近細粒度並列処理用シングルチップマルチプロセッサにおけるプロセッサコアの評価

木村啓二[†] 加藤孝幸[††] 笠原博徳[†]

1チップ上に搭載可能なトランジスタ数の増加にともない，これらの資源を活用してスケーラブルな性能向上を果たせる次世代マイクロプロセッサの開発が大きな課題となっている．このようなプロセッサの開発においては，命令レベル以外の並列性を抽出しチップ内で利用するアーキテクチャおよびソフトウェア技術が重要となる．このように，命令レベル並列性以外の並列性も有効に活用できるアーキテクチャとして，シングルチップマルチプロセッサ（SCM）が大きな注目を集めている．本論文では，複数粒度を階層的に組み合わせて利用するマルチグレイン並列処理に適したSCMのプロセッサコアの開発を目指し，マルチグレイン並列処理の主要技術の1つである近細粒度並列処理を，プロセッサコアの同時命令発行数を変えたSCMアーキテクチャに適用して評価を行った．その結果，1命令発行のプロセッサコアを4基搭載したSCMは，4命令発行in-orderプロセッサに対して最大2.48倍の性能を得ることができた．さらに，1命令発行のプロセッサコアを4基搭載したSCMは6命令発行out-of-orderプロセッサとほぼ同程度あるいは若干良い性能であったが，SCMはプロセッサコアを6に増やすことで，6命令発行out-of-orderプロセッサに対し1.39倍の性能を得ることができ，その一方でout-of-orderプロセッサでは演算器構成等の強化を行っても性能向上は得られなかった．以上から，簡素なプロセッサコアを複数搭載するSCMでは近細粒度並列処理に対しスケーラブルな性能向上を可能とすることが確認できた．

Evaluation of Processor Core Architecture for Single Chip Multiprocessor with Near Fine Grain Parallel Processing

Keiji Kimura,[†] Takayuki Kato[††] and Hironori Kasahara[†]

With the increase of transistors integrated onto a chip, development of next-generation microprocessor architecture that can achieve scalable performance improvement using these transistors effectively has been expected. In such next-generation microprocessor architecture, exploitation of multiple grains of parallelism in addition to instruction level parallelism (ILP) inside a single chip is important. A single chip multiprocessor (SCM) architecture that can use multigrain parallelism is one of the most promising approaches. To decide suitable SCM processor core architecture for multigrain parallel processing, this paper evaluates several SCM core architectures which have different instruction issue width for near fine grain parallel processing, which is one of the key issues in multigrain parallel processing. The evaluation shows the SCM architecture having four single-issue cores gives us 2.48 times better performance than a four-issue in-order superscalar processor. Furthermore, the SCM having single-issue cores and six-issue out-of-order processors are evaluated. As a result, the SCM having four cores gives us a little better performance than the out-of-order processor. However, the SCM having six cores gives us about 1.4 times better performance than the out-of-order processor, while any extensions of out-of-order can not achieve performance improvement.

1. はじめに

半導体技術の向上にともない，現在広く用いられているスーパスカラプロセッサは，スーパスカラ度の向上，投機実行，分岐予測等，高機能化の方向に進んでいる．しかしながら，このような高機能化をおし進めるアプローチでは，プロセッサの開発にかかる費用や期間の増大，微細加工技術の進歩によるトランジスタスイッチング速度と配線遅延による信号の伝達速度の差の増大等により，投入したトランジスタ資源に見合う性能向上が困難であることが指摘されている[1)~3)]．

このような状況から，チップ内に投入された大量の

† 早稲田大学
Waseda University
†† 本田技研
HONDA MOTER CO., Ltd

資源を有効に活用し，スケーラブルな性能向上を達成するためのアーキテクチャおよびコンパイル技術の研究が今後いっそう重要となる．とりわけ，プログラム中から，従来の命令レベルにとどまらずより多くの並列性を抽出し，チップ内で利用する技術が強く求められている．たとえば，複数のプロセッサコアを1チップ上に集積するシングルチップマルチプロセッサ（SCM）アーキテクチャが現在活発に研究されており，MAJC のようにマルチメディア用途に2基の VLIW コアを集積したもの[4]，Power4 のように高性能サーバ用途に2基のスーパスカラでオンチップ L2 キャッシュを共有する構成のもの[5]，Piranha のようにトランザクション処理を目的とし8つの簡素なプロセッサコアをチップ内に集積したもの[6] がそれぞれ Sun，IBM および Compaq から発表されている．また，Hydra，Multiscalar，Superthreaded，MUSCAT（Merlot），SKY のような，スーパスカラコアを複数搭載し，スレッド投機実行をサポートするアーキテクチャで命令レベル並列性とスレッドレベル並列性を利用するもの[7)~12)]，RAW や FlexRAM および PPRAM のように簡素なプロセッサとローカルメモリを持つプロセッシングエレメント（PE）をチップ内に多数配置し，その挙動をコンパイラで制御するもの[13)~15)] 等といった多くの研究が行われている．

これらのシングルチップマルチプロセッサに関する研究の中で，Piranha[6]，Hydra[16]，MUSCAT[11]，SKY[12] は，スーパスカラプロセッサとシングルチップマルチプロセッサの性能比較を行っている．特に Piranha では，1 GHz の4命令発行 Out-of-order プロセッサと，500 MHz の1命令発行のプロセッサコアを8基搭載したシングルチップマルチプロセッサをトランザクション処理を行って比較し，シングルチップマルチプロセッサがスーパスカラプロセッサに対し2.9倍の性能を得ている．また，Hydra は SPEC92/95 ベンチマークに手動並列化もしくは SUIF コンパイラによるループイタレーションレベルの並列化を施し，これらを2命令発行プロセッサコアを4基搭載したシングルチップマルチプロセッサで実行して，6命令発行スーパスカラプロセッサと比較している．その結果 Hydra では，ループイタレーションレベルの並列性が大きいアプリケーションにおいて，シングルチップマルチプロセッサがスーパスカラプロセッサに対し，1.5から2倍程度の性能を得ている．

一方，OSCAR マルチグレイン SCM[17] では，複数命令レベルでの並列処理である（近）細粒度並列処理[18] に加え，より大きな並列性を持つループイタレーションレベルの中粒度並列処理[19] およ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[20),21)] を階層的に組み合わせて使用し，プログラム全体から並列性を引き出して利用することができる．このマルチグレイン並列処理[22),23)] により，複雑なハードウェアを必要とすることなく自然にプログラムの持つ並列性を利用することができ，価格性能比の優れたスケーラブルなプロセッサを構築できると考えている．

本論文では，マルチグレイン並列処理をサポートする OSCAR マルチグレイン SCM アーキテクチャプロセッサコアの開発を目指し，マルチグレイン並列処理の主要構成要素の1つである近細粒度並列処理を，検討対象である命令発行数の異なるプロセッサコアに適用して評価を行った．さらに，簡素なプロセッサを複数搭載した SCM アーキテクチャと高機能 out-of-order スーパスカラプロセッサとの性能比較も行った．

