情報処理学会論文誌　コンピューティングシステム　Vol. 4　No. 2　40–58 (Mar. 2011)

共有資源の競合を考慮したチップマルチプロセッサ向け低消費電力化手法

佐々木　広[†1]　高木紀子[†1,*1]
近藤正章[†2]　中村宏[†1]

近年，プロセッサのアーキテクチャとして複数のコアを同一チップに搭載するチップマルチプロセッサ（CMP: chip multiprocessor）が主流となっている．CMP では複数のコアが下位のメモリ階層を共有しているが，これによりチップ上の限られた資源を有効に活用できる一方で，共有資源へのアクセスが競合すると性能が低下してしまうという問題がある．また，共有資源の競合の問題は従来より用いられている動的電源電圧制御（DVFS: dynamic voltage and frequency scaling）による省電力化の大きな障害となる．特に，性能制約を持つアプリケーションでは最適なクロック周波数・電源電圧の値がアプリケーションの性質だけでなく，競合の状況に依存して変動してしまうため，最適化が非常に困難になる．そこで本論文では，複数の共有資源へのアクセスを協調制御することで競合の状況を調整し，そのうえで DVFS によって消費電力を削減する手法を提案する．まず共有資源への協調制御が CMP の消費電力に与える影響をモデル化し，電力を最小化する協調制御を導出する．次に，モデルに基づいた協調制御手法を構築する．提案手法を評価した結果，従来の協調制御を用いない場合の DVFS 手法に比べ，2 コアの CMP の場合平均で 10%，4 コアの CMP の場合平均 8.5%の消費電力を削減できることが分かった．

Coordinated Control of Shared Resources for Energy Efficient Chip Multiprocessors

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

In a chip multiprocessor (CMP) architecture, multiple cores usually share resources in the memory hierarchy including the last-level cache, the memory bus, and the DRAM memory banks. When applications running on different cores have their own performance constraint and is applying DVFS control to meet their constraints, conflicts in shared resources lead to a waste of power consumption. In this paper, we derive the condition where the total CPU power consumption becomes minimum by constructing a power consumption model under resource conflicts, and propose a novel method to minimize the power consumption by a cooperative access control to multiple shared resource with DVFS. The experimental results reveals that our technique can reduce 10% of power consumption on average in dual-core CMP, and 8.5% in quad-core CMP, compared with the case where only DVFS is applied.

1. はじめに

近年の VLSI は半導体のプロセスを微細化することで集積度やクロック周波数を向上させ，高機能化・高性能化を実現してきた．ところがトランジスタ数の増大によるクロック周波数の向上は同時に消費電力の爆発的な増加を引き起こし，さらなる高機能化・高性能化を制限するという問題が発生している．

VLSI の低消費電力化に関し，従来より多くの研究がなされている．その 1 つが動的電源電圧制御（DVFS: dynamic voltage and frequency scaling）[1)–3)] である．DVFS はプロセッサのクロック周波数・電源電圧を実行時に下げることで消費電力を削減する技術であり，現在多くの商用プロセッサにおいて実装されている．また，DVFS は最低限必要な性能を維持しつつ消費電力を削減可能であるという性質から，リアルタイムシステムのような性能制約を持つシステムにおいて非常に有効である．

一方で，クロック周波数の増加による性能向上が困難になり，プロセッサアーキテクチャの主流は 1 チップ上に複数のプロセッサコアを搭載したチップマルチプロセッサ（CMP: chip multiprocessor）へと移行している．CMP は複数のプロセッサコアで並列処理を行うことで高い性能を達成できるため，従来のユニプロセッサに比べて電力あたりの性能が優れている．

CMP では，資源の制限や有効活用のために最下位キャッシュやメモリバス，主記憶 DRAM バンクなど下位のメモリ階層を複数のコアが共有することが一般的である．共有資源において複数のコアからのアクセスが競合すると，共有資源にアクセスしたコアおよび CMP 全体

†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 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
*1 現在，富士通株式会社
Presently with Fujitsu Limited

