Vol.2009-ARC-183 No.35
Vol.2009-OS-111 No.35
2009/4/24

# 性能制約下における共有資源へのアクセス制御とDVFSを用いたチップマルチプロセッサの省電力化

高木紀子[†1]　佐々木広[†2]
近藤正章[†3]　中村宏[†1,†2]

近年主流なアーキテクチャとなっているチップマルチプロセッサ（CMP）では，複数のコアが最下位レベルのキャッシュやメモリバスなどのメモリ階層を共有することが一般的である．共有資源で競合が発生すると，性能制約のあるアプリケーションの場合，制約を満たすために周波数が上昇し消費電力の増加を招くこととなる．本稿は競合下において CMP 全体の消費電力が最小となる条件を明らかにし，複数の共有資源へのアクセス制御により CMP 全体の消費電力を最小化する手法を提案する．評価の結果，DVFS のみを適用した場合に比べ，2 コアの CMP の場合では 15%，4 コアの CMP の場合では 13%の消費電力削減を達成できることがわかった．

# Shared Resource Access Control with DVFS for Low-Power Chip Multiprocessors

Noriko Takagi,[†1] Hiroshi Sasaki,[†2] Masaaki Kondo[†3] and Hiroshi Nakamura[†1,†2]

In a single Chip Multiprocessor (CMP), cores usually share resources in memory hierarchy, including the last-level cache and a memory bus. If applications on diferent cores have their own performance constraint, though the constraint can be satisfed by DVFS control of each core, conficts in shared resources lead to the waste of power consumption. Thus, in this paper, we derive the condition where the total power consumption becomes minimum by construction power consumption model under resource conficts, and propose a method to minimize the power consumption of CMPs by access control to multiple shared resource with DVFS. The experimental results reveals that our technique can reduce 15% of power consumption on average in dual-core CMP, and 13% in quad-core CMP, compared with the case where only DVFS is applied.

## 1. はじめに

近年，VLSI システムの消費電力の増大が重大な問題となっており，チップマルチプロセッサ（CMP）がその高い消費電力効率から主流のアーキテクチャとなりつつある．CMP は複数のプロセッサコアを 1 チップ上に搭載し処理を並列に行うことで，周波数の上昇に頼らず性能を上げることができるため，従来のユニプロセッサシステムに比べ低消費電力である．

CMP は，資源の有効活用の観点から最下位のキャッシュを共有することが多く，またメモリバスに関してもチップピンのバンド幅の制限により共有することが一般的である．これら共有資源において競合が発生すると，共有資源にアクセスしたコアの性能は低下し，その結果として CMP 全体の性能が低下することで，実行中のプログラムの性能予測が困難になることもある．共有資源の競合の問題に対し，従来よりいくつかの研究がなされている．共有キャッシュの競合の問題に対し，競合するスレッド間で共有キャッシュを分割するキャッシュパーティショニング[1)-4)] が提案されており，またメインメモリに関しては CMP 向け SDRAM メモリスケジューラ[11)]，またメモリバスでの競合には優先度制御[9)] が提案されている．

一方，消費電力の削減に関しては，動的電源電圧制御（Dynamic Voltage/Frequency Scaling: DVFS）が，特に実時間制約下のシステムにおいて広く用いられている．DVFS により実時間制約を満たしつつクロック周波数・電源電圧を下げることでプロセッサの消費電力を削減することが可能である．

今までに多くの DVFS 手法が提案されているが，共有資源の競合が発生する場合, 周波数・電圧の最適化は困難な課題である．実時間制約下では，競合の発生により性能が低下するため，制約を満たすために周波数を上げる必要があり，その結果コアの消費電力の増加を招くこととなる．逆に言えば，競合の影響を制御することで，各コアの消費電力を制御することが可能であると言える．

図 1 は，2 コアの CMP における共有資源の各 PU への割り当てと消費電力の関係を表した概念図である．共有資源の量は一定であると仮定する．横軸は割り当てられる資源の量

†1 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo
†2 東京大学先端科学技術研究センター
Research Center for Advanced Science and Technology, The University of Tokyo
†3 電気通信大学 大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications

Vol.2009-ARC-183 No.35
Vol.2009-OS-111 No.35
2009/4/24

を表わし，左に行くほど $PU_0$ に多くの資源が割り当てられ，右に行くほど $PU_1$ に多くの資源が割り当てられることを意味する．共有資源の割り当ては，例えばキャッシュパーティショニングやメモリバスの優先度制御により制御することを想定する．縦軸は各 PU の消費電力と CMP 全体での消費電力を表わしている．この例は，$PU_1$ の性能制約が $PU_0$ よりも厳しい場合を仮定している．

より多くの資源を $PU_1$ が使用するほど，共有資源の競合による性能の低下が小さくなり周波数・電圧が低下するため，$PU_1$ の消費電力は減少する．この場合，単純な資源割り当て戦略では，電力を最小化できないこともあると考えられる．

