Vol.2010-ARC-190 No.8
2010/8/3

情報家電用ヘテロジニアスマルチコア RP-X におけるコンパイラ低消費電力制御性能

和田康孝[†1] 林明宏[†1] 渡辺岳志[†1]
関口威[†1] 間瀬正啓[†1] 白子準[†1]
木村啓二[†1] 伊藤雅之[†2] 長谷川淳[†2]
佐藤真琴[†3] 野尻徹[†3]
内山邦男[†3] 笠原博徳[†1]

本稿では，情報家電用ヘテロジニアスマルチコア RP-X 上で，コンパイラによる低消費電力制御を適用した結果について述べる．RP-X は NEDO の “情報家電用ヘテロジニアス・マルチコア技術の研究開発” プロジェクトにおいて開発された情報家電用のヘテロジニアスマルチコアであり，汎用 CPU コアとして SH-4A コアを 8 基，アクセラレータコアとして多目的 DRP コア FE-GA 4 基と画像処理用コア MX2 2 基，さらにメディア用コア VPU5 を搭載する．また，周波数制御・電圧制御等の低消費電力化のための機構を持つ．OSCAR コンパイラによって実現される低消費電力制御手法を RP-X の低消費電力機構に適用し，リアルタイム処理時の消費電力削減効果の評価を行った．その結果，SH-4A 8 コアと FE-GA 4 コアを用いた場合，制御を適用しない場合と比較して，オプティカルフロー演算において約 70[%]，AAC エンコーダにおいて約 80[%] の電力削減を得ることができた．

Performance of Power Reduction Scheme by a Compiler on Heterogeneous Multicore for Consumer Electronics “RP-X”

Yasutaka Wada,[†1] Akihiro Hayashi,[†1]
Takeshi Watanabe,[†1] Takeshi Sekiguchi,[†1]
Masayoshi Mase,[†1] Jun Shirako,[†1] Keiji Kimura,[†1]
Masayuki Ito,[†2] Atsushi Hasegawa,[†2]
Makoto Sato,[†3] Tohru Nojiri,[†3] Kunio Uchiyama[†3]
and Hironori Kasahara [†1]

This paper reports the efficiency of power reduction scheme by OSCAR compiler applied for a heterogeneous multicore for consumer electronics “RP-X”. RP-X is a heterogeneous multicore developed in NEDO “Heterogeneous Multicore for Consumer Electronics” project. RP-X includes eight SH-4A cores, four FE-GA DRPs, two MX2 matrix processors, and one VPU5 media processor. To satisfy strong demands for low power consumption, RP-X is also equipped with mechanisms to reduce the power by changing operation frequency and voltage, or by gating clock. Power reduction scheme implemented in OSCAR compiler is applied to RP-X, and evaluated under the realtime constraint using eight SH-4A cores and four FE-GA cores. As the results, consumed power was reduced by about 70[%] for optical flow calculation, and about 80[%] for an AAC encoder program.

1. はじめに

近年，携帯電話，ゲーム機，デジタル TV 等に代表される情報家電，組み込みシステムの分野においても，年々高まる演算性能に対する要求を満たすため，1 つのチップ上に複数のプロセッサコアを集積したマルチコアプロセッサが[1)–3)] 一般的に用いられるようになって

†1 早稲田大学理工学術院
School of Fundamental Science and Engineering, Waseda University
†2 ルネサス エレクトロニクス株式会社
Renesas Electronics Corporation
†3 株式会社 日立製作所
Hitachi, Ltd.

いる．さらに現在では，演算性能への要求と同時に，いかに消費電力を抑えるかが大きな課題となっている．この課題を克服するために，特定の処理を高速化するアクセラレータをチップ上に併せて集積するヘテロジニアスマルチコアも開発・利用されるようになってきている[4)–6)]．ヘテロジニアスマルチコアでは，信号処理用プロセッサ（DSP）や動的再構成可能プロセッサ（DRP）等のアクセラレータを利用することで，高速な演算と低消費電力化を両立することが可能であるが，今後更なる低消費電力化と高性能化，プログラム容易性を実現することが求められる．

このような背景から，NEDO“情報家電用ヘテロジニアス・マルチコア技術の研究開発”プロジェクトにおいて，早稲田大学，東京工業大学，ルネサスエレクトロニクス，日立製作所によって情報家電用ヘテロジニアスマルチコア RP-X を開発した．RP-X は，汎用コア 8 基に加え，3 種類計 7 コアのアクセラレータを搭載しており，周波数制御や電圧制御などの低消費電力機構を備える．

さらに，ヘテロジニアスマルチコアの性能を最大限引き出すことはとても困難であり，ソフトウェアの生産性を高めるためには，並列化コンパイラが必要とされる．用途によって様々な構成をとりうるヘテロジニアスマルチコアアーキテクチャに，迅速かつ柔軟に対応するため，ヘテロジニアスマルチコア向けの並列化 API である OSCAR ヘテロジニアスマルチコア API，およびこの API に基づいた並列化コンパイルフレームワークが開発されている[7)]．

本稿では，従来 OSCAR コンパイラによって実現されてきた，並列化コンパイラによる低消費電力制御手法[8),9)] をヘテロジニアスマルチコア用に拡張し，OSCAR ヘテロジニアスマルチコア API を用いて RP-X の低消費電力機構に適用し，評価を行った結果について述べる．

以下，2 章でヘテロジニアスマルチコア向け並列化コンパイラについて，3 章で情報家電用ヘテロジニアスマルチコア RP-X について，4 章で RP-X への低消費電力制御の適用について，5 章でオプティカルフロー演算プログラムおよび AAC エンコーダを用いた，RP-X 上でのリアルタイム処理時の消費電力性能評価について述べる．

2. ヘテロジニアスマルチコア向け並列化コンパイラ

本章では，情報家電用マルチコア向け並列化 API である OSCAR API[10)] をヘテロジニアスマルチコア向けに拡張した OSCAR ヘテロジニアスマルチコア API[7)] の概要，OSCAR API を用いたヘテロジニアスマルチコア向け並列化コンパイルフロー，ヘテロジニアスマルチコア上での並列処理手法，OSCAR コンパイラによる低消費電力制御の基本概念について述べる．

