計算機アーキテクチャ 130－5
（1998. 8. 5）

シングルチップマルチプロセッサ上でのマルチグレイン並列処理

木村啓二[†] 尾形航[†]
岡本雅巳[††] 笠原博徳[†]

早稲田大学理工学部電気電子情報工学科[†]
（株）東芝[††]

〒169-8555 東京都新宿区大久保 3-4-1 TEL:03-5286-3371
E-mail: {kimura,ogata,okamoto,kasahara}@oscar.elec.waseda.ac.jp

あらまし　1チップ上に集積可能なトランジスタ数の増大に従い，次世代マイクロプロセッサでは，これらのトランジスタをいかに有効に利用し，プロセッサの実効性能を向上させるかが大きな課題になっている．しかし，現在主流のスーパースカラあるいはVLIW，それらの複合形のマイクロプロセッサでは，命令レベル並列性等の限界によりスケーラブルな実効性能の向上が困難と考えられている．

これに対して，筆者等は従来のチップ内命令レベル細粒度並列処理に加え，より並列性の大きいループイタレーションレベルの中粒度並列処理（ループ並列処理），サブルーチン，ループ，基本ブロック間の粗粒度並列性を階層的に組み合わせて使用するマルチグレイン並列処理をチップ内で実現できるシングルチップマルチプロセッサ（SCM）は真の実効性能を向上を可能にすると考えている．

本論文では，マルチグレイン並列処理を効果的に実現できるSCM検討の第一歩として，共有キャッシュ，グローバルレジスタ，分散共有メモリ，ローカルメモリの有効性に関する基本評価を行なった結果について述べる．

Multigrain Parallel Processing on the Single Chip Multiprocessor

KEIJI KIMURA[†], WATARU OGATA[†], MASAMI OKAMOTO[††] and HIRONORI KASAHARA[†]

Department of Electrical, Electronics and Computer Engineering, Waseda University[†]
Toshiba Corporation[††]

3-4-1 Ohkubo Shinjuku-ku, Tokyo 169-8555, Japan Tel: +81-3-5286-3371
E-mail: {kimura,ogata,okamoto,kasahara}@oscar.elec.waseda.ac.jp

Abstract　With the increase of the number of transistors integrated on a chip, how to use transistors efficiently and improve effective performance of a processor is getting an important problem. However, it has been thought that superscalar and VLIW which have been popular architectures would have difficulty to obtain scalable improvement of effective performance because of limitation of instruction level parallelism.

To cope with this problem, the authors have been proposing a single chip multiprocessor(SCM) approach to use multi grain parallelism inside a chip, which hierarchicaly exproits loop parallelism with large parallelism and coarse grain parallelism among subroutines, loops and basic blocks in addition to instruction level parallelism.

This paper describes preliminary evaluation of effectiveness of single chip multiprocessor architecture with a shared cache, global registers, distributed shared memory and/or local memory as the first step of research on SCM architecture for supporting effective realization of multi grain parallel processing.

1 はじめに

1チップ上で使用可能なトランジスタ数の増大に従い，現在様々な種類の次世代マイクロプロセッサアーキテクチャが提案されている．それらのアーキテクチャの代表的な例として，投機実行やスーパースカラ等の回路に大量のトランジスタを割き，命令レベルの並列処理性能を向上しようとするもの [1][2][3]，DRAM を大量に搭載し，ベクトル演算性能を向上しようとするもの [4]，そして DRAM と複数の CPU を混載するもの [5][6][3] が挙げられる．しかしながら，これらのアーキテクチャは命令レベルもしくはスレッドレベルといった単一の並列性しか利用しておらず，特に命令レベルの並列処理はその並列性の限界が指摘されており，今後のスケーラブルな性能向上は困難であろうと考えられている．

一方，我々は Fortran プログラムの並列処理を前提として，細粒度並列処理 [7] に加え，ループイタレーションレベルの中粒度並列処理 [8] 及び，サブルーチンあるいはループ，基本ブロック間の粗粒度並列性 [9][10] を階層的に組み合わせて使用することにより，高い実効性能を達成することを目指し，マルチグレイン並列処理 [11] を従来から提案している．このマルチグレイン並列処理をシングルチップマルチプロセッサ（SCM）に適応することにより，ピーク性能と実効性能の差の小さい真に高性能な計算機システムを実現することができると考えている [12]．

