情報家電用マルチコアSMP実行モードにおける制約付きCプログラムのマルチグレイン並列化

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[†] 笠原博徳[†]

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においてマルチコアプロセッサ採用の動きが進んでおり，多数の上質な並列アプリケーションプログラムを迅速に提供するために，高性能な自動並列化コンパイラの開発が求められている．本論文では，ルネサステクノロジ/日立製作所/早稲田大学により新たに開発された SH-X3 コアを 4 コア集積した情報家電用低電力マルチコアプロセッサ RP1 の SMP 実行モードにおいて，OSCAR コンパイラによるマルチグレイン並列化を適用した際の性能評価を行う．OSCAR コンパイラは制約付き C 言語で記述された逐次プログラムを自動並列化し，新たに提案された情報家電用マルチコア API で記述された並列プログラムを生成する．この API は各社マルチコアで標準的に利用可能とすることを目指している．性能評価の結果，AAC エンコーダ，MP3 エンコーダ，MPEG2 エンコーダ，MiBench より susan (smoothing)，SPEC2000 より art について OSCAR コンパイラによる自動並列化を適用することで 4 コア使用時に 1 コアによる逐次実行時と比較してそれぞれ 3.43 倍，3.15 倍，3.53 倍，3.66 倍，2.54 倍の速度向上が得られ，情報家電用マルチコアにおけるマルチグレイン並列化の有効性が確認された．

Multigrain Parallelization of Restricted C Programs in SMP Execution Mode of a Multicore for Consumer Electronics

Masayoshi Mase,[†] Daisuke Baba,[†] Harumi Nagayama,[†]
Hiroaki Tano,[†] Takeshi Masuura,[†] Takamichi Miyamoto,[†]
Jun Shirako,[†] Hirofumi Nakano,[†] Keiji Kimura[†]
and Hironori Kasahara [†]

Multicore processors are emerging everywhere in consumer electronics, such as games, digital TVs, car navigation systems and mobile phones. Advanced parallelizing compilers are desired to provide many high quality application programs in a short development period. In this paper, performance of automatic parallelization by OSCAR multigrain parallelizing compiler on a newly developed low power multicore for consumer electronics, Renesas/Hitachi/Waseda RP1 with four SH-X3 cores, is evaluated. The OSCAR compiler automatically parallelizes sequential programs written in restricted C language to generate parallel programs with a new multicore API for consumer electronics, which is aimed to be used for various multicores from different venders commonly. The experimental results shows the effectiveness of automatic parallelization on multicore processors for consumer electronics. Using AAC encoder, MP3 encoder, MPEG2 encoder, MiBench susan (smoothing) and SPEC2000 art written in restricted C language, the OSCAR compiler gave 3.43x, 3.15x, 3.53x, 3.66x and 2.54x speedups against sequential execution with 1 core, respectively.

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，高い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，

† 早稲田大学基幹理工学部情報理工学科
Dept. of Computer Science, Waseda University

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] 等が開発されている．このようなマルチコアプロセッサにおける並列プログラミングは一般に難易度が高いことで知られており，アプリケーションプログラムとハードウェア構成の両方を熟知している必要がある．一方で，製品開発サイクルの短い情報家電分野においては質の高いアプリケーションプログラムを短期間のうちに多数開発していくことが要求され，単にハードウェアの性能向上だけでなく，高いソフトウェア生産性の実現がそのマルチコアプロセッサの普及のために重要となっている．そのため，プログラマの負荷を大きく軽減できる自動並列化コンパイラの重要性が認識されており，従来の ILP(命令レベル並列性) やループ並列性に加えたさらなる並列性の抽出，メモリウォール問題に対処するためのデータローカリティの最適化，消費電力の削減等を達成する高性能な自動並列化コンパイラの開発が求められている．

筆者等が開発している OSCAR マルチグレイン自動並列化コンパイラ[4)~6)] では従来からのループレベル並列処理[7] 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，メモリウォール問題に対処するための複数ループにわたるキャッシュあるいはローカルメモリの最適利用[8),9)] およびデータ転送の最適化[10]，そして，チップ内の各リソースの周波数・電圧・電源制御による消費電力の削減[11] を実現している．