以下，2章ではマルチグレイン並列処理，3章では本論文で評価する SCM アーキテクチャ，4章でこれらのアーキテクチャに近細粒度並列処理を適用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

本章では，OSCAR マルチグレインシングルチップマルチプロセッサで扱う，マルチグレイン並列処理技術について述べる．

マルチグレイン並列処理[22),23)] 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粗粒度タスク並列処理（マクロデータフロー処理）[20),24)]，ループイタレーション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[18] を階層的に組み合わせて，プログラム全域にわたる並列処理を行う手法である．

2.1 粗粒度タスク並列処理（マクロデータフロー処理）

マクロデータフロー処理では，ソースとなるプログラムを疑似代入文ブロック（BPA），繰返しブロック（RB），サブルーチンブロック（SB）の3種類の粗粒度タスク〔マクロタスク（MT）〕[20] に分割する．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）[24),25)] 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[24),25)] により引き出し，その結果をマクロタスクグラフ（MTG）[24),25)] として表現する．その後，コンパイラは MTG 上の MT を，スタティックスケジュー

リングの場合にはプロセッサ間データ転送および同期オーバヘッドを最小化できるようにプロセッサあるいはプロセッサグループ（PG）に割り当て，各プロセッサ用コードを生成する．MTG 中に条件分岐がありダイナミックスケジューリングを用いる場合には，実行時に MT をプロセッサあるいは PG に割り当てる．

2.2 中粒度並列処理（ループ並列処理）

PG に割り当てられた MT が Doall 可能な RB である場合，この RB は PG 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割り当てられ並列実行される．またこの場合には，各 PE 上のローカルメモリ，あるいは分散共有メモリを有効利用するためのデータローカライゼーション技術[26] を利用可能である．

2.3 近細粒度並列処理

PG 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，PG 内の PE により並列処理される．

近細粒度並列処理では，BPA 内のステートメント，もしくは IF–THEN–ELSE 等で囲まれた複数のステートメントから構成される疑似代入文を 1 つの近細粒度タスクとして定義する．コンパイラは，BPA を近細粒度タスクに分割した後，タスク間のデータ依存を解析してタスクグラフを作成する．次に，このタスクグラフ上のタスクを，データ転送・同期オーバヘッドを考慮して実行時間を最小化できるように各 PE にスタティックにスケジューリングする．

OSCAR Fortran コンパイラ[27] における近細粒度タスクの PE へのスケジューリングにおいては，スケジューリング手法として，データ転送オーバ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，および DT/CP 法[25] の 4 手法を同時に適用し最良のスケジュールを選んでいる．また，このようにタスクをスタティックにプロセッサに割り当てることにより，BPA 内で用いられるデータのローカルメモリ，分散共有メモリ，レジスタへの配置等，データ配置の最適化やデータ転送・同期オーバヘッドの最小化[18] といった各種最適化が可能になる．

スケジューリング後，コンパイラは PE に割り当てられたタスクに対応する命令列を順番に並べ，データ転送命令や同期命令を必要な箇所に挿入し，各 PE ごとに異なるマシンコードを生成する．近細粒度タスク間の同期にはバージョンナンバー法を用い，同期フラグの受信は受信側 PE のビジーウェイトによって行われる[27]．このとき，OSCAR マルチグレイン SCM のように分散共有メモリ（DSM）を持つアーキテクチャでは，3.1 節で述べるようにデータ転送および同期フラグのセットは受信側 DSM への直接ストアの形で行われ，受信側 PE のビジーウェイトも自 PE 上の DSM へのビジーウェイトになるので，プロセッサ間相互接続網のバンド幅低下は起らない．また，3.3 節で述べる共有グローバルレジスタを持つ SCM アーキテクチャでは，可能な限りグローバルレジスタを用いてデータ転送を行う．

3. 評価対象 SCM アーキテクチャ

本章では，今回評価を行ったシングルチップマルチプロセッサ（SCM）アーキテクチャ，およびそのプロセッサコアについて述べる．評価のため，以下に述べるアーキテクチャを厳密に評価できるクロックレベルシミュレータを開発した．

3.1 ネットワークおよびメモリアーキテクチャ

本論文で評価する SCM のネットワークおよびメモリアーキテクチャは，近細粒度並列処理において従来の評価で共有キャッシュ型より良い性能を示している OSCAR マルチグレイン SCM アーキテクチャ[17),27] とする．OSCAR マルチグレイン SCM アーキテクチャとは，**図 1** のように，CPU，データ転送ユニット（DTU），ローカルプログラムメモリ（LPM），ローカルデータメモリ（LDM），および分散共有メモリ（DSM）を持つプロセッシングエレメント（PE）を相互接続網（本論文では 3 本バスを使用）で接続し 1 チップ上に搭載したアーキテクチャである．本論文の評価では，PE は 1 チップに 1〜4 基搭載されるものとする．

まず，メモリアーキテクチャについて説明する．LPM

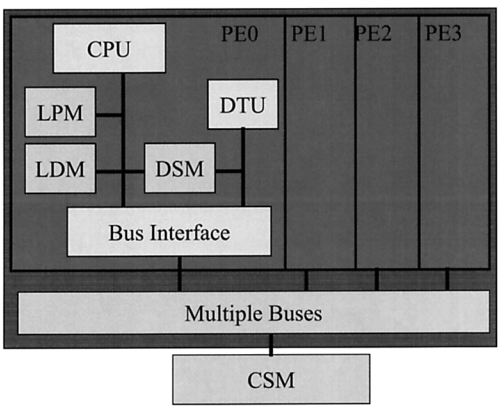

図 1 OSCAR マルチグレイン SCM アーキテクチャ
Fig. 1 OSCAR multigrain SCM architecture.

