社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

情報家電用マルチコアSMP実行モードにおけるマルチグレイン並列処理

間瀬　正啓† 馬場　大介† 長山　晴美† 田野　裕秋† 益浦　健†
宮本　孝道† 白子　準† 中野　啓史† 木村　啓二† 亀井　達也††
服部　俊洋†† 長谷川　淳†† 伊藤　雅樹††† 佐藤　真琴††† 内山　邦男†††
小高　俊彦††† 笠原　博徳†

† 早稲田大学 情報理工学科
†† 株式会社 ルネサステクノロジ
††† 株式会社 日立製作所
E-mail: {mase,kimura,kasahara}@kasahara.cs.waseda.ac.jp

あらまし　現在，ゲーム，カーナビゲーションシステム，デジタル TV，携帯電話等の情報家電機器を始め，PC からスーパーコンピュータに至る，多くの情報機器でマルチコアプロセッサ採用の動きが進んでいる．本稿では，制約付き C 言語で記述されたメディア処理等のプログラムを OSCAR マルチグレイン自動並列化コンパイラにより並列化し，NEDO“リアルタイム情報家電用マルチコア技術の研究開発” プロジェクトの一環で OSCAR 標準マルチコアメモリアーキテクチャに基づき株式会社ルネサステクノロジ，株式会社日立製作所により開発された SH-4A(SH-X3) コアを 4 コア集積した情報家電用マルチコアプロセッサ RP1 上で SMP モード実行時の性能評価を行った．評価の結果，AAC オーディオエンコーダで 4 コア使用時に 1 コア使用時の 3.34 倍の速度向上が得られた．
キーワード　マルチコア，マルチグレイン並列処理，自動並列化，コンパイラ，情報家電

Multigrain Parallel Processing in SMP Execution Mode on a Multicore for Consumer Electronics

Masayoshi MASE†, Daisuke BABA†, Harumi NAGAYAMA†, Hiroaki TANO†, Takeshi MASUURA†, Takamichi MIYAMOTO†, Jun SHIRAKO†, Hirofumi NAKANO†, Keiji KIMURA†, Tatsuya KAMEI††, Toshihiro HATTORI††, Atsushi HASEGAWA††, Masaki ITO†††, Makoto SATO†††, Kunio UCHIYAMA†††, Toshihiko ODAKA†††, and Hironori KASAHARA†

† Department of Computer Science, Waseda University
†† Renesas Technology Corp.
††† Hitachi, Ltd.
E-mail: {mase,kimura,kasahara}@kasahara.cs.waseda.ac.jp

Abstract　Currently, multicore processors are becoming ubiquitous in various computing domains, namely consumer electronics such as games, car navigation systems and mobile phones, PCs, and supercomputers. This paper describes parallelization of media processing programs written in restricted C language by OSCAR multigrain parallelizing compiler and SMP processing performance on RP1 4-core SH-4A (SH-X3) multicore processor developed by Renesas Technology Corp. and Hitachi, Ltd. based on standard OSCAR multicore memory architecture as a part of NEDO “Research and Development of Multicore Technology for Real Time Consumer Electronics Project”. Performance evaluation shows OSCAR compiler achieved 3.34 times speedup using 4 cores against using 1 core for AAC audio encoder.
Key words　Multicore, Multigrain parallel processing, Automatic parallelization, Compiler, Consumer electronics

1. はじめに

半導体集積度向上に伴うスケーラブルな性能向上，低消費電力，高価格性能比を達成するためにマルチコアプロセッサが大きな注目を集めており，ゲーム，カーナビゲーションシステム，デジタル TV，携帯電話等の情報家電機器を始め，PC からスーパーコンピュータに至る，多くの情報機器でマルチコアプロセッサ採用の動きが進んでいる．情報家電用のマルチコアプロセッサとしては，ソニー／ IBM ／東芝の Cell [1]，NEC エレクトロニクス／ ARM の MPCore [2]，富士通の FR-V [3] 等が挙げられる．一方で，このようなマルチコアプロセッサにおける並列化プログラミングは一般に難易度が高く，アプリケーションとハードウェア構成の両方を熟知している必要がある．さらに，製品開発サイクルの短い情報家電分野において国際競争力を維持するためには，従来のスーパーコンピュータ用並列アプリケーション作成のように並列化及びチューニングに数ヵ月以上をかけるわけにはいかず，質の高いアプリケーションを短期間のうちに多数開発していくことが必須である．そのため，単にハードウェアの性能だけでなく，アプリケーション生産性，すなわち開発ツールを含めたソフトウェア開発の容易性が重要となっており，並列化プログラミングの煩わしさを感じさせずに，高性能，低消費電力を達成できるコンパイラの開発が求められている [4].

