

並列回路シミュレーションマシンのプロトタイプ†

中田 登志之^{††} 田 辺 記 生^{†††} 小野塚 裕美^{††}
黒 部 恒 夫^{†††} 小 池 誠 彦^{††}

回路シミュレーションを並列に実行するアルゴリズム, ならびにアーキテクチャについて述べる. 本システムでは, モジュール分割法に基づいて, シミュレーションの対象となる回路を非線形要素を含む複数の部分回路群と部分回路群を結合する接続回路網とに分け, 部分回路ごとの計算と全体の接続回路網の計算を収束するまで交互に繰り返す. 回路シミュレーションの 90% 以上の演算量を占める部分回路群におけるモデル評価ならびに行列演算は, 各部分回路ごとに独立に演算可能である. 一方接続回路網の行列演算は逐次的な処理となるが, 全体の計算量から比べると高々 1-2% に過ぎない. したがって, 全体としては数十台程度のプロセッサを用いた並列処理が実現可能である. 今回実際に 4 台構成のプロトタイプ・システムを作成した. プロトタイプでは 4 台の MC 68020 をそれぞれ, 4 MB のメモリに結合したものをさらにグローバルバスで結合し, メモリの構成方式としては分散共有メモリ方式を採用した. 500 トランジスタの疑似 SRAM の回路のシミュレーションを実行したところ, 4 台実行時に 1 台のときと比べて 3.9 倍と高い並列性を得た.

1. はじめに

VLSI の急速な発達により VLSI チップの設計および検証が一層困難で時間のかかる作業となっている. 設計検証用のツールである回路シミュレーションは回路に関する非線形方程式を数値解析的に解くことにより, 各接点の電流・電圧を求めるものである.

回路シミュレーションは倍精度浮動小数点演算を多用するため, VLSI 回路の規模が大きくなるにつれて, そのシミュレーション時間の増加が問題になってきている. 特にメモリ回路の設計等では, 回路シミュレーション以外に有効な設計手法が確立されていないので, 各種パラメータを変えながらシミュレーションを繰り返し回路の最適化を行う方法が採られている.

米国における報告⁹⁾によると, 設計フェーズにおける計算機使用時間の 5 割以上が回路シミュレーションに費やされているという. また, 45 万素子のマイクロプロセッサの 1 命令の回路シミュレーションに 100 MIPS の計算機を用いたとしても 7 日間も必要とされている.

回路シミュレーションは科学技術計算の範疇にあるのでベクトル型のスーパーコンピュータの利用がまず考えられる⁹⁾. しかし, 回路シミュレーションでは非線

形要素を含む大規模なスパースマトリクスの微分方程式を解く必要があり, ベクトル長が短くなるので従来型のベクトル計算機がそれほど有効に働かないことが指摘されている. 一般的には汎用大型計算機に比べ高々 5 倍から 10 倍程度の速度向上にとどまっているのが現状である. 回路シミュレーションを高速化するためにはベクトル型よりむしろ MIMD 型の並列マシンによるアプローチが向いていると考えられる¹⁰⁾.

そこで, 筆者らは並列向きの数値演算アルゴリズムおよび並列処理に基づく専用のアクセラレータを開発することとした.

アルゴリズムを開発する際には従来の逐次計算機上のシミュレータと同じ結果と精度を保つこと, ならびに並列処理における隘路となり得る単一実行部の占める割合を最低限に抑えることを念頭に置いた. 今回, 実際に 4 台構成のプロトタイプ・システムを開発した¹¹⁾⁻¹⁴⁾. 4 台構成で動かしたとき, 1 台のときと比べて 3.9 倍の高速化を実現し, 提案しているアルゴリズムおよび方式は数十台規模の並列マシンにおいても有効であることを確認した. 本稿ではそのマシンアーキテクチャ, 並列アルゴリズムおよび評価結果について報告する.

2. 回路シミュレーション高速化の方策

高速な回路シミュレータを実現するためには次の点について考慮したシステム設計が重要である.

① 並列アルゴリズムの開発

従来のシミュレーションプログラムと結果および精度において互換性があり, しかも高い並列性が開拓で

† A Prototype Multiprocessor System for Modular Circuit Simulation by TOSHIYUKI NAKATA (C&C System Laboratories, NEC Corporation), NORIO TANABE, HIROMI ONOZUKA, TSUNEO KUROBE (VLSI CAD Engineering Division, NEC Corporation) and NOBUHIKO KOIKE (C&C System Laboratories, NEC Corporation).