は各々の CPU で実行するプログラムを格納し，今回の評価では 1 クロックでアクセスできるものとする．同様に，LDM は PE 固有のデータを保持するために使用し，その容量は全 PE の LDM の合計が何プロセッサの場合でも 1 M バイトとなるように設定する．すなわち，4 PE 構成のときは 1 PE あたり 256 K バイト，2 PE 構成のときは 1 PE あたり 512 K バイト，1 PE の場合は 1 M バイトとする．また，LDM のアクセスレイテンシは 1 クロックとする．DSM は自 PE と他 PE の双方から同時にアクセス可能なマルチポートメモリであり，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等に使用する．DSM の容量は 1 PE あたり 16 K バイトとし，自 PE からのアクセスレイテンシは 1 クロック，他 PE へのリモートアクセス時のレイテンシは 4 クロックとする．この DSM を使用すると，近細粒度タスク間のデータ転送や同期フラグのセットを各々4 クロック，同期フラグのチェックを 5 クロック（ロード +比較+ブランチ），データの受信を 1 クロックで行うことができる．さらに，チップ外部には集中共有メモリ（CSM）が接続され，各 PE で共有されるデータを格納する．この CSM のアクセスレイテンシは，今回の評価では 20 クロックとする．OSCAR マルチグレイン SCM アーキテクチャでは，これら 4 種類のメモリに対しコンパイラが適切なデータ配置を行うことにより，効率の良い並列処理を行うことができる．

次に，PE 間バスについて述べる．PE からバスアクセスがあると，PE 間バスは，(1) バスに対するアクセス要求と調停，(2) 使用バスの選択と獲得という順番で処理を行う．この様子を，**図 2** の 2 つの PE が同時にバスアクセスを行う例を用いて説明する．1 クロック目で 2 台の PE が 3 本のバスに対しバスアクセス要求を同時に出し，調停が行われる．バスアクセスの優先順位に関しては，各バスごとに自由に設定できるが，本論文の評価では，PE 番号の大きいものが高いバスアクセスの優先順位を持っているものとしているので，PE2 がアクセス権を獲得する．2 クロック目で，PE2 が 1 つ目のバス（BUS1）を獲得し，残りの 2 本のバスを解放する．この間，PE1 はアクセス権待ちとなる．3 クロック目で，PE2 が BUS1 上でメモリアクセスを行う一方で，PE1 が BUS2 を獲得する．4 クロック目で，PE1 は BUS2 上でメモリアクセスを行う．以上のような方式で PE 間バスは処理を行う．

3.2 プロセッサコア

各 PE が持つ CPU は，SPARC V9 規格に準拠したスーパスカラプロセッサである Sun Microsystems 社の UltraSPARC II [28),29)] をモデルにしたプロセッサである．UltraSPARC II の浮動小数点命令のレイテンシを，**表 1** に示す．本論文で評価した SCM アーキテクチャでは，この UltraSPARC II プロセッサに拡張命令として，バリア同期機構等のための特殊レジスタや共有グローバルレジスタを操作するための命令を付加した．

このプロセッサは，9 段のパイプラインを持つ 4 命令 in-order 発行のスーパスカラプロセッサである．今回の評価では**表 2** のように，命令発行数および整数演算器（Integer Execution Unit: IEU），ロード・ストアユニット（Load Store Unit: LSU），浮動小数点演算器（Floating Point Unit: FPU）の各演算器数を変更することで，1 命令発行，2 命令発行，4 命令発行のスーパスカラプロセッサとして使用する．4 命令発行時の演算器構成は，UltraSPARC II と同じ構成である．

動的スケジューリングの対象となる命令群が格納される命令バッファのエントリ数は 12 である．ただし，

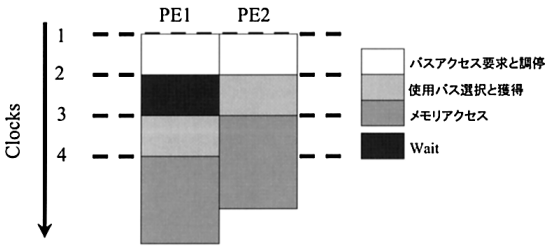

図 2 PE 間バスの動作例
Fig. 2 A example of BUS arbitration.

表 1 UltraSPARC II の浮動小数点命令レイテンシ
Table 1 Floating point instruction latencies of UltraSPARC II.

命令	レイテンシ
加算	3
乗算	3
除算（単精度）	12
除算（倍精度）	22

表 2 各プロセッサコアの演算器構成
Table 2 The function-unit compositions of each processor core.

CPU コアの命令発行数	IEU	LSU	FPU
1	1	1	1
2	1	1	1
4	2	1	2

IEU：整数演算器，LSU：ロード・ストアユニット，FPU：浮動小数点演算器

1 命令発行時は命令バッファのエントリ数は 1 とする．

また，UltraSPARC II は 512 エントリの 2 ビットカウンタ方式による分岐予測機構を備えるが，本論文の評価では，プログラム内の近細粒度並列処理の評価を目的としているために分岐命令がほとんどなく，分岐予測機構の性能は評価に影響を与えない．

3.3 共有グローバルレジスタ

本論文では，共有グローバルレジスタ（GR）を付加したアーキテクチャについても評価を行う．

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し，本論文では GR のアクセスレイテンシは 1 クロック，本数は 32 として評価を行う．これらのレジスタは近細粒度タスクのデータ転送に使用する．共有グローバルレジスタの使用により，近細粒度タスク間のデータ転送や同期フラグのセットを各々1 クロック，同期フラグのチェックを 5 クロック，データの受信を 1 クロックで行うことができる．すなわち，DSM を使用する場合に対し，データ転送・同期フラグセットのコストを 3 クロックずつ削減できる．

OSCAR 型アーキテクチャに GR を付加したときの全体図を**図 3** に示す．

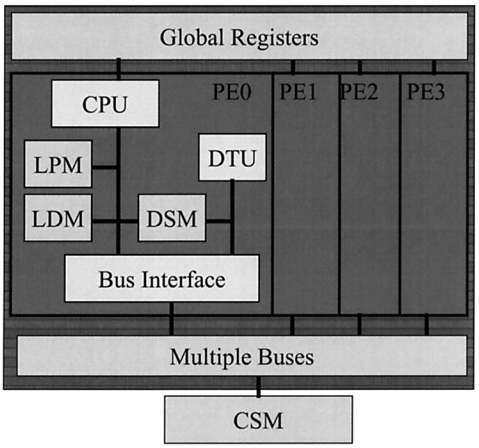

図 3 OSCAR マルチグレイン SCM アーキテクチャ ＋ 共有グローバルレジスタ
Fig. 3 OSCAR multigrain SCM architecture with shared global registers.

4. 性能評価

本章では，1 命令発行あるいは複数命令発行のプロセッサコアを持つ PE を複数搭載した SCM アーキテクチャに対し，近細粒度並列処理を大規模な基本ブロックを持つ以下の 3 つのアプリケーションプログラムに適用して評価した結果について述べる．本論文で前提としているマルチグレイン並列処理[23] では，基本ブロック間の並列性は制御依存とデータ依存を考慮して，基本ブロック，ループ，サブルーチン間の並列性を引き出す最早実行可能条件解析および粗粒度タスク間並列処理に利用されるため，ここでは近細粒度並列処理を適用する基本ブロック内の命令レベルおよびステートメントレベルの評価のみを行う．

スパースマトリクスの求解（S）[17] 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．

NS3D/SUB4 [17] このプログラムは，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」の最外側ループに内包される 4 つの Do-loop のうち，最も実行時間の大きい 2 番目の Do-loop のループボディを取り出したもので，429 個の近細粒度タスクを持つ．SUB4 の最外側ループは Doall 可能なループであり，今回の評価は，外側ループの並列性は SCM を複数接続しているマルチプロセッサの SCM 間で使用し，SCM に割り当てられたイタレーションをさらに SCM 内の PE 間で近細粒度並列処理を行うことを想定して評価を行っている．

FPPPP/FPPPP このプログラムは，SPECfp95 ベンチマーク集の「FPPPP」からサブルーチン「FPPPP」を抜き出したものである．このサブルーチンはプログラム全体の実行時間の約 35%を占める部分であり，サブルーチン全体が 333 個の近細粒度タスクから構成される．

表 3 プログラムの性質
Table 3 The character of each program.