早稲田大学の OSCAR マルチグレイン自動並列化コンパイラ [5]〜[8] では従来からのループレベル並列処理 [9]〜[11] に加え粗粒度タスク並列処理，近細粒度並列処理を組み合わせたマルチグレイン並列処理を実現しており，さらにプロセッサ近傍のキャッシュあるいはローカルメモリを有効利用しメモリウォール問題に対応するためのデータの自動分割配置 [12], [13]，サイズの小さいローカルメモリと共有メモリ間のデータ転送の DMA コントローラを用いたタスク処理とオーバーラップによるデータ転送オーバーヘッドの隠蔽 [14], [15]，そしてプロセッサの電圧・クロック周波数制御，あるいは電源シャットダウンの制御 [16] に関する研究開発が行われている．

また OSCAR コンパイラと同時に，その最適化機能を最大限サポートする，図 1 に示す OSCAR マルチコアアーキテクチャ [17] の提案も行われている．OSCAR マルチコアアーキテクチャは各コアが持つローカルデータメモリ (LDM)，分散共有メモリ (DSM)，及び各コアが共有するオンチップおよびオフチップの集中共有メモリ (CSM) を持ち，コンパイラがこれらのメモリを適切に使い分けることにより，スタティック及びダイナミックスケジューリング時の両方において効率の良い並列処理を行うことを特徴とする．この OSCAR マルチコアメモリアーキテクチャとそれに消費電力制御機能 [16] を加えたものは，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 検討委員会 (早稲田大学，株式会社日立製作所，株式会社ルネサステクノロジ，富士通株式会社，株式会社東芝，松下電器産業株式会社，日本電気株式会社) において，標準マルチコアアーキテクチャに選定されており，プロジェクト助成事業の一環として株式会社ルネサステクノロジと株式会社日立製作所がこの標準アーキテクチャに基づくマルチコアプロセッサ RP1 [18] とその評価ボードの開発を行った．本マルチコアプロセッサは OSCAR 型のメモリ構成に加え，コヒーレントキャッシュメモリも持っており，SMP マルチコアプロセッサとしても利用可能である．また，上述の委員会では情報家電用マルチコア用の標準的な並列化 API(スレッド生成，同期，排他制御，変数のメモリ上への配置，DMA 転送，消費電力制御) を策定しており，本マルチコアプロセッサ用の API 解釈コンパイラもマルチコアプロセッサと併せて開発された．

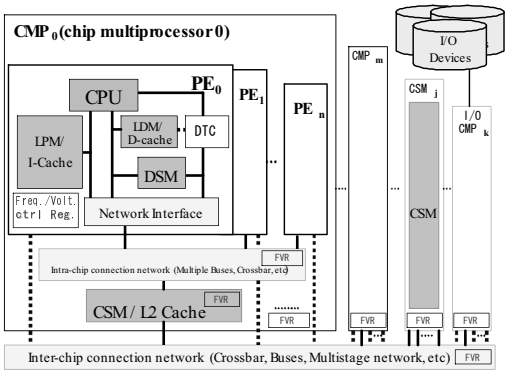


図 1　OSCAR マルチコアアーキテクチャ
Fig. 1　OSCAR multicore architecture

本稿では，制約付き C 言語 [19] で記述されたプログラムを OSCAR コンパイラで並列化し，情報家電用マルチコアプロセッサ RP1 の SMP 実行モードにおける性能評価を行った．

本稿の構成を以下に示す．まず 2. で本稿で評価を行った情報家電用マルチコアプロセッサ RP1 について述べ，3. で OSCAR コンパイラの基盤技術であるマルチグレイン並列処理について，4. で自動並列化のための制約付き C 言語について，5. で性能評価について述べる．最後に 6. で本稿のまとめを述べる．