本稿では，複数の共有資源へのアクセスを協調制御し DVFS を用いて CMP 全体の消費電力を削減する手法を提案する．対象の CMP にはメモリバスと L2 キャッシュの複数の共有資源が存在し，実行されるアプリケーションは性能性約を持っているものを想定する．各共有資源へのアクセス制御には，メモリバスは優先度制御[9]，L2 キャッシュはキャッシュパーティショニング[1)-4)] を使用し，これらを協調制御する．本稿における電力削減に向けた重要な戦略は，1）競合時の性能と消費電力のモデルを構築することにより，電力を最小化するためのキャッシュパーティショニングと優先度制御の最適制御条件を導出し，2）その条件を実現するための DVFS，キャッシュパーティショニング，優先度制御の協調制御手法を構築することである．

本稿の貢献は以下の 2 点である．

- CMP の性能と消費電力のモデルを構築し，全体の消費電力を最小化する最適制御条件を導出する．
- キャッシュパーティショニングと優先度制御，DVFS 制御を協調して行い，消費電力を最小化する制御手法を提案する．優先度制御はメモリキューを用いて実現し，キャッシュパーティショニングにはハードウェアコストの少ないプロファイルを用いた制御手法を提案する．

## 2. アクセス制御時における消費電力のモデリング

### 2.1 問題設定

本章では，提案するアクセス制御適用時における CMP の消費電力のモデリングを行う．対象とする CMP は $n$ 個の構成の等しい PU（$PU_i$: $0 \leq i \leq n-1$）で構成されているものとする．各 $PU_i$ はそれぞれ独立に L1 キャッシュを内蔵し，全 PU で L2 キャッシュを共有するものとする．また，各 $PU_i$ は独立なクロック周波数 $f_i$，電源電圧 $V_i$ で稼働するものと仮定する．全 PU は共有するメモリバスを通してメインメモリにアクセスする．

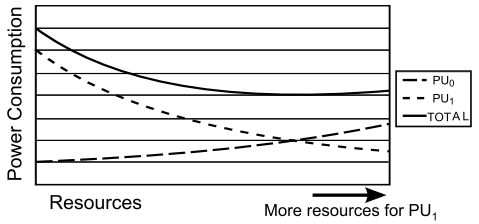


**図 1** 共有資源の分配と消費電力の関係

$PU_i$ 上ではアプリケーションタスク $T_i$ が実行されていると仮定する．各 PU で実行されているアプリケーションタスクは独立であり，他の PU の実行結果には依存しないものとする．また，$T_i$ はそれぞれ独立な実時間制約 $L_i$ が定めれているとする．これは，$T_i$ が時間 $L_i$ 内で実行される必要があることを意味する．ここで，$T_i$ 内では L2 キャッシュミス率は一定であると仮定する．

また，本稿では以下のように変数を定義する．

- $f_i$: $PU_i$ のクロック周波数
- $V_i$: $PU_i$ の電源電圧
- $T_i$: $PU_i$ のアプリケーションタスク
- $L_i$: $T_i$ の実時間制約
- $l_B$: メモリバスアクセスレイテンシ（メインメモリから L2 キャッシュへのデータ転送に要する時間）
- $l_M$: メモリアクセスレイテンシ
- $I_i$: $T_i$ における命令数
- $m_i$: $T_i$ における L2 キャッシュミス回数
- $s_i$: $T_i$ 単独実行時における L2 キャッシュミスによる $PU_i$ のストール時間

これらの変数を用いて，$PU_i$ の実効稼働時間は以下の式で与えられる．

$$t_i = L_i - s_i \tag{1}$$

この $t_i$ を用い，最適なクロック周波数は比例定数 $c$ を用いて以下の式のように書き表わすことができる．なお，この式では共有資源における競合の影響は考慮されていないことに注意する．

Vol.2009-ARC-183 No.35
Vol.2009-OS-111 No.35
2009/4/24

$$f_i = c\frac{I_i}{t_i}\ (c : const) \tag{2}$$

### 2.2 アクセス制御時における電力最適点

本節ではまず，アクセス制御（優先度制御，キャッシュパーティショニング）の影響を考慮し，競合発生時の CMP の消費電力についてモデリングを行う．次に，CMP 全体の消費電力が最小となる電力最適条件を導出する．

#### 2.2.1 優先度制御時における消費電力

まず，キャッシュのパーティションは固定し，優先度制御のみを適用した場合の電力についてモデリングを行う．つまり，優先度制御によりメモリアクセスの順番は制御されるが，メモリアクセスの回数 $m_i$ そのものは変化しない．

メモリアクセス発生時に $\mathrm{PU}_i$ からのアクセスがメモリバスを占有している確率は以下の式で与えられる．

$$p_i = \frac{m_i}{L_i} l_B \tag{3}$$

メモリバスの競合により生じる単位時間当たりの待ち時間の総和 $l_{total}$ は，$p_i$ を用いて以下の式で表わされる．

$$\begin{aligned} l_{total} = \sum_i \Bigg( & \sum_{j \neq i} \frac{1}{2} p_i p_j \Big( \prod_{k \neq i,j} (1 - p_k) \Big) \\ & + \sum_{j,k \neq i, k > j} \frac{3}{2} p_i p_j p_k \prod_{l \neq i,j,k} (1 - p_l) + \cdots \\ & + (\frac{1}{2} + n - 2) \prod_j p_j \Bigg) \end{aligned} \tag{4}$$

