計算機アーキテクチャ 134－4
（1999. 8. 2）

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の性能評価

木村啓二[†] 間中邦之[†] 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,manaka,ogata,okamoto,kasahara}@oscar.elec.waseda.ac.jp

あらまし　半導体技術の進歩にしたがい，大量の演算器，メモリ，あるいは複数のプロセッサを1チップ上に搭載することが可能となりつつある．これらの資源を有効に利用しさらなる性能向上を図るための次世代マイクロプロセッサおよびそのソフトウェア技術（特にコンパイラ技術）に関する研究が，現在活発に行なわれている．次世代マイクロプロセッサアーキテクチャの中で，シングルチップマルチプロセッサ（SCM）は従来の命令レベルのみでなく，異なる粒度の並列性を階層的に組合せプログラム全体に渡り並列性を抽出するマルチグレイン並列処理を適用可能であり，高い実効性能とスケーラブルな性能向上が可能なアーキテクチャであると考えられる．本論文では，従来のマルチプロセッサでは効果的な並列処理が困難であったアプリケーションプログラムに対し，マルチグレイン並列処理の主要技術の一つである近細粒度並列処理を適用し，SCM上での性能評価を行なったので，その結果について述べる．

Performance Evaluation of Near Finegrain Parallel Processing on the Single Chip Multiprocessor

Keiji Kimura[†], Kuniyuki Manaka[†], 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,manaka,ogata,okamoto,kasahara}@oscar.elec.waseda.ac.jp

Abstract　Advances in semiconductor technology allows us to integrate a lot of integer and floating point execution units, memory or processors on a single chip. To use these resources effectively, many researches on next generation microprocessor architectures and its software, especially compilers have been performed. In these next generation microprocessor architectures, a single chip multiprocessor(SCM) using multigrain parallel processing, which hierarchically exploits different level of parallelism from the whole program, is one of the most promising architectures. This paper evaluates performance of the SCM architectures for near fine grain parallel processing, which is one of the key issues in multigrain parallel processing, using several real application programs.

1　はじめに

半導体の集積技術の進歩により，1チップ上に演算器，メモリ，あるいはプロセッサといった資源を大量に搭載できるようになりつつある．これらの資源を活用し，さらなる性能向上を図るための次世代マイクロプロセッサアーキテクチャ[1)~5)]，およびそのコンパイラ技術[6)~10)]が，現在多く提案されている．

これらの次世代マイクロプロセッサアーキテクチャのうち，複数のプロセッサコアを1チップ上に搭載し，これにより生じるプロセッサコア間およびプロセッサコア-メモリ間の高スループット・低レイテンシデータ転送や，低マルチスレッディングオーバーヘッドを目指したシングルチップマルチプロセッサ（SCM）が，最近特に注目を集めている[3),6)~10)]．これらのアーキテクチャには，投機的実行を駆使してスレッドレベルの並列処理を行なうもの[3),6)~8)]，また簡素なプロセッシングユニットを多数チップ上に配置し，これらのプロセッシングユニットとそのネットワークをコンパイラにより制御するもの[5),9)]等がある．

一方，筆者等は手続き型言語の並列処理を前提として，複数命令レベルでの並列処理である（近）細粒度並列処理[11]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[12]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[13),14)]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ができる，マルチグレイン並列処理を提案している[15]．このマルチグレイン並列処理をSCMに適用することにより，スケーラブルな性能向上を得られる計算機システムが実現可能であると考えられる[16]．

本論文では，この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を考えるために，マルチグレイン並列処理の主要技術である近細粒度並列処理のSCM上での性能を，電子回路シミュレーション，SPECベンチマークのFPPPPといった，従来のマルチプロセッサで並列処理が非常に難しかったアプリケーションを用いて評価したので，その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

本節では，本論文で評価するシングルチップマルチプロセッサが前提とする，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[15]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),17)]，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[11]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））[13]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）[17),18)]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[17),18)]により引きだし，その結果をマクロタスクグラフ（MTG）[17),18)]として出力する．その後MTは，ダイナミックスケジューリングもしく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．

近細粒度並列処理においては，BPA内のステートメント，もしくは複数のステートメントから構成される疑似代入文を一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・同期オーバー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[18]の4手法を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，データのメモリへの最適化やデータ転送・同期オーバーヘッドの最小化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の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することにより，各PEの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なわれる．