2. 情報家電用マルチコアプロセッサ

本章では本稿で評価を行った情報家電用マルチコアプロセッサ RP1 について述べる．本マルチコアプロセッサは NEDO “リアルタイム情報家電用マルチコア技術の研究開発” プロジェクトの一環で開発された．本プロジェクトでは情報家電用マルチコア用の標準的な並列化 API の策定を進めており，その API が想定している標準マルチコアアーキテクチャに基づき株式会社ルネサステクノロジが試作チップを，株式会社日立製作所が評価ボード及び API 解釈コンパイラの開発を行った．

2.1　マルチコアアーキテクチャ

本マルチコアプロセッサは図 2 に示すように，SH-4A(SH-X3) コアを 4 コア搭載したホモジニアスマルチコアとなっており，4 コアがそれぞれ独立して周波数制御が可能である．また，従来の共有メモリモデルによるプログラミングが容易な SMP モード，リアルタイム制約が保証しやすい 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

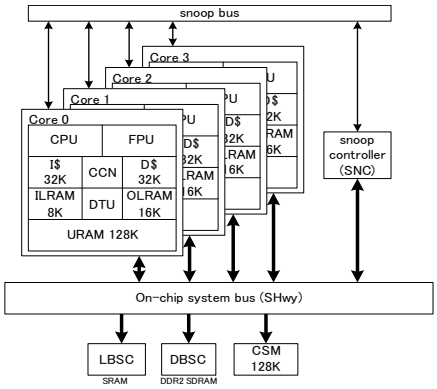


図 2　情報家電用マルチコアプロセッサ RP1

Fig. 2　multicore processor for consumer electronics

表 1　周波数設定およびメモリサイズ

Table 1　frequency configuration and memory size

CPU 周波数	600MHz
システムバス周波数	300MHz
命令キャッシュ	32KB (4 way set-associative)
データキャッシュ	32KB (4 way set-associative)
ILRAM	8KB
OLRAM	16KB
URAM	128KB
CSM	128KB / chip

して各コアのデータキャッシュの一貫性を保証する．さらに各コアは 1 クロックアクセスが可能な命令用のローカルメモリ ILRAM，データ用のローカルメモリ OLRAM，1〜数クロックアクセスで比較的大容量の URAM の 3 種類のローカルメモリを持ち，チップ内に各コアが共有する集中共有メモリ (CSM) を搭載する．これらのメモリをユーザプログラムにおいて後述の API を用いて明示的に使用し，DTU(Data Transfer Unit) を用いてプログラム実行とデータ転送をオーバーラップすることにより，ストリームデータ等の高速処理や，リアルタイム制約を満たすアプリケーションの作成が可能である．本マルチコアプロセッサを OSCAR 型アーキテクチャとして利用する際にはそれぞれ OLRAM を LDM(ローカルデータメモリ)，URAM を DSM(分散共有メモリ)，CSM を on-chip CSM(集中共有メモリ) として扱っている．本稿の評価における周波数設定および各種メモリサイズを表 1 に示す．

2.2　情報家電用マルチコア並列化 API

情報家電用マルチコア並列化 API は OpenMP API [20] のような形式のディレクティブ (C と FORTRAN に対応) により記述され，OpenMP API のサブセットおよび独自に定義したディレクティブにより構成される．OSCAR 自動並列化コンパイラを用いて並列プログラムを出力することにより，逐次コンパイラの前に API 解釈部をつけるだけの簡単な作業により，各社のマルチコア上での並列処理が可能となる．具体的には，スレッド生成，同期，排他制御は OpenMP 互換となっており，スレッド生成に parallel sections，メモリアクセス順を保証する flush，排他制御を行うための critical を用いる．これら 3 つ

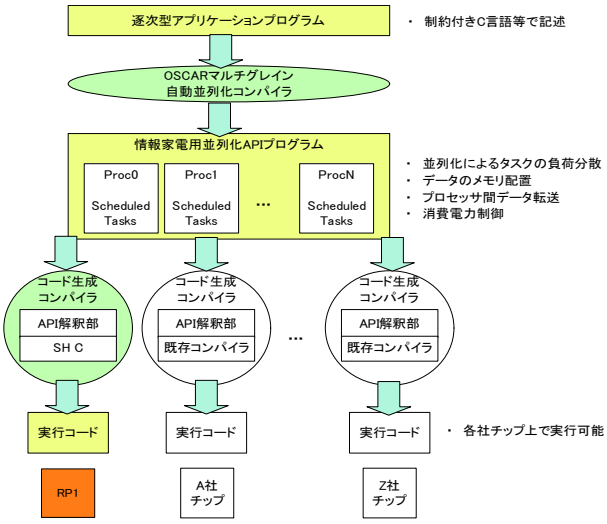


図 3　コンパイルの流れ

Fig. 3　flow of compilation

がサポートされている環境であれば，SMP モードにおけるマルチグレイン並列処理を簡単に実現できる．独自定義のディレクティブとしては，データのメモリ配置，DMAC によるデータ転送，周波数・電源制御用の API の策定を行っている．これらの情報家電用ディレクティブの評価は，今後 OSCAR コンパイラを用いて各社のマルチコア上で行われる予定である．