メモリアクセス発生時，優先度制御器は優先度を考慮し，各 PU の待ち時間の比が $\mathrm{PU}_0{:}\mathrm{PU}_1{:}\cdots{:}\mathrm{PU}_{n-1} = r_0{:}r_1{:}\cdots{:}r_{n-1}$ となるように処理するリクエストを選択する．この時，各 PU の実効稼働時間 $t_i$ は以下の式で表わすことができる．

$$\begin{aligned} t'_0 &= t_0 - r_0 l_{total} L_0 \\ t'_1 &= t_1 - r_1 l_{total} L_1 \\ &\cdots \\ t'_{n-1} &= t_{n-1} - r_{n-1} l_{total} L_{n-1} \end{aligned} \tag{5}$$

同様にして，優先度制御適用時の $f_i$, $e_i$，総消費電力 $P_{total}$ は以下の式のように表わされる．

$$f'_i = c\frac{I_i}{t_i - r_i l_{total} L_i} \tag{6}$$

$$e_i = C_2 f'^2_i + C_1 f'_i + C_0\ (C_2, C_1, C_0 :\ const) \tag{7}$$

$$P_{total} = \sum_i \frac{I_i}{L_i} e_i \tag{8}$$

ラグランジュの未定乗数法を用いて式 (8) を解くことにより，各 PU のクロック周波数が等しくなるようにメインメモリへのアクセスの優先度を制御することで CMP 全体の消費電力が最小化されることが証明され, その時の周波数は式 (10) で与えられる．

$$f'_0 = f'_1 = \cdots = f'_{n-1} = f'_{opt} \tag{9}$$

$$f'_{opt} = c\frac{\sum_i \frac{I_i}{L_i}}{\sum_i \frac{t_i}{L_i} - l_{total}} \tag{10}$$

#### 2.2.2 キャッシュパーティショニング適用時の消費電力

次に，更なる消費電力の最小化のために，L2 キャッシュのウェイのパーティションを最適化する．ウェイパーティショニングとは，各 PU にウェイの単位でキャッシュを割り振る技術であり，割り当てられる L2 キャッシュのウェイ数の変化に伴い，キャッシュミス回数 $m_i$ が変化する．

$s_i$ は $s_i = m_i l_M$ と表すことができ，これを用いて単位時間当たりの総ストール時間 $l_{stall}$ は以下のように表わされる．

$$l_{stall} = \sum_i \frac{m_i}{L_i} l_M + l_{total} \tag{11}$$

次に，式 (10) は式 (11) を用いて以下のように書き直せる．

$$f'_{opt} = c\frac{\sum_i \frac{I_i}{L_i}}{n - l_{stall}} \tag{12}$$

CMP 全体の消費電力は $f'_{opt}$ のみの関数であり，この式より，総消費電力は $l_{stall}$ が最小値をとるとき最小となることが分かる．

以上のモデリングより，アクセス制御の電力最適条件は $l_{stall}$ を最小化した下で，全 PU の周波数を一致させることであるという結論が得られる．

## 3. 制 御 手 法

本章では，2 章で導いた最適条件を，実際の CMP において実アプリケーション実行時に

Vol.2009-ARC-183 No.35
Vol.2009-OS-111 No.35
2009/4/24

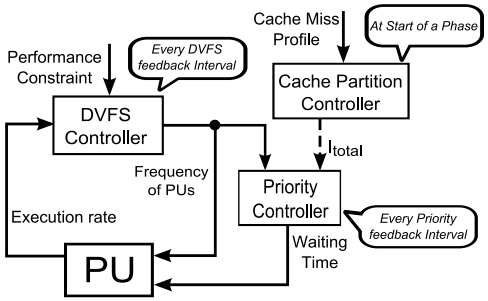


**図 2** 制御手法のブロック線図

実現するための，キャッシュパーティショニングと優先度制御の制御手法を提案する．

### 3.1 制御手法の概要

2 章において，消費電力を最小化するクロック周波数を導出した．しかしながら，実際の CMP システムで実アプリケーションを実行する場合，$T_i$ 中のキャッシュミス率が一定ではない．また選択可能な周波数・電源電圧の段階が限られているといった理由からモデリング時に使用した仮定をそのまま適用することができない．

