計算機アーキテクチャ　68－7
(1987・11・12)

並列回路シミュレーションマシンのプロトタイプ

A Multiprocessor Prototype System for Modular Circuit Simulation

小池誠彦† 中田登志之† 田辺記生†† 小野塚裕美†† 黒部恒夫††
Nobuhiko KOIKE, Toshiyuki NAKATA, Norio TANABE, Hiromi ONOZUKA and Tsuneo KUROBE

†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
†C&C Systems Laboratories, NEC Corporation
†† VLSI CAD Engineering Division, NEC Corporation

1．はじめに

VLSIの急速な発達によりVLSIチップの設計及び検証が一層困難で時間のかかる作業となっている．設計品質の向上及び設計・製造上の誤りによるチップの再作成を極力抑えるために、超LSIの設計手段としてシミュレションによる検証が必要不可欠なものとなっている．シミュレーションは計算集中的な処理が必要となるので膨大な計算機資源を消費する．各設計フェーズで用いられるシミュレータとして、①論理シミュレータ、②回路シミュレータがあげられる．論理シミュレータについては専用のエンジンが開発され高速化／大容量化が進んでいるが，回路シミュレータについては高速化が十分に実現されているとは言いがたい状況にある．

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[7]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言う．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 100MIPS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コンピュータの利用が先ず考えられる[8]．しかし，回路シミュレーションでは非線型素子を含む大規模なスパースマトリッ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留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[9]．

近年の高性能なマイクロプロセッサおよび高性能浮動小数点プロセッサの発達は目ざましいものがあり，効率の良い並列アルゴリズムを実現し，多数のマイクロプロセッサ／浮動小数点プロセッサを用いることにより従来の汎用大型計算機の１桁以上の高速化も可能となった．そこで，筆者らは並列向きの数値演算アルゴリズム及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．

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[1],[2],[3]．４台構成で動かした時、１台の時に比べ３．９倍の高速化を実現し，提案しているアルゴリズム及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，並列アルゴリズム及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及び精度において互換性が有り，しかも高い並列性が開拓出来るものが望ましい．そのためには出来るだけ粗粒度な並列性を開拓可能なアルゴリズムを用いる必要がある．

表1　並列回路シミュレーションアルゴリズムの比較

アルゴリズム	並列性	処理の性質	並列性	マシン例
直接法	モデル計算 マトリックス計算	細粒度処理	小	SX/シﾊﾞ社[7] FNAP-VP[8]
緩和法	部分回路 微分方程式レベル （波形緩和法）	粗粒度処理	大	Deutschら[4], PNAP-1 UCB
モジュール 分割法	部分回路	粗粒度処理	大	COXら[5] 日電

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が可能となる．階層設計におけるモジュール構造を利用して部分回路分割を行い実マシンへマッピングすることが考えられる．

④高速科学技術計算処理機能の強化

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性能が必要とされる．従って，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

回路シミュレーションを高速化するために並列向きあるいはベクトル向きの数値計算アルゴリズム及び汎用／専用マシンを用いたシステムが提案されている．表1はそれらのマシンをアルゴリズムによって比較分類したものである．

回路解析を並列化する場合，大別して次の3つの方式が考えられる，①直接法，②緩和法，③モジュール分割法．

直接法では回路解析処理を次のいくつかの処理ステップにわけ，各々のステップ毎に並列性を開拓する，①入出力処理，②モデル計算，③行列計算，④収束判定，⑤時間ステップ更新．全体の処理時間の大半がモデル計算と，行列計算に費やされるので，この部分の並列化が効果が大きい．モデル計算は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，マトリックス計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．この様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分解及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うので大きなスピードゲインが得られない．

緩和法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．緩和法では各反復回毎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，イベント駆動及び部分回路毎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、部分回路毎の計算と、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もので、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。並列性の面から見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、接続回路網の計算は逐次的な処理となるが、非線形素子を含まないので計算量は少なく、全体の計算量から比べると高々数％に過ぎず数十台の並列処理ではそれ程のオーバ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時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及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通りである。