2.3　開 発 環 境

本マルチコアプロセッサ用の開発環境として，SH プロセッサ用ネイティブコンパイラである通常の逐次 SH C コンパイラに情報家電用マルチコア並列化 API の解釈部を追加したコード生成コンパイラが開発されており，OSCAR 自動並列化コンパイラが出力する情報家電用マルチコア並列化 API C プログラムをコンパイルすることが可能である．OSCAR 自動並列化コンパイラおよび並列化 API を用いたプログラムのコンパイルの流れを図 3 に示す．まず，制約付き C 言語等で記述された逐次型アプリケーションプログラムを OSCAR 自動並列化コンパイラでコンパイルし，情報家電用マルチコア並列化 API プログラムを出力する．次に，この並列化 API プログラムを API 解釈コンパイラでコンパイルし並列化マシンコードを生成する．この並列化マシンコードをマルチコアプロセッサ上で実行する．

3.　マルチグレイン並列処理

本章では，OSCAR コンパイラで実現されているマルチグレイン並列処理について述べる．マルチグレイン並列処理は粗粒度タスク並列性，ループ並列性，近細粒度並列性を組み合わせ，プログラム全域から並列性を抽出する技術である．本稿では粗粒度タスク並列性とループ並列性を用いたマルチグレイン並列処理を行う．

3.1　粗粒度タスク生成

粗粒度タスク並列処理では，プログラムは基本ブロックまたはその融合ブロックで構成される疑似代入文ブロック BPA [7]，DO ループや後方分岐により生じるナチュラルループで構成される繰り返しブロック RB [7]，サブルーチンブロック SB [7] の

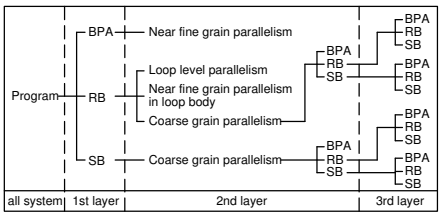
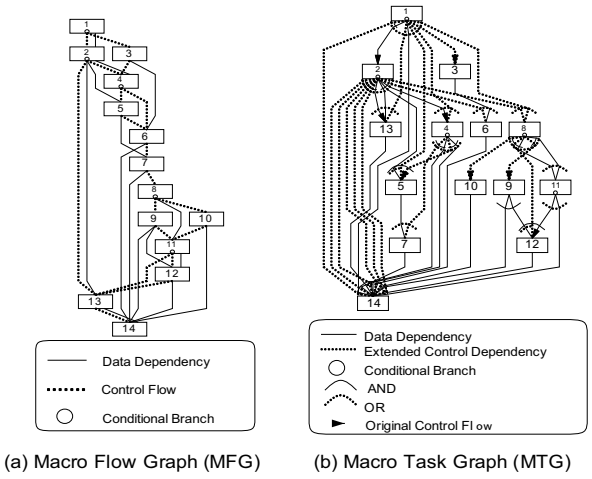


図 4　階層的マクロタスク定義

Fig. 4　hierarchical macro task difinition



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

図 5　マクロフローグラフとマクロタスクグラフ

Fig. 5　macro flow graph and macro task graph

3 種類の粗粒度タスク (マクロタスク MT [7]) に分割される．繰り返しブロック RB やサブルーチンブロック SB は図 4 に示すようにその内部をさらにマクロタスクに分割し階層的なマクロタスク構造を生成する．

3.2　粗粒度タスク並列性抽出

マクロタスク生成後，各階層においてマクロタスク間のデータ依存と制御フローを解析し，マクロタスク間のデータと制御のフローを表すマクロフローグラフ [5], [7] を生成する．

