像されるように 1 タイムステップ内で活性化（入力ピンの信号が変化したことをいう）される素子の数は非常に少ない．したがって，この方法ではデータ並列的な並列処理を直接適用しても得られる速度向上は小さい．

この場合の対処法としては以下の 2 種類の方法があげられる．

- 機能分割的な並列アルゴリズムを用いる方法
- 異なるタイムステップのシミュレーションを同時に行う方法

前者の方法はシミュレーションの処理の単位ごとにプロセッサを割り当て，パイプライン的に作動させるものである．ただし，この手法ではプロセッサ間で細かい単位で同期をとる必要があり，専用マシンには適しているが，汎用並列処理に適しているとはいい難い．

後者の方法として，分散離散系シミュレーション (Parallel Discrete Event Simulation (以後 PDES)

アルゴリズム[6]がある．PDES では，各プロセッサごとに，独立して時間を管理し，問題のもつ並列性をより多く活用することが可能となる．

PDES は，論理シミュレーションに捕らわれない広い概念であるが，以後 PDES を論理シミュレーションに適用した場合について説明する．

PDES を論理シミュレーションに適用した場合，各素子の入力端子に，イベントの生成された時間（タイムスタンプと呼ぶ）と，その論理値が伝達されることになる．

この場合注意しないといけないことは各素子の異なる端子に到着するイベントがイベント生成時間順に到着しないこともあり得ることである．この場合に正しくシミュレーションを実行できるようにするためのアルゴリズムとして，Virtual Time[7]に代表される楽観的な方法（以後バーチュアルタイム法と呼ぶ）と Chandy-Misra アルゴリズム[8]に基づく保守的な方法（Chandy-Misra 法と呼ぶ）が存在する．

バーチュアルタイム法では，イベントがきたときに（タイムスタンプ t_0 とする），評価を行い，そのイベントを接続先のイベントに伝達する．(そのときに古い状態を履歴情報として保存する．)その後タイムスタンプが t_0 より前のイベントが別の入力端子に到達した場合は，再度評価を行い，結果が異なれば接続先に新イベントをキャンセルするメッセージ（このメッセージを「アンチメッセージ」と呼ぶ）を送る（再度評価を行い，新イベントをキャンセルすることを「ロールバック」を行うと呼ぶ）．バーチュアルタイム法ではこのロールバックを行うために必要な履歴情報を格納する必要がある．一方全履歴を保存することは不可能なので，シミュレーションの実行時に時々システム内での最も小さいイベント生成時刻を求め，その時刻以前の履歴領域を解放する必要がある．したがってバーチュアルタイム法でのオーバヘッドとしては，

- ロールバックを行うためのオーバヘッド（PE 内処理ならびに PE 間通信）
- 履歴情報を保持するためのオーバヘッド
- 履歴情報を解放するためのオーバヘッド

が存在する．

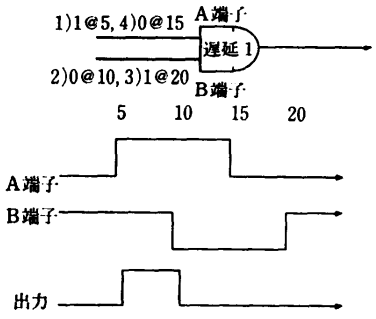
Chandy-Misra 法では，各素子の評価はイベントが全ての入力端子に最低 1 個到達するまで，待たされる．その後，最小のタイムスタンプのイベントに対して処理を行う．(**図-3** に同じ回路に対してイベントが到達したときの両者の振舞いの違いを示す．) Chandy-Misra 法の場合は楽観的な方法と異なり，ロールバックにともなうオーバヘッドは存在しない．一方，どれか 1 個の入力端子にイベントが到達しないと，その素子は金輪際シミュレーションされないことになる．この問題の回避方法として元の Chandy-Misra 法では，

- シミュレーションができなくなった時点で，システム全体での最小時刻を求め，その時点まで，システムの時間を進める方法（基本アルゴリズム）
- イベントを生成しない場合にも，出力データの有効な生成時間を更新するためのヌル (Null) メッセージを伝達する方法
- 必要に応じて，データの確定時間を問い合わせる問合せ方式

などが存在する．

3.2 実　現　例

バーチュアルタイム法の実現例としては，松本らが Multi-psi[9]上に実現したもの[10]がある．松本



イベントが	Chandy-Misra 法	バーチュアルタイム法
1) 1@5→A端子		1) 1@6 を発生
2) 0@10→B端子	2) 1@6 を発生	2) 0@11 を発生
3) 1@20→B端子		3) 1@21 を発生
4) 0@15→A端子	4) 0@11 を発生	4) 1@21 をキャンセル
の順番できたとき		

($x@y$ は y の時刻に x という値がきたことを示す)

図-3　バーチュアルタイムアルゴリズムと Chandy-Misra アルゴリズムの比較

らのシステムでは上記オーバヘッドを軽減するために以下に述べる工夫を行っている.