そこで，1 点目に対しては，アプリケーションをキャッシュミス率が一定とみなすことができる $phase$ という小さなタスクに分割することで解決を図る．性能制約は各 phase にそれぞれ与える．2 点目に対しては，フィードバック制御を用い，最適クロック周波数を実現する制御手法を提案する．提案手法のブロック線図を図 2 に示す．どれか一つの PU の phase が変化するたびに，キャッシュパーティション制御器はキャッシュミス回数のプロファイルから新しいウェイのパーティションを計算し，各 PU に割り当てる．図 2 から分かるように，優先度制御器と DVFS 制御器はフィードバックループを持つ．優先度制御器は，全 PU の周波数が等しくなるよう優先度制御の周期ごとに各 PU に $l_{total}$ を割り振る．DVFS 制御器は，DVFS の周期ごとに各 PU の周波数・電源電圧を，それぞれの性能制約を満たせる値に設定する．

以下の節において，各制御器の詳細について述べる．

### 3.2 キャッシュパーティショニングとそのアルゴリズム

キャッシュパーティション制御器は，事前のプロファイルの情報を基に L2 キャッシュのウェイを各 PU に割り振る．プロファイルの情報とは，取り得る全ウェイ数の場合の phase $P$ でのキャッシュミス回数 $m_i^P$ であり，事前に対象の CMP で phase を単独実行し，スタックディスタンスプロファイル[10] を取得することで得ることが可能である．phase 内でのキャッシュミス率は一定であると仮定しており，パーティションの変更は，全 PU の内どれかの 1 つの PU で実行中の phase が変化した時にのみ行えば必要十分である．

アルゴリズムは以下の通りである．phase の開始時，制御器は $m_i^P$ から式 (4) と式 (11) を用いて，全通りのパーティションに対し $l_{stall}$ を計算し，$l_{stall}$ が最小となるパーティションを選択する．なお，各 PU には最低でも 1 ウェイは割り当てられるものとする．

このキャッシュのウェイパーティショニングを実装するために，各キャッシュブロックのタグエントリに $log_2 n$ ビットを追加し，どの PU がそのブロックを所持しているかを識別する．$\mathrm{PU}_i$ のキャッシュミス発生時には，キャッシュパーティション制御器は，該当するキャッシュセット内に各 PU が所持しているキャッシュブロックの数をカウントする．もし $\mathrm{PU}_i$ の所持しているブロック数が割り当てられた数より少ない場合には，$\mathrm{PU}_i$ 以外の所持するブロックの中の LRU ブロックを置換する．それ以外の場合は，$\mathrm{PU}_i$ の所持するブロックの LRU ブロックを置換する．

### 3.3 優先度制御とそのアルゴリズム

優先度制御器は，全 PU の周波数が一致するようにメモリアクセスを制御する．優先度の変化は結果的に周波数の変化をもたらすものの，優先度制御器は周波数を直接制御することはできない．そこで，フィードバックベースの優先度制御手法を提案する．

提案手法では，メインメモリに対するアクセスキューを用いて優先度制御を実現する．このアクセスキューは共有資源へのアクセスを制御する際に用いられる一般的なハードウェアと考えられる．全 PU からの L2 ミスによるメモリ（とメモリバス）アクセスリクエストはキューに蓄積され，メインメモリに対して 1 個ずつ発行される．

まず，各 PU に対しメモリバスへのアクセスに関する優先度 $N_i$ を導入する．キューにおいて，$\mathrm{PU}_i$ からのメモリリクエストは，$N_i$ よりも小さな優先度を持つ PU からのリクエストに先行して処理される．なお同じ PU からのリクエストは first-come first-served (FCFS) ルールで処理される．

優先度制御器は，フィードバック周期ごとに phase $P$ の平均周波数を用いて $N_i$ の値を更新する．全 PU の周波数を一致させるため，制御器は PU の平均周波数の高い順に $n$ から 1 までの値を $N_i$ にセットする．例えば，$\mathrm{PU}_i$ の平均周波数が最も高く $\mathrm{PU}_j$ の平均周波数が最も低い場合には，$N_i$ には $n$ がセットされ，$N_j$ には 1 がセットされる．

### 3.4 DVFS 制御とそのアルゴリズム

本節では，各 phase の性能制約を満たす周波数を選択するための，PI 制御を用いた DVFS

Vol.2009-ARC-183 No.35
Vol.2009-OS-111 No.35
2009/4/24

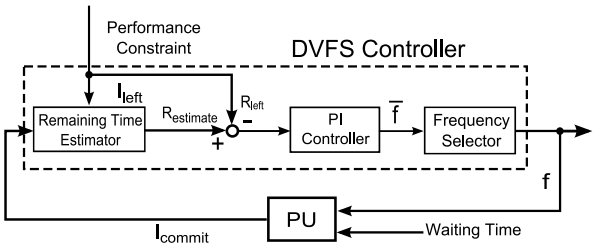


**図 3** DVFS 制御手法のブロック線図.

制御器を提案する.

