# 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

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[†,††]

半導体集積度の向上にともなう消費電力の増大，集積トランジスタ数の増化に対する処理性能向上の鈍化に対処するため，チップ上に複数のプロセッサを集積するマルチコアアーキテクチャ（チップマルチプロセッサ）が大きな注目を集めている．このようなマルチコアアーキテクチャの能力を最大限に引き出し，高実効性能・低消費電力を達成するためには，プログラムの適切な並列化に加えチップ上のリソースのきめ細かな電圧・動作周波数制御を実現するコンパイラが必要不可欠である．本論文では，各プロセッサコアが等価である OSCAR タイプのマルチコアプロセッサにおいて，各プロセッサの電源の ON/OFF・周波数電圧制御（FV 制御）をマルチグレイン並列化環境下でコンパイラが適切に判断し低消費電力化を行うコンパイル手法を提案する．提案手法を実装した OSCAR コンパイラにより，科学技術計算とマルチメディアアプリケーションに対する評価を行った結果，SPEC CFP95 applu において 4 プロセッサ使用時に最小実行時間を維持したまま 60.7%の消費エネルギー削減，MPEG2 エンコーダにおいて 4 プロセッサ使用時にデッドライン制約を保証したまま 82.7%の消費エネルギー削減が達成された．

# Compiler Control Power Saving Scheme for Multicore Processors

Jun Shirako,[†] Munehiro Yoshida,[†] Naoto Oshiyama,[†]
Yasutaka Wada,[†] Hirofumi Nakano,[†] Hiroaki Shikano,[††]
Keiji Kimura[†,††] and Hironori Kasahara[†,††]

A chip multiprocessor architecture has attracted much attention to achieve high effective performance and to save the power consumption, with the increase of transistors integrated onto a chip. To this end, the compiler is required not only to parallelize program effectively, but also to control the volatage and clock frequency of computing resources carefully. This paper proposes a power saving compiling scheme with the multigrain parallel processing environment that controls Voltage/Frequency and power supply of each core on the multiprocessor. In the evaluation, OSCAR compiler with the proposed scheme achieves 60.7 percent energy savings for SPEC CFP95 applu using 4 processors without performance degradation, and 82.7 percent energy savings for MPEG2 encoder using 4 processors added deadline constraint.

## 1. はじめに

半導体集積度向上にともなったスケーラブルな性能向上を達成できるプロセッサアーキテクチャとして，マルチコアアーキテクチャ（チップマルチプロセッサ）が大きな注目を集めている．また近年では処理性能のみでなく，増大する消費電力をいかに抑えるかが大きな課題であり，この問題を克服する手段としてもマルチコアは有望視されている．商用プロセッサにおいても，富士通 FR-V [1]，ARM MPCore [2]，IBM，ソニー，東芝で共同開発された Cell [3]，インテル Xeon dual-core [4] といったように性能向上や低電力を目標とした様々なマルチコアが開発されている．これらマルチコアプロセッサの実効性能向上のためには，プログラムからの適切なグレイン（粒度）での並列性抽出，キャッシュやローカルメモリの最適利用および DMAC 利用を含めたプロセッサ間データ転送オーバヘッドの最小化，そしてそれらの効果的なスケジューリングが必須である．これを実現するために従来より自動並列化コンパイラの研究が行われており [5)~7)]，これらの研究・開発によりループ並列化技術は大きな進歩をとげた．しかしながら現在ではループ並列化手法は成熟期に至っており，今後マルチコアシステム上での大幅な性能向上を達成するためにループ並列性以外の並列性

† 早稲田大学理工学部コンピュータ・ネットワーク工学科
Department of Computer Science, Waseda University
†† アドバンストチップマルチプロセッサ研究所
Advanced Chip Multiprocessor Research Institute

を利用する並列化手法が必要とされている．マルチレベルの並列性を利用するコンパイラとしては NANOS コンパイラ[8]，PROMIS コンパイラ[9]，そして OSCAR コンパイラ[10)~12)] があげられる．OSCAR マルチグレイン自動並列化コンパイラでは，プログラム中の粗粒度タスク並列処理，ループレベル並列処理，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世界で唯一実現している．また OSCAR コンパイラは抽出したマルチグレイン並列性に応じ，プログラムの各部分の並列性に見合った適切なプロセッサの割当てや複数のループ（すなわち粗粒度タスク）間にまたがる広域的なキャッシュメモリ最適化も実現している．この並列性に応じたプロセッサの割当て機能では，プログラム中で並列性が小さい部分においては処理の低オーバヘッド化および低消費電力化のために使用不要と判断されたプロセッサへの電源供給の遮断や，処理終了のデッドライン制約を満たす範囲での電圧・動作周波数の低減といった制御が重要となる．このような低消費電力化手法には様々なものが提案されている．たとえばキャッシュミス回数測定用カウンタや命令キューなどのハードウェアサポートにより実行時にプログラム中の各フェーズにおける負荷を判断し，不必要なリソースを停止する Adaptive Processing[13)] や，計算資源の各部分に対して実行時の負荷に応じた周波数・電圧制御（FV 制御）を行う Online Methods for Voltage and Frequency Control[14)] などがある．また，コンパイラ制御によるシングルプロセッサの低電力手法として compiler-directed DVS（dynamic voltage scaling）[15)] があげられる．

本論文では OSCAR 型チップマルチプロセッサにおいて，

- プログラム各部における不要プロセッサの停止
- プロセッサへの処理不均衡時の FV 制御
- プログラムのデッドライン制約に応じた FV 制御

をコンパイラが自動判定しプログラムに適用することでプロセッサにおける消費電力の削減を図る，コンパイラ制御低消費電力化手法を提案する．

## 2. マルチグレイン並列処理

本章では，提案する低消費電力制御で前提としているマルチグレイン並列処理における粗粒度タスク並列処理について述べる．粗粒度タスク並列処理とは逐次プログラムを階層的に粗粒度タスク分割し，生成された粗粒度タスクすなわちマクロタスクをプロセッサエレメント（PE），もしくはプロセッサグループ（PG）に割り当てて実行することによりマクロタスク間の並列性を利用する並列処理手法である[16)]．

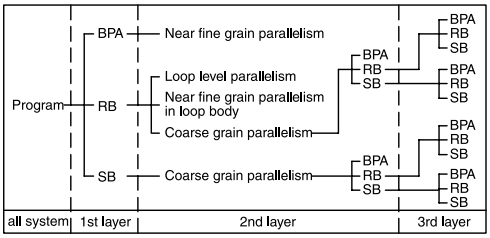


**図 1** 階層的マクロタスク定義
Fig. 1 Hierarchical macrotask definition.

### 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である BPA，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返しブロック RB，サブルーチンブロック SB の 3 種類のマクロタスク MT[12)] すなわち粗粒度タスクに分割される．繰返しブロック RB やサブルーチンブロック SB に対しては，その内部をさらにマクロタスク分割し階層的なマクロタスク構造を生成する（**図 1**）．

