社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2008－ARC－178（4）
2008／5／13

階層グルーピング対応バリア同期機構の評価

山田 海斗†‡ 間瀬 正啓† 白子 準† 木村 啓二†

伊藤 雅之†† 服部 俊洋†† 水野 弘之‡ 内山 邦男‡ 笠原 博徳†

†早稲田大学 理工学術院 〒169-8555 東京都新宿区大久保 3-4-1
‡（株）日立製作所 〒185-8601 東京都国分寺市東恋ヶ窪 1-280
††（株）ルネサス テクノロジ〒187-8588 東京都小平市上水本町 5-20-1

E-mail: †{kaito,mase,shirako}@kasahara.cs.waseda.ac.jp, {keiji,kasahara}@waseda.jp ,
‡{hiroyuki.mizuno.vp,kunoio.uchiyama.xh}@hitachi.com,
††{ito.masayuki2,hattori.toshihiro}@renesas.com

あらまし マルチコアプロセッサに搭載されつつある多数のコアを効率よく利用するため，ループやサブルーチンの内部の並列性を階層的に解析しタスクの定義を行い，プログラム全域の並列性を利用する階層的粗粒度タスク並列処理が提案され OSCAR コンパイラに実装されている．階層的粗粒度タスク並列処理では，複数のプロセッサをソフトウェアにより階層的にグルーピングし，これらのグルーピングされたプロセッサ群に対して階層的に定義された粗粒度タスクを割り当てる．この階層的粗粒度タスク並列処理を効率よくサポートする，軽量かつスケーラブルな階層グルーピング対応バリア同期機構を開発し，NEDO リアルタイム情報家電用マルチコアプロジェクトにより開発した SH4A プロセッサ 8 コア搭載の情報家電用マルチコア RP2 に実装した．本稿では，この階層グルーピング対応バリア同期機構を提案すると共に RP2 上で評価を行った結果について述べる．8 コアを使用した AAC エンコーダによる評価の結果，ソフトウェアのみによるバリア同期に対し 16%の性能向上を得ることができた．

キーワード マルチコア，自動並列化コンパイラ，バリア同期機構

An Evaluation of Barrier Synchronization Mechanism Considering Hierarchical Processor Grouping

Kaito Yamada†‡ Masayoshi Mase† Jun Shirako† Keiji Kimura†

Masayuki Ito†† Toshihiro Hattori†† Hiroyuki Mizuno‡ Kunio Uchiyama‡ and Hironori Kasahara†

†Faculty of Science and Engineering, Waseda University 3-4-1 Okubo, Shinjuku-ku, Tokyo, 169-8555 Japan
‡Hitachi Ltd., 1-280 Higashi-Koigakubo Kokubunji-shi, Tokyo, 185-8601 Japan
††Renesas Technology Corp, 5-20-1 Josuihoncho Kodaira-shi, Tokyo, 187-8588 Japan

E-mail: †{kaito,mase,shirako}@kasahara.cs.waseda.ac.jp, {keiji,kasahara}@waseda.jp ,
‡{hiroyuki.mizuno.vp,kunoio.uchiyama.xh}@hitachi.com,
††{ito.masayuki2,hattori.toshihiro}@renesas.com

Abstract In order to use a large number of processor cores in a chip, hierarchical coarse grain task parallel processing, which exploits whole program parallelism by analyzing hierarchical coarse grain task parallelism inside loops and subroutines, has been proposed and implemented in OSCAR automatic parallelizing compiler. This hierarchical coarse grain task parallel processing defines processor groups hierarchically and logically, and assigns hierarchical coarse grain tasks to each processor group. A light-weight and scalable barrier synchronization mechanism considering hierarchical processor grouping, which supports hierarchical coarse grain task parallel processing, is developed and implemented into RP2 multicore processor having eight SH4A cores with support by NEDO "Multicore Technology for Realtime Consumer Electronics". This barrier mechanism is proposed and evaluated in this paper. The evaluation using AAC encoder program by 8 cores shows our barrier mechanism achieves 16% better performance than software barrier.