さらに筆者等は，OSCAR コンパイラによる最適化機能を最大限サポートする OSCAR マルチコアメモリアーキテクチャ[12] を提案しており，これに電力制御機能[11] 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，株式会社富士通研究所，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いる．この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／株式会社日立製作所／早稲田大学によりこの OSCAR 標準アーキテクチャに基づくマルチコアプロセッサ RP1[13] が開発された．また，上述の委員会では情報家電用の標準的なマルチコア API を策定しており，各社マルチコアにおいて利用可能とすることを目指している．OSCAR コンパイラはこの API で記述された並列プログラムを逐次プログラムから自動生成することが可能である．

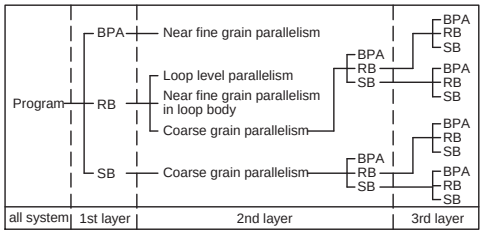


図 1 階層的マクロタスク定義
Fig. 1 Hierarchical macro task difinition

本論文では新たに開発された RP1 の SMP 実行モードにおいて，OSCAR コンパイラによる制約付き C プログラムのマルチグレイン並列化の性能評価を行う．OSCAR コンパイラはこれまで FORTRAN77 言語を対象に開発されてきたため，情報家電用のソフトウェア開発で広く用いられる C 言語へ対応するにあたり，まずは入力プログラムに一定の制約を加え，この制約を徐々に緩和していく方針で開発を進めている．

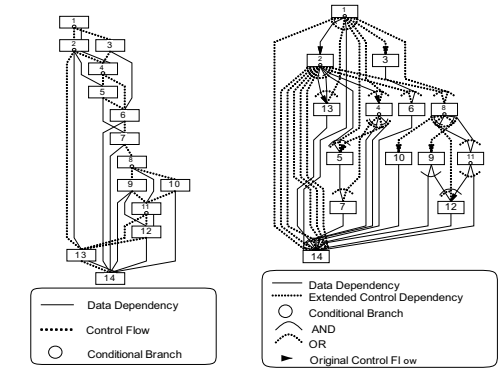
本論文の構成を以下に示す．まず 2 章では OSCAR コンパイラが実現するマルチグレイン並列処理の概要を述べる．次に 3 章では OSCAR コンパイラにおいて現在採用している制約付き C 言語について述べ，4 章では情報家電用マルチコア API について OSCAR メモリアーキテクチャの概要と併せて述べる．5 章では OSCAR コンパイラによる RP1 の SMP 実行モードにおける制約付き C プログラムのマルチグレイン並列化の性能評価について述べる．最後に 6 章で本論文のまとめを述べる．

2. 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の概要を述べる．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．本論文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．

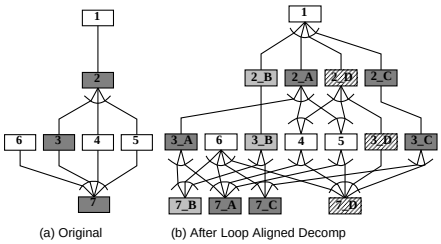
2.1 粗粒度タスク生成

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[6]，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[6]，サブルーチンブロック SB[6] の 3 種類の粗粒



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

図 2 マクロフローグラフとマクロタスクグラフ
Fig. 2 Macro Flow Graph and Macro Task Graph



(a) Original (b) After Loop Aligned Decomp

図 3 データローカライゼーションのためのループ整合分割
Fig. 3 Loop Aligned Decomposition (LAD) for data localization