2.1 OSCAR ヘテロジニアスマルチコア API

OSCAR API は情報家電用マルチコア向けの並列化 API であり，並列実行，メモリ配置，データ転送，電力制御及びタイマ制御の指示文から構成されている．また，共有メモリ型マルチプロセッサ用の並列化 API として標準的に用いられている OpenMP[11)] のサブセットを用いてスレッド生成，同期，共有変数の定義を行っているため，一般の OpenMP コンパイラでコンパイル可能であり，既存の様々なプラットフォーム上でも容易に利用することが可能である．従来の OSCAR API はホモジニアスな構成をもつマルチコアのみを想定していたが，ヘテロジニアスマルチコアへ適用するための拡張が施され，OSCAR コンパイラによって，この OSCAR ヘテロジニアスマルチコア API を用いた自動並列化を実現している[7)]．以下本節では，本ヘテロジニアスマルチコア API が対象とするマルチコアアーキテクチャおよびコンパイルフローについて概要を述べる．

2.1.1 対象ヘテロジニアスマルチコアアーキテクチャ

ここでは，OSCAR ヘテロジニアスマルチコア API が対象とするマルチコアアーキテクチャである，OSCAR ヘテロジニアスマルチコアアーキテクチャおよび OSCAR API-Applicable ヘテロジニアスマルチコアアーキテクチャについて述べる．

OSCAR ヘテロジニアスマルチコア API が対象とするヘテロジニアスマルチコアアーキテクチャは，図 1 a) に示す通り，OSCAR メモリアーキテクチャ[10)] をベースとしたものである．この OSCAR ヘテロジニアスマルチコアアーキテクチャでは，汎用 CPU を持つコアとアクセラレータを持つコアが混載されており，各コアはローカルデータメモリ (LDM)，2 ポート構成の分散共有メモリ (DSM) およびデータ転送ユニット (DTU) を持つ．さらに，アクセラレータが搭載されたコアにはコントローラ CPU も同時に搭載されている．このコントローラ CPU により，コア間の同期処理やコア内アクセラレータの起動，およびアクセラレータが対象としない処理が行われる．

ただし，実際には，当該コア内にコントローラ CPU を持たないアクセラレータが集積されているヘテロジニアスマルチコアも多く存在する．このような場合でも，チップ上の汎用 CPU コアをアクセラレータと対にし，コントローラとして使用することで本 API を適用することが可能である．このように，コントローラ CPU を持たないアクセラレータコアも含めた，より一般的な形式のマルチコアも本 API の対象アーキテクチャであり，OSCAR API-Applicable ヘテロジニアスマルチコアアーキテクチャ（図 1 b)）として定義される．

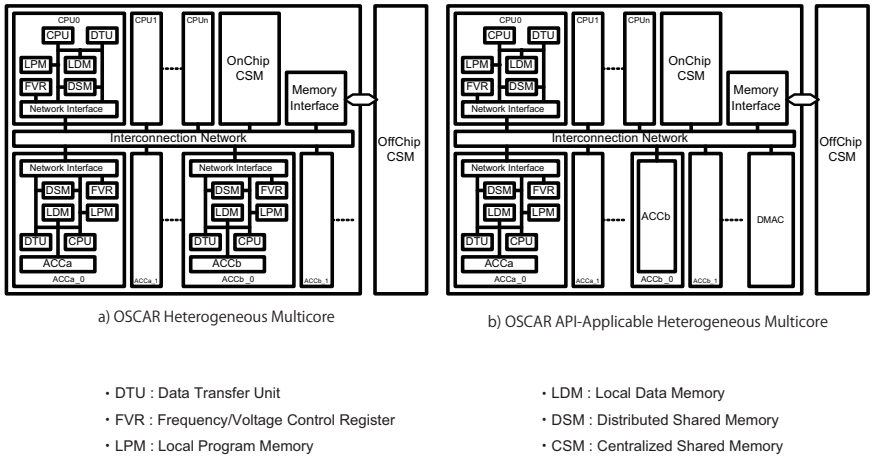


図 1 OSCAR ヘテロジニアスマルチコアおよび OSCAR API-Applicable ヘテロジニアスマルチコアの例

図中，ACCa は OSCAR メモリアーキテクチャに準拠し，コントローラ CPU を持つアクセラレータコアであるが，ACCb はアクセラレータが単体で接続される形となっている．

2.1.2 コンパイルフロー

ここでは，OSCAR ヘテロジニアスマルチコア API と OSCAR コンパイラによって実現される並列化コンパイルフローについて述べる．図 2 に示す通り，本コンパイルフローは，OSCAR コンパイラの入力以前と OSCAR コンパイラの入力以後に大きく分けることができる．

OSCAR コンパイラ入力以前の処理では，逐次の C もしくは Fortran プログラムをアクセラレータ用コンパイラに入力し，入力プログラム中でアクセラレータが実行可能な部分やアクセラレータでの処理コストなどの情報を，ヒント指示文としてソースプログラム中に挿入する．このとき，OSCAR コンパイラ用ヒント指示文はプログラマが手動で挿入することも可能である．

OSCAR コンパイラはヒント指示文入りソースプログラムを入力し，ヘテロジニアスマルチコア用並列化や低消費電力処理等を行い，その結果を OSCAR ヘテロジニアスマルチコア API の指示文入り C もしくは Fortran プログラムとして出力する．この時，OSCAR コンパイラはアクセラレータの種類ごとに各アクセラレータで実行されるプログラム部分を切り出し，これらを汎用 CPU 用コードとは別に出力する．