次に，階層的に生成されたマクロフローグラフに対し最早実行可能条件解析 [5], [7] を適用し，階層的なマクロタスクグラフ MTG [5], [7] を生成する．最早実行可能条件とは，制御依存とデータ依存を考慮したマクロタスクの最も早く実行を開始してよい条件であり，マクロタスクグラフは粗粒度タスク並列性を表す．マクロフローグラフ及びマクロタスクグラフの例を図 5 に示す．

3.3　プロセッサグループへのマクロタスク割り当て

コンパイラはマクロタスクを各プロセッサエレメント PE [21] あるいは PE を複数集めたプロセッサグループ PG [21] に割り当てる．マクロタスクグラフ上に条件分岐が無い場合はコンパイル時に静的にスケジューリングが行われ，各プロセッサグループの処理するマクロタスクが決定される．マクロタスクグラフが条件分岐等の実行時不確定性を含む場合は実行時にスケジューリングを行なうダイナミックスケジューリングルーチンをコンパイラが自動生成し，実行時にマクロタスクを PE あるいは PG に割り当てる．図 6 に示すように各マクロタスクは階層的にスタティックスケジューリングあるいはダイナミックスケジューリングされる．生成されたスタティックスケジューリングコード及び実行時スケジューラはユーザコードであり，OS のシステムコールによるスケジューラに比べ極めて低オーバーヘッドなスケジューリングが可能である．

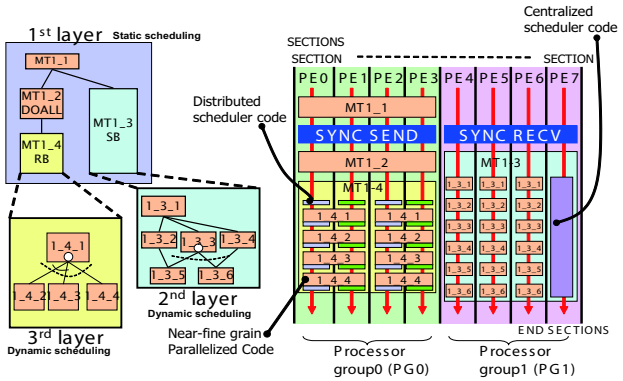


図 6　階層的コード生成イメージ

Fig. 6　hierarchical code generation image

3.4　データローカライゼーション

プロセッサとメモリの速度差の拡大によりキャッシュメモリやローカルメモリを有効利用することがマルチプロセッサシステムの性能向上にとって重要となっている．OSCAR コンパイラでは並列性とデータローカリティの両方を考慮したデータローカライゼーション手法 [12] により複数粗粒度タスク間でキャッシュあるいはローカルメモリ上のデータを効果的に用いる．

データローカライゼーション手法では，まず複数ループ間のデータ依存を解析し，データ依存する分割後の小ループ間におけるデータ授受がキャッシュあるいはローカルメモリを介して行われるようにそれらのループを整合して分割するループ整合分割 [22] を行う．分割されたループのうち同一データにアクセスする複数のマクロタスクは，データローカライザブルグループ（DLG）と呼ぶタスク集合にグループ化される．図 7 にループ整合分割を適用したマクロタスクグラフを示す．図中 (b) の同じ網掛けで塗られたマクロタスクが DLG に属するマクロタスクである．

整合分割後の粗粒度タスクスケジューリングでは，粗粒度タスク間の並列性を考慮しながら，同一 DLG に属するマクロタスクが可能な限り同一プロセッサ上で連続的に実行されるようにスケジューリングを行う．このようにループ分割と DLG 内タスクの連続実行を組み合わせることにより，複数のループに渡り再利用することを可能とすることでメインメモリアクセスを削減し，タスク間のデータ授受をキャッシュあるいはローカルメモリを用いて高速に行うことが可能となる．

4.　自動並列化のための制約付き C 言語

本章では OSCAR コンパイラで自動並列化を行うための言語記述の制約について述べる．OSCAR コンパイラはこれまで FORTRAN77 [23] を対象に開発を進めてきたため，情報家電用のソフトウェア開発で広く用いられている C 言語へ迅速に対

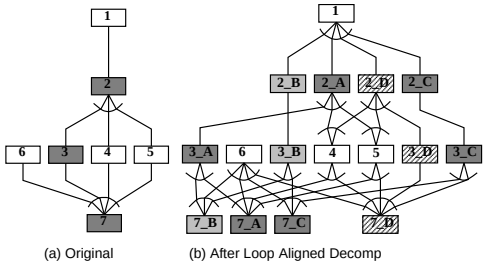


図 7　データローカライゼーションにおけるループ整合分割
Fig. 7　loop aligned decomposition for data localization