そこで，本論文では，このマルチグレイン並列処理を効率良く実現できる SCM アーキテクチャ研究の第一ステップとして，グローバルレジスタ，共有キャッシュ，ローカルメモリ，分散共有メモリの効果の基本評価を行なった結果について述べる．

以下，2節でマルチグレイン並列処理について，3節で本論文で評価する SCM について，4節でこれらのアーキテクチャにマルチグレイン並列処理を適応して評価した結果について述べる．

2 マルチグレイン並列処理

本節では，本論文でシングルチップマルチプロセッサに対して適用する，マルチグレイン並列処理手法について述べる．

マルチグレイン並列処理手法 [11] とは，ループやサブルーチン等の粗粒度タスク間の並列処理を利用するマクロデータフロー処理 [13][9]，ループレベルの並列処理である中粒度並列処理，基本ブロック内部のステートメントレベルの並列性を利用する近細粒度並列処理 [7] とを階層的に組み合わせて，並列処理を効果的に行なう手法である．

2.1 マクロデータフロー処理

マクロデータフロー処理では，ソースとなる Fortran プログラムを疑似代入文ブロック（BPA），繰り返しブロック（RB），サブルーチンブロック（SB）の三種類の粗粒度タスク（マクロタスク（MT））[9] に分割する．

ここで，BPA は基本的には通常の基本ブロックであるが，並列性抽出のために単一の基本ブロックを複数に分割したり，逆に一つの BPA の処理時間が短く，ダイナミックスケジューリング時のオーバーヘッドが無視できない場合には，複数の BPA を融合して一つの BPA を生成する．

RB は，最外側ナチュラルループである．ただし，Doall ループは，ループインデクス範囲を分割することにより複数の部分 Doall ループに分割し，分割後の部分 Doall ループを新たに RB と定義する．

また，サブルーチンは，可能な限りインライン展開するが，コード長が長くなり過ぎ効果的にインライン展開ができないサブルーチンは SB として定義する．

さらに，RB や SB の場合，これらの内部にマクロデータフロー処理を適用し，階層的マクロデータフロー処理を行なうこともある [14]．

MT 生成後，コンパイラは BPA，RB，SB 等の MT 間のコントロールフローとデータ依存を解析し，それらを表したマクロフローグラフ（MFG）[13][15] を生成する．さらに MFG から MT 間の並列性を最早実行可能条件解析 [13][15] により引きだし，その結果をマクロタスクグラフ（MTG）[13][15] として出力する．その後 MT は，条件分岐等の実行時不確定性が存在する場合にはダイナミックスケジューリングで，それ以外の場合にはスタティックスケジューリングにより各プロセッサクラスタ（PC）に割り当てられ実行される．

2.2 中粒度並列処理（ループ並列処理）

PC に割り当てられた MT が Doall 可能な RB である場合，この RB は PC 内のプロセッシングエレメント（PE）に対して，イタレーションレベルで分割され並列実行される．

2.3 近細粒度並列処理

PC に割り当てられた MT が，BPA や中粒度並列処理を適用できない RB である場合，それらはステー

トメントレベルのタスクに分割され，PC内のPEにより並列処理される．

ステートメントをPEに割り当てる際には，スケジューリング手法として，データ転送オーバーヘッドを考慮し実行時間を最小化するヒューリスティックアルゴリズムである，CP/DT/MISF法，CP/ETF/MISF法，ETF/CP法，あるいはDT/CP法[15]の4手法を適用し最良のスケジュールを選ぶ．

3 評価対象アーキテクチャ

本節では，今回評価を行なったシングルチップマルチプロセッサアーキテクチャについて述べる．

評価対象アーキテクチャとして，まずプロセッシングエレメント（PE）間の結合方式及びメモリアーキテクチャの違いにより，データキャッシュ共有型とOSCAR型（分散共有メモリ＋ローカルメモリ）の2例を用意した．さらに，これらに対してPE間グローバルレジスタを付加したものをそれぞれ用意した．