†† 日本電気(株)C&C システム研究所
††† 日本電気(株)超 LSICAD 技術本部

きるものが望ましい。

② マシントポロジの選択

問題に適合した計算機相互結合網を採用し、拡張性に優れた数十台規模まで性能低下が影響しないトポロジを採用する。

③ 構造的/時間的スパース性の利用

問題に含まれるデータの局所性などをいかすことによりプロセッサ間の通信および計算量の削減を可能とする。

④ 浮動小数点演算機能の強化

回路シミュレーションでは浮動小数点演算が主体であり、しかも倍精度の演算精度が必要とされる。したがって、各々のプロセッサの MFLOPS 性能を高めることが重要となる。

2.1 並列アルゴリズムの選択

アムダールの法則によると、全処理に対して単一プロセッサでしか処理できない部分が p ($0 \leq p \leq 1$) 存在し、全体で n 台のプロセッサが存在する場合 1 台に対する速度向上は

$$1/(p+(1-p)/n) \quad (1)$$

で表される。したがって例えば単一実行処理の部分が全体の 5% しかない場合でも得られる並列度は高々 20 になってしまう。

回路シミュレーションではほとんどの時間を占める過渡解析を 500 トランジスタの疑似 SRAM の制御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以下ようになる。

① モデル評価	83%
② 行列演算	9%
③ 時間離散化処理	6%
④ その他	2%

したがって数十程度の並列性を得るためには①, ②, ③の処理をすべて並列化する必要がある。

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⁷⁾⁻⁹⁾では回路解析処理を上述した処理ステップに分け、各々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ー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のスパース性が高く、並列性がそれほど得られない。このように直接法をそのままの形で並

列化するとデータ/行列要素間のアクセスが頻繁で、しかも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が、システムの大規模化が難しく大きなスピードゲインが得られない問題点がある。

スーパーコンピュータを用いて LU 分解およびモデル計算をベクトル化する試みも行われているが、モデルの精度を高くすると共通に計算できるモデルの数が少なくなりベクトル長が短くなってしまいうので大きなスピードゲインが得られない。

緩和法⁵⁾に基づく並列処理では回路を分割し各々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。緩和法では各反復ごとに各々の部分回路はそれまでに求めた近似解を使用するので、各々の計算量が少なく、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、イベント駆動およ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。しかし、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/精度に問題があり適用対象の回路が制約される等の問題がある。

本システムで用いたモジュール分割法⁶⁾は回路を非線形素子を含む部分回路群と、部分回路群を結ぶ全体の素子を含まない接続回路網とに分け、部分回路ごとの計算と、全体の接続回路網の計算を収束するまで交互に繰り返す。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見ると演算の 90% 以上を占める、部分回路群における、モデル評価、行列演算の大半、ならびに時間離散化処理は、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、接続回路網の行列演算は逐次的な処理となるが、非線形素子を含まないで計算量は少なく、全体の計算量から比べると高々 1-2% に過ぎず数十台の並列処理ではそれほどオーバーヘッドにならないと考えられる。

モジュール分割法によれば各々の計算処理プロセスの間のデータ転送量が比較的少なく、各々のプロセスの処理粒度が粗いので分散メモリ型のマルチプロセッサ構成が適しており、大規模システムとしたときでも資源競合による性能低下の恐れが少ない特徴がある。

2.2 マシンアーキテクチャの検討

専用の並列マシンアーキテクチャを考える場合、問題の持つ特性・トポロジが特に重要である。モジュール分割法における処理プロセス間の関係およびデータの流れを図1に示す。本アルゴリズムにおける特徴をまとめると次のとおりである。

- ① 親プロセスと複数の子プロセスがトリー状の構成となり通信は親子の間だけである。
 - ② 親と子プロセスの間は部分回路間に共通する変数に関するデータだけを授受し、親から子へ共通変数を送るフェーズと、子から親へ共通変数を送るフェーズがある。
 - ③ プロセス間の同期は親と子の間のローカルな同期と、全プロセス間のグローバルな同期が必要。
 - ④ 各々のプロセスの処理は浮動小数点演算集中で処理の単位は大きい。
 - ⑤ 親と子の関係はさらに複数レベルに階層化可能である。
- 以上の特性を損なわないアーキテクチャとして次のようなデータアクセスパターンに最適化したアーキテクチャの設計を行った。