Keyword Multicore Processor, Automatic Parallelizing Compiler, Barrier Synchronization Mechanism

1. はじめに

マルチコアプロセッサがデスクトップ PC やサーバといったハイエンドの分野から組み込みシステムに至るまであらゆる分野で活用されつつあり，それに応じて 1 チップに集積されるコア数も増加し，64 コアや 80 コアを搭載するものまで開発されている[1][2]. これら多数のコアを搭載するメニーコア上で並列化されたソフトウェアを実行する場合，バリア同期のような基本的な同期機構のオーバーヘッドがその実行効率に大きな影響を及ぼしうる.

現在のマルチプロセッサシステムあるいはマルチコアシステムにおいてバリア同期機構を実現する場合，主記憶共有型のマルチプロセッサシステムでは，メモリ上にバリア同期用のカウンタを配置し，排他制御を使用しつつバリア同期用カウンタを操作するものが一般的である[5]. またハードウェアでバリア同期機構を実現する場合，各プロセッサがバリア地点に到達したことを示すビットを書き込むレジスタを用意し，バリア同期に参加する全てのプロセッサがバリア地点に到達したかどうかを全てのビットの論理積をとることでチェックするハードウェアが提案されている[6]. ソフトウェアでバリア同期を実現する方式の場合，排他制御を使用するためにコア数増加への対応が難しい. 従来のハードウェアによる方式の場合においても，各プロセッサのビットを集約して論理積をとるなどのハードウェアが必要になり，やはりコア数増加への対応が難しい.

一方筆者等は，コア数増加に応じたスケーラブルな性能向上を可能にする並列処理方式としてマルチグレイン並列処理を提案し，これを OSCAR マルチグレイン自動並列化コンパイラに実装してきた[3]. 特にマルチグレイン並列処理の主要構成要素である階層的粗粒度タスク並列処理においては，プログラム全域の並列性を引き出して利用するために，ループやサブルーチンコールを粗粒度タスクとして定義して並列性を解析し，これらのタスク中にさらにループやサブルーチンコールがある場合はそれらをさらにサブタスクとして並列性を解析するといった，階層的なタスク定義および並列性解析を行う. 実際に並列化されたプログラムを実行する場合は，コアを階層的にグルーピングし，これらのプロセッサコアグループに対して階層的に定義された粗粒度タスクを割り当て実行する[4]. すなわち，バリア同期も階層的にグループ化されたコアに対して行えるようにする必要がある.

以上のような状況を踏まえ，筆者等はコア数が増加しても低オーバーヘッドでバリア同期を行うことができ，かつコアの柔軟なグルーピングに対応するソフトウェア・ハードウェア協調型バリア同期機構を開発し，NEDO リアルタイム情報家電用マルチコアプロジェクトにより開発した SH4A プロセッサ 8 コア搭載の情報家電用マルチコア RP2 に実装した[7]. 本稿では，この階層グルーピング対応バリア同期機構を提案すると共に RP2 上で評価した結果について述べる.

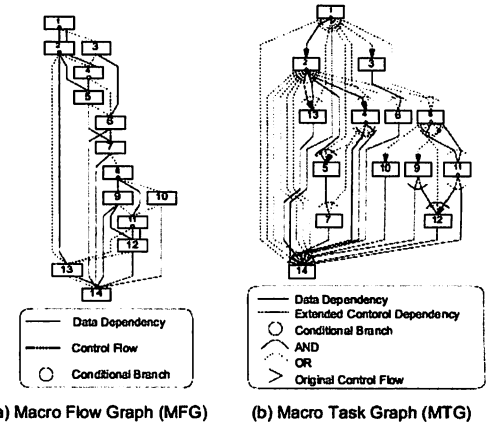
以降，第 2 章で OSCAR マルチグレイン自動並列化コンパイラについて説明する. 本章では特に階層的粗粒度タスク並列処理および OSCAR コンパイラにおけるバリア同期コードについて説明する. 第 3 章では新規に開発した階層グループ対応バリア同期機構について，第 4 章では情報家電用マルチコア RP2 についてそれぞれ説明する. 第 5 章で評価について述べ，第 6 章で本稿のまとめとする.

2. OSCAR マルチグレイン自動並列化コンパイラ

本章では OSCAR マルチグレイン自動並列化コンパイラについて，特にその主要構成要素である階層的粗粒度タスク並列処理を説明する. また，OSCAR コンパイラにより並列化されたプログラムを SMP サーバ等の通常のハードウェアで実行する際のバリア同期コードについても説明する.