### 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，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[10),12)] 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[10),12)] を適用し，階層的なマクロタスクグラフ MTG[10),12)] を生成する．ここで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が粗粒度タスク並列性を表す．

### 2.3 階層的なプロセッサグルーピング

階層的なマクロタスクグラフを効果的に処理するため，プロセッサのグルーピングを行う．複数のプロセッサエレメント PE をソフトウェア的にグループ化し，プロセッサグループ PG を定義する．この PG がマクロタスクを処理する単位であり，SB や RB など内部にマクロタスクグラフが存在する場合はプロセッサグループ内の PE をさらにグルーピングする．**図 2** に階層的なグルーピングの例を示す．図中，プログラム全体で 8 プロセッサ利用可能であるとする．第 1 階層（1st layer）では 4 つの PE を持った 2 つの PG を定義し，これを (2PG, 4PE) の構成と表記する．第 1 階層ではこの 2 つの PG を用い 2 並列の粗粒度タスク並列処理を行い，さらに 4 PE を割り当てられた第 2 階層（2nd layer）ではそれぞれ (4PG, 1PE) で 4 並

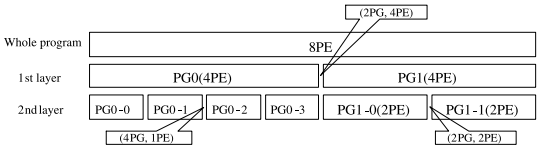


**図 2** プロセッサグループ・プロセッサエレメントの階層的定義
Fig. 2 Hierarchical definition of processor groups and processor elements.

列，(2PG, 2PE) で 2 並列という粗粒度タスク並列処理を行っている．このように階層的なプロセッサ構成を定義することでそれぞれのマクロタスクグラフの並列性を有効に利用することが可能である．

### 2.4 並列処理階層自動決定手法

抽出された各階層の粗粒度タスク並列性をプロセッサに効率良く割り当てることが，マルチグレイン並列処理の性能向上において重要である．OSCAR コンパイラでは並列処理階層自動決定手法[16),17)] により，推定した各階層のマクロタスクグラフの並列度を用いて適切な PG 数・PE 数を決めることで，抽出されたマルチグレイン並列性の効果的な利用を実現している．さらに各マクロタスクの並列性の最大値を推定し，そのマクロタスクを処理するのに必要十分なプロセッサ数を推定する．これにより過剰と判断されたプロセッサにはオーバヘッド最小化のため処理を割り当てず，不要な並列処理オーバヘッドを削減している．

### 2.5 プロセッサグループへのマクロタスク割当て

上述のように各マクロタスクグラフに合わせて生成したプロセッサグループがそのマクロタスクグラフを処理する単位となり，当該マクロタスクグラフ上のマクロタスクを各プロセッサグループに割り当てる．マクロタスクグラフ上に条件分岐がな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コンパイル時に決定される．マクロタスクグラフが条件分岐などの実行時不確定性を含む場合は，実行時にタスクをプロセッサグループに割り当てる動的スケジューリングルーチンをコンパイラが自動生成し，並列プログラム中に埋め込むダイナミックスケジューリング方式がとられる．

本論文ではスタティックスケジューリングを利用した場合の低消費電力制御手法について述べる．下記説明においては，MT はマクロタスク，MTG はマクロタスクグラフ，PG はプロセッサグループ，PE はプロセッサエレメントを表すものとする．また疑似代入文ブロックを BPA，繰返しブロックを RB，サブルーチンブロックを SB と表記する．

## 3. コンパイラによる低消費電力化手法

2 章で述べたマルチグレイン並列処理により，プログラム中に存在するマルチレベル並列性を最大限引き出すことが可能となる．しかし利用可能な計算資源に対しつねに十分な並列性が抽出されるとは限らず，このような場合には必要以上のプロセッサを動作させるための無駄な電力消費が発生してしまう．また，定められた時刻までに処理を終了すればよいようなリアルタイム処理では，ゆっくりと低動作周波数・低電圧で処理を行った方が消費電力を低く抑えられる場合がある．このような最小時間で処理を行う場合と，リアルタイム制約を満たす場合の両方に対する低消費電力化制御手法を提案する．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低消費電力制御を行わず，本手法適用前の最小実行時間を保証する．また，デッドライン制約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で，電力消費を最小になるように制御を行う．

提案手法は，プログラム全域にマルチグレイン並列処理が適用され，各プロセッサに MT およびその部分タスクがスケジューリングされた状態を前提としている．すなわち得られた粗粒度タスクスケジューリング結果に対し，粗粒度タスク間の負荷バランスやマルチグレイン並列化された際の MTG の実行終了時間などを考慮して各 MT の適切な周波数を決定する．さらに各 MT は内部に MTG を含むネスト構造であり，この内部 MTG に対してもマルチグレイン並列化における粗粒度タスクスケジューリングが適用されているため，提案手法による動作周波数の決定および電源制御が可能となる．このため各プロセッサに対し，プログラム全体から最もネストレベルの深いベーシックブロックまでネストされた各 MTG 上のマクロタスクが割り当てられ，どの粒度（ネストレベル）でも FV 制御・電源 ON/OFF の制御が可能となる．提案手法においては，FV 切替えのオーバヘッドやそれぞれの MT のとりうる周波数などを考慮し，適切と判断される粒度に対し FV 制御・電源制御を適用する．

### 3.1 低消費電力制御の対象モデル

本手法の制御対象マルチコアの 1 つとして，OSCAR（Optimally Scheduled Advanced Multiprocessor）[18)~21)] チップマルチプロセッサがあげられる．**図 3** に示すようにチップ上の各コアはローカルメモリ，分散共有メモリ，DMAC を持ち，各コアおよび集中共有メモリ間はマルチバスまたはクロスバーで接続されている．さらに本手法による低消費電力化を実

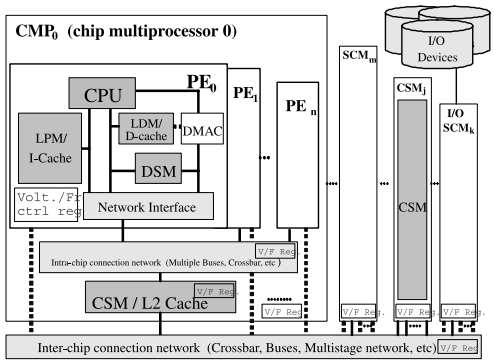


**図 3** OSCAR アーキテクチャ（チップマルチプロセッサ）
Fig. 3 OSCAR architecture (Chip multiprocessor).

**表 1** 各動作周波数におけるパラメータの比率
Table 1 Rates of parameters at each frequency.

| state | FULL | MID | LOW | OFF |
|---|---|---|---|---|
| frequency | 1 | 1/2 | 1/4 | 0 |
| voltage | 1 | 0.87 | 0.71 | 0 |
| dynamic energy | 1 | 3/4 | 1/2 | 0 |
| static power | 1 | 1 | 1 | 0 |