PE 間通信削減 従来 PE 間通信を削減する方法として，複数個のアンチメッセージが存在した場合，最もタイムスタンプの小さいアンチメッセージを送る方法が福井によって提案されていた[11]. 松本らはこれを改良し，タイムスタンプの最小のキャンセルメッセージより小さいイベントが存在する場合は，そのメッセージのみを送る.

負荷分散方法の工夫 PE 間通信とロールバックの回数削減，ならびに並列性の開拓のために，以下に述べる“縦割り指向戦略”を用いて静的に回路をプロセッサに割り当てる. 縦割り指向戦略では，連結したゲートは可能なかぎり同一 PE に割り当てるとともに，1個のゲートの出力が複数のゲートに接続されているときは，接続先のゲートを異なるプロセッサに割り当てる.

また複数個の未処理イベントが PE に存在する場合は，最もイベント生成時刻が小さいメッセージから処理することにより，ロールバックの頻度低減を図っている.

松本らの評価では，以上の工夫により，

- ロールバック処理のオーバヘッドが小さい.
- 回路が十分大きければ，PE 間通信のオーバヘッドが無視できる.

としている. 実際最良のケースでは64台時に1台のときの48.4倍の速度向上を達成している.

松本らはまた文献10)で同一マシン上で，3.1の ZYCAD 社の LE のところで説明したグローバルタイムホイール法ならびに保守的な方法のヌルメッセージを用いたものの比較も行っている. 図-4に同一回路に対するバーチュアルタイム法とグローバルタイムホイール法との速度の比較を示す. プロセッサ台数が増加したときのバーチュアルタイム法のグローバルタイムホイールに対する優位性を示す.

今後の課題としては現在は問題となっていないが，メモリ素子などのより複雑な素子のモデリングを行ったときのロールバックのために必要な退避領域の低減化があげられる.

Chandy-Misra 法の実現例として，Soule らは文献12)で基本的な Chandy-Misra Algorithm に以下のような最適化を施した.

デッドロック時の処理の軽減 入力端子に未処理のイベントのある素子のリストを保持すること，ならびに，各素子の最小時間を毎回更新することにより，デッドロック発生時の処理のオーバヘッドを削減する. これにより，単一プロセッサ上で，イベントホイール法と比較して4倍かかっていた処理時間を1.4倍に軽減した.

デッドロック発生の軽減 たとえば2入力 AND 素子は，片方の入力の信号値が0であれば，たとえ，もう一方の信号値が未知でも出力値は決定できる. 同様に FF 素子の出力はクロックの入力が変化しないかぎり，変化しない. このようなノウハウを用いることにより，デッドロックの発生自体を軽減させるとともに，得られる並列性を高めることが可能となる. このような最適化により，64台のプロセッサ時に，1台のイベントホイール法のアルゴリズムのシミュレータと比べて2.5倍から16倍の速度向上が得られるとしている[12].

もう一つ Chandy-Misra 法の実現

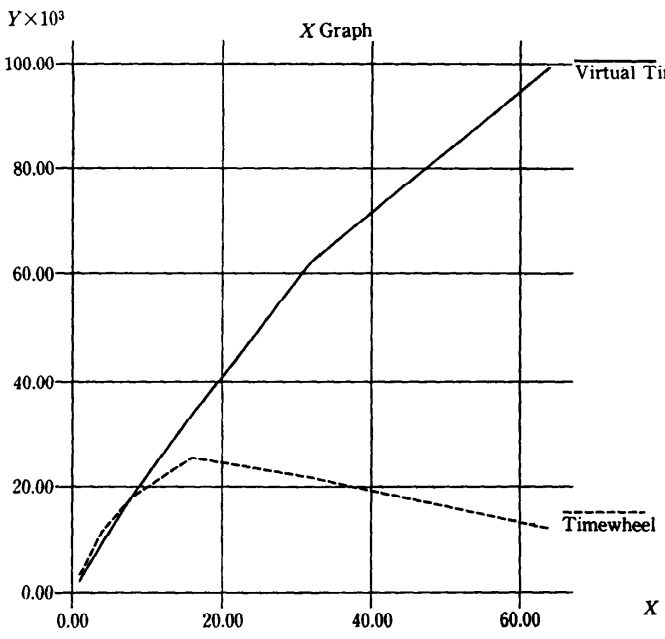


図-4 バーチュアルタイム法とグローバルイベント法の比較（文献10)のデータを元に作成）横軸が PE 台数，縦軸が実行時間を表す.