2.1. 粗粒度タスク並列処理

階層的粗粒度タスク並列処理では，ソースとなる C あるいは Fortran プログラムを基本ブロック(BPA)，ループ(RB)，サブルーチンコール(SB)の 3 種類のマクロタスク（MT）に分割する. MT 生成後，コンパイラは MT 間のコントロールフローとデータ依存関係を表現したマクロフローグラフ（MFG）を生成し，さらに MFG から MT 間の並列性を再早実行可能条件解析により引き出した結果をマクロタスクグラフ（MTG）として表現する. MFG および MTG の例を図 1 に示す.



(a) Macro Flow Graph (MFG)　(b) Macro Task Graph (MTG)

図 1　マクロフローグラフとマクロタスクグラフ

あるMTを構成するサブルーチンコールやループ内部に粗粒度タスク並列性が存在する場合，図2に示すようにそのサブルーチンコールやループ内部でさらにMTGを生成する．

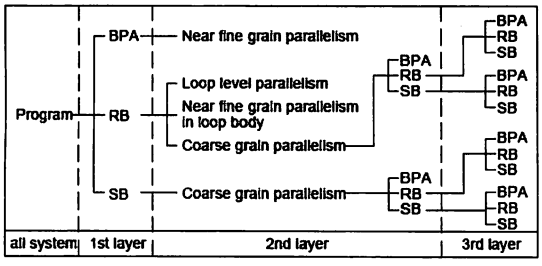


図2　マクロタスクの階層的な定義

このように階層的に生成されたMTGを処理するために，一つ以上のプロセッサエレメント（PE：コアと同義）からなるプロセッサコアグループ（PG）を図3のように階層的に定義し，MTにPG単位で割り当てて実行する．

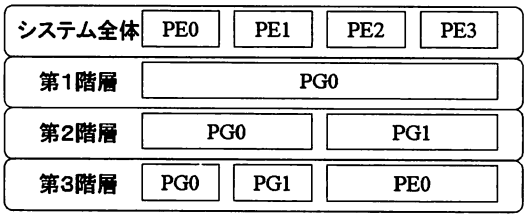


図3　階層的なプロセッサコアグルーピング

階層的粗粒度タスク並列処理を施した後，ループのMTにループ並列性が存在する場合はループイタレーションレベルの並列処理（ループ並列処理）を施す．また，ソースコードのステートメントレベルの並列処理が抽出可能であれば近細粒度並列処理を施す．

図4に階層的なマクロタスクグラフ（図4(a)）と8コア用に生成されたコードイメージ(図4(b))を示す．図4のマクロタスクグラフでは，第1階層中のMT1_3とMT1_4の中に粗粒度タスク並列性があるので，各々のMTの中に第2階層のマクロタスクグラフが生成される．このマクロタスクグラフを8コアのシステムにコード生成する場合，まず8コアを4コアずつの二つのグループに分割して第1階層のMTを割り当てる．MT1_1とMT1_2はループ並列処理あるいは近細粒度並列処理により4コアで処理される．MT1_3内部の第2階層マクロタスクグラフ中のMTは3コアでダイナミックスケジューリングにより並列処理される．PE7にはコンパイラが生成した集中スケジューラコードが生成され，MT1_3内部のMTのスケジューリングを行う．MT1_4が割り当てられたPE0-PE3はさらに2コアずつのグループに分割され，MT1_4内部の第2階層マクロタスクグラフのMTが割り当てられる．MT1_4内部の各MTの末尾には次に実行すべきMTをスケジューリングする分散スケジューラコードがコンパイラにより生成される．

図4の例で階層的なバリア同期に着目すると，まずMT1_4内部の第2階層を実行する際，PE0-PE1のバリア同期とPE2-PE3のバリア同期は独立して処理される必要がある．さらに，第2階層終了後再びPE0-PE3のグループで第1階層の並列処理を行う際も第1階層と第2階層のバリアを独立して処理しなければならない．なぜならば，例えばPE2-3が第2階層を先に終了し第2階層脱出のためのバリア同期成立を待っている際に，PE0-1が第2階層を実行中というような状況が存在するからである．このように，階層的粗粒度タスク並列処理を行う場合は，階層内のプロセッサコアグループ間バリア同期と階層間のバリア同期をそれぞれ独立に行えることが必要となる．