本論文では転送データおよび同期フラグの授受は，データキャッシュ共有型アーキテクチャ（3.2節）では共有キャッシュを，OSCAR型アーキテクチャ（3.3節）では分散共有メモリ（DSM）を介して行なわれるが，共有グローバルレジスタ（3.4節）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行なう．共有グローバルレジスタへのデータ転送の割り当てアルゴリズムには，単一プロセッサの汎用レジスタ割り当てに用いられるレジスタカラーリングを拡張したものを使用する．

3 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．

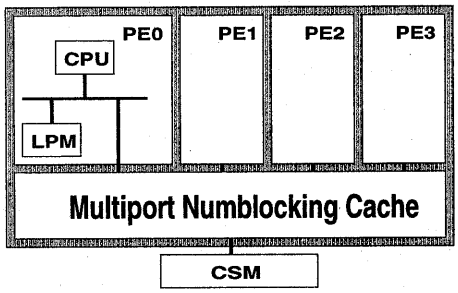

図 1: データキャッシュ共有型アーキテクチャ

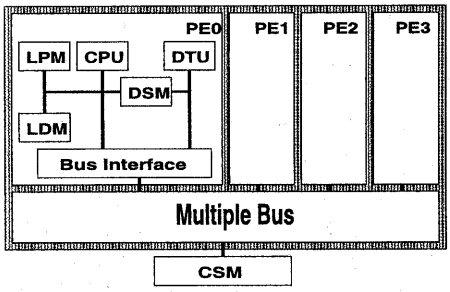

図 2: OSCAR 型アーキテクチャ

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を用いるものとする．

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1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[19]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．LDMは1ポートメモリであり，共有キャッシュやDSMに比べ少ないチップ面積で大容量化が可能である．そのため，コンパイラがデータのLDMへの割当を効果的に行なえば，価格性能比の良いシステムを構成できると考えられる．また，DSMは他PEからもアクセスできる2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に使用する．LDMのメモリアクセスにかかるクロック数は1，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1クロック，他PEからのアクセスには4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．

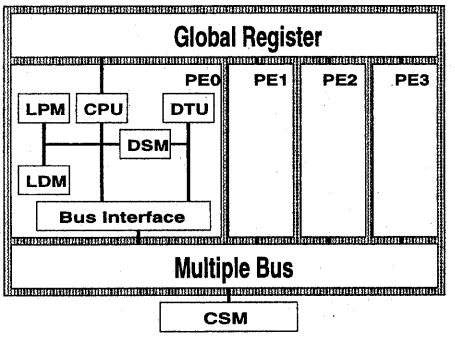

図3: OSCAR型アーキテクチャ＋グローバルレジスタ

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を適用して評価した結果について述べる．

評価に使用したプログラムは，従来のマルチプロセッサシステムではほとんど性能向上が望めなかった以下の2例である．

FPPPP/FPPPP このプログラムは，SPECfp95ベンチマーク集のプログラム「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．

電子回路シミュレーション ここで用いた回路シミュレーションプログラムは，Berkeley大学のSpice3f.4とほぼ同等なコードであり，線形方程式直接解法の一つであるクラウト法を用いてシンボリック生成によりFORTRANループフリーコードを出力する．ここでは，このFORTRANループフリーコードを近細粒度並列処理する．このコードはSpice3f.4添付の回路サンプルrtlin-v.cirシミュレーション用のループフリーコードであり，最内側の収束ループのボディが221個の近細粒度タスクから構成される．このプログラムの一部を図4に示す．図中，1ステートメント，もしくはIFからENDIFで囲まれた部分（疑似代入文）が一つの近細粒度タスクとなる．

これらのプログラムに近細粒度並列処理を適用し，OSCAR型アーキテクチャ（OSCAR），データキャッシュ共有型（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型SCMにおける1プロセッサ上での実行時間に対する速度向上率で表すと，図5および図6のグラフのようになる．

図より，OSCAR型がPE数の増加とともに性能

向上を得ることができ，図5のFPPPPでは2PEで1.48倍，4PEで2.27倍の速度向上率を得ていることがわかる．一方CACHE-WB型では，図5，図6共に2PEでは性能が向上しているが，4PEでは性能が低下してしまう．特に図6の電子回路シミュレーションでは，CACHE-WB型の2PEでの実行時間がOSCAR型1PE時の実行時間の1.49倍であったものが，4PEでは0.911倍にまで性能が落ちてしまい，OSCAR型4PEと比較して0.59倍の性能となってしまう．