方法として，デッドロック発生時にイベント未到着の入力端子に接続されている素子に対して問合せを行うシミュレータ LOGIQUEを 工藤らが ATTEMPT0[13] 上に実現している．通常問合せ方式は問合せに関するオーバヘッドが大きいとされているが，工藤らはこれを以下に述べる方法で軽減した．

● 各素子は出力が確定していることを保証する保証時刻を保持する．問合せ処理を行うためには基本的には，各素子の保証時刻を共有するメモリを介して読み出すことにより実現する．（したがって LOGIQUE は共有メモリマシン固有のアルゴリズムであると言える．）一レベルの問合せで，値が確定しない場合は，さらに上流の素子に遡って問合せを実行する．

● 不必要に上流に遡って問い合わせることを避けるために，問合せの過程にある素子の保証時刻を更新する．

これらの改良により，単一プロセッサ上での LOGIQUE の実行速度は通常のイベント駆動シミュレータの 1/2 程度であるという．一方 8 台で並行させた場合の速度向上率は 5 倍で飽和した小さい回路を除くと，約 8 倍から 10 倍程度である．（8 台のときの速度向上率が 1 台のときの 8 倍以上である理由は記述されていないが，おそらくキャッシュのヒット率が良くなったためではないかと思われる．）．

4. 回路シミュレーション

回路シミュレーションは回路の素子特性を元に回路の各節点に関する非線形常微分方程式を解いて，回路の振舞いをアナログレベルで解析するものである．回路シミュレーションでは非線形常微分方程式を，後退オイラー法を高次に拡張したものと，Newton-Raphson 法を組み合わせて解く[15]．過渡解析の流れは図-5 のようになる．この中で最も時間を占めるのはステップ 4 と 5 の Newton-Raphson 法に基づくループの中である．500 トランジスタのスタティック RAM の回路に対して単一プロセッサで実行した場合の計算時間の分布は表-1 のようになる[16]．したがって数十程度の並列性を得るためには図-5 の処理の少なくともステップ 3:，4:，5: を並列化する必要がある．

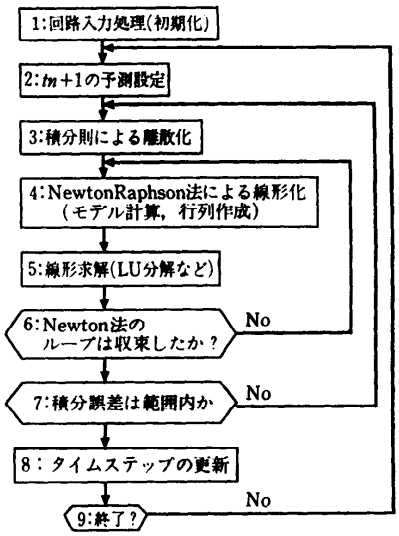


図-5 回路シミュレーションの処理手順

表-1 500 トランジスタの DRAM 回路での処理の分布

処理の内容	処理の比率
モデル評価	83%
線形方程式の求解	9%
時間離散化処理	6%
その他	2%

回路シミュレーションを並列化する方法としては，大別して a）直接法，b）緩和法，c）モジュール分割法の 3 種類が存在する．

直接法[17]では回路解析処理を上述した処理ステップにわけ，おのおののステップごとに並列性を開拓する．モデル計算は比較的容易に並列に処理可能であるが，しかし行列への代入過程でメモリのアクセス競合が起こりメモリのロック処理などでオーバヘッドが大きくなる．また，行列演算においては一般に LU 分解が用いられるが行列要素間の依存性が高く，並列性がそれほど得られない（図-6 参照）．したがって 10 台程度までの比較的少ない数の細粒度な並列処理が要求される．メモリ共有型の密結合システムが向いていると考えられる MIMD 型並列計算機システムに適した方法であると言える．