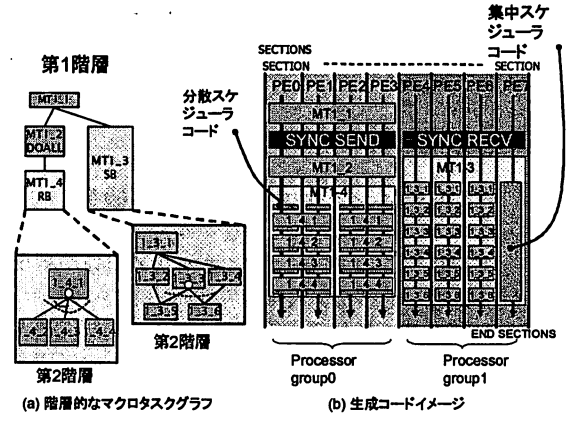


図4 階層的なコード生成イメージ

2.2. OSCARコンパイラにおけるバリア同期コード

OSCARコンパイラでは使用プロセッサコア数の増加に応じたスケーラブルな性能向上を目指しているため，バリア同期においてもプロセッサ数増加によらず低レイテンシで処理が行える必要がある．そのため，ロックを使用しないバージョンナンバを用いたバリア同期コードを生成する．図5にOSCARコンパイラが生成するバリア同期コードのイメージを示す．

図5では一つのプロセッサコアグループ内に3コアある様子を示している．図中vnは各コアが階層ごとに固有に持っているバージョンナンバ変数であり，プログラム実行時に0に初期化される．またf0, f1, f2は各コアが同期地点に達したことを示すフラグであり，全

コアで共有される．フラグ変数もプログラム開始時に0で初期化される．

PE1あるいはPE2がバリア同期地点に達すると，各自のvnとf1あるいはf2をインクリメントし，f0とvnを比較することでバリア同期成立を待つ．PE0は自分のvnとf1及びf2を比較し，PE1及びPE2がバリア同期地点に到達したかどうかをチェックする．バリア同期に参加する全てのコアがバリア同期地点に到達したら，f0をインクリメントし，各コアにバリア同期成立を通知する．ここで，バリア同期ごとに別々のvnの値と比較することになるので，vn及びフラグ変数のリセットが不要であることに着目されたい．

このように，OSCARコンパイラでは排他制御のようなオーバーヘッドの大きい処理を用いずにバリア同期を行うことができる．

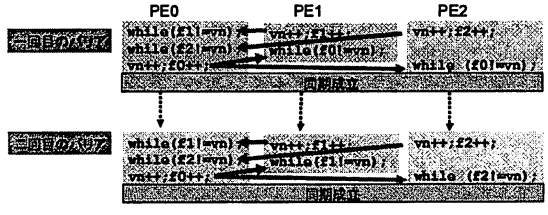


図5 OSCARコンパイラによるバリア同期コードのイメージ

3. 階層グルーピング対応バリア同期機構

本章では新規に開発した階層グルーピング対応バリア同期機構について説明する．本バリア同期機構は，2.2節で説明したバリア同期コードの一部をハードウェア化したものと見なすことができる．

3.1. バリア同期ハードウェアの概要

図6にバリア同期ハードウェアの概念図を示す．図ではPE0-PEnの各コアに1ビットの書き込み専用レジスタBWとnビットの読み出し専用レジスタBRがmセット設置されている．なお，図では1セット分の配線のみ示している．プロセッサが自コアのBWレジスタに書き込むと，その情報はブロードキャストされ全PEのBRレジスタの該当ビットに反映される．本稿では，各コアの持つBWレジスタのLSB側から順に，PE0, PE1, ...がBWレジスタに書き込んだ値を読めるものとする．このハードウェアを用いて前述の階層的粗粒度タスク並列処理におけるバリア同期機構を効率よく実現することができる．

本バリア同期機構のハードウェア量は$m \times n$に比例する．3.3節で述べるとおりレジスタセット数mはOSCARコンパイラのコード生成では$\log_2(n)$で十分である．