現するための電力制御用レジスタ（FV 制御レジスタ）を持つ．本手法は OSCAR アーキテクチャに限らず，以下の機能を持つマルチプロセッサに適用できる．

- プロセッサコアごとに周波数が可変
- 周波数に応じて電圧も低減可能
- プロセッサコアごとに電源の ON/OFF が可能

周波数・電圧（FV）制御には様々なアプローチがあるが，ここではとりうる周波数の状態は離散的とし，各周波数状態に対して適切な電圧がハードウェア制約により定められているものとする．**表 1** に各周波数における電圧，動作電流による動的消費エネルギー，リーク電流による消費電力の比率の 1 例（FULL を 400 MHz，MID を 200 MHz，LOW を 100 MHz とし 90 nm テクノロジを仮定した例）を示す．ここで，動的消費エネルギーとは同じ clock の処理を行った場合に消費されるエネルギーを表す．電源制御に関しては，完全に電源を遮断するため OFF の状態ではリーク電流による電力消費もないものとするが，プロセッサコアへのクロック供給のみを遮断できる場合は動作電力が 0，リークによる消費電力が通常と同じという状態を追加することができる．これらパラメータ類および状態の数は任意に変えることが可能である．また周波数を切り替えた際に，安定状態となるまでの状態遷移分のオーバヘッドが存在するが，これもパラメータとして与えられるものとする．

このようにプロセッサコアごとに電源の ON/OFF

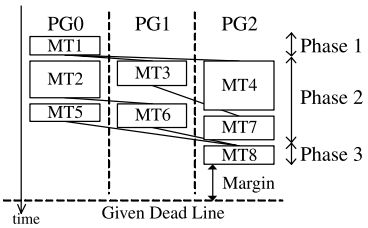


**図 4** MTG のスケジューリング結果
Fig. 4 Scheduling result of MTG.

を実現しているマルチコアプロセッサの例として，SH-Mobile [22),23)] があげられる．SH-Mobile では複数の CPU，DSP が搭載されたチップを 20 個のパワードメインに分け，CPU コアごとの動的な電源シャットダウンが可能である．またコアごとの動作周波数・電圧を変更する技術は現状の商用マルチコアプロセッサにおいてはまだないが，一般に CPU の内部状態を破棄してしまう電源停止に比べて動作周波数・電圧の変更自体は容易に実現可能と考えられる．

### 3.2 低消費電力制御の対象 MTG

条件分岐などの実行時不確定性の有無により各 MTG に対してダイナミックスケジューリングかスタティックスケジューリングかが選択されるが，前述のように本論文ではスタティックスケジューリング適用 MTG のみを制御対象とする．ただし MTG の一部のみが条件分岐を含む場合，条件分岐のない部分的 MTG を擬似的に別階層として扱うことで本手法が適用可能となる．また各動作周波数における演算命令の処理コスト（クロック数）と消費エネルギーのリストを用いることで，各 MT の処理コストと消費エネルギーを算出する．さらにプロファイルを適用することにより，これら推定値の精度を高めることも可能である．

### 3.3 FV 制御適用 MTG の実行終了制約

以下ではスタティックスケジューリングの適用結果をもとに，各 MT に対して適切な動作周波数・電圧を決定する手順を述べる．すなわち**図 4** のように，スタティックスケジューリングにより処理時間が最小となるよう MT が各 PG へ割り当てられているものとする．図中の Time が時間経過を表しており，周波数が FULL の場合の 1 clock を単位時間とする．また MT 間の実線はデータ依存を表している．まず FV 制御適用時の当該 MTG の実行終了時間を表すため，以下の定義を行う．$MT_i$ に対して，

$T_i$：FV 制御適用後の $MT_i$ の処理時間
$T_{start_i}$：$MT_i$ の実行開始時刻
$T_{end_i}$：$MT_i$ の実行終了時刻

を定義する．この時点では $T_i$ は未定であり，当該

MTG の開始時刻を 0 とする．entry node である $MT_1$ のように，当該 PG が最初に実行する MT でありかつ他の MT にデータ依存されていない $MT_i$ の実行開始時刻 $T_{start_i}$ は

$$T_{start_i} = 0$$

となり，実行終了時刻は

$$T_{end_i} = T_{start_i} + T_i = T_i$$

となる．一方それ以外の $MT_i$ に関しては，当該 PG が先行的に実行するマクロタスク $MT_j$ と $MT_i$ がデータ依存するマクロタスク集合 $\{MT_k, MT_l, ...\}$ が終了した時点で $MT_i$ の実行が開始されるため，実行開始時刻は

$$T_{start_i} = max(T_{end_j}, T_{end_k}, T_{end_l}, ...)$$

となり，終了時刻は

$$T_{end_i} = T_{start_i} + T_i$$

となる．図 4 を例として考えると $MT_2$，$MT_3$ は $MT_1$ が終了した時点で実行されるため開始時刻は $T_{start_2} = T_{start_3} = T_{end_1} = T_1$ となり，終了時刻は $T_{end_2} = T_{start_2} + T_2 = T_1 + T_2$，$T_{end_3} = T_{start_3} + T_3 = T_1 + T_3$ である．また $MT_6$ は $MT_2$ と $MT_3$ が終了した時点で実行が開始されるため $T_{start_6} = max(T_{end_2}, T_{end_3})$ より，

$$T_{start_6} = max(T_2, T_3) + T_1$$

となる．同様に計算し，exit node である $MT_8$ の終了時刻は $T_{end_8} = T_1 + T_8 + max(T_2 + T_5, T_6 + max(T_2, T_3), T_7 + max(T_3, T_4))$ と表される．

ここで一般形を考えると，exit node の終了時刻は

$$T_{end_e} = T_m + T_n + ... + max_1(...) + max_2(...) + ...$$

と表記される．entry node の実行開始時刻を 0 としたため，この $T_{end_e}$ が FV 制御を適用した際の当該 MTG の実行終了時間となり，$T_{MTG}$ と表記する．当該 MTG に対して定められた処理終了時間を $T_{MTG\_deadline}$ とし，

$$T_{MTG} \leq T_{MTG\_deadline}$$

を満たすように各 $MT_i$ の動作周波数を決定することが本手法の基本方針である．

### 3.4 FV 制御による低消費電力化

本節では，FV 制御適用時の $MT_i$ の処理時間を $T_i$，FV 制御適用 MTG に定められた処理終了の時間を $T_{MTG\_deadline}$ として

$$T_{MTG} = T_m + T_n + ... + max_1 + max_2 + ... \quad (a)$$

$$T_{MTG} \leq T_{MTG\_deadline} \quad (b)$$

を満たす範囲で各 MT に適切な周波数を決める手法について述べる．便宜上式 (a) の各項 $T_m$，$T_n$，... および $max_1$，$max_2$，... に対応する MT の集合それぞれを Phase と呼ぶこととする．これらの項は当該 MTG の実行時間の一部分であり，各項が表す時間帯が重なることはない．このため，異なる Phase が並列に実行されることもない（図 4）．ここで動作周波数を $F_n$ と表し，$Phase_i$ に対して以下のパラメータを定義する．