緩和法[18),19]に基づく並列処理では回路を分割しおのおのの部分回路を別個に解き全体の解が収束するまで計算を繰り返す．回路をどのレベルで

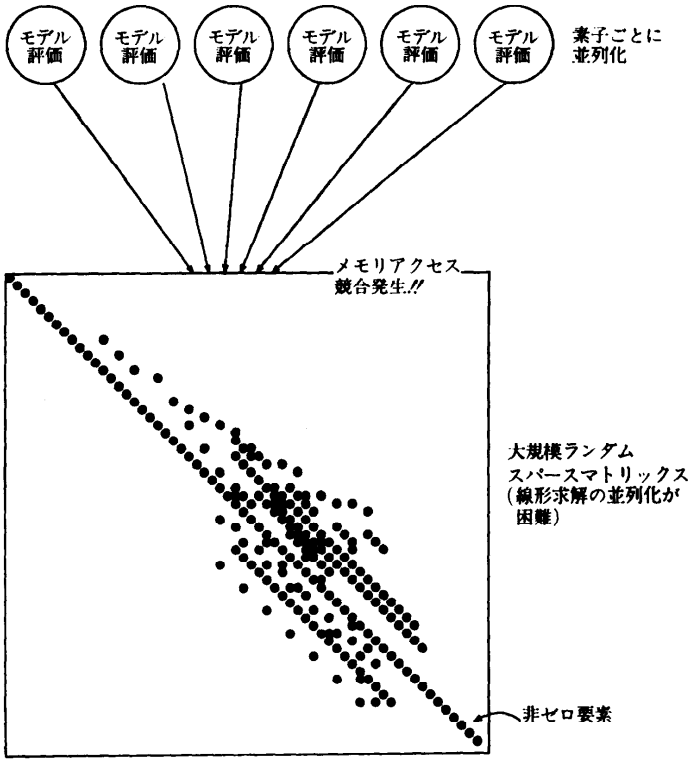


図-6 直接法での並列化の問題点

分割するかによって反復タイミング解析法や波形緩和法などがある．緩和法では各反復回ごとにおのおのの部分回路はそれまでに求めた近似解を使用するので，おのおのの計算量が少なく，しかも部分回路の計算を信号の伝播順に行うなどして並列に／パイプライン式に同時処理可能である．さらに，イベント駆動及び部分回路ごとに時間刻みを変えるなどの手法を併用し計算量を削減することもできる利点がある．文献 19) では NOT 回路を 50 個直列に結合したような理想的な回路を対象として，1 台のときの 20 倍程度の速度向上を得たものが報告されている．

一方，大きなフィードバックループを含む回路では反復回数が増大し，収束性／精度に問題がある．したがって適用対象の回路をある程度限定した場合に有効な手法であると言える．

モジュール分割法を用いた並列化[20,21]は回路を非線形素子を含む部分回路群と，部分回路群と境界変数を共有し，部分回路群全体を結ぶ素子を含まない接続回路網とに分ける（図-7 参照）．Newton-Raphson 法のループ内では直接法と同じ

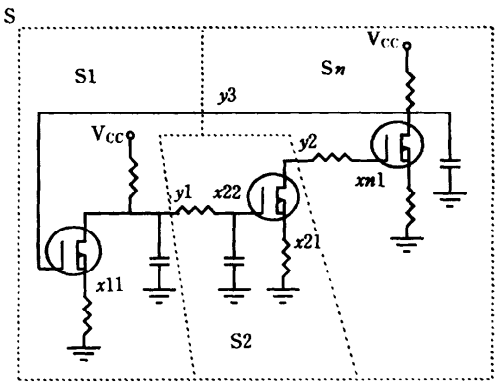


図-7 モジュール法の概念

処理を以下の 3 ステップで行う．

並列ステップ 1 各部分回路ごとにプロセッサを割り当て，モデル評価を行い，各部分回路ごとの行列を作成する．さらに接続回路網に関する境界変数のみの方程式を作成する．

逐次ステップ 接続回路網に関する境界変数のみの方程式を 1 台のプロセッサで解き，境界変数の解を得る．

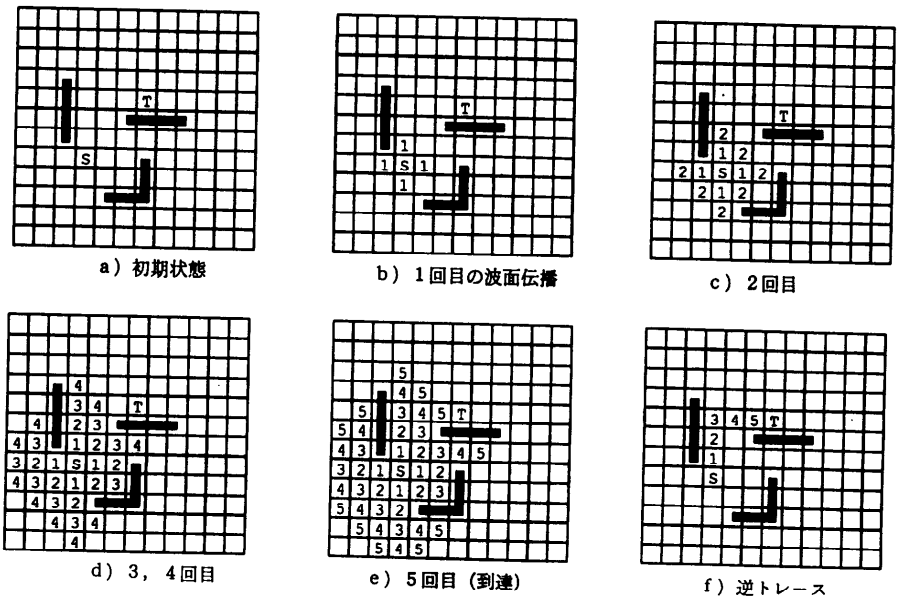


図-8　Lee-Moore アルゴリズム

並列ステップ2　境界変数の解を元に，各プロセッサで部分回路ごとに線形方程式を解く．

モジュール分割法では，解の収束性・精度は従来の直接法と同等であることが保証されている．さらに並列性の面からみると部分回路の計算では互いに独立に同時処理可能であり高い並列性がある．一方，逐次ステップが逐次的な処理となる．