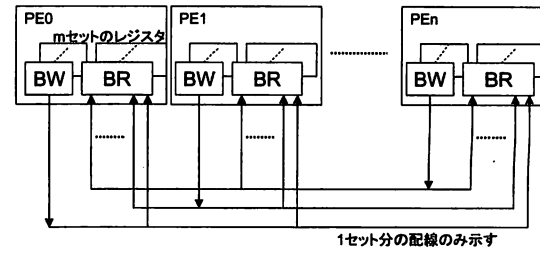


図7 階層的バリア同期ハードウェアの概要

3.2. 単階層におけるバリア同期

図8に図7で示したバリア同期ハードウェアを用いた場合の，図5と等価なバリア同期コードのイメージを示す．図中，BW, BRは図7中のBW, BRレジスタ，vn0は各PEがBWレジスタに設定するバージョンナンバ変数，vn1はバリア同期に参加する各PEがバリア同期地点に達したかどうかをPE0がチェックするためのバージョンナンバ変数，vn2はPE1以降のPEがPE0のバリア同期地点到達をチェックするためのバージョンナンバ変数をそれぞれ示す．また，0bxxxxは2進数の表記を示す．なお，初期状態では，vn0, vn1, vn2, BW, BRは全て0に設定されているものとする．

PE1あるいはPE2がバリア同期地点に達すると各自のvn0のLSBを反転し，その値をBWに書き込む．さらにvn2のPE0に該当するビットを反転し，これをBRの値と比較することでバリア同期成立を待つ．PE0はvn1とBRから読み出した値を比較することで，PE1及びPE2がバリア同期地点に到達したかどうかをチェックする．バリア同期に参加する全てのコアがバリア同期地点に到達したら，vn1のPE1及びPE2に対応するビットを反転する．さらにvn0のLSBを反転し，BWに書き込む．

各PEがバリア同期成立のチェックを行うために BRレジスタを読み込む際，適切にバリア同期に参加するPEに対応するビットをマスクするため，同一階層内の異なるグループのバリア同期が干渉することはない．

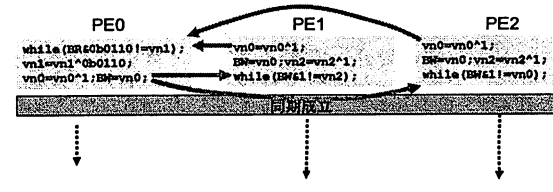


図8 バリア同期ハードウェアを使用した場合のコ

3.3. 複数階層におけるバリア同期

複数階層におけるバリア同期に対しては，階層ごと

に異なるバリア同期レジスタセットを用いることで対応する．このようにして，階層間でバリア同期が干渉することがなくなる．

必要なレジスタのセット数はプロセッサ数 n と同じ数だけあれば十分であるが，現在の OSCAR コンパイラの実装では，プロセッサを 2 の冪乗単位でグルーピングしていく．そのため，実用上は $\log_2(n)$ セットあれば十分である．

4. 情報家電用マルチコア RP2

本章では情報家電用マルチコア RP2 について説明する．RP2 は NEDO “リアルタイム情報家電用マルチコア技術の研究開発” プロジェクトにおけるマルチコアアーキテクチャ・API 委員会にて標準的マルチコアメモリアーキテクチャとして決定された早稲田大学 OSCAR アーキテクチャをベースに，ルネサステクノロジ・日立製作所・早稲田大学が共同開発したマルチコアである．