$T_{sched_i}(F_n)$：$F_n$ におけるスケジューリング長

$Energy_i(F_n)$：$F_n$ における総消費エネルギー

$T_{sched_i}(F_n)$ は $Phase_i$ 全体を周波数 $F_i$ で処理した際にかかる処理時間であり，$T_{sched_i}(FULL)$ が式 (a) における各項のとりうる最小値である．また $Energy_i(F_n)$ は $Phase_i$ 内に存在する MT を周波数 $F_n$ で処理した場合の消費エネルギーの総和である．ここで $F_n$ から $F_m$ へ $Phase_i$ の周波数を 1 段階落とした場合を考える．スケジューリング長は $T_{sched_i}(F_n)$ から $T_{sched_i}(F_m)$ へ増加し，消費エネルギーは $Energy_i(F_n)$ から $Energy_i(F_m)$ へ減少する．これらを用いて，次のような利得 $Gain_i(F_m)$ を定義する．

$$Gain_i(F_m) = -\frac{Energy_i(F_m) - Energy_i(F_n)}{T_{sched_i}(F_m) - T_{sched_i}(F_n)}$$

$Gain_i(F_m)$ は動作周波数を変化させた際の，スケジューリング長の単位時間あたりの増加に対する消費エネルギーの減少量を表す．すなわちスケジューリング長の増加分が等しい場合，$Gain_i(F_m)$ が大きい $Phase_i$ に優先して FV 制御を適用することで，より大きな消費電力削減が望める．

次に $T_{MTG}$ がとりうる最小値を求めるが，これは $T_{sched_i}(FULL)$ の総和である．この最小値を $T_{MTG\_deadline}$ から引いた値を当該 MTG の余裕度 $T_{MTG\_margin}$ と定義する．すなわち

$$T_{MTG\_margin} = T_{MTG\_deadline} - \sum T_{sched_i}(FULL)$$

である．当該 MTG が最小実行時間で終了しなければならない場合，$T_{MTG\_margin} = 0$ であり各 Phase は動作周波数 FULL で実行と判定される．$T_{MTG\_margin} > 0$ である場合は $T_{MTG\_margin}$ の範囲内で，$Gain_i(F_m)$ に応じて各 Phase の電圧・動作周波数を低減させる．つまり消費エネルギーがより低減しやすい箇所の周波数を優先して落とすことで，効果的な低消費電力化が可能となる．ここで Phase が単一の MT ならば MT の動作周波数は Phase と同じとなる．Phase が複数 MT の集合であり max の項に対応する場合は max の各引数に対して同様に Phase を定義し，動作周波数を決定する．以下に提案する FV 制御手法の手順を述べる．各 Phase の初期動作周波数はすべて FULL と

する．

#### 3.4.1 各 Phase に対する FV 制御アルゴリズム

**step.1 対象 Phase の選択**

各 Phase において，現在の動作周波数を $F_n$，それより 1 段階低い周波数を $F_m$ とする．周波数の決定していない中で（初期状態では全 Phase），利得 $Gain_i(F_m)$ が最も大きい $Phase_i$ を対象 Phase として選択する．**step.2** へ

**step.2 対象 Phase の FV 制御適用判定**

対象 Phase に対して，$F_n$ から $F_m$ へ周波数を変更する条件を以下のとおりとする．

(1) 周波数切り替え時の状態遷移のオーバヘッドも含め，余裕度 $T_{MTG\_margin}$ の範囲内で周波数 $F_m$ で処理が終了する．
(2) $F_m$ 時の電力消費が FV 制御非適用時や電源遮断時よりも小さい．

両方の条件を満たすか？

Yes：対象 Phase を $F_m$ に変更．**step.3** へ

No：対象 Phase を $F_n$ と決定．**step.4** へ

**step.3 余裕度 $T_{MTG\_margin}$ の更新**

対象 Phase をオーバヘッドも含め動作周波数 $F_m$ で処理するのに必要な時間を計算し，$T_{MTG\_margin}$ から引く．また対象 Phase の最低の動作周波数が $F_m$ でありこれ以上落とせない場合，動作周波数を $F_m$ と決定する．**step.4** へ

**step.4 終了判定**

終了条件は以下のとおりである．

(1) すべての Phase の周波数が決定する，
(2) $T_{MTG\_margin}$ が 0 となる，

のいずれかを満たすか？

Yes：終了．

No：継続．**step.1** へ

終了時に $T_{MTG\_margin}$ が 0 でない場合，残りの $T_{MTG\_margin}$ は以下の条件を満たす $Phase_i$ の余裕度として与えられる．

- 周波数が最低でない
- 利得 $Gain_i(F_m)$ が最も大きい

この時点ですべての Phase の周波数が決定する．次に各 Phase 内 MT に対する動作周波数決定に移る．

#### 3.4.2 Phase 内に対する FV 制御アルゴリズム

**step.1 Phase を構成する MT による分類**

Phase が単一の MT であるか？

Yes：Phase の周波数をその MT の周波数とし，終了．

No：**step.2** へ

**step.2 $max$ 項に対する FV 制御**

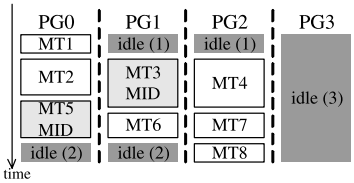


**図 5** FV 制御適用結果
Fig. 5 Result of FV control.

Phase が複数の MT の集合であり $max_i$ と対応する場合は，定められた動作周波数で Phase 全体を実行した際の実行時間を求め，これを $T_{max_i\_deadline}$ とする．$max_i$ と，その引数 $arg_{i\_j}$ は以下のとおり．

$$max_i = max(arg_{i\_1}, arg_{i\_2}, ...)$$
$$arg_{i\_j} = T_{i\_j\_m} + T_{i\_j\_n} + ... + max_{i\_j\_1} + ...$$