筆者らが並列シミュレーションマシン Cenju[21] 上で7000トランジスタの回路のシミュレーションを64台の並列プロセッサシステムで実行したところ，得られた速度向上は15.8倍であった．これは前述の逐次ステップを1台のプロセッサで実行したため，そこが並列処理での隘路になったためである．

接続回路網の線形方程式に相当する行列の非零要素率は20-30%と元の行列の非零要素率よりかなり高い．そこで，逐次ステップの主な処理である LU 分解を並列化したところ，速度向上率は25.8倍となった[22]．

5. 配線処理

5.1 配線処理のアルゴリズム

配線処理は LSI チップやプリント基板の，論理素子の端子間の信号線を障害物を避けながら，配線していく処理である．

配線処理のアルゴリズムとしては，a）迷路法，b）線分探索法，c）チャネル法などが存在する．

迷路法[23]では，メッシュを2次元格子状にマップし，接続したい端子から，到達できる端子に順にラベルを付けながら，探索範囲を広げていく（図-8参照）．迷路法での探索プロセスはいわば，石を水面に落としたときの波面の伝播に類似させることができる．この場合，プロセッサを格子にマップさせて並列処理を行うことが考えられる．しかし，迷路法における並列性の粒度は非常に細かい．したがって，迷路法を並列化した例としては，MAPLE-RP[24]のように専用ハードウェアを用いたものや SIMD 的なものが多い．

線分探索法は図-9に示すとおり，次の手順で

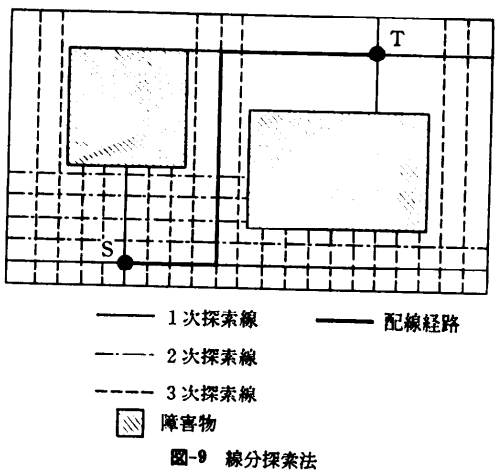


図-9　線分探索法

配線処理を行う.

1. 始点と終点から，X，Y 方向に探索線を出す．これをレベル 0 の探索線とする.

2. もし両方の探索線が交差すれば経路が見つかったことになり，5)へいく．見つからない場合は

3. そのレベルの探索線群上にあり，かつ設計規則を満足する全ての線上にある点（ベース線）を求め，

4. 各点から直角方向に同様に探索線を引き，これを次の探索線群とし，2)からの処理を繰り返す.

5. 発見された経路を新たに禁止領域として図形を更新し，次に配線すべき線があれば1)から繰り返す.

線分探索法は迷路法に比べて，メモリ容量が少なく，高速に処理が可能となる．線分探索法を並列化した例としては，Multi-PSI 上で並列化されたもの[26]，Coral 68 K[27]上で並列化したもの[28]，Cenju 上で並列化したもの[29]などが存在する．紙面の都合上 Cenju 上で並列化したものについてのみ，後で述べる.

迷路法や線分探索法が一般的なカスタム LSI，ゲートアレイ，プリント基板などに広く用いられるアルゴリズムであるのに対して，チャネル法は，ASIC の一種であるスタンダードセルやゲートアレイで用いられる手法である．スタンダードセルの場合は，セル群が水平方向に密に列をなし，列と列との間がチャネルと呼ばれる配線領域になる（**図-10** 参照）．チャネル法の並列処理の例として，LocusRoute[25]の場合について後述する.

5.2 配線処理で得られる並列性

一般的に配線処理で得られる並列性には以下の 2 種類の並列性が存在する.