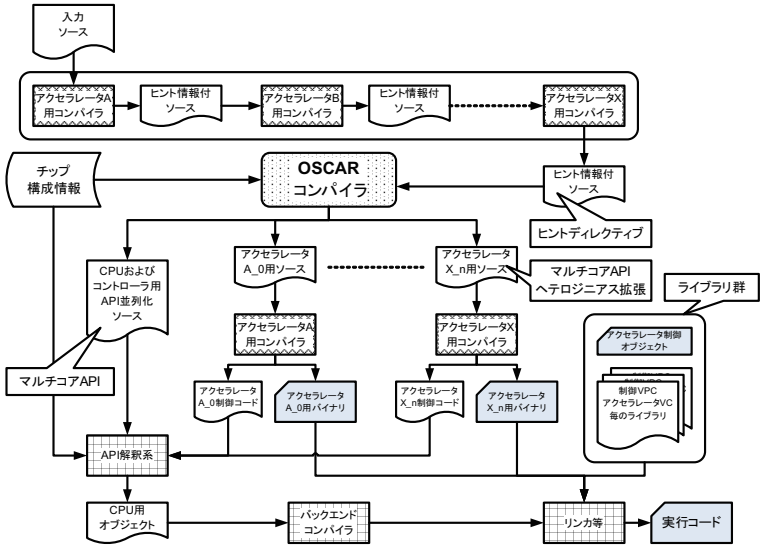


図 2 ヘテロジニアスマルチコア API を用いた並列化コンパイルフロー

OSCAR コンパイラ出力以後は，汎用 CPU コードは API 解釈系により OSCAR API を対象 CPU 用の実行時ライブラリに変換され，バックエンドコンパイラでオブジェクトコードに変換される．アクセラレータ用コードはそれぞれ対象とするアクセラレータコンパイラによりアクセラレータ用バイナリに変換され，同時にアクセラレータを制御するコントローラ CPU 用コードが生成される．このコントローラ CPU コードは，汎用 CPU 用バックエンドコンパイラによりオブジェクトコードに変換される．

最終的に，これらの汎用 CPU 用コードとアクセラレータコードがリンクされて単一の実行オブジェクトが生成される．

なお，既存のアクセラレータ用ライブラリコードを有効活用するため，ヒント指示文を手動で挿入して当該ライブラリにより実行可能な部分を指定し，当該ライブラリをリンクして実行オブジェクトを生成することもできる．

2.2 ヘテロジニアスマルチコア上での並列処理

ここでは，2.1 で述べた OSCAR ヘテロジニアスマルチコア API を用いて OSCAR コンパイラによって実現される，ヘテロジニアスマルチコア向けの並列処理手法[12] について述

べる．本手法は，粗粒度タスク並列処理をベースとしており，これをヘテロジニアスマルチコアに適用したものである．なお，本手法では，コンパイル時に各コアへのタスク割当てを決定するスタティックスケジューリングが用いられる．

2.2.1 粗粒度タスク並列処理

粗粒度タスク並列処理では，対象とするプログラムはまず擬似代入文ブロック (BPA)，繰り返しブロック (RB)，サブルーチンブロック (SB) の 3 種類の粗粒度タスク (マクロタスク (MT)) に分割される．MT 生成後，MT 間のコントロールフローおよびデータ依存を解析し，マクロフローグラフ (MFG) が生成される．さらに最早実行可能条件解析によって MFG から MT 間の並列性を抽出してマクロタスクグラフ (MTG) を生成する[13),14)]．この際，RB および SB はネスト構造を持つ場合があり，その場合には内部ボディ部に対して階層的に MT および MTG 生成を行う．

その後，各階層の MT は 1 つ以上のプロセッサコアを論理的にグルーピングしたプロセッサグループ (PG) に割り当てられる．このとき，MTG 内に条件分岐等がなければ，プロセッサ間の同期やデータ通信などのオーバーヘッドを最小化するために静的に MT を割り当てるスタティックスケジューリング手法が適用される．本稿ではこのスタティックスケジューリング手法を対象としており，実行時のタスク割り当て等のオーバーヘッドを最小化することができる．

さらに MTG 内の SB や RB 内部に粗粒度並列性が存在する場合，その SB や RB 内部で階層的に MTG を生成し，階層的に粗粒度タスク並列処理を適用する．

2.2.2 ヘテロジニアスマルチコア上での粗粒度タスク並列処理

ヘテロジニアスマルチコア上での粗粒度タスク並列処理においては，マクロタスクの決定，プロセッサグルーピング，およびマクロタスクスケジューリングのそれぞれがコンパイルフローと対象アーキテクチャを考慮して拡張されている．

まずマクロタスクの決定においては，OSCAR コンパイラの入力となるソースに付加されているヒント情報と，コントローラ CPU とアクセラレータコアの連携が考慮される．

プロセッサグルーピングにおいては，アクセラレータを最大限活用するようにグルーピングが行われる．もし，アクセラレータを階層的なプロセッサグルーピングの対象としてしまうと，特定の MTG からしか利用できなくなってしまうため，プログラム全体としてアクセラレータコアの利用効率が下がってしまう．これを避けるため，本手法では，汎用 CPU コアはプログラムの階層的な並列性に従って階層的なグルーピングを適用するが，アクセラレータコアはこれとは独立して個別に取り扱う．このようにして，プログラム中のいずれの

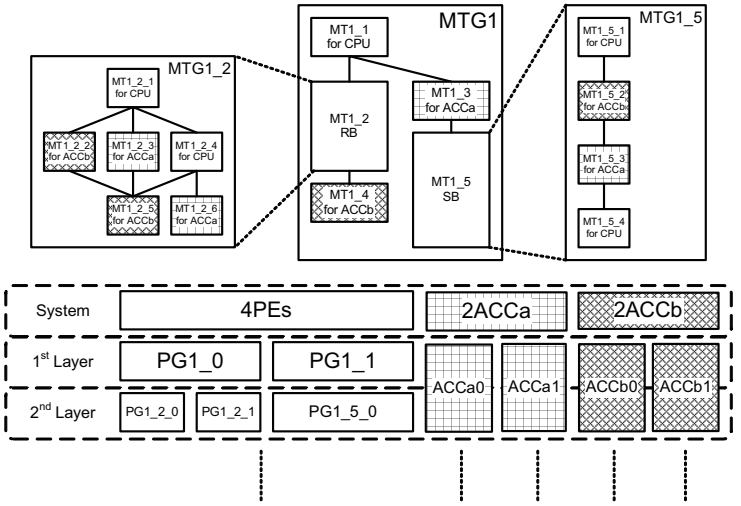


図 3 ヘテロジニアスマルチコア向けプロセッサグルーピング

階層からもタスクを割り当てることができるようにすることで，アクセラレータコアの利用効率を向上することができる．

各プロセッサグループあるいはアクセラレータコアへの粗粒度タスクのスケジューリングにおいては，各コアにおける処理コストの違いが考慮される．さらに，アクセラレータによって高速化可能な部分を含むタスクであっても，アクセラレータコアの負荷状況によっては汎用 CPU コアに割り当てることも考慮する．このようにして，プログラム全体の処理効率を向上させる．