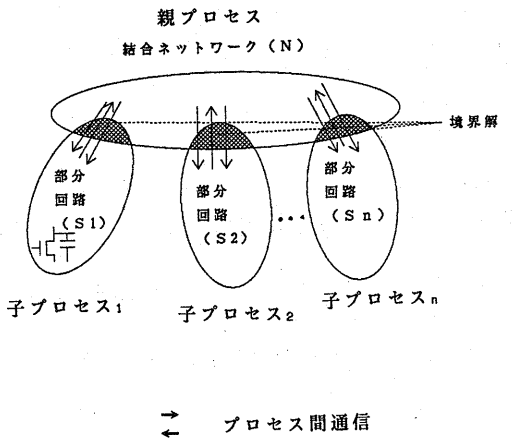


図１　処理プロセスと通信パターン

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返すフェーズがある。

③プロセス間の同期は親と子の間のローカルな同期と、全プロセスの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様なデータアクセスパターンに特化したアーキテクチャの設計を行った．

①プロセスの内部処理においては各々のプロッサは他のプロセッサとアクセス競合を起こさない様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。

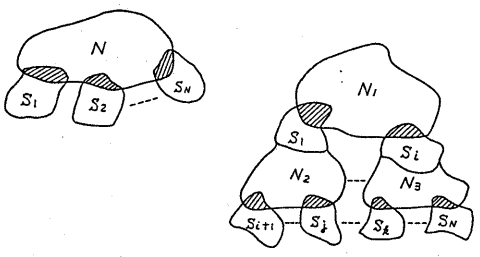
【ローカルメモリ方式】

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／リモートメモリの全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経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ッフィクの増大を抑制する．

【分散共有メモリ方式】

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セッサ及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４ＭＢを持たせる。



(1)単一レベル分割　　(2)マルチレベル分割

図２　モジュール分割法

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

本システムで用いたモジュール分割法は図２に示すとおり，シミュレーション対象となる回路を次の様にして解く，①部分回路（Ｓ１，Ｓ２，---,ＳＮ）に分割し，②部分回路間の接続ネットワーク計算部（Ｎ：親プロセス）と部分回路内計算部（子プロセス群）にわけ，③親プロセスと子プロセス群の計算を交互に解く．親プロセスと子プロセス群の階層は図２．１の様な単一レベル以外にも図２．２．のマルチレベルの分割も考えられる．本システムでは主に単一レベルについて検討を進めるが，後述するように親プロセスの負荷が重くなるときはマルチレベルの方式も検討に入れる必要があるのでプログラム上は双方の方式が実行可能としている

図３が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したものである．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のに対し、本方式では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，に分割して解く．分割を行っ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遙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ることになる．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．その時はマルチレベル化を行うなどしてネットワーク処理の圧縮を行う必要がある．

３．２．並列アルゴリズム

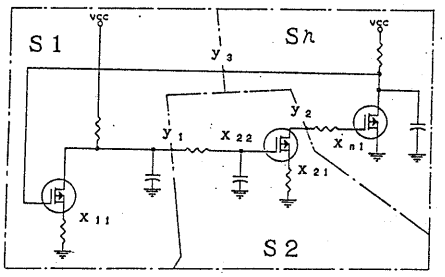
本システムで用いたモジュール分割並列回路シミュレーション処理の実行は次の２つのフェーズからなり，その両方の処理に於て出来るだけ並列処理を適用している．

<u>従来</u>

$S(x_{11}, x_{21}, x_{22}, x_{n1}, \dot{x}_{11}, \dot{x}_{21}, \dot{x}_{22}, \dot{x}_{n1}, y_1, y_2, y_3, \dot{y}_1, \dot{y}_2, \dot{y}_3, t) = 0$

→大規模，ランダム，
スパース，非線型

S



<u>本方式</u>　全体回路Ｓを部分回路に分割

• ステップ１　（部分回路の内部計算）

$S1(x_{11}, \dot{x}_{11}, y_1, y_3, \dot{y}_1, \dot{y}_3, t) = 0$
$S2(x_{21}, \dot{x}_{22}, y_1, y_2, \dot{y}_1, \dot{y}_2, t) = 0$　　• 非線型常微分の求解
$Sn(x_{n1}, \dot{x}_{n1}, y_2, y_3, \dot{y}_2, \dot{y}_3, t) = 0$