これは，変数へのアクセス，データ転送および同期フラグチェック時のビジーウェイトの全てを共有キャッシュに対して行なうCACHE-WB型では，PE数の増加にともなうバンクコンフリクトの増加が性能低下を引き起こしてしまうのに対し，ローカル変数へのアクセスをローカルメモリに，共有データの転送・同期をDSMに対して行なうOSCAR型では，他PEのメモリアクセスを妨げることなくメモリアクセスコンテンションを最小化して並列処理を行なうことができるためと考えられる．

また，グローバルレジスタの使用に関しては，OSCAR型の4PE時でグローバルレジスタなしに対し，FPPPPでは11.7%，電子回路シミュレーションでは13.7%の性能向上を得ることができた．グローバルレジスタの使用によりデータ転送・同期のオーバーヘッドを削減でき，近細粒度並列処理をより効果的に行なえることを確認できた．

5 まとめ

本論文では，従来のマルチプロセッサでは並列化困難であったSPICE同等の回路シミュレーションおよびSPECベンチマーク集のFPPPPといった実アプリケーションに対し近細粒度並列処理を適用し，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シングルチップマルチプロセッサアーキテクチャの評価を行なった．

その結果，PEローカルに使用するデータはローカルメモリへ，データ転送や同期用のデータをDSM上と，コンパイラがメモリの使用方法をきめ細かく制御できるOSCAR型アーキテクチャが，近細粒度並列処理を適用した実アプリケーションに対しても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13.7%の速度向上を得ることができた．

```
      T3 = ABS ( v129 - v130 )
      T4 = ( v129 - v130 ) * 2.587435e-02 + 1
      IF(v129.GT.8.497211e-01.AND.T3.GT.5.174871e-02)THEN
        IF(v130.LE.0)THEN
      v198 = 3.864831e+01 * ALOG ( v129 * 2.587435e-02 )
        ELSE IF(T4.LE.0)THEN
      v198 = 8.497211e-01
        ELSE
      v198 = 3.864831e+01 * ALOG ( T4 ) + v130
        END IF
      ELSE
      v198 = v129
      END IF
      IF(v198.GE.1.201000e+00)THEN
      v198 = 1.201000e+00
      END IF
C     #2 exp_v&dexp_v
      IF(v198.GT.-1.293718e-01)THEN
      v213 = EXP ( c12 * v198 )
      v214 = c12 * EXP ( c12 * v198 )
      ELSE
      v213 = 0
      v214 = - 1 / v198
      END IF
      T06312 = v129 - v139
      T06311 = v130 - v140
C     #1 pnjilim
      T06313 = ABS ( T06312 - T06311 )
      T06314 = ( T06312 - T06311 ) * 2.587435e-02 + 1
      IF(T06312.GT.8.497211e-01.AND.T06313.GT.5.174871e-02)THEN
        IF(T06311.LE.0)THEN
      v202 = 3.864831e+01 * ALOG ( T06312 * 2.587435e-02 )
        ELSE IF(T06314.LE.0)THEN
      v202 = 8.497211e-01
        ELSE
      v202 = 3.864831e+01 * ALOG ( T06314 ) + T06311
        END IF
      ELSE
      v202 = T06312
      END IF
      IF(v202.GE.1.201000e+00)THEN
      v202 = 1.201000e+00
      END IF
C     #2 exp_v&dexp_v
      IF(v202.GT.-1.293718e-01)THEN
      v215 = EXP ( c12 * v202 )
      v216 = c12 * EXP ( c12 * v202 )
      ELSE
      v215 = 0
      v216 = - 1 / v202
      END IF
C     #4 qd0
      IF(v198.LT.3.750000e-01)THEN
      v190 = 9.000000e-13 * ( 1 - EXP ( - 3.300000e-01
     - * ALOG(1 - v198 * 1.333333e+00)) * (1 -
     - v198 * 1.333333e+00 )) * 1.492537e+00 * 7.500000e-01
      ELSE
      v190 = 9.000000e-13 * ( 4.158507e-01 + 2.514027e+00 * ( ( v198
     - * v198 - 3.750000e-01 * 3.750000e-01)* 3.300000e-01 *
     - 6.666667e-01 + 3.350000e-01
     - * (v198 - 3.750000e-01)))
      END IF
      v190 = v190 + 1.000000e-10 * ( 1.000000e-16 *
     - (v213 -1) + v198 *1.000000e-12)
C     #6  cd
      IF(v198.LT.3.750000e-01)THEN
      v188 = 9.000000e-13 * EXP ( - 3.300000e-01 *
     - ALOG(1 - v198 * 1.333333e+00))
      ELSE
      v188 = 9.000000e-13 * 2.514027e+00 * ( 3.300000e-01 *
     - v198 * 1.333333e+00 + 3.350000e-01)
      END IF
      v188 = v188 + 1.000000e-10 * ( 1.000000e-16
     - * v214 + 1.090000e-12)
```