2.3 コンパイラによる低消費電力制御

ここでは，OSCAR コンパイラによって実現される低消費電力制御[8),9)] の概念について述べる．本手法では．OSCAR コンパイラによって対象アプリケーションを並列化する際に，プログラムの特性を考慮して，デッドライン制約に応じた周波数・電圧制御やプロセッサを停止するなどの処理を自動的に挿入する．

入力プログラムの持つ特性や並列性によっては，チップ上の資源に対して必ずしも充分な並列性が得られない場合があり，この時，処理の割り当てられていない資源は無駄な電力を消費してしまうことになる．本手法は，そのような場合に周波数・電圧制御や電源制御等を適用することで，無駄に消費される電力を削減するものである．

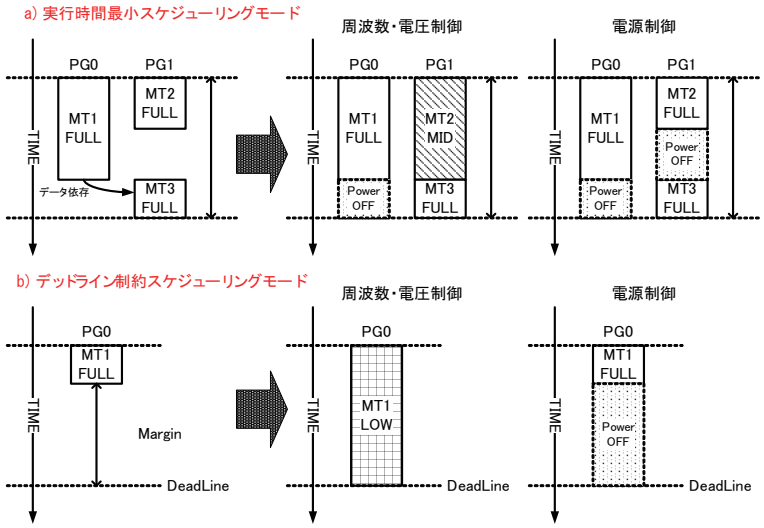
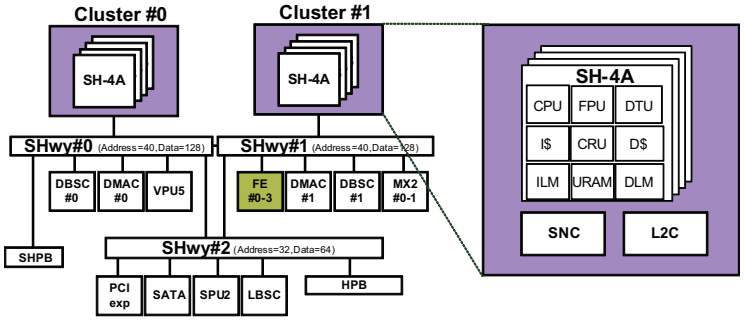


図 4 OSCAR コンパイラによる低消費電力制御の基本概念

OSCAR コンパイラによって実現される消費電力制御手法には，実行時間最少スケジューリングモードとデッドライン制約スケジューリングモードの 2 つのモードがある．図 4 にこれらの基本概念を示す．図中の実行時間最小スケジューリングモードでは，プログラムのクリティカルパスにあたる処理に対しては消費電力制御を行わず，それ以外の部分に対して消費電力制御を行うものである．これにより，プログラムの実行時間を延ばすこと無く低消費電力化を実現する．また，図中のデッドラインスケジューリングモードでは，プログラム終了のデッドラインを保証する範囲内で電力消費が最小となるように低消費電力制御を行うものである．

なお，本手法では，プログラム内の各 MTG に対してスタティックスケジューリングが適用された結果に対し，粗粒度タスク間の負荷バランスや各 MTG の実行終了時間などを考慮して各 MT を実行する際の周波数を決定する．この際，周波数・電圧切り替えのオーバヘッドやそれぞれの MT の取り得る周波数などを考慮して周波数・電圧／電源制御を適用する．

コンパイラによって決定された動作周波数および切り替えのタイミングは，消費電力制御用 API をプログラムの当該箇所に挿入することにより実現される．



SHwy	: On-chip system bus	SNC	: Snoop controller
DBSC	: DDR-SDRAM controller	L2C	: L2 cache/On-chip SRAM
LBSC	: SRAM controller		
PCIexp	: PCI express	CRU	: Cache/RAM control unit
SATA	: Serial ATA	DTU	: Data transfer unit
VPU5	: Video processing unit	I$	: Instruction cache
SPU2	: Sound processing unit	D$	: Data cache
FE	: Dynamically reconfigurable processor	ILM	: Instruction local memory
MX2	: 1024-way matrix processor	DLM	: Data local memory
SHPB/HPB	: Peripheral bus	URAM	: User memory

図 5 情報家電用ヘテロジニアスマルチコア RP-X のブロック図

3. 情報家電用ヘテロジニアスマルチコア RP-X

本章では，本稿で評価の対象とする情報家電用ヘテロジニアスマルチコア RP-X のアーキテクチャおよび消費電力制御機能について述べる．RP-X は，NEDO“情報家電用ヘテロジニアス・マルチコア技術の研究開発” プロジェクトにおいて，早稲田大学，東京工業大学，ルネサスエレクトロニクス，日立製作所によって開発されたヘテロジニアスマルチコアである．

3.1 アーキテクチャ概要

RP-X は汎用 CPU コアとして SH-4A コアを 8 基，アクセラレータコアとして多目的 DRP コア FE-GA[4] 4 基と画像処理用コア MX2[15] 2 基，さらにメディア用コア VPU5 を搭載したヘテロジニアスマルチコアである[16]．RP-X のブロック図を図 5 に示す．

RP-X 内の各 SH-4A は OSCAR 型メモリアーキテクチャをベースに構成されており，命令キャッシュ，データキャッシュ，ローカルメモリ（ILM, DLM），分散共有メモリ（URAM），データ転送ユニット（DTU）を持つ．また，各アクセラレータはコントローラ CPU を持た