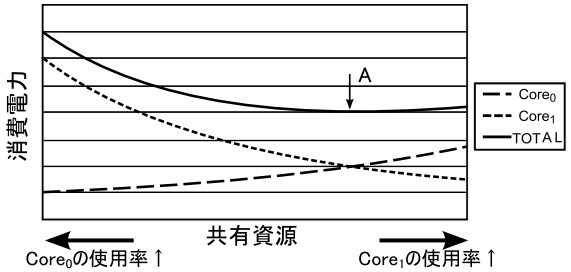


図 1　共有資源の単位時間あたりの使用率と消費電力の関係
Fig. 1　Relationship between power consumption and usage of shared resources of each core.

の性能低下や，実行中のプログラムの性能予測が困難になるなどの問題が発生する．これら共有資源の競合の問題を解決するため従来より数多くの研究がなされている[4)–14)]．

性能制約を持つシステムの DVFS による省電力化ではクロック周波数・電源電圧を下げつつも制約を厳守しなければならず，適切なクロック周波数を選択するためには高い精度で DVFS 時の性能を予測することが重要となる．そのため性能が競合の状況に依存し，かつ性能予測が困難な CMP において性能制約を守りつつ消費電力を最小化するようにクロック周波数・電源電圧を最適化することは容易ではない．

我々は，この問題に対して最下位キャッシュやメモリバスという複数の共有資源へのアクセスを協調して制御することで共有資源における競合の状況を調整し，そのうえで DVFS を行うことで CMP 全体の消費電力を削減する手法を提案している[15)]．本論文ではさらに，実際のハードウェアをより詳細にモデル化したシミュレーションによる評価を加え提案手法の有効性を示すとともに，評価結果の詳細な分析と考察を行い提案する制御手法の効果を解析した．

図 1 は，2 コアの CMP における各コアの共有資源の単位時間あたりの使用率と消費電力の関係を表した概念図である．横軸は共有資源の使用率を表し，左に行くほど $Core_0$ が共有資源の多くを使用し，右に行くほど $Core_1$ が共有資源を多く使用する．つまり，左端では $Core_0$ が共有資源をすべて使用し，$Core_1$ は共有資源を使用しないことを表す．縦軸は各コアの消費電力および CMP 全体の消費電力を表している．また，この例では $Core_1$ の性能制約が $Core_0$ よりも厳しい場合を仮定している．図 1 のように，より多くの共有資源を $Core_1$ が使用するほど，共有資源の競合による $Core_1$ の性能低下が小さくなり $Core_1$ のクロック周波数・電源電圧を下げることが可能となる．それにより $Core_1$ の消費電力は減少する．逆に $Core_0$ の性能は低下し消費電力は増加してしまうが，$Core_1$ の性能制約の方が厳しいため $Core_1$ の消費電力の削減分が $Core_0$ の消費電力の増加分を上回ることとなり，使用率を $Core_1$ 側に偏らせた A 点で消費電力は最小化される．このようにアクセス制御により共有資源の使用率を適切に調整することで消費電力を最小化することが可能である．

本論文では共有資源へのアクセス制御として，(1) 最下位共有キャッシュ（本論文においては L2 キャッシュ）に対しては，スレッド間でキャッシュを分割するキャッシュパーティショニングによって各コアが使用可能なキャッシュ領域の大きさを制御し，(2) メモリバスに対しては，DRAM メモリコントローラ上で DRAM へのアクセスの順番を調整することでメモリバスへのアクセスの優先度を制御する．メモリバスへのアクセス回数はそれぞれのコアの L2 キャッシュミスの回数に依存するためキャッシュパーティショニングにより変化する．そのため，DRAM アクセスの優先度とキャッシュパーティショニングを独立に決めることによって最適化を行うことはできず，両アクセスを協調して制御する必要がある．このように，消費電力を最小化する制御を行うためには，これら 2 つのアクセス制御を協調して行う必要があることに注意しなくてはならない．