これらのアーキテクチャを，クロックレベルの精密なシミュレータを用いて評価を行なう．

3.1 共通仕様

本論文で評価するアーキテクチャは，32bit固定命令長，ロード/ストアアーキテクチャのシンプルなシングルイシューRISCアーキテクチャのCPUを1チップ上に4基搭載するものとした．このCPUは整数演算及び浮動小数点演算の両方に使用することができる汎用レジスタを64本持ち，また，FMUL，FADDを含む全命令の実行を1クロックで処理することができるものとする．

プロセッサの内部には各々のCPUで実行するプログラムが格納されるローカルプログラムメモリ（LPM）があり，LPMには1クロックでアクセスできるものとする．また，プロセッサの外部には各PEで共有するデータを格納する集中共有メモリ（CSM）が接続される．このCSMのレイテンシは20クロックとする．

3.2 データキャッシュ共有型アーキテクチャ

データキャッシュ共有型アーキテクチャとは，CPUとローカルプログラムメモリ（LPM）を持つプロセッシングエレメント（PE）が，一つのデータキャッシュを共有するアーキテクチャである．データキャッシュ共有型アーキテクチャの全体図を図1に示す．

ここでデータキャッシュは，4つのポートを持つノンブロッキングキャッシュを使用する．キャッシュメモリは4つのバンクを持ち，各々のバンクが各ポートとスイッチを介して接続される構成とし，同一バンクへのアクセスが生じた場合はいずれかのアクセスのみが優先されるが，それ以外の場合は各ポートが独立にキャッシュメモリにアクセスできる．キャッシュの連想方式は4-way set associativeとし，ライトアクセスの際はWrite Backもしくは Write Through方式のいずれかを用いるものとする．

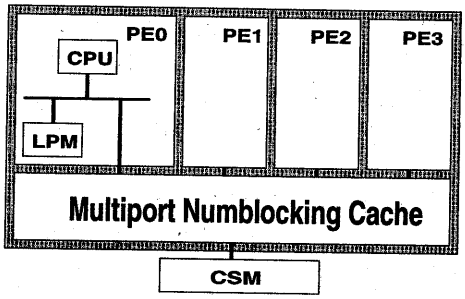

図 1: データキャッシュ共有型アーキテクチャ

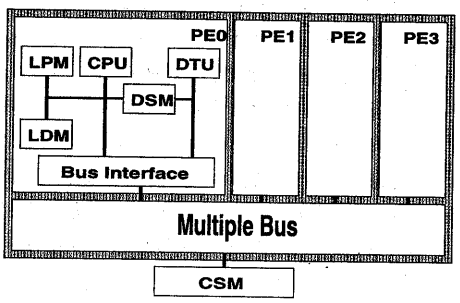

図 2: OSCAR型アーキテクチャ

このようにデータキャッシュを共有することにより，キャッシュのコヒーレンスを気にすることなく各PE間のデータ転送，とりわけ近細粒度タスク間のデータ転送を効率良く行なうことができる．

この共有データキャッシュは，ヒット時のアクセスタイムは最短3クロック，キャッシュメモリの容量は4Mbyteとした．

3.3 OSCAR型アーキテクチャ

OSCAR型アーキテクチャとは，マルチプロセッサシステムOSCAR[16]を基に構成されたアーキテクチャである．OSCAR型アーキテクチャの全体図を図2に示す．

OSCAR型アーキテクチャは，CPU，LPM，データ転送ユニット（DTU），ローカルデータメモリ（

LDM），そしてデュアルポートメモリで構成された分散共有メモリ（DSM）を持つプロセッシングエレメント（PE）を複数バスを介して接続したアーキテクチャである．複数バスは3本のバスで構成されており，PE間のデータ転送を効率良く行なうことができる．

LDMは自PEからのみアクセスできるメモリであり，PE固有のデータを保持するために使用する．また，DSMは他PEからもアクセスできるメモリであり，近細粒度タスク間のデータ転送や，マクロデータフロー処理におけるダイナミックスケジューリング時のスケジューリング情報の通知に使用する．LDMのメモリアクセスにかかるクロック数は3，容量は1PEあたり1Mbyteとし，DSMのメモリアクセスにかかるクロック数は自PEからはLDMと同様に3クロック，他PEからのアクセスには6クロックかかり，容量は1PEあたり16Kbyteとした．このように，メモリの用途をコンパイラがきめ細かく制御することにより，効率の良い並列処理を行なうことができる．