表 1　RP-X の周波数状態

	設定可能な周波数状態
SH-4A	FULL（648[MHz]）
	MID（324[MHz]）
	LOW（162[MHz]）
	VLOW（81[MHz]）
FE-GA	FULL（チップ全体の設定に依存）
チップ全体	FULL（SH-4A:648[MHz]，FE-GA:324[MHz]）
	MID（SH-4A:324[MHz]，FE-GA:162[MHz]）
	LOW（SH-4A:162[MHz]，FE-GA:81[MHz]）

ず，オンチップバス（SHwy#1）に直接接続されている．つまり，RP-X は OSCAR API-Applicable ヘテロジニアスマルチコアである．また，8 基の SH-4A コアは 4 基ずつで 2 つのクラスタを構成しており，同一クラスタ内でのみハードウェアでキャッシュのコヒーレンシ制御が行われる．そのため，5 コア以上の SH-4A を同時に使用する場合など，別クラスタの SH-4A を同時に利用する際には，キャッシュを明示的に走査する命令を用い，ソフトウェアによりコヒーレンシを維持する必要がある．なお，SH-4A のデータキャッシュのサイズは 32[KBytes] であり，ラインサイズは 32[Bytes] である．

3.2　消費電力制御機構

RP-X は，動作周波数や電源電圧を実行時に動的に変化させることによって消費電力を削減するための機能を持つ．ここでは，RP-X の消費電力制御機能について述べる．

RP-X では，各 SH-4A コアの動作周波数を 648[MHz]，324[MHz]，162[MHz]，81[MHz] のいずれかに独立して変更することが可能である．また各 SH-4A コアは，コア内の CPU のみクロックを停止する Light Sleep，コア内の URAM と DTU 以外のクロックを遮断する Normal Sleep のいずれかの状態を個別にとることもできる．それに対し，4 基の FE-GA は通常 324[MHz] で動作するが，個別に周波数制御を適用することができない．ただし，チップ全体の動作周波数を一括して変更する場合にのみ，それに依存して動作周波数が変化する．チップ全体の動作周波数に関しては，通常時を 100[%] とした場合，50[%]，25[%] に設定することが可能である．また，電源電圧はチップ全体一括での変更が可能となっている．

SH-4A および FE-GA の周波数状態を表 1 に示す．

また，電源電圧はチップ一括での制御が可能であり，その時点で最も高速に動作するコアにあわせて設定される．最も高速に動作しているコアが FULL（100[%]）の状態であった場合には 1.308[V]，MID（50[%]）の状態であった場合には 1.164[V]，LOW（25[%]）以下の状態であった場合には 1.020[V] となる．

4. RP-X への低消費電力制御の適用

ここでは，2.3 で述べた OSCAR コンパイラによる低消費電力制御手法を，RP-X に対して適用するための手法について述べる．

2.3 で述べてた消費電力制御手法は，チップ上の各コアが個別に周波数・電圧の状態をとることができ，クロックゲーティング・電源遮断も個別に適用可能なマルチコアの制御を実現している．しかしながら，3.2 で述べた通り，RP-X の消費電力制御機能は，電源遮断やアクセラレータに対するクロックゲーティングをサポートしていないため，RP-X の消費電力制御機能に限定して改変を行った．

3 章で述べた通り，RP-X 汎用 CPU コアである SH-4A に対して個別に周波数制御およびクロック停止を適用することが可能であるが，アクセラレータである FE-GA に対しては，クロック停止や個別の周波数制御を適用することはできない．また，電圧の制御はチップ全体に対してのみ可能という制限がある．そのため，そのような制限を考慮した上で，消費電力制御手法を適用する必要がある．

FE-GA による電力消費について考慮すると，チップ全体の動作周波数および電圧を下げることでアクセラレータの消費電力を極力削減し，さらに SH-4A に対してクロック停止を適用することで，チップ全体の消費電力をより効率よく低減することが可能になると考えられる．そのため，RP-X に対しては，チップ上のコア全てが処理に参加する MTG に対して，

- MTG 全体をデッドラインまでに終了可能な最低の動作周波数を探索する．
- MTG 開始時にチップ全体を当該周波数となるように制御
- 周期等により待ち時間が発生する汎用コアに対し，制御オーバヘッドを考慮してクロックゲーティングを適用する．
- デッドライン制約スケジューリングモードの場合，デッドラインまでの時刻制御を行う間，時刻制御を行うマスタコア以外の SH コアにクロックゲーティングを適用し，マスタコアはデッドラインを迎えるまでタイマを監視する．
- 周期タスクにおいてデッドラインを迎えたタイミングで，上記のマスタコアがチップ全体を制御して動作状態にし，次の周期の処理に移る．

という形で消費電力制御が適用されるように OSCAR コンパイラに改変を施した．さらに，このような制御を API を介して可能とするため，消費電力制御 API において，チップ全体を一括して指定できるよう，記述を拡張した．具体的には，消費電力制御 API において制御対象を指定するモジュールの記述をする際，“OSCAR_ENTIRECHIP()” を用いて

チップ全体に対する制御を行った．

5. RP-X 上での性能評価

4 章で述べた低消費電力制御手法を OSCAR コンパイラに実装し，RP-X 上での電力削減効果の評価を行った．

5.1 評価条件

本評価では，周波数・電圧制御において，チップ全体での一括制御を効果的に適用するため，OSCAR コンパイラから利用可能な全てのコア（8 基の SH-4A と 4 基の FE-GA）を用いた場合について評価を行った．このとき，リアルタイム制御の対象となっている箇所のみを評価および測定の対象とし，ファイル読み込み等の前処理および後処理は評価の対象外としている．なお，あらかじめ取得したプロファイル結果をタスクスケジューリングに用いた．

また，OSCAR コンパイラに実装されているコヒーレンシ制御機能[17] を適用し，SH-4A のクラスタにまたがる並列処理を実現している．全コアで共有するオンチップ CSM が，コア間通信用のノンキャッシャブルバッファとして用られる．

5.2 評価アプリケーション

本評価で用いたアプリケーションは，オプティカルフロー演算プログラムおよび AAC エンコーダプログラムである．これらに対して，リアルタイム制約モードによる低消費電力制御を適用し，RP-X 上で評価を行った．