プログラム	IPC	平均タスクコスト（1PE）[clock]	平均タスクコスト（4PE）[clock]	同期・データ転送コスト [%]	タスク数
S	0.51	17.8	15.6	36.4	94
NS3D	0.62	22.6	20.0	51.1	429
FPPPP	0.74	42.0	42.7	23.9	333

これらのプログラムの 1 命令発行 × 1PE で計測したときの IPC，1 命令発行 × 1PE および 1 命令発行 × 4PE のそれぞれで計測した近細粒度タスクの平均コスト（平均タスクコスト（1PE），平均タスクコスト（4PE））， 4PE の実行時間の合計に対する近細粒度データの転送や同期フラグ待ちに要した時間の合計の割合（同期・データ転送コスト），および近細粒度タスクのタスク数を**表 3** に示す．

表 3 より，3 つのプログラムの中で FPPPP の平均

タスクコストが最も大きく，42.7 クロックかかることが分かる．スパースマトリクスの求解（S）と NS3D で，4 PE 時の平均タスクコストが 1 PE のときより減少しているのは，4 基の PE を使用することで総レジスタ数が 1 PE のときよりも増えたことにより，レジスタのスピルコードが減ったためである．また，同期・データ転送の割合は NS3D が最も大きく，実行時間の約半分がデータ転送や同期フラグ待ちに費やされていることが分かる．

4.1 命令発行数と PE 数に関する評価

ここでは，まずはじめに，命令発行数がそれぞれ 1, 2, 4（1 Issue, 2 Issue, 4 Issue）であるプロセッサコアを，1 基から 4 基まで変えて搭載した SCM アーキテクチャの評価を行う．このとき，各アーキテクチャの上記 3 種のアプリケーションに対する性能を，単一の 1 命令発行プロセッサでの実行時間に対する速度向上率として，**図 4**，**図 5**，**図 6** に示す．

まず，図 4 のランダムスパースマトリクスにおいて，1 Issue のアーキテクチャでは 2 PE 時で 1 Issue × 1 PE の 1.70 倍，4 PE 時で 2.83 倍の速度向上が得られている．同様に，2 Issue のアーキテクチャでは，1 PE 時で 1 Issue × 1 PE の 1.13 倍，2 PE 時で 1.91 倍，4 PE 時で 3.10 倍の速度向上，4 Issue のアーキテクチャでは，1 PE 時で 1 Issue × 1 PE の 1.14 倍，2 PE 時で 1.94 倍，4 PE 時で 3.23 倍の速度向上をそれぞれ得ている．また，図 5 の NS3D では，1 Issue アーキテクチャの 2 PE 時で 1.56 倍，4 PE 時で 2.20 倍，2 Issue アーキテクチャの 1 PE 時で 1.12 倍，2 PE 時で 1.74 倍，4 PE 時で 2.46 倍，4 Issue アーキテクチャの 1 PE 時で 1.14 倍，2 PE 時で 1.78 倍，4 PE 時で 2.52 倍の速度向上を得ている．同じく，図 6 の FPPPP では，1 Issue アーキテクチャの 2 PE 時で 1.74 倍，4 PE 時で 2.98 倍，2 Issue アーキテクチャの 1 PE 時で 1.21 倍，2 PE 時で 2.11 倍，4 PE 時で 3.54 倍，4 Issue アーキテクチャの 1 PE 時で 1.24 倍，2 PE 時で 2.15 倍，4 PE 時で 3.62 倍の速度向上を得ている．以上より，各アーキテクチャ上で近細粒度並列処理を行うことにより，どのアーキテクチャにおいても，PE 数の増加に応じてスケーラブルな性能向上を得られることが分かる．一方，1 PE 時で命令発行数を増加させたときに着目すると，図 4 のランダムスパースマトリクスでは命令発行数が 2，4 と増加するとともに，1 Issue に対する性能が 1.13 倍，1.14 倍となる．同様に，図 5 の NS3D では 1.12 倍，1.14 倍，図 6 の FPPPP では，1.21 倍，1.24 倍となり，PE 数を増加させたときのような効果が得られていない．このように，PE 数の増加と命令発行数の増加で得られる性能向上が異なる理由として，in-order 発行およびスーパスカラプロセッサにおける演算器の非対称性があげられる．すなわち，今回比較対象とした UltraSPARC II は in-order 発行であり，並列性抽出の範囲が隣接する命令間と非常に狭い．また，UltraSPARC II の演算器構成は整数演算ユニットが 2 基，浮動小数点ユニットが 2 基であり，しか

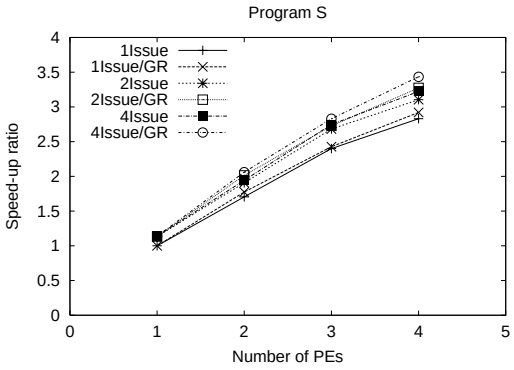

図 4 ランダムスパースマトリクスにおける速度向上率
Fig. 4 Speed up ratio of random sparse matrix solution.

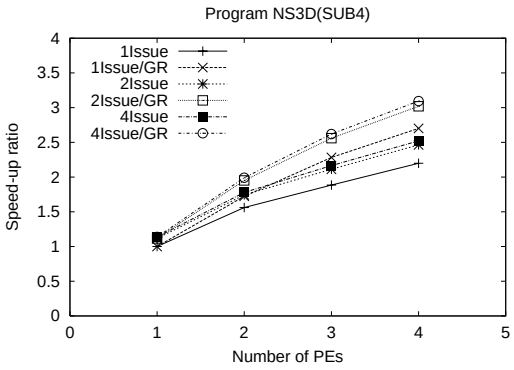

図 5 NS3D における速度向上率
Fig. 5 Speed up ratio of NS3D.