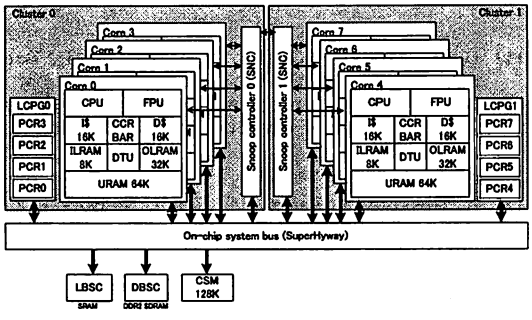


図 9　情報家電用マルチコア RP2

RP2 のブロック図を図 9 に示す．RP2 は 600MHz で動作する SH4A(SH-X3)コアを 8 コア搭載したマルチコアとなっており，SMP モード，AMP モード，およびそのハイブリッドモードでの利用が可能である．各コアは命令キャッシュおよびデータキャッシュを持ち，SMP モードではスヌープコントローラが専用のスヌープバスを介して 4 コア毎にデータキャッシュの一貫性を保証するため，5 コア以上使用時にはスヌープコントローラを無効にしてソフトウェアで一貫性の保証を行う．さらに各コアはローカルプログラムメモリ(ILRAM)，ローカルデータメモリ(OLRAM)，および分散共有メモリ(URAM)の 3 種のローカルメモリをもつ．集中共有メモリは on-chip のもの(CSM)と off-chip のもの(DDR2SDRAM)があり，各コアとスプリットトランザクションバス(SHwy)で接続されている．

バリア同期機構に関しては各コアが書き込み用と読み込み用の 32 ビットレジスタを 3 セット搭載している．読み込み用レジスタには LSB から PE0～PE7 の順に 1 ビットずつ各コア用の領域が割り当てられており，あるコアが書き込み用レジスタに書き込んだ値(1 or 0)は即座に全てのコアの読み込み用レジスタの対応する領域に反映される．

5. 性能評価

本章では RP2 上での階層グルーピング対応バリア同期機構の評価について述べる．

5.1. 評価環境

評価は 2.2 節で示したソフトウェアバリアと比較して行う．なお，RP2 におけるバリア同期機構の効果の最大値を計測するため，キャッシュを無効化してプログラムを実行した．ハードウェアバリアに用いる変数は OLRAM に配置し，それ以外のプログラム及びデータは DDR2SDRAM に配置した．OLRAM と DDR2SDRM のメモリアクセスレイテンシはそれぞれ 1 クロック，約 55 クロックである．

評価に用いたプログラムは以下の 3 つである．

- Barrier loop(Bloop)
  全 PE でのバリア同期だけをループで 5000 回繰り返すサンプルプログラム
- Barrierjacobi
  バリア同期の性能を測る Java のベンチマークプログラムを制約付き C に書き換えたもの．Jacobi 法で行列の固有値を求めるプログラムで，行列計算，バリア同期，収束判定，バリア同期，配列のスワップ，バリア同期の順に実行し，これをループで繰り返す．収束は起こらないように調整しており，処理の中核部分だけを評価対象とする．行列サイズは 32x32 と 64x64 の二種類で評価した
- AAC encoder
  音声圧縮アプリケーション．ルネサステクノロジ提供のアプリケーションであり，製品ミドルウェア仕様を並列性抽出が可能となるように制約付き C で参照実装したものとなっている．8 フレームごとにデータを読み込みフレーム間で並列処理をするもので，8 フレームのエンコードが終わるとバリア同期をとり，次の 8 フレームに移る．入力データは約 20[s]の音声データを使用する

5.2. 評価結果

Bloop，Barrierjacobi，AAC encoder の評価結果を図 10, 11, 12 にそれぞれ示す．各図の横軸は使用コア数，縦軸は実行時間をそれぞれ示す．またソフトウェアバリアを SW バリア，開発したバリア同期機構によるバ

リアをHWバリアとしてそれぞれ示す．

図10をみると，2コアを使用したときにHWバリアがSWバリアの15%の時間で処理していることがわかる．さらに，SWバリアではコア数の増加と共に実行時間が増加しているのに対し，HWバリアはほぼ一定の時間で処理が完了している．このことより，開発したバリア同期機構の低いオーバーヘッドと高いスケーラビリティがわかる．図11においてもHWバリアはSWバリアに対して高い性能を得ており，行列サイズ32x32の問題に対してもSWバリアに対して32%の性能向上を得ている．図12のAACエンコーダプログラムは並列処理しているタスクの粒度に対してバリア同期の比率が比較的小さいアプリケーションであるが，このような実用的なアプリケーションに対してもHWバリアは良い性能を得ており，8コア使用時にSWバリアに対して16%の性能向上を得ることができた．