• ステップ２（部分回路間の共通ネットワークの計算）

$N(y_1, y_2, y_3) = 0$　　• 線型方程式求解

図３　並列回路シミュレーション方式

①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，コンパイル及びリンク処理を並列に行う．

②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．従って，入力フェーズを並列化しても其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時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．従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予め実際のプロセッサ群に静的に割付けておく．）

【並列実行部】

各プロセッサが並列に動作して各部分回路Ｓｉの内部変数の疑似解を部分回路の非線型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】

各々のプロセッサが並列に動作し各々の部分回路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４に示した．

図５は過渡解析における計算時間の分布を測定したものである（単一プロセッサ構成時）．モデル評価時間が全体の80%以上を占めており，マトリックス計算が９%，時間離散化処理が６%と少ない．この比率は対象回路の規模，使用するモデルの精度に大きく依存する．対象回路が大規模になると後２者の比率が増大して行くが，それでも依然として前者のモデル計算の占める比率の方が大きいと考える．従って，モデル計算の並列化が最も効果が大きいと言える．

以上，本アルゴリズムの特徴をまとめると次のとおりである．

①並列実行プロセス（子プロセス）が処理の大半を占める．

②子プロセス間の通信が不要で，子プロセスの実行は各プロセッサが独立して処理可能である．

③親プロセス（単一実行フェーズ）の実行はは計算量は少ないが，プロセッサ間の通信が必要となり，本質的に１台のプロセッサしか動かない．

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及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．

親プロセスの処理	子プロセスの処理
①$t_{n+1}=t_n+d_t$　とし後退積分の行列係数を求める，近似解を与え，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４　過渡解析における処理フロー

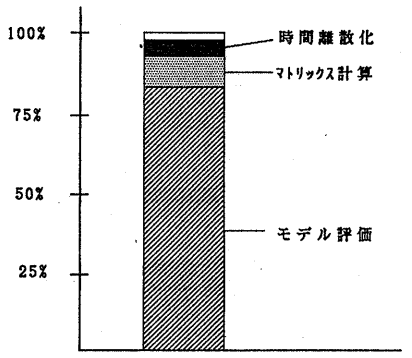


図５　過渡解析における計算時間分布

表２　入力処理における実行時間

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

4．プロトタイプの構成

アルゴリズム及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エアは基本アーキテクチャの設計方針に基づき小規模なシステムに適した構成としている。そのため，そのままの形で大規模システム化は困難であるが，プログラムの構成，プロセッサ間通信方式は大規模化を意識した作り方をしている．従って，大規模システムを構成した時にもプロトタイプでの評価を外挿して考えることができる．

プロトタイプ・システムは図６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出来るので、各々のプロ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・バス側から見ると全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出来る。

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．

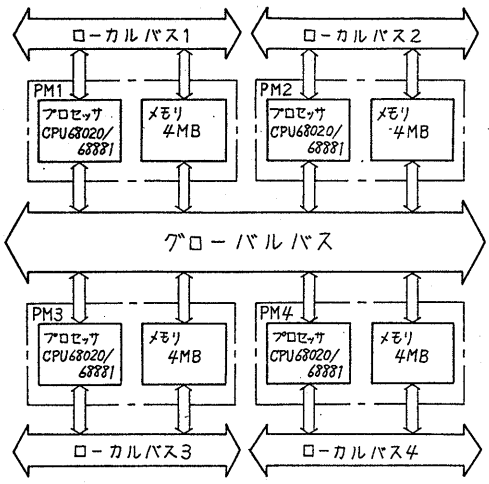


図６　プロトタイプの構成

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スタティック RAMの回路の回路解析を行った結果について評価する．

5．1．入力処理における並列台数効果

表２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と比較的良い値を得ている．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．従って，入力フェーズにおける並列処理の効果は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言う意味で意義が大きいと思われる．

表３　ｼﾐｭﾚｰｼｮﾝ処理における実行時間

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

５．２．ｼﾐｭﾚｰｼｮﾝ処理における並列台数効果

表３及び図７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３で注目すべきは単一実行フェーズの全体の処理時間に占める割合が少ない点である．４台構成時でも 3.9%と少ない．実用規模の対象回路は本実験で用いた約10倍の規模を想定しており，規模の増大につれてこの単一実行フェーズの比率が当然増して行くが，それでも数十台規模のシステムでは並列性の限界にはならず十分な高速化が達成出来ると言える．