ネット内並列性 これは，一つのネットの配線処理に内在する並列性であり，線分探索法，迷路法などのアルゴリズム自体の並列性であるとも言える．たとえば，迷路法の場合には，配線領域の格子に波面伝搬のラベル付けをするという処理の繰り返しなので，2 次元メッシュ状にプロセッサを並べ，波面伝搬の処理を並列処理させるという手法が数多く提案されている．しかし，波面伝搬に内在する並列性は Wave Front にしか存在しないので，N 台のプロセッサに対して $\sqrt{N}$ のオーダの並列性しか得られないという問題点があり，プロセッサのマッピング方法を工夫している例もある.

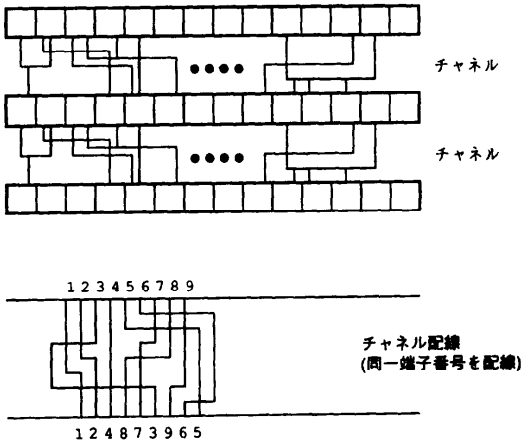


図-10 チャネル法

ネット間並列性 複数のネットを同時に配線する

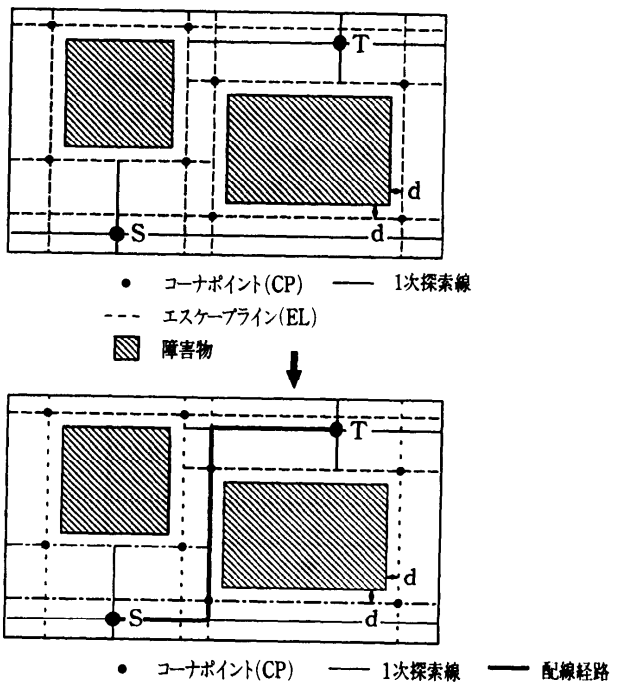


図-11 改良線分探索法

ことによって得られる並列性である．ネット間並列性を用いた種々の並列配線アルゴリズムも提案されているが，高い並列性を得ようとすると配線率，配線の品質が低下するというものもある．ネット間並列性を活用しても配線の品質の低下は避けるべきであると考えられる．

ネット内並列性，ネット間並列性のいずれを活用する場合にも通常は，なんらかの領域分割を用いてデータ並列的な処理を用いる場合が多い．

5.3 配線処理の並列化の実現例

山内らが Cenju 上で並列化したルータ PROTON[29] では配線アルゴリズムは「改良線分探索法」[30] を基本としたものである．このアルゴリズムでは前節で述べた線分探索法での無駄を省くために，各レベルでの探索線の候補として障害物から d（障害物と配線の許容距離）だけ離れた位置における点 (CP) を結んだエスケープ・ライン (EL) を用いる(**図-11** 参照)．そして，それらの EL の中から最少折れ曲がりの経路を求める．

PROTON での並列処理は以下の 2 種類の並列処理を組み合わせて用いる．

ネット内並列処理　改良線分探索法の処理の中で最も重い部分は，探索線と交差（直交）するエスケープ・ラインを探す部分である．PROTON では配線領域を配線方向に従って帯状に分割して各 PE にそれぞれの配線領域内での線分の探索を担当させることにより，配線処理の並列化を行っている．並列に探索している様子を**図-12** に示す．