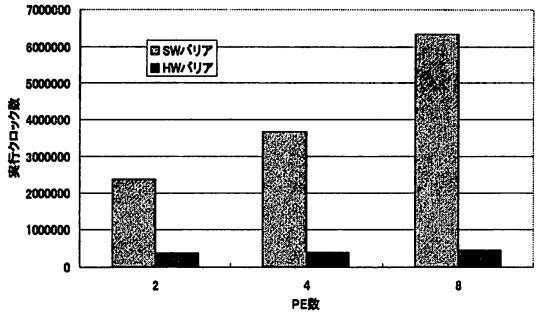


図10 Bloopの性能評価

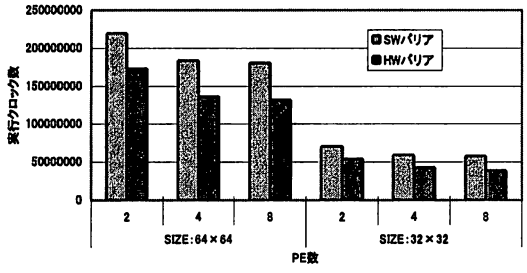


図11 Barrierjacobiの性能評価

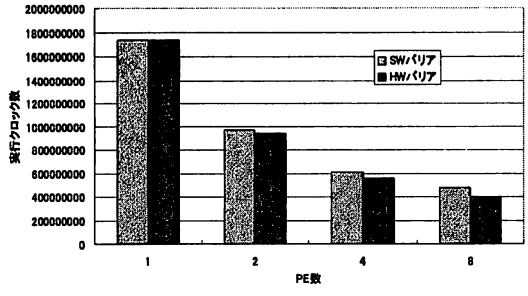


図11 AACエンコーダの性能評価

6. まとめ

本稿ではOSCAR自動並列化コンパイラの主要要素技術である粗粒度タスク並列処理と協調動作可能な，軽量かつスケーラブルなバリア同期機構を提案すると共にRP2上で評価を行った．本同期機構を実装した情報家電用マルチコアプロセッサRP2により評価を行った結果，ソフトウェアのみによるバリア同期に対して，8コア使用時に16%の性能向上を得ることができた．キャッシュを有効化した場合のRP2によるバリア同期機構の評価，シミュレータ等によるメニーコアを想定したバリア同期機構の評価などが今後の課題としてあげられる．

謝辞

本研究の一部はNEDO“リアルタイム情報家電用マルチコア技術”の支援により行われた．

文　献

[1] D. Wentzlaff, P. Griffin, H. Hoffmann, L. Bao, B. Edwards, C. Ramey, M. Mattina, C.-C. Miao, J. F. Brown III, and A. Agarwal, “On-Chip Interconnection Architecture of the Tile Processor”, IEEE MICRO, vol. 27, no 5, pp.15-31, September/October 2007.

[2] A. Hoskote, S. Vangal, A. Singh, N. Borkar, and S. Borkar, “A 5-GHz Mesh Interconnect for a Teraflops Processor”, IEEE MICRO, vol. 27, no 5, pp.51-61, September/October 2007.

[3] H. Kasahara, H. Honda, A. Mogi, A. Ogura, K. Fujiwara, and S. Narita, “A Multi-grain Parallelizing Compilation Scheme for OSCAR (Optimally Scheduled Advanced Multiprocessor)”, 4th Intl. Workshop on LCPC, pp. 283-297, August, 1991.

[4] M. Obata, J. Shirako, H. Kaminaga, K. Ishizaka, and H. Kasahara, “Hierarchical Parallelism Control for Multigrain Parallel Processing”, 15th Intl. Workshop on LCPC, August, 2002.

[5] D. E. Culler, J. P. Singh, and A. Gupta, “Parallel Computer Architecture – A Hardware/Software Approach”, Morgan Kaufmann Pub., 1999.

[6] C. J. Beckmann, and C. D. Polychronopoulos, “Fast Barrier Synchronization Hardware”, Prof. of Supercomputing ’90, pp. 180-189, November, 1990.

[7] M. Ito, T. Hattori, Y. Yoshida, K. Hayashi, T. Hayashi, O. Nishii, Y. Yasu, A. Hasegawa, M. Takada, M. Ito, H. Mizuno, K. Uchiyama, T. Odaka, J. Shirako, M. Mase, K. Kimura, H. Kasahara, “An 8640 MIPS SoC with Independent Power-off Control of 8 CPUs and 8 RAMs by an Automatic Parallelizing Compiler”, ISSCC2008, February, 2008.