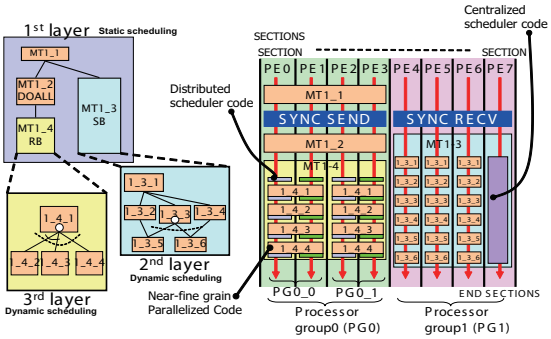


図 4 階層的コード生成イメージ
Fig. 4 Hierarchical code generation image

度タスク (マクロタスク MT[6]) 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 1 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．

2.2 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，図 2(a) に示すようなマクロフローグラフ[4),6)] 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[4),6)] を適用し，図 2(b) に示すようなマクロタスクグラフ MTG[4),6)] 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，マクロタスクが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．

2.3 データローカリティ最適化

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[8)] 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[15)] を行う．

図 3 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．整合分割後の粗粒度タスクスケジューリングにおいて，粗粒度タスク間の並列性を考慮しながら，同一データにアクセス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とで，複数のループに渡りキャッシュあるいはローカルメモリ上のデータをそのまま利用することが可能となり，メインメモリアクセスを削減することができる．

2.4 階層的な粗粒度タスクスケジューリング

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[6)] あるいは PE を複数集めたプロセッサグループ PG[6)] 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 PG 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ラコード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 4 に示すように各マクロタスクに対して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が適用される．

このようにして階層的なマクロタスクグラフから各 PE 専用のコードを生成し，プログラム開始時に PE 数だけスレッドを生成するワンタイム・シングルレベルスレッド生成を行う．また，生成されるダイナミックスケジューラはユーザコードであり，OS のスケジューラ等に比べて極めて低オーバーヘッドな並列処理が可能となる．

3. 自動並列化のための制約付き C 言語

本章では自動並列化のために本論文で採用している制約付き C 言語について述べる．C プログラムの解析については長年にわたり多くの研究がなされてきたが，特にポインタ解析については未だ解決に至っていない．現状においてこの問題に一時的に対処するためには，C99 標準[16] の “restrict” ポインタ修飾子のように，プログラマがヒント情報を付加することでコンパイラの解析精度を向上させることも選択肢の一つである．実際，多くの商用コンパイラはポインタエイリアスに関するヒント情報をユーザが指定可能な枠組み[17)~19)] を用意している．本論文では，C プログラムを対象とした開発環境において OSCAR コンパイラによる自動並列化技術の適用を早期に実現するため，制約付き C 言語を採用した．

3.1 現在の制約

これまで OSCAR コンパイラは FORTRAN77 を対象に開発が進められてきた．そこで，情報家電分野における早期の自動並列化コンパイラ実現への要求を満たすために，まず最初の段階として FORTRAN77 と同様の記述を行う制約付き C 言語を採用した．これにより，必要最低限の開発で OSCAR コンパイラを用いた C 言語による並列コード生成を実現した．現在の制約を以下に示す．

分割コンパイル

プログラム全域から並列性抽出とデータローカリティ最適化を行うために，全てのユーザプログラムを一度にコンパイルする．ライブラリ関数を用いる場合，コンパイラは内部状態を持たない数学ライブラリ等の標準ライブラリ関数のみを考慮する．その他のライブラリ関数を含む部分の並列化は行わない．

関数の再帰呼び出し

関数の再帰呼出しは行わない．

ポインタおよび構造体

以下に示すような関数のポインタ引数を除いて，ポインタおよび構造体は原則的に使用しない．そのようなポインタおよび構造体アクセスは，コンパイラで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．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．