【ローカルメモリ方式】

① プロセスの内部処理においては各々のプロセッサは他のプロセッサとアクセス競合を起こさないようにローカルメモリをおく。さらにローカルメモリを2ポート化し、他のプロセッサからもアクセス可能とする。

【分散共有メモリ方式】

② プロセッサ間でデータを効率良く転送するためにメモリ共有機構を設ける。各々のローカルメモリはシステムにユニークなアドレスを与え各プロセッサはローカル/リモートメモリのすべてをアクセス可能とする。ただし、自ローカルメモリへはグローバルなバスを經由せずローカルなバスを介してアクセス可能とし、グローバルバスのトラフィックの増大を抑制する。

③ プロセス間の論理的なトポロジ(木構造)は問題によって変わるので、親と子のプロセスを任意のプロセッサに割当可能とするために物理的なプロセッサのトポロジは均質な構造とする。

④ 各々のプロセッサの処理は倍精度浮動小数点演算が主体となるので、高性能 32 ビットマイクロプロ

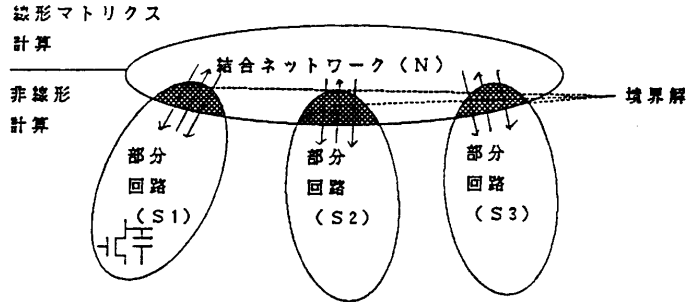
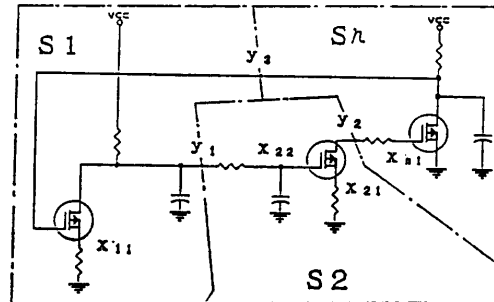


図1 処理プロセスと通信パターン
Fig. 1 Communication pattern among processes.

従来方式:

$$S(x_{11}, x_{21}, x_{n1}, x_{11}, x_{21}, x_{n1}, y_1, y_2, y_3, y_1, y_2, y_3) = 0$$

S



本方式

ステップ1 (部分回路の内部計算)

$$S1(x_{11}, x_{11}, y_1, y_3, y_1, y_3, t) = 0$$

$$S2(x_{21}, x_{22}, x_{21}, x_{22}, y_1, y_2, y_1, y_2, t) = 0$$

$$S3(x_{n1}, x_{n1}, y_2, y_3, y_2, y_3, t) = 0$$

ステップ2 (部分回路間の共通ネットワークの計算)

$$N(y_1, y_2, y_3) = 0 \text{ 線形方程式求解}$$

ステップ3 (部分回路の線形求解)

$$N1(x_{11}, y_1, y_3) = 0$$

$$N2(x_{21}, x_{22}, y_1, y_2) = 0$$

$$N3(x_{n1}, y_2, y_3) = 0$$

図2 並列シミュレーション方式

Fig. 2 Strategy for parallelizing circuit simulation.

セッサおよび浮動小数点演算コプロセッサを用いる。各々のローカルメモリはコード領域、変数領域、テーブル領域等のために最低 4 MB を持たせる。

⑤ 高速化を主眼とし、OSでのオーバヘッドを最小限にとどめるためにOS機能を軽装化しプロセス間通信だけをカーネルとしてサポートする。

3. モジュール分割並列回路シミュレーションアルゴリズム

3.1 モジュール分割法

図2に従来の直接解法とモジュール分割並列解法の

違いを図式的に示す。従来法は1つの大規模・非線形・ランダム・スパースな常微分方程式を解く。一方本方式ではシミュレーション対象となる回路をn個の部分回路に分割し、解析処理を①複数の非線形な小規模な部分回路の常微分方程式群の求解、②接続ネットワークに相当する線形マトリクス計算と、③部分回路ごとの線形計算、の3段階を繰り返すことにより、実行する。処理を3段階に分けたことにより従来法に比べ、プロセス間同期処理、疑似解の生成、内部変数の再計算、接続ネットワークの計算などがオーバーヘッドとして加わる。しかし、前述の大半の処理は全プロセッサで並列に処理されるので並列処理の効果の方がはるかに高くなるので問題とはならない。ただし、接続ネットワークの処理が非並列な処理として残り、部分回路の分割数が多くなった場合に無視できなくなる可能性がある。本稿では詳しく述べないが本システムではマルチレベル化（部分回路をさらに接続ネットワークと部分回路に分割する）により、この問題に対処することを可能にしてある。