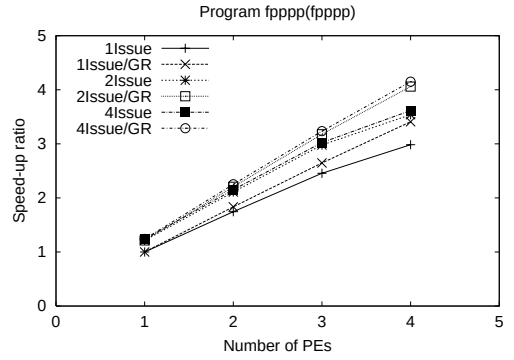

図 6 FPPPP における速度向上率
Fig. 6 Speed up ratio of FPPPP.

も各々の演算器で実行できる命令にも制限がある．このため，十分に並列性があるプログラムでも，同時に発行できる命令数には上限が生じてしまう．また，プロセッサコアのロード・ストアユニットは 1 基のみであるのに対し，SCM アーキテクチャでは，LDM や自 DSM へのアクセスは各 PE が他の PE に干渉することなく並列に行うことができることも前述の性能差の一因となっている．結果として，1 Issue × 4 PE の SCM アーキテクチャは，4 Issue × 1 PE のアーキテクチャに対し，ランダムスパースマトリクスでは 2.48 倍，NS3D では 1.93 倍，FPPPP では 2.40 倍の性能を得ている．

チップ内 PE 数 4 のときに注目すると，図 4 では 1 Issue × 4 PE に対し，2 Issue × 4 PE が 9.7%，4 Issue × 4 PE は 13.9%の性能向上を達成している．プロセッサコア単体の同時命令発行数が増加すると，レジスタファイルのポート数の増加，フォワーディング機構や命令発行系の複雑化による回路規模および遅延の増大等が生じる．以上のことから，プロセッサコア単体の命令発行数を増加させても，その回路規模に見合う性能向上が得られていないと考えられる．この結果は NS3D と FPPPP でも同様で，4 Issue × 4 PE では 1 Issue × 4 PE の 14.7〜21.5%の性能向上しか得られず，回路規模の拡大に見合う性能向上が得られていないことが分かる．

ここで，4 Issue × 1 PE の 1 Issue × 1 PE に対する性能向上率と，4 Issue × 4 PE の 1 Issue × 4 PE に対する性能向上率が，ランダムスパースマトリクスと NS3D で約 14%，FPPPP で約 22%と，1 PE 時と 4 PE 時でほぼ同じものとなっている．表 3 の IPC および平均タスクコストより，各プログラムの近細粒度タスク内命令数はおよそ 8〜32 命令であり，スーパスカラプロセッサをプロセッサコアとしたときに，各々のコアがこれらの命令列からさらに並列性を抽出しているために，このような結果となる．またこれは，比較対象のスーパスカラプロセッサが抽出できる並列性の範囲はせいぜい近細粒度タスク程度の大きさであるという，前述した in-order 発行の並列性抽出能力の低さを示しているともいえる．

1 Issue × 4 PE のときに着目すると，図 4 では 1 PE のときの 2.83 倍の性能を得ているが，これは 2 Issue × 3 PE および 4 Issue × 3 PE と同等の性能である．図 5 および図 6 でも同様に，1 Issue × 4 PE は，2 Issue × 3 PE および 4 Issue × 3 PE と同等の性能となっている．これはすなわち，1 Issue × 4 PE がトランジスタ数やチップ面積において 2 Issue × 3 PE 以下であれば，1 Issue × 4 PE が価格性能比に優れたアーキテクチャとなることを意味する．1 Issue プロセッサコアを複数搭載するアーキテクチャは，簡素なプロセッサコアを用いるのでクロック周波数を上げやすいことや，今後マルチグレイン並列処理による複数粒度での並列処理により PE 数増加とともにさらなる性能向上を望める，細粒度のスタティックスケジューリングによる最適化を行いやすいといった点から，単純な 1 Issue を用いた SCM 方式が，より価格性能比に優れたマイクロプロセッサの構築を可能とすると期待できる．

4.2 Out-of-order プロセッサとの比較

次に，1 Issue × 4 PE の SCM と，out-of-order プロセッサの比較を行った結果について述べる．比較対象の out-of-order プロセッサは，Compaq 社の Alpha21264 [30] とほぼ同等の演算器構成を持つ，整数演算 4 命令，浮動小数点演算 2 命令の最大 6 命令同時発行可能なスーパスカラプロセッサである．このプロセッサは整数演算器を 4 基，ロード・ストアユニットを 2 基，浮動小数点演算器を 2 基搭載し，各々の演算のレイテンシおよびスループットは UltraSPARC II と同じものとする．レジスタは，整数演算および浮動小数点演算用の各々32 本の論理レジスタに加え 40 本ずつのリネーミング用レジスタを持つ．動的スケジューリングの対象となる命令バッファは，整数命令用に 20 エントリ，浮動小数点命令用に 15 エントリ用意されている．ただし，実際の Alpah21264 プロセッサは 4 命令同時フェッチであるが，ここではより命令フェッチ数を増やした 8 命令同時フェッチとして評価を行った．また，メモリシステムは理想的なマルチポートメモリを仮定し，複数のメモリアクセス要求が起こってもすべて同時に処理できるものとした．

さらに，この out-of-order プロセッサ（ooo(BASE)）を基にして，浮動小数点演算器を 4 基構成にして，整数演算・浮動小数点演算合わせて 6 命令まで同時に発行できるようにしたもの（ooo(FPU4)），ロード・ストアユニットを 4 基構成にしたもの（ooo(LSU4)），命令バッファを 2 倍（整数 40 エントリ，浮動小数点 30 エントリ）にしたもの（ooo(IBUF2)），リネーミング用のレジスタを 2 倍の 80 本ずつにしたもの（ooo(REN2)）に対しても評価を行い SCM アーキテクチャと比較した．この結果を，単一の 1 命令発行プロセッサでの実行時間に対する速度向上率として，**図 7**，**図 8**，**図 9** に示す．図では左から順に，1 Issue × 1 PE，4 Issue × 1 PE の単一プロセッサアーキテクチャ，1Issus × 4 PE，1 Issue × 6 PE の SCM アーキテクチャならびに，Alpha21264 相当の out-of-order スーパスカラプ

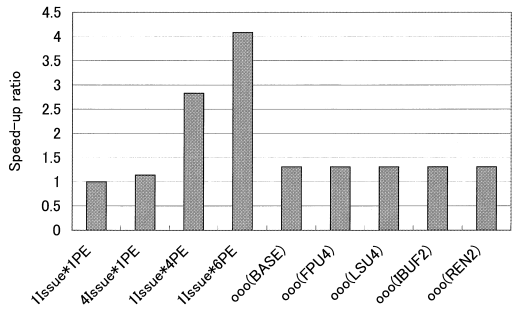

図 7 ランダムスパースマトリクスによる SCM と out-of-order の性能評価
Fig. 7 Performance evaluation of SCM and out-of-order processor by randam sparse matrix solusion.