関数のポインタ引数

配列を実引数とした関数呼び出しを想定し，関数のポインタ引数は利用可能とする．ただし，ポインタ値の再代入は行わない，ポインタ引数を用いた参照先がエイリアスしない，という制約があるものとする．これら 2 つの制約により，コンパイラのインタープロシージャ解析において，実引数と仮引数が静的にエイリアスするものとして扱うことが可能となる．さらに，ポインタ引数を用いた参照先は，全ての呼び出し元における実引数の配列宣言の境界を越えないという制約も設ける．ポインタ引数に対応する配列次元情報をディレクティブにより明示的に宣言することも可能であり，その場合はディレクティブによる宣言に従う．この制約により，C 言語のポインタ記述では表現できないポインタによるアクセスの境界を保証でき，コンパイラによる範囲解析の精度を高めることができる．

3.2 今後の制約緩和に向けて

筆者等は今後，制約付き C 言語においてポインタおよび構造体の一部に対応することを計画している．Ribeiro ら[20] は SPEC CINT2000，MediaBench の各 C プログラムに対して context-, flow-sensitive ポインタ解析を適用した際の解析精度の定量的な評価を行っており，その結果，C プログラムのポインタには静的な解析が不可能なものが含まれる一方で，十分に解析可能なプログラムが存在することを示唆した．Ryoo ら[21] はコンパイラによる解析で MPEG4 エンコーダのリファレンスプログラムに含まれる粗粒度並列性が抽出可能かどうか分析しており，インタープロシージャ配列解析, heap-sensitive, field-sensitive ポインタ解析，変数の取りうる値の範囲解析等を組み合わせることで並列性の抽出が可能と結論づけている．これらの研究成果は，制約付き C 言語における今後の制約緩和について期待ができる結果であると考えられる．

4. 情報家電用マルチコア API

本章では情報家電用マルチコア API について述べる．OSCAR 自動並列化コンパイラにより逐次プログラムから本 API を用いた並列プログラムを自動生成することが可能であり，既存の逐次コンパイラに API 解釈部を追加することにより，各社マルチコア上での並列処理が実現可能となる．

4.1 OSCAR マルチコアアーキテクチャ

本節では本マルチコア API の標準モデルとなっている OSCAR マルチコアアーキテクチャ[12] の概要を述べる．図 5 に示すように，OSCAR マルチコアアーキテクチャは複数のプロセッサコア (PE) および複数バスやクロスバ等の内部接続ネットワーク，オンチップ

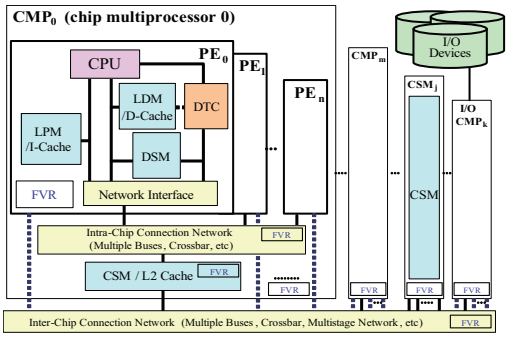


図 5 OSCAR マルチコアアーキテクチャ
Fig. 5 OSCAR multicore architecture

およびオフチップの集中共有メモリ (CSM) を持つ．各 PE は CPU コア，ローカルプログラムメモリ (LPM) または命令キャッシュ，ローカルデータメモリ (LDM) またはデータキャッシュ，分散共有メモリ (DSM)，およびデータ転送コントローラ (DTC) を持つ．コンパイル時の情報に応じて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．さらに，コンパイラによる低消費電力制御を実現するために，周波数・電圧制御レジスタ (FVR) が追加されている[11]．また，多くの実在するマルチコアプロセッサは OSCAR マルチコアのモデルで近似することができ，機能の有無やパラメタを変更することで，多様なマルチコアプロセッサへの対応が可能となる．

4.2 ディレクティブの構成

本マルチコア API は C と FORTRAN に対応しており，共有メモリマルチプロセッサを対象とした OpenMP[14] のサブセットに，OSCAR マルチコアアーキテクチャ用の新たなディレクティブを追加したものとなっている．以下にディレクティブの構成を示す．