図4: 電子回路シミュレーションのコード片

今後は，ループ並列処理，粗粒度並列処理も含めたマルチグレイン並列処理のシングルチップマルチプロセッサ上での評価，スーパースカラをコアとしたSCMとの性能比較，VLSI実装の際に問題となるハードウェア量やバスドライブ能力に関する検討等が課題として挙げられる．

本研究の一部は，通産省次世代情報処理基盤技術開発事業並列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．

参考文献

[1] C.Kozyrakis and D.Patterson. A New Direction for Computer Architecture Research. *Computer*, Vol. 31, No. 11, pp. 24–32, 1998.

[2] J.Smith and S.Vajapeyam. Trace processors: Moving to fourth generation microarchitectures. *Computer*, Vol. 30, No. 9, pp. 68–74, 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] M.Lipasti and J.Sben. Superspeculative microarchitecture for beyond ad 2000. *Computer*, Vol. 30, No. 9, pp. 59–66, 1997.

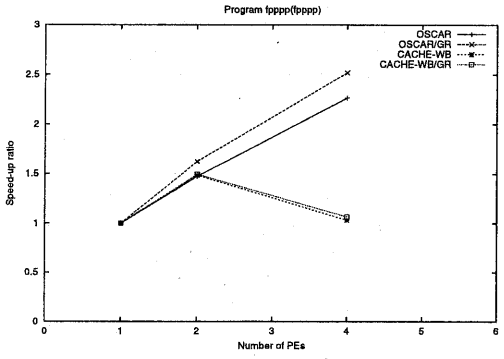

図 5: FPPPP における速度向上率

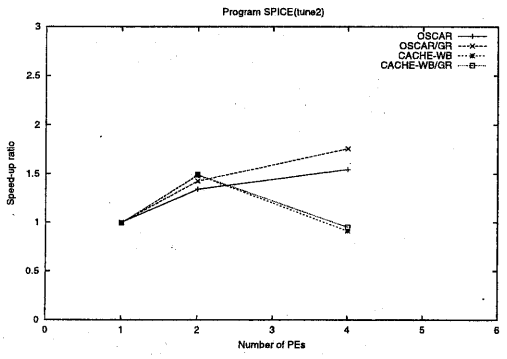

図 6: 電子回路シミュレーションにおける速度向上率

[5] Y.Kang, M.Huang, S.Yoo, Z.Ge, A.Keen, V.Lam, P.Pattnaik, and J.Torrellas. Flexram:an advanced intelligent memory system. In *Proc. Int'l Conf. on Computer Design*, Oct 1999.

[6] T.N.Vijaykumar and G.S.Sohi. Task Selection for a Multiscalar Processor. In *31th Int'l Conf. on Microarchitecture (MICRO-31)*, Nov-Dec 1998.

[7] K.Okulotun, L.Hammond, and M.willey. Improving the performance of speculatively parallel applications on the hydra cmp. In *Proc. of the 1999 ACM Int'l Conf on Superconmuting*, June 1999.

[8] J.G.Steffan and T.C.Mowry. The Potential for Using Thread-Level Data Speculation to Facilitate Automatic Parallelization. In *Proc. of the 4th Int'l Conf. on High-Performance Computer Architecture (HPCA-4)*, Feb 1998.

[9] R.Barua, W.Lee, S.Amarasinghe, and A.Agarwal. Maps:a compiler-managed memory system for raw machines. In *Proc. of ISCA-26*, June 1999.

[10] 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, May 1999.

[11] 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.

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

[13] 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.

[14] 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.

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

[16] 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.

[17] 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.

[18] 笠原. 並列処理技術. コロナ社, 1991.

[19] 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.