すなわち $arg_{i\_j}$ は次の条件を満たすものとする．

$$T_{i\_j\_m} + T_{i\_j\_n} ... + max_{i\_j\_1} + ... \leq T_{max_i\_deadline}$$

上式の各項に対応する MT の集合を Phase と定義し，Phase に対する周波数決定 3.4.1 項に進む．

この際，各 $arg$ の動作周波数 FULL 時の実行時間を計算し実行時間の大きい順（余裕度の小さい順）に FV 制御を適用する．複数の $arg$ 中に同一の MT を表す項が含まれる場合があるが，一度 FV 制御が適用された MT の周波数はすでに決定されているため，変更はされない．

3.4.1 項，3.4.2 項を再帰的に適用することで全 MT に対して適切な周波数が決定される．

### 3.5 電源制御による低消費電力化

3.4 節のように当該 MTG に対して FV 制御が適用され，次に電源の ON/OFF を制御することでプロセッサが処理を行っていない idle 部分の省電力化を図る．MTG 内に idle 部分が発生するのは次の 3 種類である．

(1) データ依存がある MT の開始前
(2) 全割当て MT の終了時
(3) 並列処理階層決定手法による idle

**図 5** にそれぞれの場合の idle を示す．ここで PG3 は階層決定手法により不要と判断されたプロセッサであり，処理は割り当てられていない．以下の条件を満たす idle 部分に対し，電源遮断を適用する．

- 状態遷移オーバヘッドも含め，idle の時間で電源 ON/OFF の切替えが可能
- 電源遮断により電力消費が削減可能

### 3.6 MT の内部 MTG に対する低消費電力化

以上により当該 MTG の各 $MT_i$ に対し適切な動作周波数が決定される．しかし $MT_i$ 内部にさらに $MTG_i$ が定義される場合，$MTG_i$ 上に存在する MT

をさらに細かく制御した方がより消費電力を抑えられる場合がある．このため $MT_i$ の実行時間 $T_i$ より $MTG_i$ が満たさなければならない実行終了時間 $T_{MTG_i\_deadline}$ を求め，同様に FV 制御と電源制御を適用する．このように $MTG_i$ の内部をさらに細かく制御した場合と $MTG_i$ 内部は制御せず $MT_i$ 全体を同一の周波数で処理をした場合の消費エネルギーを比較し，より省電力効果の高い方を選択する．以下に具体的なアルゴリズムを示す．

**step.1** $T_{MTG_i\_deadline}$ **の決定**

$T_i(FULL)$ を $MT_i$ 全体を周波数 FULL で実行した際の処理時間，$num\_of\_iter_i$ を $MT_i$ が RB である場合は回転数（RB 以外の場合は 1）と定義し，以下のように定める．

$$T_{MTG_i\_deadline} = (T_i - T_i(FULL)) \ / \ num\_of\_iter_i$$

**step.2** $MTG_i$ **の総消費エネルギー算出**

$MTG_i$ 上の j 番目の MT を $MT_{i\_j}$ とする．**step.1** で定められたデッドラインをもとに 3.4 節の FV 制御，3.5 節の電源制御を適用し，$MT_{i\_j}$ の消費エネルギー $Energy_{MT_{i\_j}}$ を決定する．これらを総和し，$MTG_i$ の総消費エネルギーは $\sum Energy_{MT_{i\_j}}$ となる．

**step.3 消費エネルギーの比較**

$MT_i$ が RB であったときはその回転数 $num\_of\_iter_i$ を考慮し，FV の切替えが行われた回数をカウントすることで求めたオーバヘッド分のエネルギーを $overhead_i$ として，2 つの場合の総消費エネルギー

- $MT_i$ を同一周波数で実行：$Energy_{MT_i}$
- $MTG_i$ の内部を制御：$(\sum Energy_{MT_{i\_j}} + overhead_i) \times num\_of\_iter_i$

を比較することでよりエネルギーの低い方を選択する．

## 4. 性能評価

本章では提案する低消費電力化制御手法を OSCAR コンパイラに実装し，コンパイラ内で見積もられた電力消費推定結果について述べる．評価に用いた周波数，電力の比率などのパラメータは表 1 に，平均消費電力および FV 切替え時の状態遷移時間を**表 2** に示す．90 nm のテクノロジの組み込み向け低電力用途プロセッサを想定し，通常時（FULL）の周波数は 400 MHz とする．今回はプロセッサに対する電力消費の評価であるため，メモリやバスの消費電力は含めていない．表 2 中，動的電力は Wattch[24] を用いた電力シミュレータより測定し，静的電力については本評価で前提にしている低電力用途プロセッサの一例として SH-mobile の電力評価論文[25] を参考にした．

**表 2** 消費電力および状態遷移時間
Table 2 Power and frequency transition overhead.

| | |
|---|---|
| dynamic power | 220 [mW] |
| static power | 2.2 [mW] |
| delay(FULL - MID - LOW) | 0.1 [ms] |
| delay({FULL, MID, LOW} - OFF) | 0.2 [ms] |

130 nm の SH-mobile において CPU コアのみの動作電流は 94.5 mA，CPU コア，URAM，バックアップレジスタを含めたリーク電流が 2.2 mA との測定値が記載されており，本評価で想定しているプロセッサにおいても同程度の比率であるとして CPU コアのみに対するリーク電流が動作電流の 100 分の 1 と設定した．状態遷移時間や電圧・動作周波数の関係についてはシングルプロセッサ（SH-4）上での Cooperative Voltage Scaling 手法[26] において，動作電圧を 1.2 V から 2.0 V に切り替える際の状態遷移時間が 0.1 [ms]，2.0 V から 1.2 V への遷移時間が 0.2 [ms] という値が実測されており，これを参考にした．評価には MediaBench に収録されている MPEG2 エンコードプログラム “mpeg2encode” を Fortran で参照実装したプログラム，および SPEC CFP95 より 101.tomcatv，102.swim，110.applu を用いた．今回は組み込み系を想定しているため，プロファイルを適用して回転数推定の精度などを高めている．また applu では，実行時間のほとんどを占めるループボディから条件分岐のない部分を手動で別階層に分離し，この部分にインライン展開とループ分割を行いデータローカリティを高めるようリストラクチャリングを行った．また MPEG2 エンコーダについては，エンコーディング自体の性能評価を行うため，対応する 7 つのステージ（動き推定，動き予測，DCT モード選択，データ変換，ビットストリーム出力，逆量子化，逆データ変換）を性能評価の対象とした[27),28)]．

### 4.1 実行時間最小スケジューリングモード

実行時間最小スケジューリングモードでの速度向上率を**図 6** に，総消費エネルギーを**図 7** に示す．それぞれ横軸がプロセッサ数であり，左のバーが本手法を用いない場合，右が本手法を適用した結果である．図 6 のように最小スケジューリングモードでは本手法適用による性能低下は見られず，適用前の処理性能を維持していることが分かる．また消費エネルギーについては，提案手法を用いることで applu では 2 プロセッサでの実行時に 36.3%（102 [J] から 65.0 [J]），4 プロセッサ時に 60.7%（174 [J] から 68.4 [J]），mpeg2enc では 2 プロセッサ時に 5.55%（1045 [mJ] から 987 [mJ]），4 プロセッサ時に 27.2%（1336 [mJ] から 973 [mJ]）の