OpenMP のサブセット

スレッド生成，同期，排他制御ディレクティブは OpenMP 互換となっており，スレッド生成には “parallel sections”，メモリアクセス順の保証には “flush”，排他制御には “critical” を用いる．これら 3 つ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

独自定義のディレクティブとしては，データのローカルメモリ配置，DTC(DMAC) によるデータ転送，そして周波数・電圧・電源制御用の API の策定が行われている．これらのディレクティブについては，現在 OSCAR コンパイラを用いて各社のマルチコア上で評価を行っている．

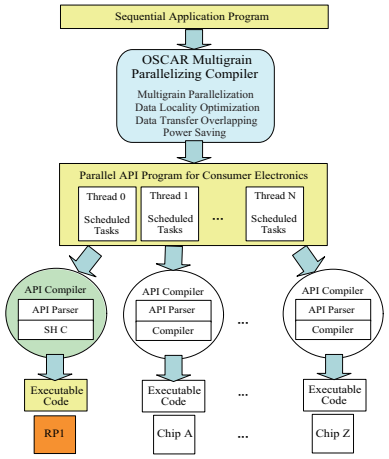


図 6 コンパイルの流れ
Fig. 6 Flow of compilation

4.3 コンパイルの流れ

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 6 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 API で記述された並列プログラムを生成する．次に，この並列化されたプログラムを API 解釈コンパイラ (通常の 1 プロセッサ用のコンパイラに API 解釈部を加えたもの) でコンパイルしマシンコードを生成する．これにより OSCAR コンパイラを用いて異なる企業のマルチコアプロセッサ用の並列コードを自動生成することが可能となる．

5. 性能評価

本章では情報家電用マルチコア RP1 の SMP 実行モードにおける，OSCAR コンパイラによる制約付き C プログラムのマルチグレイン並列化の性能評価について述べる．

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[22] 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フレーム間の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．

MP3 エンコーダは UZURA3: MPEG1/LayerIII

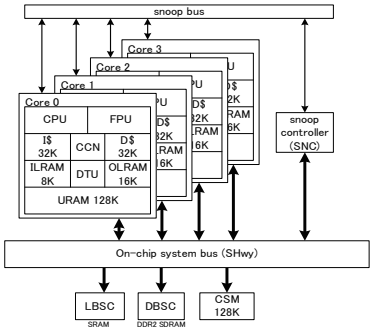


図 7 ルネサステクノロジ／日立製作所／早稲田大学 RP1
Fig. 7 Renesas/Hitachi/Waseda RP1

Encoder in FORTRAN90[23] をフレーム間の並列性抽出が可能となるように，複数フレームをバッファリングしてエンコードするように制約付き C 言語で参照実装したプログラムを用いた．

MPEG2 エンコーダは MediaBench[24] に収録されている “mpeg2encode” をマクロブロック間の並列性およびデータローカリティ[25] が抽出可能となるように制約付き C 言語で参照実装したプログラムを用いた．

susan は 組み込み向けベンチマーク MiBench[22] 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．本評価では，画像の平滑化フィルタ処理である “smoothing” オプション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り，データサイズは ref を用いた．

なお，MPEG2 エンコーダ，MP3 エンコーダについては，現在 OSCAR コンパイラにおけるデータローカライゼーション機能の C 言語対応が開発中のため，ループ整合分割のためのディレクティブによるヒント情報の指定を行っている．

5.2 ルネサステクノロジ／日立製作所／早稲田大学 RP1

情報家電用マルチコアプロセッサ RP1[13] 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，早稲田大学の OSCAR 標準マルチコアメモリアーキテクチャに基づき開発された．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 SH C コンパイラ用の API 解釈部の開発を行った．

RP1 は 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．図 7 に示すように，RP1 はコヒーレントキャッシュとローカルメモリを持っており，比較的プログラミングが容易な SMP，ローカルメモリを持ちリアルタイム処理に適した AMP(Asymmetric Multi Processor) の両実行モードをサポートしている．