3.2 並列アルゴリズム

本システムで用いたモジュール分割並列回路シミュレーション処理の実行は、次の3つのフェーズからなる。本システムでは後者の2つのフェーズについて並列処理を適用している。

① 回路分割フェーズ

入力された回路をモジュール構造を有する部分回路に分割する。(なおこのフェーズは本プロトタイプでは作成していない。後に述べる評価では設計者が手で分割したデータを利用した。)

② 入力フェーズ

モジュール構造を持つ回路データを入力とし、部分回路に分割し各々のプロセッサに部分回路群をアロケートし、コンパイルおよびリンク処理を並列に行う。

③ 解析フェーズ

DC 解析、トランジェント解析を並列に実行する。

2つのフェーズを処理時間で比較すると解析フェーズが処理の大半を占め、入力フェーズは全体の 10% 以下に過ぎない。したがって、入力フェーズ

を並列化してもそれほど効果が得られないように見える。しかし、入力フェーズは今後扱う回路の規模が大きくなったときここで費やされる処理時間が無視できなくなると判断した。したがって入力フェーズにおけるコンパイル/リンク処理も並列化を行うこととした。

解析フェーズでは次の処理を繰り返すことになる。(子プロセス群と親プロセスはあらかじめ実際のプロセッサ群に静的に割り付けておく。)

【並列実行部-1】

各プロセッサが並列に動作して各部分回路 S_i の内部変数の疑似解を部分回路の非線形常微分方程式を解くことによって並列に求める。

【単一実行部】

結合ネットワーク（親プロセス）を担当するプロセッサが各々の部分回路の境界解をもとに線形方程式を解く。

【並列実行部-2】

各々のプロセッサが並列に動作し各々の部分回路

親プロセスの処理	子プロセスの処理
① $t_{n+1} = t_n + dt$ とし後退積分の行列係数を求める、 近似的解を与え、後退積分を行いチャージ電流を計算。	
② 独立なソースの値を解ベクトルにロード。	
③ 親と子プロセス群の同期。	
④ 結合ネットワークの境界変数を子プロセスに送出。	
	親プロセスより境界変数を入力し解ベクトルにロード。
⑤ 親と子プロセス群の同期、続いてモデル評価、マトリクスのセットアップ	
⑥	内部変数の求解、ノートン等価値を生成、親に送出
ノートン等価値群をマージし接続ネットワークの境界変数を求解、解を子プロセス群に送出	
収束判定	親プロセスより境界変数を入力し解ベクトルにロード、内部変数の修正、収束判定。
⑦ 親と子プロセス群の同期	
⑧ 全回路が収束したか？ 収束しなければ⑤へ、 収束すれば⑨へ。	
⑨ 親と子プロセス群の同期。	
⑩ 後退積分式の更新、積分誤差を計算。	
⑪ 全回路の最大積分誤差を求め、新しい時間刻みと積分次数を決定、子プロセス群に連絡。	
⑫ 最終時刻に到るまで①からの処理を繰り返す。	

図 3 過渡解析における処理フロー
Fig. 3 Processing flow in transient simulation.

Si の内部変数の真の値を求めるために線形方程式を解く。

以上の処理過程を処理フローとして図3に示した。

本方式で高速化を実現するためには、単一実行フェーズ処理時間の短縮および並列実行フェーズでの各々のプロセッサの負荷均衡化をはかることが重要となる。

4. 並列回路シミュレーションに適したアーキテクチャならびにそのプロトタイプ構成

4.1 並列回路シミュレーションに適したアーキテクチャ

3章でまとめたアルゴリズムを並列システム上で実現する場合、プロセッサ間通信は主に以下の2つの時点で生じる。

① 並列実行部-1の終わりで、各部分回路を担当するプロセッサがノートン等価値を生成して親のプロセスを担当するプロセッサに送出するとき。

② 単一実行部の終わりで、親プロセスを担当するプロセッサが線形方程式を解いた結果得られた境界解の値を各部分回路を担当するプロセッサに分配するとき。

特に良く用いられる単一レベルのモデルの場合、①の通信は n 台のプロセッサから1台のプロセッサへの書き込み、②は1台のプロセッサから n 台のプロセッサへの書き込みとなる。