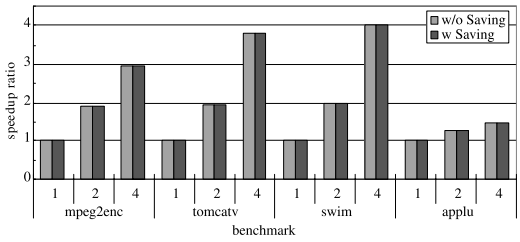


**図 6** 実行時間最小スケジューリングモードでの速度向上率
Fig. 6 Speedup in fastest execution mode.

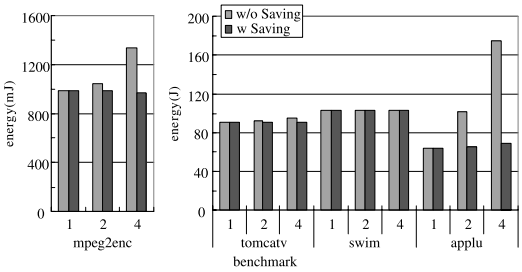


**図 7** 実行時間最小スケジューリングモードでの総消費エネルギー
Fig. 7 Energy in fastest execution mode.

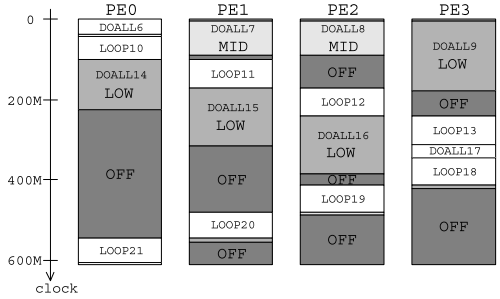


**図 8** applu の低消費電力制御適用結果（4PE）
Fig. 8 Power saving result of applu (4PE).

電力削減が見られた．これらのアプリケーションには十分な並列性が引き出せずプロセッサがアイドルとなる箇所が存在し，提案手法を適用することでアイドル状態の不要な電力消費を削減されるためである．**図 8** は applu でローカリティ抽出のリストラクチャリングを適用した階層に対し，提案手法が FV・電源制御を適用した結果である（4 プロセッサ時）．クリティカルパスに相当する DOALL6，LOOP10〜13，DOALL17，LOOP18〜21，DOALL22 は余裕度が 0 であるため動作周波数 FULL と決定され，それ以外の MT はそれぞれの余裕度に応じて MID および LOW と決定された．さらに idle となる部分に対しては電源遮断が適用され，低消費電力化が図られている．一方，tomcatv，swim はプログラム全体を通して十分な並列性が存在し，最小の処理時間を保証するためにはつねに FULL で動作しなければならないため電力削減は見られなかった．

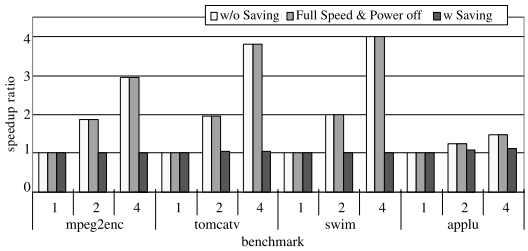


**図 9** デッドライン制約モードでの速度向上率
Fig. 9 Speedup in deadline mode.

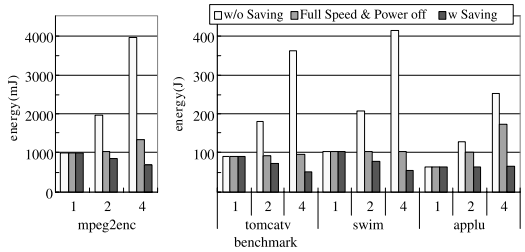


**図 10** デッドライン制約モードでの総消費エネルギー
Fig. 10 Energy in deadline mode.

### 4.2 デッドライン制約モード

**図 9** にデッドライン制約モードでの速度向上率，**図 10** に総消費エネルギーを示す．本評価では 1 プロセッサの処理時間をデッドラインとした．左のバーは省電力手法を適用しない場合であり，プログラム終了後も各プロセッサはデッドラインに達するまで電力を消費し続ける．中央のバーはプログラムの実行は最速の周波数 FULL で行い，実行終了後電源を OFF にする省電力制御を行ったものである．右のバーが提案手法による低消費電力化を行った場合を表す．図 9 より，提案手法を用いた場合の速度向上率は全ベンチマーク，全プロセッサ数において 1 以上となり，デッドライン制約条件を満たしている．また図 10 より，提案手法を用いることで mpeg2enc で 2 プロセッサ時に 56.4%（1973 [mJ] から 861 [mJ]），4 プロセッサ時に 82.7%（3945 [mJ] から 683 [mJ]），tomcatv で 2 プロセッサ時に 60.1%（181 [J] から 72.2 [J]），4 プロセッサ時に 85.6%（361 [J] から 51.9 [J]），swim で 2 プロセッサ時に 62.0%（207 [J] から 78.7 [J]），4 プロセッサ時に 86.7%（414 [J] から 55.2 [J]），applu で 2 プロセッサ時に 50.0%（127 [J] から 63.3 [J]），4 プロセッサ時に 74.0%（253 [J] から 65.8 [J]）消費エネルギー削減という結果となった．また実行終了後に電源を OFF にするという省電力制御を行った場合に比べて mpeg2enc で 2 プロセッサ時に 17.6%

**表 3** 主要 MTG に内包される Phase
Table 3 Phase in dominant MTG.

| ベンチマーク | MTG | Phase 数 | MT の clock |
|---|---|---|---|
| mpeg2enc | メインループ | 148 | $37.8 \times 10^5$ |
| tomcatv | プログラム全体 | 5 | $85.5 \times 10^8$ |
| | メインループ | 9 | $59.4 \times 10^5$ |
| swim | プログラム全体 | 6 | $77.4 \times 10^8$ |
| | メインループ | 4 | $12.8 \times 10^6$ |
| | CALC1 | 5 | $39.5 \times 10^5$ |
| | CALC2 | 5 | $42.4 \times 10^5$ |
| | CALC3 | 2 | $43.0 \times 10^5$ |
| applu | メインループ | 35 | $34.1 \times 10^6$ |