DVFS 制御器のブロック線図を図 3，アルゴリズムを表 1 に示す．DVFS 制御器は残り時間予測器，PI 制御器，周波数選択器で構成される．以下アルゴリズムを示す.

phase $P$ の $k$ 番目の DVFS 周期の最後において，現在の周波数で phase の最後まで実行した場合の残り実行時間 ${T_{estimate}}_i^k$ を予測する（Step 1）．次に，次の周期で用いられるクロック周波数を PI 制御器を用いて計算する（Step 2）．${R_{estimate}}_i^k$ は ${T_{total}}_i^P$ に対する ${T_{estimate}}_i^k$ の割合を表わし，${R_{left}}_i^k$ は実際の残り時間 ${T_{left}}_i^k$ の ${T_{total}}_i^P$ に対する割合を表わす．PI_control 関数は以下の式で表わされる.

$$\bar{f}_i^k = \bar{f}_i^{k-1} + K_P(e_i^k - e_i^{k-1} + \frac{1}{T_I}e_i^k) \tag{13}$$

PI 制御器より得られた $\bar{f}_i^k$ は連続値であり，使用可能な周波数段階から適切な周波数を選択する必要がある．周波数選択のためにシュミットトリガ関数 (Step 3) を使用する．DVFS にはオーバーヘッドが存在するため，頻繁な周波数の遷移は消費電力の増加につながるが，この関数を用いることで周波数の遷移回数を減少させることが可能である．選択された周波数が現在の周波数と異なる場合，DVFS のオーバーヘッドを考慮して周波数を再計算する (Step 4)．最終的に得られた値を次の周期の周波数とする.

**表 1** DVFS 制御のアルゴリズム.

**Initialize** (At the start of an phase $P$ of $\mathrm{PU}_i$)
  set ${I_{total}}_i^P$: the total number of instructions in $P$
  set ${t_{deadline}}_i^P$: the deadline of the phase $P$
  $e_i^0 = 0$
  ${T_{total}}_i^P = {t_{deadline}}_i^P - t_{now}$

**DVFS Routine** (At the end of $k$-th interval period)
For each PU $i$:
/* **Step 1: Remaining time estimate** */
  ${T_{left}}_i^k = {t_{deadline}}_i^P$ - $t_{now}$, ${I_{left}}_i^k$ -= ${I_{commit}}_i^k$
  ${T_{estimate}}_i^k = T_{DVFS}$ / ${I_{commit}}_i^k$ * ${I_{left}}_i^k$

/* **Step 2: PI Controller** */
  ${R_{estimate}}_i^k = {T_{estimate}}_i^k / {T_{total}}_i^P$
  ${R_{left}}_i^k = {T_{left}}_i^k / {T_{total}}_i^P$
  $e_i^k = {R_{estimate}}_i^k - {R_{left}}_i^k$
  $\bar{f}_i^k = \mathrm{PI\_control}(e_i^{k-1}, e_i^k, \bar{f}_i^{k-1})$

/* **Step 3: Frequency select** */
  $f_i^k = \mathrm{schmitt\_trigger\_func}(\bar{f}_i^k)$

/* **Step 4: Recalculate frequency** */
if ($f_i^k$ != $f_i^{k-1}$){
  ${T_{left}}_i^k$ -= $T_{penalty}$
  goto Step 2 and Step 3
}

## 4. 評　　価

### 4.1 評価の仮定

#### 4.1.1 アプリケーション

提案手法の評価のために，ソフトリアルタイムアプリケーションである H.264/AVC デコーダ[7] と SPEC CPU2000 ベンチマーク[8] から選択した art，mcf，twolf，vpr，mesa，bzip2 の各プログラムを同時に実行する．各アプリケーションの初期化部分の実行は評価せず，その後の H.264 が 2 億命令の実行を完了するまでを評価した．図 4 は，単独実行時においてキャッシュのウェイ数を 1 から 16 に変化させた場合の，評価に用いた各アプリケーションの 1000 命令あたりのキャッシュミス回数（MPKI）を示したものである．横軸は割り当てられたウェイ数を表わす．図 4 より，twolf と vpr はキャッシュサイズが増加すると大きくキャッシュミス回数が減少しており，*way-sensitive* なアプリケーションであることが分かる．また art と mcf はキャッシュミス率が高く，逆に mesa と bzip2 はキャッシュミス率の低いアプリケーションであることが分かる.

#### 4.1.2 CMP の仮定

CMP は 2 コアの場合と 4 コアの場合について評価を行う．CMP のコアはアウトオブオーダー，スーパースカラプロセッサを仮定し，非共有の L1 命令/データキャッシュと共有