５．３．処理粒度の影響

回路分割数を増加し，部分回路の処理の粒度を細かくした時に，結合ネットワークの処理（単一実行部）の影響を調べたものを表４に示した．同一の対象回路を分割数を変えた場合の単一実行部の処理時間を示している．分割数を増すと其れにつれて単一実行部の処理時間の増加が見られる．一方，並列実行フェーズは処理単位が細かくなるので処理時間が短くなると考えられる．全体の処理時間は並列実行フェーズでの実行時間と単一実行フェーズでの実行時間の和で与えられるので処理の粒度が細かくなると両者の時間配分を考慮した分割を行う必要が出て来る．なお本実験で使用した回路は小規模であるので，回路分割数を多くすると，粒度が細かくなりすぎてしまい単一実行フェーズの比率が増大してしまう傾向にある．しかし，数千素子レベルの実用規模の回路を扱う場合は数十個程度の分割では各々の部分回路は其れほど細かくならないので処理粒度の影響をあまり考えなくて良いと思われる．

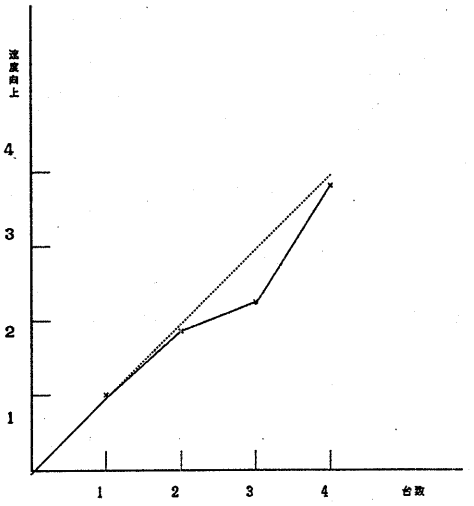


図７　シミュレーション時間とプロセッサ数

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５に示した．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．プログラム転送を行っているので通信オーバヘッドは転送語数に比例する，

表４　回路分割数と単一実行フェーズの処理時間

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

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．転送量は親から子への転送が少なく，それに比べ子から親への転送量は５から８倍程度多くなっている．これは後者の場合，境界解と付帯データを一緒に転送していることによる．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．

表５　プロセス間通信量

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

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．従って，処理の粒度が粗く転送によるオーバーヘッドは殆ど無視することができると言える．

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の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及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．

今後は大規模にした時の問題点，特にプロセッサ間接続方式，単位プロセッサの高性能化，回路分割／割当方式などを検討する必要がある．

最後に，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，森野部長，　C&Cシステム研究所石黒所長，山本部長に深謝します．

参考文献

[1]Nakata,T.,Tanabe,N.,Onozuka,H., and Koike,N.,"A Multiprocessor System for Modular Circuit Simulation", Proc. ICCAD 87

[2]小池ほか，”並列回路ｼﾐｭﾚｰﾀ･ﾌﾟﾛﾄﾀｲﾌﾟのｱｰｷﾃｸﾁｬ”、情処35回大会予稿集，pp197-198

[3] 中田ほか、”並列回路ｼﾐｭﾚｰﾀ･ﾌﾟﾛﾄﾀｲﾌﾟの性能評価”、情処35回大会予稿集，pp199-200

[4] Deutsch,J.T.,Newton,A.R.,"A Multiprocessor Implementation of Relaxation-based Electrical Circuit Simulation", Proc.21st DA Conf. pp.350-357, 1984

[5] Cox,P., Burch,R., Epler,B.,"Circuit Partitioning for Parallel Processing", Proc.ICCAD 86, pp.186-189

[6] 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, VLD87-15

[7] Deutsch,T.,et al,"Parallel Computing for VLSI Circuit Simulation, VLSI SYSTEMS DESIGN, July 1986, pp46-52

[8] 鹿毛，”VLSI回路ｼﾐｭﾚｰｼｮﾝ”，電学論Ｃ，107 巻６号，昭62，PP519-524

[9] 小池，”ｼﾐｭﾚｰｼｮﾝｴﾝｼﾞﾝ” 電子情報通信学会誌Vol.70,No.7, pp728-737