したがって回路シミュレーションに適した並列プロセッサのアーキテクチャを構成する上での指針としては以下の4点が挙げられる。

a) ハイパキューブのように多数のプロセッサが細い線で同時に多数のプロセッサと通信できる形態よりは、たとえ、同時には数少ないプロセッサとしか通信できないにせよバス結合のように太い線で結合されている形態の方が有利である。

b) ①の通信で1台のプロセッサが処理の隘路にならないためにはメッセージ通信に基づくプロセッサ結合形態よりは、直接相手のメモリに書き込むことが可能な共有メモリに基づくプロセッサ結合形態の方が有利である。

c) 一方商用のバス結合型マルチプロセッサシステムのように、システムバスに共有メモリを配置して各プロセッサがそのメモリにアクセスする形態では、並列実行部-1ならびに並列実行部-2で、共有メモリが

隘路となり、各プロセッサでの並列処理を妨げる可能性が存在する。したがって各プロセッサがローカルにメモリを有する形態が望ましい。

d) 1本のバスに結合できるプロセッサ台数は高々十数台程度である。

これらの条件を考慮して、2.2節でも述べたように以下のようなアーキテクチャの選択を行った。

1) 分散共有メモリ方式の採用

プロセッサ間でデータを効率良く転送するためにメモリ共有機構を設ける。各々のローカルメモリはシステムにユニークなアドレスを与え各プロセッサはローカル/リモートメモリのすべてをアクセス可能とする。ただし、自ローカルメモリへはグローバルなバスを經由せずローカルなバスを介してアクセス可能とし、グローバルバスのトラフィックの増大を抑制する。

2) 階層バス構成の採用

プロセッサを8から十数台ごとにクラスタにまとめ、クラスタ内はグローバルバスで結合し、さらにクラスタ間の結合に新たな結合網を用いる。ただしクラスタ内・クラスタ間の通信はハードウェアならびにシステムソフトウェアで吸収し、シミュレーションプログラムにはシステム全体をユニークなアドレスでアクセス可能にする。

4.2 プロトタイプ・システムの構成

アルゴリズムおよびアーキテクチャの有効性を検証・評価するために、4台構成の小規模なプロトタイプ・システムを構築した。プロトタイプ・システムに用いたハードウェアは4.1節で述べた基本アーキテクチャの設計方針に基づきクラスタ数が1の小規模なシステムに適した構成としている。今後の大規模化に備えて、プログラムの構成、プロセッサ間通信方式はそのまま大規模システムでも利用できるようにした。

特にクラスタ間のメモリ読み出しは効率が悪くなる可能性があるため、Producer-Consumerのモデルに基づき、可能な限りデータの生成者がデータの消費者にデータを書き込むようにプログラムを作成するように工夫している。したがって、大規模システムを構成したときにもプロトタイプでの評価を外挿して考えることが可能である。

プロトタイプ・システムでは、図4に示すとおりグローバル/ローカルバスの2バス方式を採用した。各々のプロセッサは各々のローカルバスを介してローカルメモリにアクセスすることができるので、各々のプロ

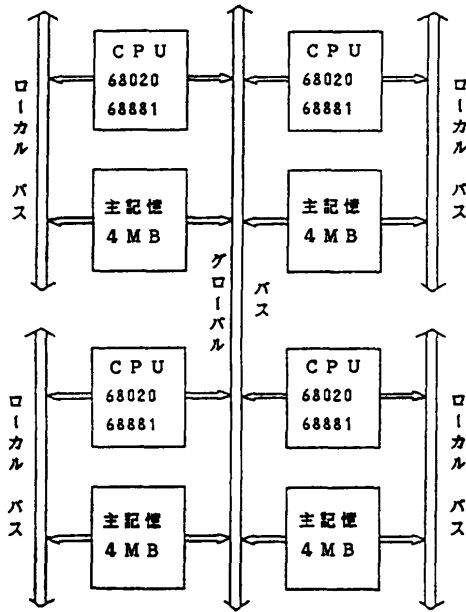


図4 プロトタイプシステムの構成
Fig. 4 System configuration of the prototype system.