3.4 グローバルレジスタ

本論文では，3.2節及び3.3節で述べたデータキャッシュ共有型アーキテクチャ，OSCAR型アーキテクチャの各々に，グローバルレジスタ（GR）を付加したアーキテクチャついても評価を行なう．

GRには，各PE内のCPUが同時にアクセスすることができるものとする．このときのアクセスタイムは1クロックとする．

GRの本数は16本とし，これらは近細粒度タスクのデータ転送に使用する．

OSCAR型アーキテクチャにGRを付加した時の全体図を図3に示す．

4 評価プログラム

本節では，データキャッシュ共有型アーキテクチャとOSCAR型アーキテクチャに対して近細粒度並列処理もしくはマルチグレイン並列化処理を適用して評価した結果について述べる．

4.1 近細粒度並列処理

近細粒度並列処理時の性能評価に際して使用したプログラムは，ランダムスパースマトリクスを係数に持つ線形方程式の求解プログラム（s.f）である．このプログラムは算術代入文のみからなるFortranループフリーコードであり，94個の近細粒度タスク（ステートメント）を持つ．また，このプログラムは配列変数を一切使用しないので，OSCAR型アーキテクチャではコンパイラが全てのデータを各PEのLMに初期配置することができ，以後DSMを使用したデータ転送を行なうことによりCSMにアクセスすることなく処理を行なうことができる．

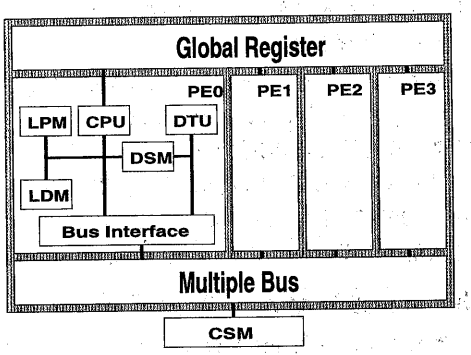

図3: OSCAR型アーキテクチャ＋グローバルレジスタ

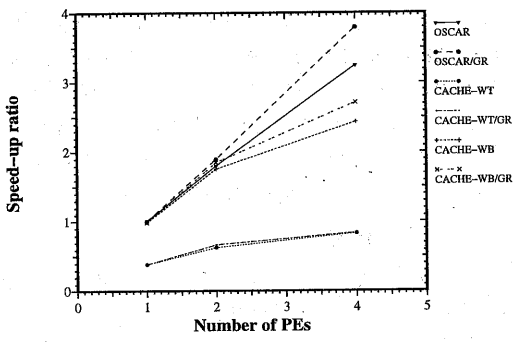

図4: ランダムスパースマトリクスの求解における速度向上率

このプログラムに近細粒度並列処理を施し，OSCAR型アーキテクチャ（OSCAR），データキャッシュ共有型のWrite Through（CACHE-WT）とWrite Back（CACHE-WB），および，これらに対して共有グローバルレジスタを付加したアーキテクチャ（GR）のそれぞれで，PE数1，2，4で実行した．この結果を，OSCARでの1プロセッサ上での実行時間を基準とした時の速度向上率を表すグラフとして図4に示す．

図4より，まずCACHE-WTの性能が著しく低いことがわかる．これは，今回のアーキテクチャには

L2-Cache が無く，Write Back ではプロセッサ外部へのメモリアクセスが頻繁に生じてしまうためである．

逆に，プロセッサ外部へのメモリアクセスがほとんど生じない OSCAR と CACHE-WB は，ほぼ同じ傾向を示している．しかしながら，PE 数 2 では CACHE-WB の方が優勢であったものが，PE 数 4 になると OSCAR の方が優位に立つ．これは，PE 数が増加するとキャッシュメモリの同一バンクへのアクセスが生じやすくなるためであると考えられる．