Vol.2009-ARC-183 No.35
Vol.2009-OS-111 No.35
2009/4/24

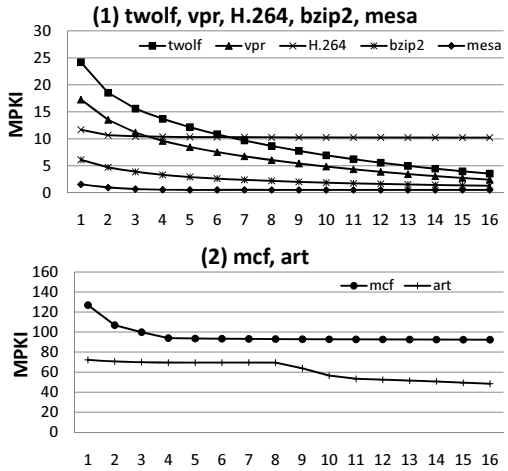


**図 4** アプリケーションの MPKI

**表 2** ハードウェアの構成

| | |
|---|---|
| Cores | 2 and 4 |
| Clock frequency | 600 - 2400 MHz (200 MHz step) |
| Supply voltage | 0.988 - 1.441 V |
| L1-Inst (private) | 32 KB, 2-way, 32 B line, 1-cycle |
| L1-Data (private) | 32 KB, 2-way, 32 B line, 2-cycle |
| L2 (unified) | 1 MB, 16-way, 64B line, 10-cycle (2 core)<br>2 MB, 16-way, 64B line, 10-cycle (4 core) |
| Memory latency | CAS, RCD, RP 4-bus cycle<br>RAS 12-bus cycle |
| Memory bus | 8 B, 400 MHz |
| DVFS penalty | 10 us |

の 16-way L2 キャッシュを持つ．表 2 にシミュレーションで用いたハードウェア構成を示す．DVFS は全 10 段階のクロック周波数・電源電圧を選択可能とする．クロック周波数は 600 MHz から 200 MHz 刻みで 2400 MHz までである．シミュレーションは CMP 用に拡張した SimpleScalar Tool Set[5] を用いて行い，電力評価には Wattch[6] を用いる． 3 章で述べた制御手法のアルゴリズムに用いる各パラメータは表 3 の通りである．

**表 3** アルゴリズムで用いたパラメータ．

| | |
|---|---|
| Partitioning $l_B$, $l_M$ | 22.5 ns |
| Priority feedback interval | 400 us |
| DVFS interval | 200 us |
| PI control gain | $K_P$: 0.1, $T_I$: 0.1 (2 core)<br>$K_P$: 0.3, $T_I$: 0.2 (4 core) |

### 4.2 2 コアにおける評価

#### 4.2.1 性能制約

`H.264` の性能制約は 30fps とする．これは 1 フレームを 33ms 以内に処理することを意味する．SPEC ベンチマークは実時間制約のあるリアルタイムアプリケーションではないため，IPS（1 秒あたりに実行すべき命令数）を性能制約として任意に付加する．性能制約は各アプリケーションの各 phase に対し定められており，全 phase の平均 IPS を表 4 に示す．`twolf`, `vpr`, `bzip2`, `mesa` の性能制約は各アプリケーションを 1400 MHz で単独実行した際の IPS であり，`art` と `mcf` の性能制約は 600 MHz で単独実行した際の IPS である．`art` と `mcf` は L2 キャッシュミス率が高く，他のアプリケーションと実行した際に競合により周波数が大きく上昇する傾向がある．そのため，これらのアプリケーションの性能制約は他のアプリケーションよりも緩く設定してある．

#### 4.2.2 評価結果

本節では，以下の 6 種類のケースについての評価結果を示す．

- *only DVFS (share)*: DVFS 制御のみ適用．L2 キャッシュは共有．
- *PRIORITY (share)*: DVFS 制御と優先度制御を適用．L2 キャッシュは共有．
- *only DVFS (even)*: DVFS 制御のみ適用．L2 キャッシュは各コアに均一に分割．
- *PRIORITY (even)*: DVFS 制御と優先度制御を適用．L2 キャッシュは各コアに均一に分割．
- *PARTITION*: DVFS 制御とキャッシュパーティショニングを適用．
- *PRIORITY+PARTITION*: DVFS 制御，優先度制御，キャッシュパーティショニングを適用（提案手法）．

図 5 に上記の 6 種類のケースにおける各コアの消費電力及び，CMP 全体の消費電力の評価結果を示す．縦軸の値は電力を表わし，*only DVFS* の全消費電力の値で正規化した値である．評価した全アプリケーションの組み合わせにおいて提案手法 (*PRIORITY+PARTITION*) は，*only DVFS* に対し最大で 23%，平均で 15%の電力を削減でき，また，*only DVFS (even)*

Vol.2009-ARC-183 No.35
Vol.2009-OS-111 No.35
2009/4/24

**表 4** 性能制約（2 コア）.

| Application | IPS |
|---|---|
| H.264 | 865 M (30 fps) |
| twolf | 1486 M |
| vpr | 1649 M |
| bzip2 | 2127 M |
| mesa | 2350 M |
| art | 620 M |
| mcf | 329 M |

に対しては，最大で 46.8%，平均で 21%の電力を削減できていることが分かる．H.264 とキャッシュミス率の高い art や mcf を組み合わせて実行した場合には，全消費電力は提案手法を適用した場合に最小となっている．一方，twolf や vpr などの *way-sensitive* なアプリケーションと実行した際には，キャッシュパーティショニングを適用することで消費電力が大きく削減されており，優先度制御の適用によっては電力を削減できていないことが分かる．また，キャッシュミス率が低く，キャッシュサイズの増加によるキャッシュミス率の減少が少ない mesa や bzip2 などのアプリケーションと H.264 を実行した際には，もとから共有資源において競合がほとんど発生しないため，消費電力を削減できていない．