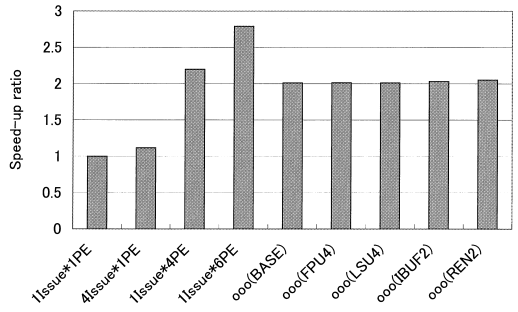

図 8 NS3D による SCM と out-of-order の性能評価
Fig. 8 Performance evaluation of SCM and out-of-order processor by NS3D.

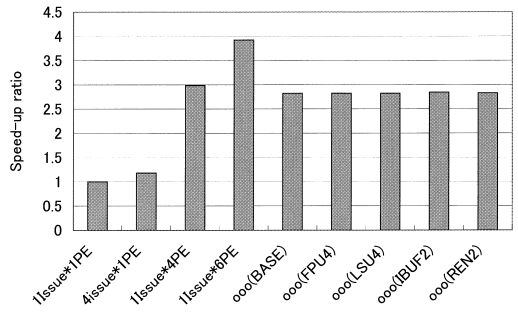

図 9 FPPPP による SCM と out-of-order の性能評価
Fig. 9 Performance evaluation of SCM and out-of-order processor by FPPPP.

ロセッサとその機能拡張アーキテクチャの実行結果を示している．

まず，図 7 のランダムスパースマトリクス求解では，1 Issue × 4 PE の SCM アーキテクチャが 2.83 倍の速度向上を得ているのに対し，ooo(BASE) は 1.31 倍の速度向上しか得ていない．これは，このプログラム内のほとんどすべての近細粒度タスクが除算を持っており，パイプライン化されていない除算を実行するたびに浮動小数点演算器がストールしてしまうためであり，他の条件の out-of-order プロセッサでも結果はほぼ同じであった．一方，図 8 の NS3D では 1 Issue × 4 PE の SCM で 2.20 倍，ooo(BASE) で 2.01 倍の速度向上，図 9 の FPPPP では 1 Issue × 4 PE の SCM で 2.98 倍，ooo(BASE) で 2.82 倍の速度向上をそれぞれ得ており，6 命令発行の out-of-order スーパスカラプロセッサが 1 Issue × 4 PE の SCM とほぼ同等の性能を示すことが分かる．これは，out-of-order 発行，レジスタリネーミング機構および演算器構成の強化により，4.1 節で評価対象とした in-order 命令発行の UltraSPARC II をベースとしたスーパスカラプロセッサよりも並列性抽出能力が大幅に向上したためである．しかしながら，SCM アーキテクチャでは PE 数を 6 に増やすことで，6 命令発行 out-of-order スーパスカラプロセッサに対し，NS3D では 1.39 倍の性能，FPPPP でも 1.39 倍の性能が得られることが確かめられた．以上より，SCM アーキテクチャでは out-of-order スーパスカラプロセッサと比較し，投入した資源（PE）に応じた性能向上を得られることが分かる．

4.3 共有グローバルレジスタに関する評価

次に，各々のアーキテクチャにおいて 32 本の共有グローバルレジスタを付加したアーキテクチャ（GR）の性能に関する評価を行う．

図 4 のランダムスパースマトリクスにおける共有グローバルレジスタの効果に着目すると，チップ内に 4 PE 集積されているとき，1 Issue で 3.2%，2 Issue で 5.5%，4 Issue で 6.5%の性能向上が得られている．同様に，図 5 の NS3D では，グローバルレジスタの付加により 1 Issue で 22.8%，2 Issue で 22.6%，4 Issue で 22.7%，また，図 6 の FPPPP では，1 Issue で 14.2%，2 Issue で 14.7%，4 Issue で 14.8%の性能向上が得られた．

1 Issue × 4 PE 構成において，共有グローバルレジスタを使用したときの近細粒度データの転送や同期フラグ待ちに要した時間を計測したところ，共有グローバルレジスタなしのときに対して，NS3D では 40.3%，FPPPP では 50.7%まで削減されていた．NS3D では，表 3 のように，実行時間の約半分が同期およびデータ転送に費やされている．このようなアプリケーションでは，特に共有グローバルレジスタを用いた同期オーバヘッド削減の効果が大きい．

次に共有グローバルレジスタのレイテンシを 2 クロックおよび 3 クロックに増やして評価を行った．1 Issue × 4 PE に共有グローバルレジスタを付加したアーキテクチャ上で NS3D と FPPPP を実行した結果を，共

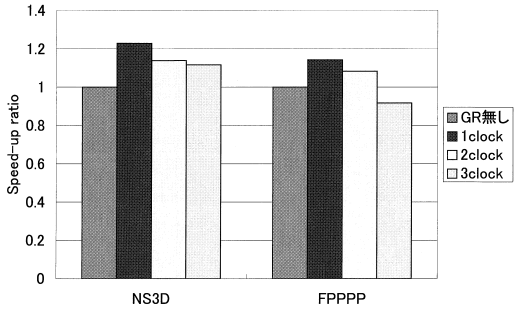

図 10 共有グローバルレジスタのレイテンシを変えての評価
Fig. 10 Performance evaluation by varying latency of shared global registers.

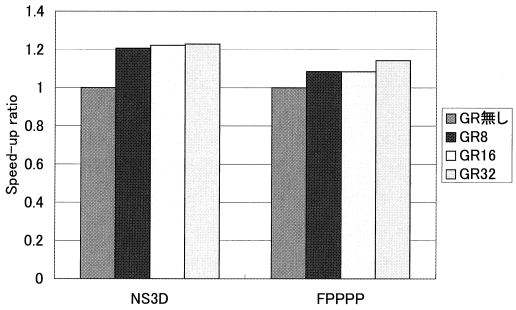

図 11 共有グローバルレジスタの本数を変えての評価
Fig. 11 Performance evaluation by varying number of shared global registers.