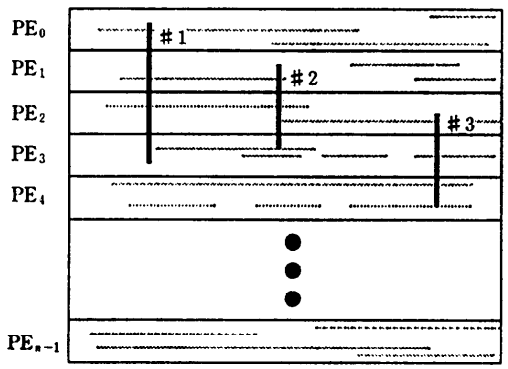


図-12 PROTON でのネット内並列性

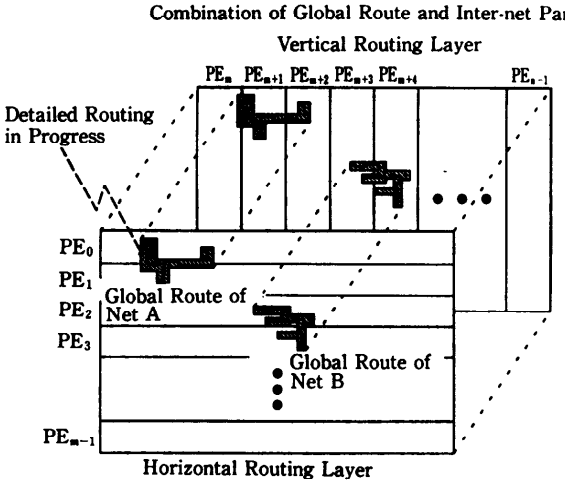


図-13 PROTON でのネット間並列性

たとえば，PE 0～PE 3 は，探索線 #1 と交差するエスケープ・ラインを並列に探すといったぐあいに探索処理を並列化している．

ネット間並列処理　一般的に，LSI の自動配線では探索空間を絞り込むために，概略配線を用いて，あらかじめ各ネットの大まかな経路を定める．詳細配線は概略配線の領域内で行われるということを考えると，概略配線領域が重ならないネットは同時に詳細配線処理を行うことが可能である．PROTON では，概略配線の領域が重ならないネットを並列に配線することにより配線品質を低下させることなくネット間の並列性を引き出している．PROTON でどのようにネット間の並列性を用いているかを**図-13** に示す．

PROTON では上記した 2 種類の並列性を活用することにより中規模のチャネルレス・ゲートアレイ (1,537×1,790 グリッド，12,591 ピンペア) を配線対象とした場合に 64 台のプロセッサで 43 倍の速度向上を得ている．

LocusRoute[25] は，チャネル法を並列化したものである．LocusRoute では，チャネル内の各配線格子点に対応した要素からなるコスト表を用いている．並列に配線する場合は共有メモリ上に配置されたコスト表を複数のプロセッサが排他的にアクセスして，配線を行う．

チャネル法では 1) ネット間並列処理，ならびにネット内並列処理として，2) セグメント間の

並列処理，ならびに 3) 異なる複数経路間の並列処理が活用できる．LocusRoute では，15 プロセッサ構成の Encore/MULTIMAX を用いて，1)のネット間並列処理で，15 台で 6.9 倍から 13 倍，3)の異なる複数経路間の並列処理で 8 台で 4.6 倍の速度向上を達成している．2)に関しては処理のばらつきによりあまり高い速度向上は得られなかった．したがって 1)と 3)を組み合わせて 120 台で 55 倍の速度向上が見込まれるとしている．実際には共有メモリマシンを仮定しているため，ここまで高い速度向上が得られるとは考え難い．

6. おわりに

電気系 CAD 分野における，MIMD 型並列計算機のアルゴリズムの例として，論理シミュレーション，回路シミュレーション，配線処理を例にとりあげ，解説した．

CAD 分野では問題自体にもともと並列性を有するものが多く，また処理速度の向上の要求が高いことから，科学技術計算とともに並列処理の研究が盛んに進められている．実際にはどの応用でも，1)逐次処理のオーバヘッド，2)付加分散のオーバヘッド，3)通信のオーバヘッドを削減する種々の工夫を行って，初めて実用的な速度向上が得られることが分かっていただければ幸いである．

CAD 分野ではこのほかに故障シミュレーションや DRC (Design Rule Check) で並列処理の研究が行われている．また，本解説でのべたアーキテクチャとはまったく異なった，より超並列処理に適した観点から，機能メモリを用いた CAD 処理の研究もなされている．これらの研究については，文献 31) に詳しく解説されているので，合わせて参考にしていただければ幸いである．