図 5 に，提案手法適用時の *PRIORITY (share)* の場合に対する，単位時間当たりの総ストール時間 $l_{stall}$ の削減率を示す．図より，すべてのアプリケーションの組み合せに対し $l_{stall}$ が削減出来おり，特にキャッシュパーティショニングによる電力削減率の大きい twolf や vpr との実行の場合には削減率が大きく，それぞれ約 18%，20%削減されている．逆に削減率の低い mesa や bzip2 との実行の場合には $l_{stall}$ もほとんど削減されない．

図 6 に H.264 と art を同時に実行した際の各 PU のクロック周波数の推移を示す．図 6-(1)，図 6-(2)，図 6-(3) はそれぞれ，*only DVFS (share)*, *PRIORITY (share)*, *PRIORITY+PARTITION* の場合の結果である．図中の各点は 5 ms ごとの平均周波数を表わしている．図 6-(2) より優先度制御を適用することで両コアの周波数が近づいていることが分かる．これは優先度制御により，art からのアクセスが H.264 からのアクセスに対し優先されたためである．図 6-(3) より，提案手法を適用することで両コアの周波数はさらに接近し，両コアの平均周波数も *PRIORITY (share)* の場合より低くなっていることが分かる．これらは期待された挙動であり，提案手法によりモデルから得られた電力最適条件を達成できていることが分かる．

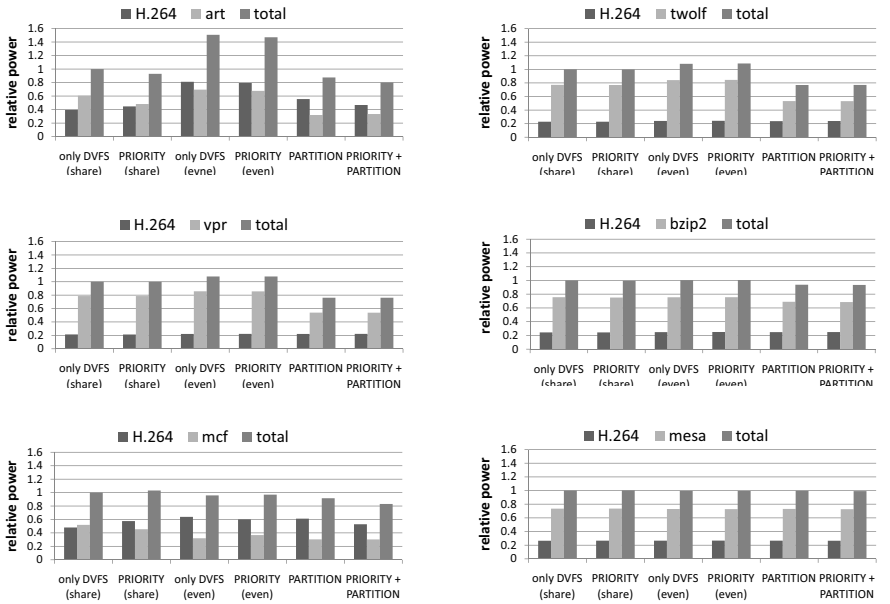


**図 5** 消費電力の評価結果（2 コア）.

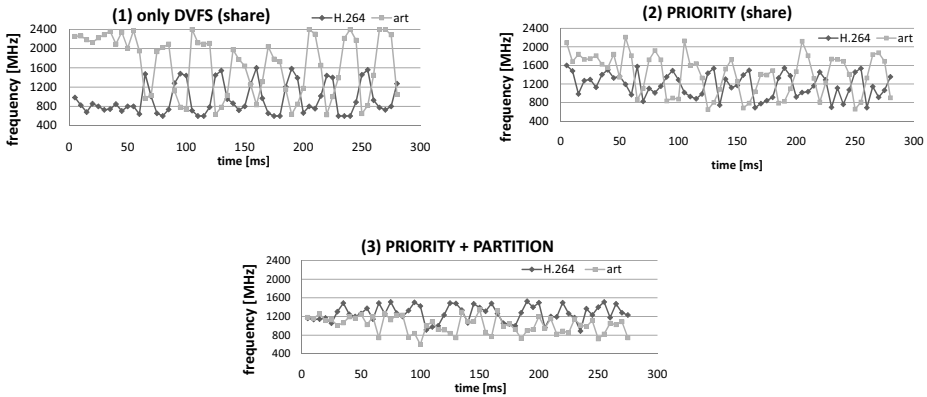


**図 6** 平均周波数の推移（H.264 − art）.

Vol.2009-ARC-183 No.35
Vol.2009-OS-111 No.35
2009/4/24

**表 5** 総ストール時間の削減率

| Application | stall time reduction |
|---|---|
| H.264 and twolf | 17.8 % |
| H.264 and vpr | 19.7 % |
| H.264 and bzip2 | 2.6 % |
| H.264 and mesa | 1.2 % |
| H.264 and art | 8.7 % |
| H.264 and mcf | 6.6 % |