有グローバルレジスタなしのときに対する速度向上率として**図 10** に示す．図では，各々のプログラムに対して左から順に，共有グローバルレジスタなしの場合（GR なし），共有グローバルレジスタのレイテンシが 1 クロックの場合（1clock），2 クロックの場合（2clock）および 3 クロックの場合（3clock）での実行結果をそれぞれ示している．図 10 より，NS3D では，レイテンシが 1 のときには共有グローバルレジスタなしに対し 22.8%の性能向上を得ていたが，レイテンシの増加に従い，性能向上率が 2 クロックの場合で 14.2%，3 クロックの場合で 11.6%まで低下している．さらに，FPPPP では，レイテンシが 1 のときには 14.2%の性能向上を得ていたのが，レイテンシが 2 クロックの場合は性能向上率が 8.3%，レイテンシが 3 クロックの場合では共有グローバルレジスタなしに対し 91.7%の性能になってしまっている．DSM を使ったデータ転送では，リモート DSM アクセスで 4 クロック，ローカル DSM アクセスで 1 クロックの，計 5 クロックでデータの授受が行われる．一方，共有グローバルレジスタのレイテンシが 3 クロックの場合は，データの書き込みおよび読み出しで計 6 クロックかかるため，FPPPP のような性能低下が起こる．しかしながら，NS3D のようにデータ転送のオーバヘッドが大きいアプリケーションでは，同期フラグや近細粒度データの授受に，PE 間バスと共有グローバルレジスタの 2 つの経路を有効に利用できるために，レイテンシの増加による性能低下が少ないものと考えられる．

さらに，共有グローバルレジスタの本数を 16 本および 8 本に変えて評価を行った．図 10 と同様に，1 Issue × 4 PE に共有グローバルレジスタを付加したアーキテクチャ上で NS3D と FPPPP を実行した結果を，共有グローバルレジスタなしのときに対する速度向上率として**図 11** に示す．図では，各々のプログラムに対して左から順に，共有グローバルレジスタなしの場合（GR なし），共有グローバルレジスタの本数が 8 本の場合（GR8），16 本の場合（GR16）および 32 本の場合（GR32）での実行結果をそれぞれ示している．図 11 より，NS3D では，共有グローバルレジスタの本数が 8 のときに，共有グローバルレジスタなしの場合に対し 20.7%の性能向上を得ており，以降はレジスタの本数を増やしても性能はあまり変わらず，レジスタが 32 本で 22.8%となっている．一方で，FPPPP では，レジスタ本数が 8 の場合に 8.5%の性能向上であったのが，レジスタ本数の増加とともに徐々に性能が向上し，32 本のときに 14.2%の性能向上を得ている．NS3D と FPPP におけるこのような性能向上の違いを調べるため，レジスタ本数が 8 の場合に各々のアプリケーションで共有グローバルレジスタに割り当てられた近細粒度データ転送や同期フラグ授受の個数，およびこれらのデータ転送が 1 回の転送でレジスタを占有する時間を計測した．その結果 NS3D では，335 の近細粒度データ転送や同期フラグ授受の中から 33 が 8 つのレジスタに対して割り当てられており，このうちの 12 のデータ転送が 696 ページの表 3 に示した平均タスクコストである 20 クロック以下の期間レジスタを占有していた．同様に FPPPP では，316 の近細粒度データ転送や同期フラグ授受の中の 23 がレジスタに対して割り当てられており，このうちの 4 つのデータ転送が平均タスクコストである 43 クロック以下の期間レジスタを占有していた．このような短期間だけ共有グローバルレジスタを占有するデータ転送では，先行タスクが生成したデータを後続タスクがすぐに利用するという緊急度の高いデータを転送していると考えられ，NS3D ではこのようなデータ転送が多いために少ないレジスタ本数でも同期・データ転送オーバヘッド削減の効果が大きいと考えられる．

以上より，特に近細粒度並列処理における同期・データ転送のオーバヘッドが大きいアプリケーションにお

いて，共有グローバルレジスタの利用による同期オーバヘッド削減効果が大きいことが確認できた．

5. まとめ

本論文では，マルチグレイン並列処理をサポートするシングルチップマルチプロセッサ（SCM）アーキテクチャのプロセッサコア構成を検討する基礎研究として，近細粒度並列処理を命令発行数の異なるプロセッサコアを持つ SCM アーキテクチャに適用して性能評価を行った．その結果，1 命令発行 × 1 PE 構成のアーキテクチャと比較すると，ランダムスパースマトリクスの求解プログラムにおいて，1 命令発行のアーキテクチャでは 4 PE 時で 2.83 倍，また，2 命令発行のアーキテクチャでは 4 PE 時で 3.10 倍，4 命令発行のアーキテクチャでは 4 PE 時で 3.23 倍の速度向上をそれぞれ得ることができた．同様に，数値流体解析プログラム（NS3D）では，1 命令発行アーキテクチャの 4 PE 時で 2.20 倍，2 命令発行アーキテクチャの 4 PE 時で 2.46 倍，4 命令発行アーキテクチャの 4 PE 時で 2.52 倍の速度向上をそれぞれ得ることが分かった．また，SPECfp95 の FPPPP では，1 命令発行アーキテクチャの 4 PE 時で 2.98 倍，2 命令発行アーキテクチャの 4 PE 時で 3.54 倍，4 命令発行アーキテクチャの 4 PE 時で 3.62 倍の速度向上をそれぞれ得ることができ，いずれの場合でも，SCM アーキテクチャは近細粒度並列処理において，PE 数の増加とともにスケーラブルな性能向上を得ることができることを確認できた．

その一方で，1 PE のときに命令発行数を 2，4 と増加させたときは，ランダムスパースマトリクスの求解プログラムにおいて，1 命令発行 × 1 PE に対しそれぞれ 1.13 倍，1.14 倍，NS3D では 1.12 倍，1.14 倍，FPPPP では 1.21 倍，1.24 倍の性能しか得られない．4 PE のときに命令発行数を増加させてもほぼ同様の結果であり，以上から，プロセッサコア内の命令発行数の増加よりも，チップ内の PE 数の増加の方が性能向上に貢献することが確かめられた．結果として，1 命令発行 × 4 PE の SCM は 4 命令発行 × 1 PE に対して，ランダムスパースマトリクスの求解で 2.48 倍，NS3D では 1.93 倍，FPPPP では 2.40 倍の性能を得ることができた．

さらに，SCM アーキテクチャと高機能 out-of-order プロセッサの比較も行った．その結果，1 命令発行 × 4 PE 構成の SCM は，6 命令発行 out-of-order プロセッサとほぼ同程度の性能であったのが，SCM では PE 数を 6 に増やすことで 6 命令発行 out-of-order プロセッサに対し，NS3D では 1.39 倍，FPPPP でも 1.39 倍の性能を得ることができ，SCM アーキテクチャでは投入した資源（PE）に応じた性能向上を得ることが確かめられた．

1 命令発行の簡素なプロセッサコアを用いることにより，クロック周波数の向上を見込めることや，PE を多数チップ内に搭載することにより，今後マルチグレイン並列処理を適用する際にさらなる性能向上を期待できるといった点から，マルチグレイン並列処理用シングルチップマルチプロセッサのコアには，1 命令発行のものの使用が有望であると考えられる．

共有グローバルレジスタの有効性に関しては，上記 3 種のプログラムにおいて 1 命令発行 × 4 PE の SCM に 32 本の共有グローバルレジスタを付加したときに最大で 22.8%の性能向上を得ることができ，その有効性を確認できた．

今後の課題として，音声や画像処理といったマルチメディアアプリケーションや SPEC ベンチマーク等の各種アプリケーションに対し，マルチグレイン並列処理を適用した際の SCM 上での性能評価，今後の半導体技術を見越したうえでのメモリアーキテクチャや共有グローバルレジスタの構成およびそのレイテンシ等のパラメータのさらなる検討等があげられる．