(1045 [mJ] から 861 [mJ])，4 プロセッサ時に 48.9% (1336 [mJ] から 683 [mJ])，tomcatv で 2 プロセッサ時に 21.6% (92.1 [J] から 72.2 [J])，4 プロセッサ時に 45.4% (95.0 [J] から 51.9 [J])，swim で 2 プロセッサ時に 23.7% (103 [J] から 78.7 [J])，4 プロセッサ時に 46.5% (103 [J] から 55.2 [J])，applu で 2 プロセッサ時に 37.8% (102 [J] から 63.3 [J])，4 プロセッサ時に 62.2% (174 [J] から 65.8 [J]) 電力消費を抑えている．このようにデッドライン制約の範囲で動作周波数・電圧を適切に低減させることにより，大幅な電力削減を達成した．また 4 プロセッサで提案手法を適用した場合，swim において 1 プロセッサ時の処理性能を維持したまま 46.5%消費エネルギーを抑制し，mpeg2enc, tomcatv においても逐次処理性能を達成したうえでそれぞれ 30.8%，42.6%消費エネルギーを削減した．これらのエネルギー削減は FV 制御の効果である．今回の評価ではリーク電流の小さい低電力用途プロセッサを仮定しており，消費エネルギーの大部分は動的電力によるものである．動的電力は次式のように動作電圧の 2 乗と周波数に比例し，

$$dynamic\ power \propto frequency \times voltage^2$$

周波数を FULL から LOW に変えた場合，表 1 より動的電力は $0.71^2 \times 1/4 = 1/8$ となる．一方，処理時間は動作周波数に比例するとして 4 倍となり，FULL から LOW へ FV 制御を適用した場合の消費エネルギーは $1/8 \times 4 = 1/2$ に低減する．このため 4 コアによる低消費電力制御並列処理においては，デッドライン制約に応じた FV 制御を適用することにより 1 プロセッサ時よりも大幅に低い電力消費での実行が可能となった．

### 4.3 各ベンチマークプログラムにおけるフェーズ数と粒度について

**表 3** に 4 プロセッサで並列化した際の各ベンチマークの主要な MTG におけるフェーズ数と，フェーズの最小単位である MT の周波数 FULL における平均実行クロック数を示す．mpeg2enc ではメインループのループボディにあたる MTG においてマクロブロックレベルの並列性が粗粒度タスク並列性として利用でき，これら MT を各プロセッサにスケジューリングした結果 148 個の Phase が定義された．提案手法を適用した結果，これら Phase における負荷のロードバランス不均衡やプログラム全体のデッドライン制約に応じた FV 制御，電源の ON/OFF の制御が当該 MTG 中の MT に対して適用された．また当該 MTG 上に存在する MT の平均実行クロックは $37.8 \times 10^5$ であるが，FV 制御における状態遷移オーバヘッドは表 2 より $0.1\,[\mathrm{ms}] \times 400\,[\mathrm{MHz}] = 40 \times 10^3$，電源 ON/OFF に関しては $0.2\,[\mathrm{ms}] \times 400\,[\mathrm{MHz}] = 80 \times 10^3$ であり，状態遷移オーバヘッド分の電力消費は提案手法による省電力効果に比べ十分小さいと推定される．applu においても同様に，図 8 のようにスケジューリング結果に対し FV 制御・電源シャットダウンが適用される．この際の Phase 数は 35，平均実行クロック数は $34.1 \times 10^6$ となった．

tomcatv ではメインプログラム中に処理のほとんどを占めるメインループがあるが，提案手法ではプログラムの上位階層すなわちネストレベルの浅い方から順に FV 制御を適用する．このためプログラム全体を MTG とした場合にメインループは 1 つの MT となり，周波数 MID と判定された．次にメインループのループボディにあたる MTG に対して提案手法が適用され，当該 MTG 上の MT ごとに周波数を制御した場合の総電力消費とメインループ全体を MID で実行した場合の電力消費を比較した結果，MT ごとの FV 制御を行うと判断された．さらにこれら MT 内部の MTG に対しても提案手法が適用されるが，この場合はより粒度の細かい FV 制御は適用しないと判定された．swim については，メインプログラムにあたる MTG 上に処理のほとんどを占めるメインループが存在し，このループ内にサブルーチン CALC1, CALC2, CALC3 が内包される．上位階層から順に提案手法を適用した結果，メインループ全体を周波数 LOW で処理すると判定された．

## 5. ま と め

本論文では，コンパイラの制御による低消費電力化手法を提案した．提案手法には実行時間最小スケジューリングモードとデッドライン制約モードが存在し，本手法適用前の実効性能の保証，および与えられたデッドライン制約を満たす範囲内での電力最小化といった様々な要求にフレキシブルに対応可能である．

提案手法を OSCAR コンパイラに組み込み，コン

パイラ内で推定された消費エネルギーを検証したところ SPEC CFP95 の applu で最小処理時間での実行を保証したまま最大 60.7%の消費エネルギー削減，MPEG2 エンコーダではデッドライン制約を単一プロセッサでの処理時間とした場合において，デッドライン制約を満たしつつ最大 82.7%の消費エネルギー削減を達成した．

今後の課題としてはシミュレータ上で電力など様々なパラメータを変化させた場合の詳細な評価，ダイナミックスケジューリング適用時での低消費電力制御などがあげられる．

**謝辞** 本研究の一部は NEDO “先進ヘテロジニアスマルチプロセッサ研究開発”，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”，および NEDO “リアルタイム情報家電用マルチコア技術” Multi core processors for real time consumer electronics の支援により行われた．