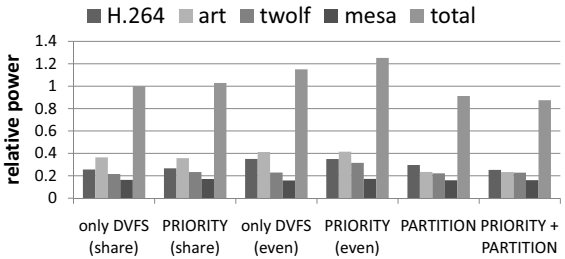


**図 7** 消費電力の評価結果（4 コア）

### 4.3 4 コアでの評価

`H.264`, `art`, `twolf`, `mesa` を用いて，4 コアの CMP における消費電力を評価する．`H.264` の性能制約は 2 コアの場合と同様 30 fps であり，その他のアプリケーションの性能制約は評価対象の CMP において 600 MHz で単独実行した際の IPS である．

評価結果を図 7 に示す．図より，提案手法は *only DVFS (share)* の場合に比べ 13%の電力を削減出来ており，*only DVFS (even)* と比較すると 24%の電力を削減出来ていることが分かる．この結果より，提案手法は 2 コアの場合だけでなく 4 コアにおいても有効であると言える．

## 5. まとめと今後の課題

本稿では，メインメモリへのアクセスの優先度を制御する優先度制御器と，L2 キャッシュへのアクセスをキャッシュパーティショニング[1)–4)] により制御するキャッシュパーティション制御器を導入し，複数の共有資源へのアクセスを協調制御することで CMP 全体の消費電力を最小化する手法を提案した．まず消費電力のモデリングにより，共有資源競合時の最適電力条件を導出し，その条件を，1）単独実行時のプロファイルを用いたキャッシュパーティショニングにより，総ストール時間を最小化し，2）優先度制御により全 PU の周波数を一致させることで実現した．評価の結果，提案手法は DVFS 制御のみを適用した場合に比べ大きく消費電力を削減できることが分かった．

今後の課題としては，提案手法をプロファイルを使用しない完全に動的な手法に拡張することがあげられる．

**謝辞** 本研究の一部は，科学技術振興機構・戦略的創造研究推進事業（CREST）の研究プロジェクト「革新的電源制御による超低電力高性能システム LSI の研究」，および文部科学省科学研究費補助金（基盤研究（A）No. 18200002）の支援によって行われた．

## 参 考 文 献

1) “G. Edward Suh, Srinivas Devadas, Larray Rudolph, A New Memory Monitoring Scheme for Memory-Aware Scheduling and Partitioni ng,” *In Proceedings of the 8th International Symposium on High-Performance Computer Architecture*, pp.117–128, Feb. 2002.
2) Seongbeom Kim, Dhruba Chandra, Yan Solihin, “Fair Cache Sharing and Partitioning in a Chip Multiprocessor Architecture,” *In Proceedings of the 13th International Conference on Parallel Architec ture and Compilation Techniques*, pp.111–122, Sep. 2004.
3) Moinuddin K. Qureshi, Yale N. Patt, “Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches,” *In Proceedings of the 39th Annual IEEE/ACM International Symposium on Mi croarchitecture*, pp.423–432, Dec. 2006.
4) Jichuan Chang, Gurindar S. Sohi, “Cooperative cache partitioning for chip multiprocessors,” *In Proceedings of the 21st annual international conference on Supercompu ting*, pp.242–252, Jun. 2007.
5) Todd Austin, Eric Larson, Dan Ernst, “SimpleScalar: An Infrastructure for Computer System Modeling,” *Computer*, 35(2):59–67, Feb. 2002.
6) David Brooks, Vivek Tiwari, Margaret Martonosi, “Wattch: A framework for architectural-level power analysis and optimizations,” *In Proceedings of the 27th Annual International Symposium on Computer Architecture*, pp.83–94, Jun. 2002.
7) H.264/AVC Software Coordination, H.264/AVC reference software, http://iphome.hhi.de/suehring/tml.
8) Standard Performance Evaluation Corporation (SPEC), SPEC CPU2000, http://www.specbench.org.
9) Ryo Watanabe, Masaaki Kondo, Hiroshi Nakamura, Takashi Nanya, “Power reduction of chip multi-processors using shared resource control cooperating with DVFS,” *In Proceedings of the 27th International Conference on Computer Design* , pp.365–379, 1999.
10) Calin Cascaval, Luiz Derose, David A. Padua, Daniel A. Reed, “Compile-time based performance prediction,” *In Proceedings of the 12th International Workshop on Langua ges and Compilers for Parallel Computing*, pp.365–379, 1999.
11) Kyle J. Nesbit, Nidhi Aggarwal, James Laudon, James E. Smith, “Fair Queuing Memory Systems,” *In Proceedings of the 39th Annual IEEE/ACM International Sy mposium on Microarchitecture* , pp.208–222, 2006.