SMP 実行モードでは，スヌープコントローラが専用のスヌープバスを介して各コアのデータキャッシュの一貫性を保証する．AMP 実行モードでは，各コアが持つ 3 種類のローカルメモリ，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，分散共有メモリ URAM およびチップ内の集中共有メモリ CSM が利用可能である．これらのメモリにユーザプログラムに記述された API を用いてデータを明示的に配置し，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．

本論文では SMP 実行モードにおける性能評価を行う．RP1 の周波数設定および各種メモリサイズを表 1 に示す．

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

5.3 評価手順

対象の逐次プログラムに対し OSCAR コンパイラによるマルチグレイン並列化を適用し，SMP 用マルチコア API(OpenMP のサブセット) で記述された並列 C プログラムを生成する．この並列化された情報家電用マルチコア API プログラムを，API 解釈部を追加した SH C コンパイラでコンパイルすることで実行バイナリを生成し，RP1 の評価ボード上で実行した．なお，入出力についてはメイン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．また，OS について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，時間計測にはパフォーマンスカウンタの経過クロックサイクル数を用いた．

5.4 評価結果

図 8 に OSCAR コンパイラで並列化したプログラ

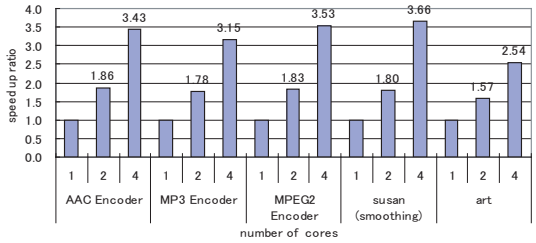


図 8 RP1 SMP 実行モードにおける速度向上率
Fig. 8 Speedup ratio in SMP execution mode of RP1

ムを RP1 の SMP 実行モードで実行した際の，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プログラムおよび使用したコア数を示し，縦軸が 1 コアによる逐次実行時の所要時間に対する速度向上率を示す．

AAC エンコーダ (図中の AAC Encoder) については 4 コア使用時に 3.34 倍，susan (smoothing) については 4 コア使用時に 3.66 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並列処理可能な一つの DOALL ループが占めており，OSCAR コンパイラでこのループを DOALL ループと解析し，粗粒度タスク並列化を行ったためこのような速度向上が得られた．

MPEG2 エンコーダ (図中の MPEG2 Encoder) については 4 コア使用時に 3.53 倍と非常に大きな速度向上，MP3 エンコーダ (図中の MP3 Encoder) についても 4 コア使用時に 3.15 倍と大きな速度向上が得られた．この 2 つのプログラムはいずれも演算処理の大半を占めるエンコード処理の逐次ループ内部が DOALL ループおよび逐次ループが連続したプログラム形状となっている．そのため，これらのループに対しループ整合分割を適用することで，粗粒度タスク並列性とデータローカリティの利用が可能となった．OSCAR コンパイラを用いて生成した MPEG2 エンコーダおよび MP3 エンコーダの主要部分のマクロタスクグラフを図 9 に示す．これらの図はともに分割数 4 の場合を表している．図 9 を見ると，MPEG2 エンコーダ，MP3 エンコーダ共に，ループ整合分割後に同一データにアクセスするマクロタスクを連続的にスケジューリングすることが可能となっている．これより，プログラムの持つ粗粒度タスク並列性とデータローカリティの有効利用による速度向上が得られた．

art については 4 コア使用時に 2.54 倍の速度向上が得られた．art は主要演算ループ内に並列化が困難な逐次処理部を含むため，このような速度向上率となっている．

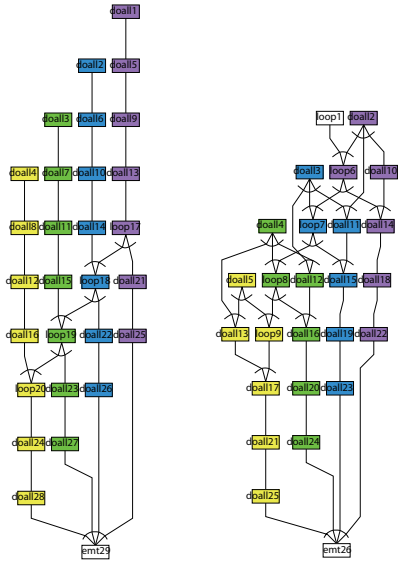


図 9 MPEG2 エンコーダおよび MP3 エンコーダの主要部分の MTG
Fig. 9 MTGs from the dominant parts of MPEG2 encoder and MP3 encoder