## 参 考 文 献

1) Suga, A. and Matsunami, K.: Introducing the FR 500 embedded microprocessor, Vol.20, pp.21–27 (2000).
2) Cornish, J.: Balanced Energy Optimization, *International Symposium on Low Power Electronics and Design* (2004).
3) Pham, D., et al.: The Design and Implementation of a First-Generation CELL Processor, *Proc. IEEE International Solid-State Circuits Conference* (2005).
4) Intel. http://www.intel.com/multi-core/
5) Wolfe, M.: *High Performance Compilers for Parallel Computing*, Addison-Wesley Publishing Company (1996).
6) Eigenmann, R., Hoeflinger, J. and Padua, D.: On the Automatic Parallelization of the Perfect Benchmarks, *IEEE Trans. parallel and distributed systems*, Vol.9, No.1 (1998).
7) Hall, M.W., Anderson, J.M., Amarasinghe, S.P., Murphy, B.R., Liao, S., Bugnion, E. and Lam, M.S.: Maximizing Multiprocessor Performance with the SUIF Compiler, *IEEE Computer* (1996).
8) Gonzalez, M., Martorell, X., Oliver, J., Ayguade, E. and Labarta, J.: Code Generation and Run-time Support for Multi-level Parallelism Exploitation, *Proc. 8st International Workshop on Compilers for Parallel Computing* (2000).
9) Saito, H., Stavakos, N. and Polychronopoulos, C.: Multithreading Runtime Support for Loop and Functional Parallelism, *Proc. International Symposium on High Performdance Computing* (1999).
10) 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手法，電子情報通信学会論文誌，Vol.J73-D-I, No.12, pp.951–960 (1990).
11) Kasahara, H., et al.: A Multi-grain Parallizing Compilation Scheme on OSCAR, *Proc. 4th Workshop on Language and Compilers for Parallel Computing* (1991).
12) 笠原博徳：最先端の自動並列化コンパイラ技術，情報処理，Vol.44, No.4, pp.384-392 (2003).
13) Albonesi, D.H., et al.: Dynamically tuning processor resources with adaptive processing, *IEEE Computer* (2003).
14) Wu, Q., Juang, P., Martonosi, M. and Clark, D.W.: Formal Online Methods for Voltage/Frequency Control in Multiple Clock Domain Microprocessors, *11th International Conference on Architectural Support for Programming Languages and Operating Systems* (2004).
15) Hsu, C. and Kremer, U.: The Design, Implementation, and Evaluation of a Compiler Algorithm for CPU Energy Reduction, *The ACM SIGPLAN Conference on Programming Language Design and Implementation* (2003).
16) 小幡元樹，白子 準，神長浩気，石坂一久，笠原博徳：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，Vol.44, No.4 (2003).
17) 白子 準，長澤耕平，石坂一久，小幡元樹，笠原博徳：マルチグレイン並列性向上のための選択的インライン展開手法，情報処理学会論文誌，Vol.45, No.5 (2004).
18) 笠原博徳，成田誠之助，橋本 親：OSCAR（Optimally Scheduled Advanced Multiprocessor）のアーキテクチャ，電子情報通信学会論文誌，Vol.J71-D, No.8 (1988).
19) Kasahara, H., Honda, H., Iwata, M. and Hirota, M.: A Compilation Scheme for Macrodataflow computation on Hierarchical Multiprocessor System, *Proc. Int Conf. on Parallel Processing* (1990).
20) Kasahara, H., Honda, H. and Narita, S.: Parallel Processing of Near Fine Grain Tasks Using Static Scheduling on OSCAR, *Proc. Supercomputing '90* (1990).
21) 木村啓二，尾形 航，岡本雅巳，笠原博徳：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 (1999).
22) Kanno, Y., et al.: Hierarchical Power Distribution with 20 Power Domains in 90-nm Low-

Power Multi-CPU Processor, *IEEE ISSCC* (2006).
23) Hattori, T., et al.: A Power Management Scheme Controlling 20 Power Domains for a Single-Chip Mobile Processor, *IEEE ISSCC* (2006).
24) Brooks, D., Tiwari, V. and Martonosi, M.: Wattch: A Framework for Architectural-Level Power Analysis and Optimizations, *Proc. 27th ISCA* (2000).
25) Ishikawa, M., et al.: A 4500 MIPS/W, 86 $\mu$A Resume-Standby, 11 $\mu$A Ultra-Standby Application Processor for 3G Cellular Phones, *IEICE TRANS. ELECTRON.*, Vol.E88-C (2005).
26) Kawaguchi, H., Shin, Y. and Sakurai, T.: $\mu$ITRON-LP: Power-Conscious Real-Time OS Based on Cooperative Voltage Scaling for Multimedia Applications, *IEEE Trans. multimedia* (2005).
27) 中野啓史ほか：マルチコアプロセッサ上でのデータローカライゼーション，情報処理学会 ARC 研究報告 (2005).
28) 小高 剛，中野啓史，木村啓二，笠原博徳：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，Vol.46, No.9 (2005).

(平成 18 年 1 月 27 日受付)
(平成 18 年 5 月 24 日採録)

**白子 準**（学生会員）
昭和 54 年生．平成 14 年早稲田大学理工学部電気電子情報工学科卒業．平成 16 年同大学大学院修士課程修了．平成 16 年同大学院博士課程進学．平成 17 年早稲田大学理工学部助手，現在に至る．

**吉田 宗弘**
昭和 58 年生．平成 18 年早稲田大学理工学部電気電子情報工学科卒業．平成 18 年同大学大学院修士課程進学，現在に至る．

**押山 直人**
昭和 58 年生．平成 17 年早稲田大学理工学部電気電子情報工学科卒業．平成 17 年同大学大学院修士課程進学，現在に至る．

**和田 康孝**（学生会員）
昭和 54 年生．平成 14 年早稲田大学理工学部電気電子情報工学科卒業．平成 16 年同大学大学院修士課程修了．平成 16 年同大学院博士課程進学．平成 18 年早稲田大学理工学部助手，現在に至る．

**中野 啓史**（学生会員）
昭和 52 年生．平成 13 年早稲田大学理工学部電気電子情報工学科卒業．平成 15 年同大学大学院修士課程修了．平成 15 年同大学院博士課程進学，現在に至る．

**鹿野 裕明**（学生会員）
昭和 52 年生．平成 12 年中央大学理工学部情報工学科卒業．平成 14 年同大学大学院修士課程修了．平成 14 年（株）日立製作所入社．平成 18 年早稲田大学大学院博士課程入学，現在に至る．

**木村 啓二**（正会員）
昭和 47 年生．平成 8 年早稲田大学理工学部電気工学科卒業．平成 13 年同大学大学院理工学研究科電気工学専攻博士課程修了．博士（工学）．平成 11 年早稲田大学理工学部助手．平成 16 年同大学理工学部コンピュータ・ネットワーク工学科専任講師．平成 17 年同助教授，現在に至る．マルチグレイン並列処理用チップマルチプロセッサアーキテクチャに関する研究に従事．

**笠原　博徳**（正会員）

昭和 32 年生．昭和 55 年早稲田大学理工学部電気工学科卒業．昭和 60 年同大学大学院博士課程修了，工学博士．昭和 58 年〜60 年早稲田大学助手．昭和 60 年学振第 1 回特別研究員．昭和 61 年早稲田大学理工学部専任講師．昭和 63 年助教授．平成 9 年教授．現在 CS 学科教授，アドバンストチップマルチプロセッサ研究所所長．昭和 60 年カリフォルニア大学バークレー，平成元年〜2 年イリノイ大学 Center for Supercomputing R & D 客員研究員．昭和 62 年 IFAC World Congress 第 1 回 Young Author Prize，平成 9 年情処坂井記念特別賞，平成 16 年 STARC 共同研究賞受賞．主な著書『並列処理技術』（コロナ社）．本会 ARC 主査，論文誌 HG 主査，会誌 HWG 主査，ACM ICS Program Vice Chair，IEEE CS Japan Chair，文科省地球シミュレータ中間評価委員，経産省/NEDO コンピュータ戦略 WG 委員長，“アドバンスト並列化コンパイラ”，“リアルタイム情報家電用マルチコア” 等プロジェクトリーダ．