セッサがローカルメモリを用いて内部処理を行っている間は互いのアクセス競合は発生しない。ローカルメモリはそれぞれ4MBの容量を持ち2ポート化されグローバル/ローカル両方のバスからアクセス可能である。グローバル・バス側から見るとすべてのローカルメモリは4MB単位の別のアドレス空間に割り当てられどのプロセッサからも共通にアクセスできる。

各々のプロセッサはCPUとメモリの2ボードで構成される。CPUボードはMC68020(12.5MHz)を主プロセッサとし浮動小数点コプロセッサMC68881(12.5MHz)などで構成される。

5. プロトタイプの性能評価

プロトタイプを用いて実際に約500トランジスタからなるMOSの疑似SRAMの制御回路の回路解析を行った結果について評価する。なお本回路は、設計者により負荷分散を考慮して4個の部分回路ならびにそれを結合するネットワークに分割されている。

5.1 入力処理における並列台数効果

表1は同対象回路の入力処理におけるCPU時間をプロセッサ数が1台から4台までの構成時について見たものである。4台構成時に性能向上は3.3と比較的良好な値を得ている。しかし、単一実行フェーズが4台構成時に既に30%と全体に占める比率が高くなっている。したがって、入力フェーズにおける並列処理の

表1 入力処理における実行時間
Table 1 Execution time for the input phase.

台数	実行時間(秒)	単一実行フェーズの割合(%)
1	54.14	9.6
2	30.06	16.4
3	23.42	20.8
4	16.53	30.3

表2 シミュレーション処理における実行時間
Table 2 Execution time for the simulation phase.

台数	実行時間(秒)	単一実行フェーズの割合(%)
1	5,112	1.0
2	2,630	1.9
3	2,263	2.3
4	1,311	3.9

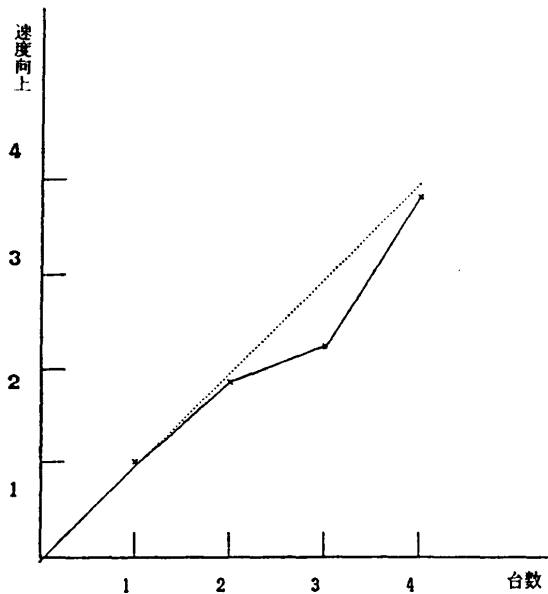


図5 シミュレーション時間とプロセッサ数
Fig. 5 Relationship between the number of processors and simulation time.

効果は10台程度が限界と考えられる。この数値は今まであまり試みられていなかった入力フェーズを並列化したという意味で意義が大きいと思われる。

5.2 シミュレーション処理における並列台数効果

表2および図5はシミュレーション処理におけるCPU時間を同様に1台から4台までの構成時について見たものである。4台構成時に3.9倍と高い速度向上が得られている。一方3台構成時には2.2倍と悪くなっている。これは部分回路の分割が4と少なく3台のプロセッサへのアロケーションが均等に行かず、負

荷不均衡によるものと思われる。

表2で注目すべきこととしては単一実行フェーズの全体の処理時間に占める割合が1台構成時で1.0%、4台構成時でも3.9%と少ないことが挙げられる。これは我々が採用したモジュール分割法によって行列演算部も並列化したことにより、単一実行部の占める割合を大幅に減少することができたことを示している。

5.3 処理粒度の影響

5.2節の結果からでは、一見この疑似SRAMの制御回路のような小さなデータでも100に近い並列性が存在すると思われる。しかし、以下に示すように実際は部分回路の個数が増えるにつれて単一実行部の比率が増えるため、それほどの並列性は存在しない。