謝辞 本研究の一部は，STARC「自動並列化コンパイラ協調型シングルチップ・マルチプロセッサの研究」により行われた．また，本論文作成にあたり，有益なコメントをいただいた STARC 研究員である，STARC 小澤時典氏，平田雅規氏，東芝 浅野滋博氏，富士通 高橋宏政氏，ソニー 倉田隆弘氏，松下 高山秀一氏に感謝いたします．

参考文献

1) Hennessy, J.: The Future of Systems Research, *Computer*, Vol.32, No.8, pp.27–33 (2000).
2) Palacharla, S., Jouppi, N.P. and Smith, J.E.: Quantifying the Complexity of Superscalar Processors, Technical Report, Univ. of Wisconsin-Madison (1996).
3) Agarwal, V., Hrishikesh, M.S., Keckler, S.W. and Burger, D.: Clock Rate versus IPC: The End of the Road for Conventional Microarchitectures, *Proc. ISCA 00* (2000).
4) Sun Microsystems, Inc.: MAJC Home Page (2000). http://www.sun.com/microelectronics/MAJC/.
5) Diefendorff, K.: Power4 Focuses on Memory Bandwidth, *Microprocessor Report*, Vol.13, No.13 (1999).

6) Barroso, L.A., Gharachorloo, K., McNamara, R., Qadeer, A.N.S., Sano, B., Smith, S., Stets, R. and Verghese, B.: Piranha: A Scalable Architecture Based on Single-Chip Multiprocessing, *Proc. ISCA 00* (2000).
7) Hammond, L., Hubbert, B., Siu, M., Prabhu, M.K., Chen, M. and Olukotun, K.: The Stanford HYDRA CMP, *IEEE MICRO Magazine*, Vol.20, No.2, pp.71–84 (2000).
8) Vijaykumar, T.N. and Sohi, G.S.: Task Selection for a Multiscalar Processor, *31st Int'l Conf. on Microarchitecture* (*MICRO-31*) (1998).
9) Tsai, J.-Y., Jiang, Z., Ness, E. and Yew, P.-C.: Performance Study of a Concurrent Multithreaded Processor, *Proc. 4th Int'l Conf. on HPCA-4* (1998).
10) NEC Corporation: MP98 Project (2000). http://www.labs.nec.co.jp/ MP98/.
11) 鳥居, 近藤, 本村, 池野, 小長谷, 西：オンチップ制御並列プロセッサ MUSCAT の提案, 情報処理学会論文誌, Vol.39, No.6, pp.1622–1631 (1998).
12) 小林, 岩田, 安藤, 島田：非数値計算プログラムのスレッド間命令レベル並列を利用するプロセッサ・アーキテクチャSKY, *JSPP'98* (1998).
13) Barua, R., Lee, W., Amarasinghe, S. and Agarwal, A.: Maps: A Compiler-Managed Memory System for Raw Machines, *Proc. ISCA-26* (1999).
14) Kang, Y., Huang, M., Yoo, S., Ge, Z., Keen, D., Lam, V., Pattnaik, P. and Torrellas, J.: FlexRAM: An Advanced Intelligent Memory System, *Proc. Int'l Conf. on Computer Design* (1999).
15) 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, 情報処理学会研究報告, Vol.96, No.80, pp.161–166 (1996).
16) Olukotun, K., Nayfeh, B.A., Hammond, L., Wilson, K. and Chang, K.: The Case for a Single-Chip Multiprocessor, *Proc. ASPLOS VII* (1999).
17) 木村, 尾形, 岡本, 笠原：シングルチップマルチプロセッサ上での近細粒度並列処理, 情報処理学会論文誌, Vol.40, No.5, pp.1924–1934 (1999).
18) 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 (1996).
19) Padua, D. and Wolfe, M.: Advanced Compiler Optimization for Super Computers, *C.ACM*, Vol.29, No.12, pp.1184–1201 (1996).
20) 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).
21) 本多, 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).
22) Kasahara, H., Honda, H. and Narita, S.: A Multigrain Parallelizing Compilation Scheme for OSCAR, *Proc.4th Workshop on Lang. and Compilers for Parallel Computing* (1991).
23) Kasahara, H., Obata, M. and Ishizaka, K.: Automatic Coarse Grain Task Parallel Processing on SMP using OpenMP, *Proc. 13th International Workshop on Language and Compilers for Parallel computing* (*LCPC'00*) (2000).
24) 本多, 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).
25) 笠原：並列処理技術, コロナ社 (1991).
26) 吉田, 越塚, 岡本, 笠原：階層型粗粒度並列処理における同一階層内ループ間データローカライゼーション手法, 情報処理学会論文誌, Vol.40, No.5, pp.2054–2063 (1999).
27) 笠原, 尾形ほか：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).
28) Lev, L.A., et al.: A 64-b Microprocessor with Multimedia Support, *IEEE Journal of SOLID-STATE CIRCUITS*, Vol.30, No.11, pp.1227–1238 (1995).
29) Sun Microelectronics: UltraSPARC^{TM} User's Manual (1997).
30) Kessler, R.E.: The Alpha 21264 Microprocessor, *IEEE MICRO Magazine*, Vol.19, No.2, pp.24–36 (1999).

(平成 12 年 9 月 14 日受付)
(平成 13 年 2 月 1 日採録)

木村 啓二（正会員）

昭和 47 年生．平成 8 年早稲田大学理工学部電気工学科卒業．平成 13 年同大学大学院理工学研究科電気工学専攻博士課程修了．博士（工学）．平成 11 年同大学同学部助手，現在に至る．マルチグレイン並列処理用プロセッサアーキテクチャに関する研究に従事．

加藤 孝幸

昭和 49 年生．平成 10 年早稲田大学理工学部電気工学科卒業．平成 12 年同大学大学院修士課程修了．現在，本田技術研究所に勤務．在学中は，マルチグレイン並列処理用プロセッサアーキテクチャに関する研究に従事．

笠原 博徳（正会員）

昭和 32 年生．昭和 55 年早稲田大学理工学部電気工学科卒業．昭和 60 年同大学大学院博士課程修了．工学博士．昭和 58 年同大学同学部助手．昭和 60 年学術振興会特別研究員．昭和 61 年早稲田大学理工学部電気工学科専任講師．昭和 63 年同助教授．平成 9 年同大学同学部電気電子情報工学科教授，現在に至る．平成元年〜2 年イリノイ大学 Center for Supercomputing Research & Development 客員研究員．昭和 62 年 IFAC World Congress 第 1 回 Young Author Prize．平成 9 年度情報処理学会坂井記念特別賞受賞．著書「並列処理技術」（コロナ社）．電子情報通信学会，IEEE 等会員．