5.2.1 オプティカルフロー

オプティカルフロー演算とは，時間的に連続な動画像を入力とし，画像中の物体の速度場を求める計算である．

本評価では，日立製作所および東北大学張山研究室提供のプログラムを Parallelizable C[18] で参照実装したプログラムを用いた．本プログラムでは，入力画像を 16 × 16 ピクセルのブロックに分割し，各ブロックに対して，連続する 2 フレーム間で対応する点を探索して速度ベクトルを生成する．速度ベクトルの計算においては，ある対象ブロックについて，次フレーム中で対応する探索範囲に対してピクセル単位で差分計算（SAD 演算）を行い，最も差分の小さい箇所を対応する点とする．なお，探索を行う範囲は，対象ブロックを中心として上下左右 4 ピクセルの範囲，つまり 24 × 24 ピクセルの範囲となっている．図 6 に速度ベクトル算出の処理イメージを示す．

本プログラムは，画像を縦方向に走査するループおよび横方向に走査するループの 2 重

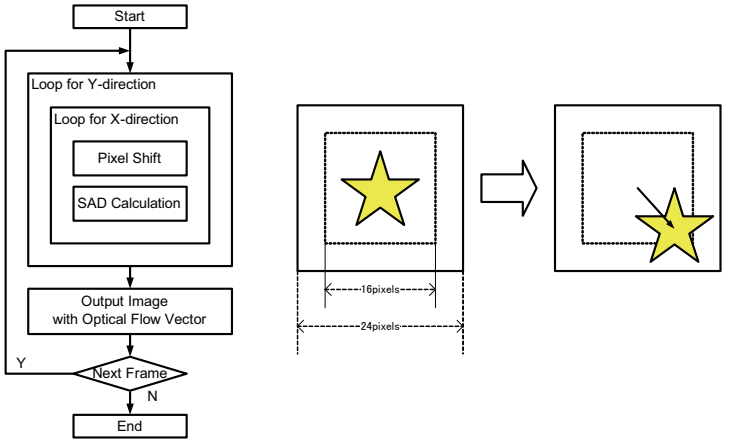


図 6 オプティカルフロー演算における速度ベクトル算出のイメージ

ループ構造となっており，各ブロックのマッチング処理は互いに依存せず並列に行うことが可能である．つまり，上記の 2 重ループは DOALL ループである．本評価では，画像の縦方向に相当するループを分割することで粗粒度タスク並列性が抽出される．OSCAR コンパイラは，分割後の横方向の処理を 1 つの粗粒度タスクとしてヘテロジニアス並列化したコード生成する．

なお，SAD 演算によって最も差分の小さい箇所を算出する部分が FE-GA で高速化可能であり，日立製作所および東北大学張山研究室提供のプログラムをライブラリとして用いた．また，本評価においては，352 × 240 ピクセルのグレースケール画像 450 フレームを入力データとして用いた．

5.2.2 AAC エンコーダ

AAC エンコーダは，CD 等に含まれる音楽データを圧縮するプログラムである．本評価で用いる AAC エンコーダはルネサスエレクトロニクスおよび日立製作所提供の AAC-LC エンコードプログラムを Parallelizable C で参照実装したものである．AAC エンコードプログラムの構造を図 7 に示す．本プログラムは音声データの読み込み後，各フレームに対してフィルタバンク処理，MS ステレオ処理，量子化およびハフマン符号化を行う．

本プログラムに実装されているアルゴリズムでは，各フレームに対する処理を並列に行うことが可能であり，粗粒度並列性抽出の為，エンコード処理はソースレベルで 20 フレーム

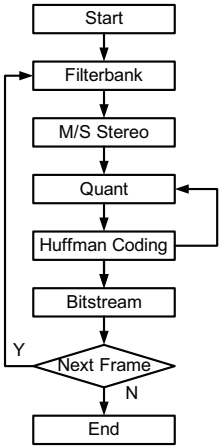


図 7 AAC エンコーダの処理フロー

分アンローリングされている．OSCAR コンパイラは 1 つのフレームのエンコード処理を 1 つの粗粒度タスクとしてヘテロジニアス並列化を行い，ヘテロジニアスマルチコア API を用いて並列化されたプログラムを生成する．

なお，アクセラレータである FE-GA で高速化可能な処理はフィルタバンク処理，MS ステレオおよび量子化の処理であり，日立製作所提供のプログラムをライブラリとして用いた．本評価におけるエンコードパラメータは，サンプリングレートが 44.1[kHz]，ビットレートが 128[kbps] であり，約 19 秒のステレオ PCM データが入力として用いられた．

5.3 評価結果

ここでは，オプティカルフロー演算および AAC エンコーダに対して，リアルタイム制御モードにおける低消費電力制御を適用し，RP-X 上で評価を行った結果について述べる．

図 8 に，オプティカルフロー演算プログラムにコンパイラによる低消費電力制御を適用しない場合，適用した場合それぞれのチップの消費電力波形を示す．図 8 において，横軸は時間，縦軸は消費電力 [W] である．本評価においては，30[fps] で動画表示するため，1 周期あたりのデッドラインはおよそ 33[ms] となっている．

低消費電力制御を適用しない場合，演算処理を行っている間は各コアが活性化して消費電力が一時的に増大するが，演算処理が終わり次の周期を開始するまでの間も，ビジーウェイト等のために電力を消費してしまっており，平均 1.75[W] となった．一方，コンパイラによる低消費電力制御を適用した場合，後述の AAC と比較して演算そのもののコストと画像表示のコストが大きいことから，演算処理を行っている間の電力状態として MID が選択された．さらに，演算処理および画像表示を行った後，次の周期を開始するまでの間，時刻制御を行うコア以外の CPU は Sleep 状態となる．この結果，消費電力は平均 0.53[W] となり，およそ 70[%] の電力削減効果を得た．

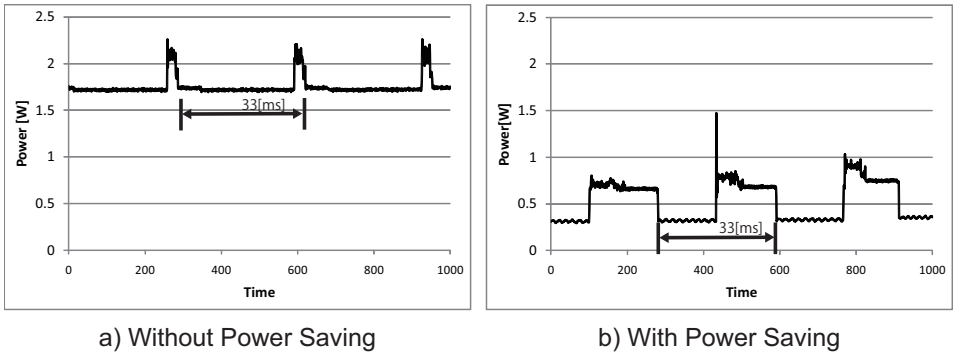


図 8 リアルタイム処理時のオプティカルフロー演算の電力波形

図 9 に，AAC エンコーダにコンパイラによる低消費電力制御を適用しない場合，適用した場合それぞれのチップの消費電力波形を示す．図 9 において，横軸は時間，縦軸は消費電力 [W] である．本評価においては，1 周期に 20 フレームが処理されるため，1 周期あたりのデッドラインはおよそ 0.46[s] となっている．