応するにあたり，まずは FORTRAN77 のレベルまで言語仕様を制限した制約付き C 言語を規定し，これを並列化することから C 言語対応を開始した．この制約は今後緩和していく計画であるが，現時点での制約を以下に示す．

4.1　分割コンパイル

OSCAR コンパイラはプログラム全域からの並列性，データローカリティの抽出を行うためコンパイル時に全てのユーザプログラムを一度にコンパイルする必要がある．ライブラリ関数の使用については，数学ライブラリ等の内部状態を持たない標準ライブラリ関数を除き，ライブラリ関数を含む部分の並列化は行わない．

4.2　関数の再帰呼び出し

関数の再帰呼出しは行わない．

4.3　ポインタ・構造体

ポインタ・構造体は原則的に使用しない．ヒープについても可能な限り単純な多次元配列を用いて代替する．ただし，後述する関数のポインタ引数については例外とする．その他のポインタ・構造体アクセスについては，全てのメモリ領域に対してアクセスする可能性があるものとして扱う．

4.4　関数のポインタ引数

配列を実引数として関数呼出しを行う場合，C 言語の言語仕様では仮引数はポインタとなり，FORTRAN77 のように実引数と仮引数を静的にエイリアスすることができない．そこで，ポインタ引数については値の再代入は行わない，C99 [24] の restrict 修飾子と同様に複数のポインタ引数を用いた参照先が重ならない，という制約をつけている．この仮定により，コンパイラではポインタ仮引数を FORTRAN77 における参照渡しによる関数呼出しの仮引数と同様に扱うことができる．

5.　性 能 評 価

本章では OSCAR マルチグレイン自動並列化コンパイラで並列化した制約付き C プログラムの情報家電用マルチコア RP1 における SMP 実行モードの性能評価結果について述べる．

5.1　対象アプリケーション

音声圧縮プログラムである AAC エンコーダ，組み込み向けベンチマーク MiBench [25] より susan (smoothing)，SPEC2000 より art を用いて評価を行った．

AAC エンコーダは株式会社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 言語で参照実装したものとなっている．入力データはサンプリングレート 44.1kHz のステレオ PCM データ，出力データのビットレートは 128kbps とした．

susan は組み込み向けベンチマーク MiBench に収録されている “susan” を制約付き C 言語仕様を満たすように修正したプログラムを用いた．susan は画像認識用のアプリケーションであり，シャープネス，画像の粗さ変更を行う smoothing, エッジ加工を施す edges, 画像の角を丸くする corners の 3 種類のモードがあるが，本稿の評価では特に大きな並列性のある smoothing について評価を行った．

art は SPEC2000 に収録されている，“179.art” を制約付き C 言語仕様を満たすように修正したプログラムを用いた．art はニューラルネットワークを用いた画像認識アプリケーションである．トレーニングデータを用いて学習した後，スキャンデータの中からトレーニングデータと一致する部分を探索する．データサイズは ref を用いた．

5.2　評 価 手 順

対象のプログラムを OSCAR コンパイラで並列化し，OpenMP C バックエンドを用いて出力した OpenMP C プログラムを，本マルチコアのネイティブコンパイラである API 解釈機能を追加した SH C コンパイラを用いてコンパイルし，SH 用統合開発環境であるルネサステクノロジ HEW(High-performance Embedded Workshop) を用いて実行バイナリを作成した．情報家電用マルチコア評価ボード上で実行する際は，エミュレータで接続し，統合開発環境 HEW よりプログラム実行に要するクロックサイクル数を測定した．入出力については，主メモリ上の特定の領域を入出力用の領域とし，参照元プログラムでファイル I/O となっている部分を当該領域へのメモリコピーに置き換えて評価を行った．なお，OS には各プロセッサコア用のスレッド生成をサポートする簡易 OS を用い，OS に関するその他の機能は利用していない．

5.3　評 価 結 果

図 8 に OSCAR コンパイラで並列化したプログラムの使用したコア数に対する速度向上率を示す．図中，横軸が評価を行ったアプリケーションおよび使用したコア数を示し，縦軸が 1 コア使用時に対する速度向上率を示す．