本疑似SRAMの回路の分割数を替えてVax 780で実行させた場合の単一実行部の処理時間を表3に示す。分割数を増すとそれにつれて単一実行部の処理時間が増加する。

これを $y = a * m^x$ (m は部分回路数) で近似すると $x = 0.56$ となる。

したがってこの回路を理想的に n 台の部分回路に分割して、 n 個のプロセッサで実行した場合、単一実行部の比率 p は、

$$p = 0.00467 * n^{0.56} \quad (2)$$

となる。これを第2章で述べたアムダールの法則に適用すると、 n が45のときの速度向上が16.5で最大となる。したがってプロセッサ間での負荷の不均衡を考慮するとこの500トランジスタの疑似SRAMの制御回路のシミュレーションにおいて得られる速度向上は1台に対して約10倍程度と予想される。

一方実際のシミュレーションの対象となる回路としては数千トランジスタ程度とこの評価で用いた疑似SRAMの制御回路の約10倍程度のを想定している。その場合は数十個程度の分割では各々の部分回路はそれほど細かくならず、プロセッサ台数が64台の場合30-40程度の速度向上が得られると予測される。

表3 回路分割数と単一実行フェーズの処理時間
Table 3 Execution time for the serial phase with respect to the number of circuits.

回路分割数	処理時間(秒)
1	19.5
2	27.5
4	37.3
8	55.7
16	89.8

表4 プロセス間通信量
Table 4 Amount of inter processor communication.

フェーズ	転送方向	転送語数
DC シミュレーション	親→子	3,268
	子→親	26,344
過渡解析	親→子	98,212
	子→親	590,248

プロセッサ台数を100台以上に増やす場合には前述したマルチレベル化等による単一実行部分の処理の軽減化等を考慮する必要がある。

5.4 プロセス間通信

プロセス間通信量を同じモデルを用いて調べたものを表4に示す。親プロセスと子プロセス間のデータ転送語数(1語は4バイト)を測定したものである。プログラム転送で実現しているので通信オーバーヘッドは転送語数に比例する。また共有メモリ方式を採っているので転送はメモリとメモリの間で行われるので負荷はそれほど重くない。プロセス間の通信量は両者の和となり同一モデルではプロセッサ数によらない。

処理時間が最も短い4台構成の場合を考えると、シミュレーション時間が1,380秒に対し転送語数は691,728であるので平均1.9m/秒に1語の割合で転送が起きていることになる。今後プロセッサ台数を増やした場合、例えば64台(16倍)のプロセッサで、4台のときの16倍の速度向上を得た場合でも平均7.4μ秒に1語の転送となり、転送によるオーバーヘッドはそれほど深刻な問題にはならない。

6. おわりに

モジュール分割アルゴリズムに基づく並列回路シミュレーションマシンのプロトタイプ構成および性能評価結果について述べた。4台構成のプロトタイプを実際に動作させ並列化の効果を確認した。本システムでは回路シミュレーション処理における入力フェーズおよびシミュレーションフェーズの双方について並列性を開拓した。特にシミュレーションフェーズにおいては高い並列性が得られ、数十台規模のシステムにおいても本方式が有効であることを確認した。

現在本研究の評価をもとに8台構成のクラスターを8本結合した階層バス構成による64台構成の並列マシンを構築中である¹¹⁾。今後はこのマシンを用いて以下のような点について評価していく予定である。

① 逐次部分の隘路の評価

5章でも述べたように、実際のシミュレーションの対象となる回路としてはこの評価で用いた SRAM の約 10 倍程度のものを想定している。その場合は本論文で述べた評価よりもはるかに並列部分の処理量が大きいので、逐次部分が隘路にならないと期待されるが、この点について実際のシステムで評価する必要がある。

② 単一レベルモデルとマルチレベルモデルのトレードオフの評価

マルチレベルモデルの方が処理のオーバーヘッドは大きいとその分根元のネットワークが小さくなり、逐次部分を小さくすることが可能となる。

③ 負荷の偏りの評価

本システムは粗い粒度による並列処理に基づいているため、静的な負荷分散しか行っていない。したがって負荷の分散をいかに均等に行うかがシステムの効率に大きな影響を与えることになる。

④ 回路分割方式

本プロトタイプでは、回路図に基づき負荷バランスを考慮して手で回路分割を行ったが、③との兼ね合いもあり回路分割方式の確立は急務である。この実現方式としては、大規模回路の設計で用いられる階層設計を活用して、階層設計に基づくブロック構造データを直接活用する方法や、フラットな回路の分割と類似する LSI レイアウトの配置問題で適用されるカーニハン・リン法¹²⁾などの 2 分割、ミニカット法が適用可能であると考えられる。