AAC エンコーダにおいては，オプティカルフロー演算と比較して演算のコストが小さいことから，エンコード処理を行っている間の電力状態として LOW が選択された．さらに，オプティカルフロー演算と同様，エンコード処理後次の周期を開始するまでの間，時刻制御を行うコア以外の CPU は Sleep となる．この制御のイメージを図 10 に示す．図 10 では，汎用コアである SH-4A（CPU0 から CPU3），およびアクセラレータである FE-GA とそれを制御する SH-4A の組（ACCEL0 から ACCEL3）それぞれの特性と性能差を考慮して負荷分散がなされ，効率よく並列処理が行われている．さらに，リアルタイム制御においてデッドラインが与えられているため，それを考慮して周波数および電圧を下げ，LOW の電力状態で処理が行われている．この例では，ACCEL0 が最も遅い時刻までエンコード処理を行っているため，ACCEL0 内の SH-4A コア（SH#4）がマスタコアとなって時刻制

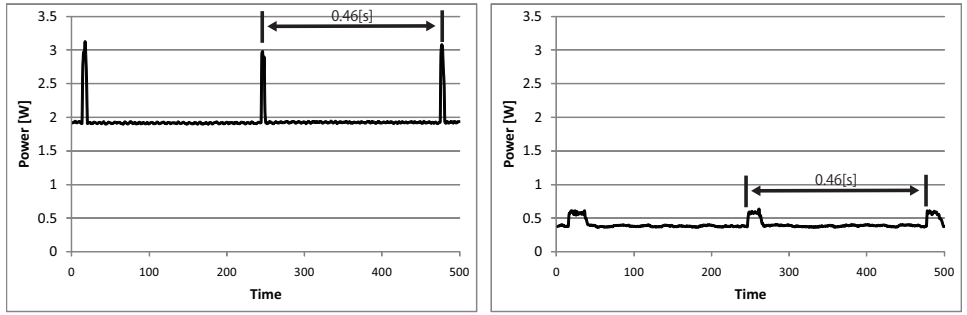


図 9 リアルタイム処理時の AAC エンコーダの電力波形

御を行っており，デッドラインを迎えるまでの間，それ以外の SH-4A コアは Sleep となる．ただし，FE-GA はクロックゲーティングを適用できないため，チップ全体の周波数・電圧状態に従い，その電力状態は LOW となる．最後に，デッドラインを迎えたタイミングでマスタコアがチップ全体を LOW の状態になるよう制御し，他の SH-4A コアが動作状態となり次の周期の処理が始まる．以上のようにして，コンパイラによる低消費電力制御が適用される．この結果，制御を適用しない場合の消費電力が平均 1.9[W] であるのに対し，適用した場合の消費電力は平均 0.38[W] となり，およそ 80[%] の電力削減効果を得た．

6. ま と め

本稿では，SH-4A コア 8 コア，DRP アクセラレータ FE-GA 4 コア，画像処理用コア MX2 2 コア，メディア用コア VPU5 1 コアの計 15 コアを搭載した情報家電用ヘテロジニアスマルチコア RP-X においてコンパイラによる低消費電力制御を適用し，評価を行った結果について述べた．本評価では，OSCAR ヘテロジニアスマルチコア API の枠組みに準拠した並列化機能および低消費電力化機能を OSCAR コンパイラに実装し，オプティカルフロー演算プログラムおよび AAC エンコーダを用いてリアルタイム制御時の消費電力削減効果を評価した．その結果，汎用コア SH-4A 8 コアと FE-GA アクセラレータ 4 コアを用いてリアルタイム実行を行ったところ，電力制御を適用しない場合に比べ，オプティカルフロー演算において約 70[%]，AAC エンコーダにおいて約 80[%] の電力削減に成功した．

謝辞 本研究の一部は NEDO“情報家電用ヘテロジニアス・マルチコア技術の研究開発”

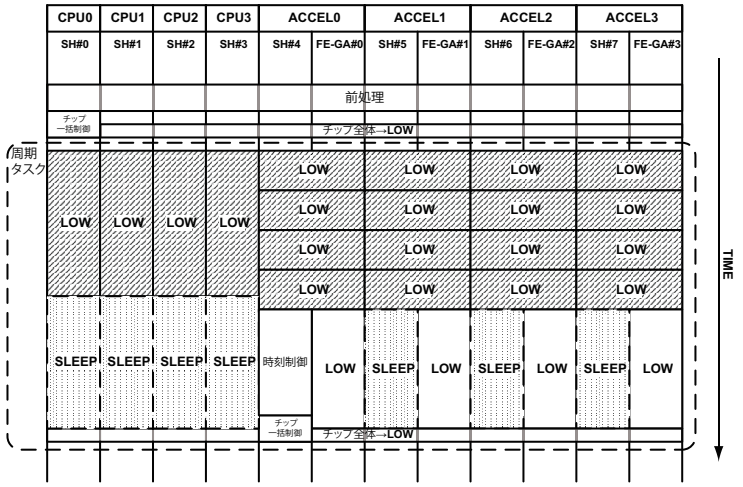


図 10 AAC エンコーダに対する電力制御のイメージ

および早稲田大学グローバル COE“アンビエント SoC” の支援により行われました．また，本研究を進めるにあたり，東北大学大学院情報科学研究科の張山研究室より，オプティカルフロー演算プログラムおよび FE-GA 用ライブラリをご提供いただきました．心より御礼申し上げます．