次にグローバルレジスタの有無について述べる．OSCAR 型では，データ転送を行なうために送信先の PE の DSM に 6 クロックかけてデータを書き込まなければならないところを，グローバルレジスタを使用することによって 1 クロックで送信先の PE にデータを転送することができる．このため，グローバルレジスタの付加によって効率的に近細粒度タスクのデータ転送を行なうことができ，OSCAR 型では 4PE 時に 3.2 倍の速度向上率であったものが，グローバルレジスタの付加により 3.8 倍となり，17%速度が向上している．同様にして CACHE-WB 型でも 4PE 時の速度向上率が 2.4 倍であったものがグローバルレジスタの付加により 2.7 倍となり，11%速度が向上している．

4.2 マルチグレイン並列処理

マルチグレイン並列処理時の性能評価に際して使用したプログラムは，航空宇宙技術研究所の CFD プログラム「NS3D」のサブルーチン「SUB4」を取りだし，更に配列サイズを 1/1000 に縮小したものである．このプログラムは，中粒度並列処理が不可能な 3 つの Do-loop と一つの Doall loop を内包する Do-loop によって構成されており，中粒度並列処理のみではほとんど台数効果が得られない．なお，このプログラムでは配列を多用しているので，OSCAR 型ではスカラ変数全てと配列変数の一部をコンパイルが各 PE の LM に初期配置し，それらのデータは以後 DSM を使用したデータ転送を行なうことにより各 PE に転送されれる．

このプログラムにマルチグレイン並列処理を施し，OSCAR 型アーキテクチャ（OSCAR），データキャッシュ共有型のライトスルー（CACHE-WT）とライトバック（CACHE-WB），および，これらに対してグローバルレジスタを付加したアーキテクチャ（GR）のそれぞれで，PE 数 1，2，4 で実行した．この結果を，OSCAR 型アーキテクチャでの 1 プロセッサ上での実行時間を基準とした時の速度向上率を表すグラフとして図 5に示す．

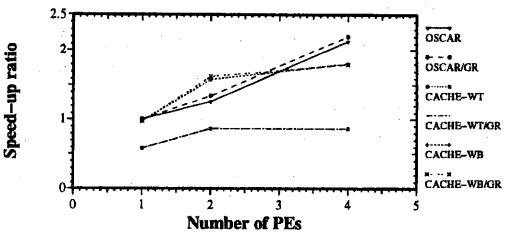

図 5: NS3D における速度向上率

図 5より，4.1節と同様に CACHE-WT の性能が著しく低いことがわかる．一方，OSCAR と CACHE-WB は PE 数 2 と 4 で完全に立場が逆転し，CACHE-WB では PE 数が 4 になってもほとんど性能が向上しない．

まず，OSCAR で PE 数 2 の時にほとんど台数効果が出ない理由であるが，LM の容量制限により LM に配置できる配列変数のデータ量が少ないためである．そのため，マルチグレイン並列化による効果をメモリアクセスコストの増加が打ち消してしまい，台数効果が得られない．PE 数が増えると，各 PE の LM に配列変数が分散されるために，LM に配置できる総データ量が増え，結果として台数効果を得ることができる．

CACHE-WB で PE 数が 4 になってもほとんど性能が向上しない原因であるが，一つには 4.1節で述べた，PE 数の増加によるキャッシュメモリの同一バンクへのアクセスの増加が挙げられる．それに加えて，NS3D の SUB4 ではキャッシュが苦手とするストライドアクセスを多用しており，ミスヒットペナルティが増加してしまうためであると考えられる．

グローバルレジスタの付加に関しては，図 4程には影響が見られず，OSCAR 型 4PE 時で 2.1 倍であった速度向上率がグローバルレジスタの付加により 2.2 倍となり，5%の速度向上に留まっている．これは，NS3D の SUB4 には大量の近細粒度タスクのデータ転送があり，グローバルレジスタに収まり切らなかったデータ転送が大量にあるためである．今後，割り当てアルゴリズム等を見直すことによって，グローバルレジスタの効果を高めることができると考える．

5 まとめ

本論文では，マルチグレイン並列処理に適したシングルチップマルチプロセッサアーキテクチャの基本検討を行なうことを目的として，データキャッシュ共有型アーキテクチャ，OSCAR型アーキテクチャ，およびこれらのアーキテクチャにグローバルレジスタを付加したアーキテクチャを用意した．