謝辞 本研究の機会を与えて頂き、また有益な示唆を頂いた当社超 LSICAD 技(本)柳川本部長、晴山部長、C&C システム研究所石黒所長、森野所長代理、大野部長に深謝します。

参 考 文 献

- 1) Nakata, T., Tanabe, N., Onozuka, H. and Koike, N.: A Multiprocessor System for Modular Circuit Simulation, *Proc. ICCAD 87*, pp. 364-367 (1987).
- 2) 小池ほか: 並列回路シミュレータ・プロトタイプのアーキテクチャ, 第 35 回情報処理学会全国大会論文集, pp. 197-198 (1987).
- 3) 中田ほか: 並列回路シミュレータ・プロトタイプの性能評価, 第 35 回情報処理学会全国大会論文集, pp. 199-200 (1987).
- 4) 小池ほか: 並列シミュレーションマシンのプロトタイプ, 信学会, CPSY 87-19, pp. 49-56 (1987).

- 5) Deutsch, J. T. and Newton, A. R.: A Multiprocessor Implementation of Relaxation-based Electrical Circuit Simulation, *Proc. 21st DA Conf.*, pp. 350-357 (1984).
- 6) Cox, P., Burch, R. and Epler, B.: Circuit Partitioning for Parallel Processing, *Proc. ICCAD 86*, pp. 186-189 (1986).
- 7) 吉田ほか: 並列計算機による回路シミュレーションの高速化について, 信学技法, Vol. 87, No. 55, VLD 87-15 (1987).
- 8) Deutsch, T. et al.: Parallel Computing for VLSI Circuit Simulation, *VLSI Systems Design*, Vol. 7, No. 7, pp. 46-52 (July 1986).
- 9) 鹿毛哲郎: VLSI 回路シミュレーション, 電学論 C, Vol. 107, No. 6, pp. 519-524 (1987).
- 10) 小池誠彦: シミュレーションエンジン, 電子情報通信学会誌, Vol. 70, No. 7, pp. 728-737 (1987).
- 11) 松下ほか: 並列シミュレーションマシン, 第 37 回情報処理学会全国大会論文集, pp. 97-98 (1988).
- 12) Kernighan, B. W. and Lin, S.: An Efficient Heuristic Procedure for Partitioning Graphs, *Bell Syst. Tech. J.*, Vol. 49, No. 2, pp. 291-307 (Feb. 1970).

(平成元年 4 月 25 日受付)

(平成元年 10 月 11 日採録)



中田登志之 (正会員)

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。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に勤務。並列計算機システムの研究に従事。電子情報通信学会会員。

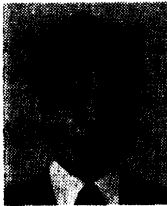


田辺 記生

昭和 26 年生。昭和 48 年東京理科大学電気工学科卒業。昭和 51 年東京都立大学電気工学科修士課程卒業。同年日本電気(株)入社。昭和 57 年-59 年西独アーヘン工科大学研究員。現在日本電気(株)超 LSICAD 技術本部にて LSICAD の開発に従事。シミュレーションアルゴリズム、デバイスモデリング、並列処理、LSI 設計法に興味を持つ。電気学会会員。

**小野塚裕美**

昭和 35 年生。昭和 60 年東京女子大学数理学科卒業。同年日本電気(株)入社。現在日本電気(株)超 LSI-CAD 技術本部にて LSICAD の開発に従事。

**黒部 恒夫 (正会員)**

昭和 21 年生。昭和 45 年早稲田大学理工学部卒業。同年日本電気(株)に入社。入社以来 LSI の CAD, 特に論理設計・テスト設計用 CAD の開発に従事。

**小池 誠彦 (正会員)**

昭和 22 年生。昭和 45 年東京大学工学部電気工学科卒業。昭和 47 年同大学院修士課程修了。同年日本電気(株)に入社。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、部長。この間、マルチプロセッサシステム、専用マシンシステム、高並列アーキテクチャ等の研究に従事。並列処理、ハードウェア技術、CAD 技術、AI システムに興味を持っている。情報処理学会昭和 59 年度論文賞受賞。電気通信学会会員。