6. ま と め

本論文では，新たにルネサステクノロジ/日立製作所/早稲田大学により開発された情報家電用低電力マルチコアプロセッサ RP1 の SMP 実行モードにおいて，OSCAR コンパイラによるマルチグレイン並列化の性能評価を行った．OSCAR コンパイラは制約付き C 言語で記述された逐次プログラムを自動並列化し，新たに提案された情報家電用マルチコア API で記述された並列 C プログラムを自動生成する．これを既存の逐次用コンパイラに API 解釈部を追加した API 解釈コンパイラでコンパイルすることで実行バイナリを生成し，RP1 の SMP 実行モードで実行した．評価の結果，SH-X3 コアを 4 コア集積した RP1 において，制約付き C 言語で記述された AAC エンコーダ，MP3 エンコーダ，MPEG2 エンコーダ，MiBench より susan (smoothing)，SPEC2000 より art に対して OSCAR コンパイラによる並列化を適用することで，コア数に応じた速度向上を得ることができ，4 コア使用時に 1 コアによる逐次実行と比較してそれぞれ 3.43 倍，3.15 倍，3.10 倍，3.66 倍，2.54 倍の速度向上が得られた．これより，情報家電用マルチコアにおけるマルチグレイン並列化の有効性が確認された．

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” およびNEDO“先進ヘテロジニアス・マルチプロセッサ技術” の支援により行われた．また，本研究に関して多大なご協力をいただき，貴重な議論をさせていただきました株式会社日立製作所の小高俊彦氏，内山邦男氏，伊藤雅樹氏，水野弘之氏，鹿野裕明氏，株式会社ルネサステクノロジの長谷川淳氏，服部俊洋氏，亀井達也氏，伊藤雅之氏，“リアルタイム情報家電用マルチコア技術の研究開発” プロジェクト関係者の皆様に感謝申し上げます．

参 考 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) 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).
5) H.Kasahara and et al: “A multi-grain parallizing compilation scheme on oscar”, Proc. 4th Workshop on Language and Compilers for Parallel Computing (1991).
6) 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).
7) M.Wolfe: “High performance compilers for parallel computing”, Addison-Wesley Publishing Company (1996).
8) 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).
9) 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).
10) 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).
11) 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).
12) 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).
13) 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).
14) “OpenMP Application Program Interface Version 2.5” (2005).
15) 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).
16) “ISO/IEC 9899:1999 - Programming Language C” (1999).
17) “IBM XL C/C++ Advanced Edition V8.0 for Linux Compiler Reference” (2005).
18) “Sun Studio11 C User’s Guide” (2005).
19) “Intel C++ Compiler Documentation” (2007).
20) C. G. Ribeiro and M. Cintra: “Quantifying uncertainty in points-to relations”, International Workshop on Languages and Compilers for Parallel Computing(LCPC) (2006).
21) S. Ryoo, S.-Z. Ueng, C. I. Rodrigues, R. E. Kidd, M. I. Frank and W. W. Hwu: “Automatic discovery of coarse-grained parallelism in media applications”, Transactions on High-Performance Embedded Architectures and Compilers (2007).
22) 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).
23) “UZURA3:MPEG1/LayerIII Encoder in FORTRAN90”. http://members.at.infoseek.co.jp/kitaurawa/index_e.html.
24) C. Lee, M. Potkonjak and W. H. Mangione-Smith: “MediaBench: a tool for evaluating and synthesizing multimedia andcommunications systems”, In 30th Annual IEEE/ACM International Symposium on Microarchitecture (1997).
25) 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).