OSCAR コンパイラによる並列処理の速度向上率は，1 コア使用時と比較して，AAC エンコーダで 2 コア使用時に 1.84 倍，3 コア使用時に 2.62 倍，4 コア使用時に 3.34 倍の速度向上が得られた．susan (smoothing) では 2 コア使用時に 1.91 倍，3 コア使用時に 2.95 倍，4 コア使用時に 3.82 倍の速度向上，art では 2 コア使用時に 1.84 倍，3 コア使用時に 2.53 倍，4 コア使用時に 2.90 倍の速度向上が得られており，初期評価データではあるが，使用するコア数に応じた速度向上が得られている．

AAC エンコーダについては 4 コア使用時に 3.34 倍と大きな速度向上，susan (smoothing) についても 4 コア使用時に 3.82 倍と非常に大きな速度向上を得ることができた．これらのアプリケーションは演算処理の大部分を一つの doall ループが占めており，OSCAR コンパイラでこのループを doall ループと判定できたためこのような速度向上が得られた．

art については 4 コア使用時に 2.90 倍の速度向上が得られ

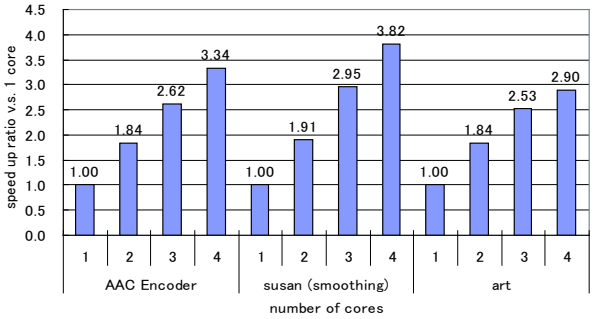


図 8　情報家電用マルチコア SMP モードの速度向上率
Fig. 8　speedup ratio in SMP mode on RP1

た．art は主要演算ループ内に逐次処理部を含むため，プログラムの持つ並列性を最大限抽出できた結果と考えられる．

6. ま と め

本稿では OSCAR マルチグレイン自動並列化コンパイラで並列化したプログラムの NEDO“リアルタイム情報家電用マルチコア技術の研究開発” プロジェクトで開発した情報家電用マルチコアプロセッサ RP1 の SMP 実行モードにおける並列処理性能について述べた．制約付き C 言語で記述されたメディア処理等のプログラムを OSCAR コンパイラで並列化し，OpenMP C バックエンドを用いて出力した並列コードを，API 解釈機能を追加した逐次 SH C コンパイラを用いてコンパイルし，SH-4A(SH-X3) コアを 4 コア集積した情報家電用マルチコアプロセッサ RP1 上で実行した．その結果，使用するコア数に応じた速度向上が得られ，4 コア使用時に 1 コア使用時と比較して，AAC エンコーダで 3.34 倍，susan (smoothing) で 3.82 倍，art で 2.90 倍の速度向上が得られた．これより，OpenMP 互換の情報家電用マルチコア SMP 用並列化 API を用いた，OSCAR コンパイラと情報家電用マルチコアプロセッサ RP1 の SMP 実行モードにおける協調動作の有効性が示された．

今後の課題としては OSCAR コンパイラによるローカルメモリ管理機能 [13] を利用した，データのメモリ配置 API，DMAC によるデータ転送 API を用いた AMP モードでの評価，OSCAR コンパイラによる低消費電力化制御機能 [16] を利用した，周波数・電源制御 API を用いた電力性能評価等が挙げられる．

7. 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研究開発”, 及び STARC(半導体理工学研究センター)“並列化コンパイラ協調型チップマルチプロセッサ技術” の支援により行われた．また，本稿で性能評価に用いた AAC エンコーダプログラムをご提供いただきました株式会社ルネサステクノロジの皆様に感謝申し上げます．