参 考 文 献

1) Vangal, S., Howard, J., Ruhl, G., Dighe, S., Wilson, H., Tschanz, J., Finan, D., Iyer, P., Singh, A., Jacob, T., Jain, S., Venkataraman, S., Hoskote, Y. and Borkar, N.: An 80-Tile 1.28TFLOPS Network-on-Chip in 65nm CMOS, *Digest of Technical Papers of the 2007 IEEE International Solid-State Circuits Conference*, pp.98–589 (2007).
2) Shiota, T., ichi Kawasaki, K., Kawabe, Y., Shibamoto, W., Sato, A., Hashimoto, T., Hayakawa, F., ichirou Tago, S., Okano, H., Nakamura, Y., Miyake, H., Suga, A. and Takahashi, H.: A 51.2GOPS 1.0GB/s-DMA Single-Chip Multi-Processor Integrating Quadruple 8-Way VLIW Processors, *Digest of Technical Papers of the 2005 IEEE International Solid-State Circuits Conference*, pp.194–593 (2005).
3) Torii, S., Suzuki, S., Tomonaga, H., Tokue, T., Sakai, J., Suzuki, N., Murakami, K., Hiraga, T., Shigemoto, K., Tatebe, Y., Ohbuchi, E., Kayama, N., Edahiro, M.,

Kusano, T. and Nishi, N.: A 600MIPS 120mW 70μA Leakage Triple-CPU Mobile Application Processor Chip, *Digest of Technical Papers of the 2005 IEEE International Solid-State Circuits Conference*, pp.136–589 (2005).
4) Shikano, H., Ito, M., Onouchi, M., Todaka, T., Tsunoda, T., Kodama, T., Uchiyama, K., Odaka, T., Kamei, T., Nagahama, E., Kusaoke, M., Nitta, Y., Wada, Y., Kimura, K. and Kasahara, H.: Heterogeneous Multi-Core Architecture That Enables 54x AAC-LC Stereo Encoding, *IEEE Journal of Solid-State Circuits*, Vol.43, No.4, pp.902–910 (2008).
5) Nakajima, M., Yamamoto, T., Yamasaki, M., Hosoki, T. and Sumita, M.: Low Power Techniques for Mobile Application SoCs Based on Integrated Platform "UniPhier", *Proceedings of the 2007 Conference on Asia South Pacific Design Automation*, pp.649–653 (2007).
6) Seiler, L., Carmean, D., Sprangle, E., Forsyth, T., Abrash, M., Dubey, P., Junkins, S., Lake, A., Sugerman, J., Cavin, R., Espasa, R., Grochowski, E., Juan, T. and Hanrahan, P.: Larrabee: A Many-Core x86 Architecture for Visual Computing, *ACM Transactions on Graphics*, Vol.27, No.3, pp.1–15 (2008).
7) 林　明宏，和田康孝，渡辺岳志，関口　威，間瀬正啓，木村啓二，伊藤雅之，長谷川淳，佐藤真琴，野尻　徹，内山邦男，笠原博徳：情報家電用ヘテロジニアスマルチコア用自動並列化コンパイラフレームワーク，情報処理学会研究報告計算機アーキテクチャ，Vol.2010-ARC-190, No.7 (2010).
8) 白子　準，吉田宗弘，押山直人，和田康孝，中野啓史，鹿野裕明，木村啓二，笠原博徳：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コンピューティングシステム，Vol.47, No.SIG12(ACS15), pp.147–158 (2006).
9) 間瀬正啓，中川　亮，大國直人，白子　準，木村啓二，笠原博徳：マルチコア上でのOSCAR API を用いた並列化コンパイラによる低消費電力化手法，情報処理学会論文誌コンピューティングシステム，Vol.2, No.3, pp.96–106 (2009).
10) : Optimally SCheduled Advanced multiprocessoR Application Program Interface (OSCAR API) Version 1.0, `http://www.kasahara.cs.waseda.ac.jp/`.
11) : OpenMP Application Program Interface, `http://www.openmp.org/`.
12) 和田康孝，林　明宏，益浦　健，白子　準，中野啓史，鹿野裕明，木村啓二，笠原博徳：ヘテロジニアスマルチコアプロセッサ上でのスタティックスケジューリングを用いたMP3 エンコーダの並列化，情報処理学会論文誌コンピューティングシステム，Vol.1, No.1, pp.105–119 (2008).
13) 本多弘樹，岩田雅彦，笠原博徳：Fortran プログラム粗粒度タスク間の並列性検出法，電子情報通信学会論文誌 D-I，Vol.J73-D-I, No.12, pp.951–960 (1990).
14) 笠原博徳，合田憲人，吉田明正，岡本雅巳，本多弘樹：Fortran マクロデータフロー処理のマクロタスク生成手法，電子情報通信学会論文誌 D-I，Vol.J75-D-I, No.8, pp. 511–525 (1992).
15) Kondo, H., Otani, S., Nakajima, M., Yamamoto, O., Masui, N., Okumura, N., Sakugawa, M., Kitao, M., Ishimi, K., Sato, M., Fukuzawa, F., Imasu, S., Kinoshita, N., Ota, Y., Arimoto, K. and Shimizu, T.: Heterogeneous Multicore SoC With SiP for Secure Multimedia Applications, *IEEE Journal of Solid-State Circuits*, Vol.44, No.8, pp.2251–2259 (2009).
16) Yuyama, Y., Ito, M., Kiyoshige, Y., Nitta, Y., Matsui, S., Nishii, O., Hasegawa, A., Ishikawa, M., Yamada, T., Miyakoshi, J., Terada, K., Nojiri, T., Satoh, M., Mizuno, H., Uchiyama, K., Wada, Y., Kimura, K., Kasahara, H. and Maejima, H.: A 45nm 37.3GOPS/W Heterogeneous Multi-Core SoC, *Digest of Technical Papers of the 2010 IEEE International Solid-State Circuits Conference*, pp.100–101 (2010).
17) 間瀬正啓，木村啓二，笠原博徳：並列化コンパイラによるソフトウェアコヒーレンシ制御，情報処理学会研究報告計算機アーキテクチャ，Vol.2010-ARC-189, No.7 (2010).
18) 間瀬正啓，木村啓二，笠原博徳：マルチコアにおける Parallelizable C プログラムの自動並列化，情報処理学会研究報告計算機アーキテクチャ，Vol.2009-ARC-184, No.15 (2009).