参考文献

1) 富田眞治：並列計算機構成論，昭晃堂 (1986).
2) 小池誠彦：CAD マシン，オーム社 (1989).
3) Pfister. et al.: *The Yorktown Simulation Engine: Introduction*, Proc. 19th DA Conf., pp. 50-54 (1982).
4) 小池他：論理シミュレーションマシンのアーキテクチャ，情報処理学会論文誌，Vol. 25, No. 5, pp. 864-872 (1984).
5) 山田他：シミュレーションプロセッサ *SP*，電子情報通信学会論文誌 D, Vol. J71-D, No. 4, pp. 644-651 (1988).
6) Fujimoto, R. M.: *Parallel Discrete Event Simulation*, Comm. ACM, Vol. 33, No. 10, pp. 30-53 (1990).
7) Jefferson, D. R.: *Virtual Time*, ACM Trans. Prog. Lang. Syst., Vol. 7, No. 3, pp. 404-425 (1985).
8) Chandy, K. M. and Misra, J.: *Distributed Simulation: A Case Study in Design and Verification of Distributed Programs*, IEEE Trans. Software Eng. Vol. 5, No. 5, pp. 440-452 (1979).
9) Taki, K.: *The Parallel Software Research and Development Tool: Multi-PSI System*, Programming of Future Generation Computers, Fuchi, K. and Nivat, M. (eds.), pp. 411-426, North Holland (1988).
10) 松本幸則他：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).
11) 福井：バーチャルタイムアルゴリズムの改良，情報処理学会論文誌，Vol. 30, No. 12, pp. 1547-1554 (1989).
12) Soule, L. et al.: *Parallel Distributed-Time Logic Simulation*, IEEE Design and Test of Computers, No. 6, pp. 32-48 (1989).
13) 鳥居，天野：並列計算機テストベッド ATTEMPT の交信機構の評価，並列処理シンポジウム JSPP '91 論文集，pp. 205-212 (1991).
14) 工藤他：問い合わせに基づく並列論理シミュレーションアルゴリズム，電子情報通信学会論文誌，Vol. J75-D-I, No. 4, pp. 221-231 (1992).
15) Wiliam, J. M.: *Fundamentals of Computer-Aided Circuit Simulation*, Kluwer Academic Publishers (1988).
16) Nakata, T. et al.: *A Multiprocessor System for Modular Circuit Simulation*, Proc. ICCAD 87, pp. 364-367 (1987).
17) 鹿毛：VLSI 回路シミュレーション，電学論C，Vol. 107, No. 6, pp. 519-524 (1987).
18) Deutsch, T. et al.: *Parallel Computing for Circuit Simulation*, VLSI SYSTEMS DESIGN, pp. 46-52 (July 1986).
19) Jacob, G. K., Newton, A. R. and Pederson, D. O.: *An Empirical Analysis of the Performance of a Multiprocessor-Based Circuit Simulator*, Proc. 23rd DA Conf. pp. 588-593 (1986).
20) Cox, P., Burch, R. and Epler, B.: *Circuit Partitioning for Parallel Processing*, Proc. ICCAD '86, pp. 186-189 (Nov. 1986).
21) 中田，松下，小池他：並列回路シミュレーションマシン Cenju，情報処理学会 30 周年記念論文，情報処理，Vol. 31, No. 5 (1990).
22) Nakata, T. et al.: *Cenju: A Multiprocessor System with a Distributed Shared Memory Scheme for Modular Circuit Simulation*, Proc. International Symposium on Shared Memory Multiprocessing, pp. 82-90 (Apr. 1991).
23) Lee, C. Y.: *An Algorithm for Path Connec-*

tions and Its Applications, IEE Trans. on. Electrical Computers, Vol. EC-10, pp. 346-365 (1961).
24) 河村他：超並列配線マシン MAPLE-RP, 並列処理シンポジウム JSPP '91 予稿集, pp. 373-379 (1991).
25) Rose, J.: The Parallel Decomposition and Implementation of an Integrated Circuit Global Router, Proc. ACM SIG-PLAN 88, pp. 138-145 (1988).
26) 伊達他：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 378-379 (1992).
27) 高橋他：二進木並列計算機, *Coral 68 K* の開発と性能評価, 情報処理学会論文誌, Vol. 30, No. 1. pp. 46-57 (1989).
28) 佐野他：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, pp. 369-378 (1992).
29) Yamauchi. et al.: *Proton: A Parallel Detailed Router on an MIMD Parallel Machine*, Proc. ICCAD '91 (1991).
30) Suzuki, K., Ohtsuki, T. and Sato, M.: A Gridless Router: Software and Hardware Impelmentations, VLSI '87, pp. 121-131 (1987).
31) 佐藤他：特集機能メモリのアーキテクチャとその並列計算への応用, 6: LSI CAD 分野への応用, 情報処理, Vol. 32, No. 12, pp. 1276-1286 (1991).

(平成4年6月2日受付)

中田登志之（正会員）

昭和32年生. 昭和57年京都大学大学院工学研究科修士課程修了. 昭和60年同大学院博士後期課程単位取得退学. 同年日本電気(株)入社. 京都大学工学博士. 現在同社C&Cシステム研究所コンピュータシステム研究部研究課長. 並列計算機システムの研究に従事. 昭和61年度本学会論文賞受賞. 30周年記念論文入賞, 電子情報通信学会会員.