文　　献

[1] D. Pham, et al.: “The design and implementation of a first-generation cell processor”, 2005 IEEE International Solid-State Circuits Conference (2005).
[2] “ARM11 MPCore Processor Technical Reference Manual” (2005).
[3] T. Shiota, et al.: “A 51.2 gops 1.0 gb/s-dma single-chip multi-processor integrating quadruple 8-way vliw processors”, 2005 IEEE International Solid-State Circuits Conference (2005).
[4] 笠原：“並列化コンパイラ協調型低消費電力・高実効性能マルチコアプロセッサの動向”, 情報処理学会　第 158 回 計算機アーキテクチャ研究会 (2006).
[5] 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手法”, 電子情報通信学会論文誌, **J73-D-1**, 12, pp. 951–960 (1990).
[6] H.Kasahara and et al: “A multi-grain parallizing compilation scheme on oscar”, Proc. 4th Workshop on Language and Compilers for Parallel Computing (1991).
[7] 笠原：“最先端の自動並列化コンパイラ技術”, 情報処理, Vol.44 No. 4(通巻 458 号), pp.384-392 (2003).
[8] 岡本, 小幡, 松崎, 笠原, 成田：“マルチグレイン並列化 fotran コンパイラ”, 情報処理学会論文誌, **40**, 12, pp. 4296–4308 (1999).
[9] M.Wolfe: “High performance compilers for parallel computing”, Addison-Wesley Publishing Company (1996).
[10] R. Eigenmann, J. Hoeflinger and D. Padua: “On the automatic parallelization of the perfect benchmarks”, IEEE Trans. on parallel and distributed systems, **9**, 1 (1998).
[11] M. W. Hall, J. M. Anderson, S. P. Amarasinghe, B. R. Murphy, S. Liao, E. Bugnion and M. S. Lam: “Maximizing multiprocessor performance with the suif compiler”, IEEE Computer (1996).
[12] 石坂, 中野, 八木, 小幡, 笠原：“共有メモリマルチプロセッサ上でのキャッシュ最適化を考慮した粗粒度タスク並列処理”, 情報処理学会論文誌, **43**, 4 (2002).
[13] 三浦, 田川, 村松, 池見, 中川, 中野, 白子, 木村, 笠原：“マルチグレイン並列化コンパイラにおけるローカルメモリ管理手法”, 情報処理学会研究会報告 2007-ARC-172/HPC-109-11 (2007).
[14] 小高, 中野, 木村, 笠原：“チップマルチプロセッサ上での mpeg2 エンコードの並列処理”, 情報処理学会論文誌, **46**, 9 (2005).
[15] 宮本, 中川, 浅野, 内藤, 仁藤, 中野, 木村, 笠原：“マルチコアプロセッサ上での粗粒度タスク並列処理におけるデータ転送オーバラップ方式”, 第 159 回計算機アーキテクチャ・第 105 回ハイパフォーマンスコンピューティング合同研究発表会 (2006).
[16] 白子, 吉田, 押山, 和田, 中野, 鹿野, 木村, 笠原：“マルチコアプロセッサにおけるコンパイラ制御低消費電力化手法”, 情報処理学会論文誌, **47**, ACS15 (2006).
[17] 木村, 加藤, 笠原：“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”, 情報処理学会論文誌, **42**, 4 (2001).
[18] Y. Yoshida, T. Kamei, K. Hayase, S. Shiratori, O. Nishii, T. Hattori, A. Hasegawa, M. Takada, N. Irie, K. Uchiyama, T. Odaka, K. Takada, K. Kimura and H. Kasahara: “A 4320mips four-processor core smp/amp with individually managed clock frequency for low power consumption”, 2007 IEEE International Solid-State Circuits Conference (2007).
[19] 間瀬, 馬場, 長山, 田野, 益浦, 深津, 宮本, 白子, 中野, 木村, 笠原：“Oscar コンパイラにおける制約付き c プログラムの自動並列化”, 情報処理学会研究会報告 2006-ARC-170-01 (2006).
[20] “OpenMP Application Program Interface Version 2.5” (2005).
[21] 小幡, 白子, 神長, 石坂, 笠原：“マルチグレイン並列処理のための階層的並列処理制御手法”, 情報処理学会論文誌, **44**, 4 (2003).
[22] 吉田, 前田, 尾形, 笠原：“Fortran マクロデータフロー処理におけるデータローカライゼーション手法”, 情報処理学会論文誌, **35**, 9, pp. 1848–1994 (1994).
[23] “ISO 1539-1980 - Programming Language Fortran” (1980).
[24] “ISO/IEC 9899:1999 - Programming Language C” (1999).
[25] M. R. Guthaus, J. S. Ringenberg, D. Ernst, T. M. Austin, T. Mudge and R. B. Brown: “Mibench: A free, commercially representative embedded benchmark suite”, IEEE 4th Annual Workshop on Workload Characterization (2001).