そして，これらのアーキテクチャのシミュレータ上でマルチグレイン並列処理を適用し，シングルチップマルチプロセッサアーキテクチャの性能評価を行なった．

その結果，メモリの使用方法をコンパイラがきめ細かく制御できるOSCAR型アーキテクチャがマルチグレイン並列処理に対して有効であることが確認された．また，グローバルレジスタの付加により，近細粒度タスクのデータ転送を効果的に行なうことができ，最大17%の速度向上を得ることができた．

今後は，スーパースカラ，VLIWといった命令レベル並列処理プロセッサとマルチグレイン並列処理を適用したシングルチップマルチプロセッサとの性能比較や，SPARC，PA-RISC等の商用RISCプロセッサをCPUコアにしたシングルチップマルチプロセッサでの性能評価を行なう予定である．

本研究の一部は，通産相次世代情報処理基盤技術開発事業並列処理分散分野マルチプロセッサコンピューティング領域研究の一環として行なわれた．

最後に，CFDプログラムNS3Dを御提供いただいた航空宇宙技術研究所の皆様に感謝致します．

参考文献

[1] Patt et al. One billion transistors, one uniprocessor, one chip. *Computer*, Vol. 30, No. 9, pp. 51–57, 1997.

[2] Lipasti and Sben. Superspeculative microarchitecture for beyond ad 2000. *Computer*, Vol. 30, No. 9, pp. 59–66, 1997.

[3] J.-Y. Tsai, Z. Jiang, E. Ness, and P.-C. Yew. Performance study of a concurrent multithreaded processor. In *Proc.4th Int'l Conf. on HPCA-4*, Fec 1998.

[4] Kozyrakis et al. Scalabel processors in the billion-transisteor era: Iram. *Computer*, Vol. 30, No. 9, pp. 75–78, 1997.

[5] Hammond, Nayfeb, and Olukotun. A single-chip multiprocessor. *Computer*, Vol. 30, No. 9, pp. 79–85, 1997.

[6] 岩下, 宮嶋, 村上. リファレンスppram「$ppram^R$」に基づく「$ppram^R_{mf}$」アーキテクチャの概要. 情処研報, Vol. 96, No. 80, pp. 161–166, 1996.

[7] 笠原. マルチプロセッサシステム上での近細粒度並列処理. 情報処理, Vol. 37, No. 7, pp. 651–661, Jul 1996.

[8] Padua and Wolfe. Advanced compiler optimization for super computers. *C.ACM*, Vol. 29, No. 12, pp. 1184–1201, 1996.

[9] 笠原, 合田, 吉田, 岡本, 本多. Fortranマクロデータフロー処理のマクロタスク生成手法. 信学論, Vol. J75-D-I, No. 8, pp. 511–525, 1992.

[10] 本田, 合田, 岡本, 笠原. Fortranプログラム粗粒度タスクのoscarにおける並列実 行方式. 信学論（D-I）, Vol. J75-D-I, No. 8, pp. 526–535, 1992.

[11] Kasahara, Honda, and Narita. A multigrain parallelizing compilation scheme for oscar. In *Proc.4th Workshop on Lang. And Compilers for Parallel Computing*, Aug 1991.

[12] 笠原, 尾形等. マルチグレイン並列化コンパイラとそのアーキ テクチャ支援. 信学技報, IDC98-10, CPSY98-10, FTS98-10, pp. 71–76, 1998.

[13] 本田, 岩田, 笠原. Fortranプログラム粗粒度タスク間の並列性検出法. 信学論(D-I), Vol. J73-D-I, No. 12, pp. 951–960, 1990.

[14] 岡本, 合田, 宮沢, 本田, 笠原. OSCARマルチグレインコンパイラにおける階層型マクロデータフロー処理. 情報処理学会論文誌, Vol. 35, No. 4, pp. 513–521, 1994.

[15] 笠原. 並列処理技術. コロナ社, 1991.

[16] 笠原, 成田, 橋本. OSCAR（Optimally SCheduled Advanced multiprocessoR）のアーキテクチャ. 信学論D, Vol. J71-D, No. 8, 1988.