まずはじめに，両アクセス制御が CMP 全体の消費電力に与える影響をモデル化し，そのモデルを用いて消費電力を最小化するキャッシュパーティショニングと DRAM アクセスの優先度を導出する．次に，導出した消費電力を最小化するキャッシュパーティショニングと DRAM アクセスの優先度を CMP 上で達成するような協調制御手法を構築する．モデルの正確さおよび制御手法の有用性を確認するために，モデルに近い理想化したハードウェアおよび，実際のハードウェアを詳細にモデル化したハードウェアの 2 つの場合についてシミュレーションによる評価を行う．提案手法を評価した結果，従来の協調制御を用いない場合の DVFS 手法に比べ，理想化したハードウェアにおいては 2 コアの CMP の場合平均で 12%，4 コアの CMP 場合 8%，また実際のハードウェアを詳細にモデル化したハードウェアにおいては 2 コアの CMP の場合平均で 10%，4 コアの CMP 場合 8.5%の消費電力を削減できることが分かった．

本論文の構成は以下のとおりである．次章において，アクセス制御時の CMP の消費電力のモデル化を行い，CMP の消費電力を最小化するキャッシュパーティショニングと優先度の条件を理論的に導出する．3 章では 2 章で導出した条件を実際の CMP において達成する協調制御手法を提案する．4 章では評価環境について述べ，シミュレーションによる提案手法の評価を行う．5 章で関連研究について言及し，最後に 6 章で本論文のまとめと今後の課題について述べる．

2. アクセス制御時の消費電力のモデリング

2.1 問題設定

本節では，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い，消費電力を最小化する協調制御（キャッシュパーティショニング・DRAM アクセスの優先度）を導出する．モデル化に用いた変数は**表 1** のとおりである．

対象とする CMP は以下のとおりである．CMP は構成の等しい n 個のアウトオブオーダ・スーパスカラプロセッサコア（$Core_i$: $0 \leq i \leq n-1$）で構成されており，各コア $Core_i$ は独立なクロック周波数 f_i，電源電圧 V_i で動作する．また，各コアはそれぞれ独立に L1 キャッシュを搭載し，全コアで L2 キャッシュおよびメモリバスを共有する．本論文では $Core_i$ 上でシングルスレッドなタスク $Task_i$ が実行されている状況を想定する．シングルスレッドであるため，各コア上で実行されているタスクは他のコアの実行状況に依存しない．

本章では以下の仮定[⋆1]をおいたうえで，DRAM アクセスの優先度とキャッシュパーティショニングの影響を考慮した CMP の消費電力のモデル化を行い，CMP 全体の消費電力を最小化する優先度・キャッシュパーティショニングの条件を導出する．

(1)　全タスクの命令レベル並列度は等しい．
(2)　ライトバックによるメモリバスの占有は無視できる．
(3)　主記憶 DRAM バンクは競合しない．
(4)　主記憶アクセスのレイテンシは一定であり，主記憶アクセスの順番には依存しない．
(5)　優先度を変えることでバスにおける待ち時間を各コアに任意に割り振れる．
(6)　$Task_i$ 内では L2 キャッシュミス率は一定である．
(7)　f_i，V_i は任意の連続値をとれる．

2.2 CMP の消費電力のモデル化

タスクの実行時間 T_i は，クロック周波数に反比例して変化する実効稼働時間 t_i と，メモリバスの周波数に比例して変化する L2 キャッシュミスを原因とするストール s_i に分けることができる．

$$T_i = t_i + s_i \tag{1}$$

⋆1 なお，これらの仮定は現実との乖離があるが，(5)–(7) の仮定が実際には成立しない点については，導出された条件をもとに協調制御を提案する際に考慮しており（3.1 節），4 章ではその他の仮定も成立しない条件下の評価も行っている．評価結果から，これらの仮定が成立しない現実の状況においても，本論文が提案する手法の有効性が示されていることに留意されたい．

表 1　モデル化に用いた変数
Table 1　Parameters used for constructing the model.

変数	説明
n	コア数
f_i	$Core_i$ のクロック周波数
V_i	$Core_i$ の電源電圧
P_i	$Core_i$ の消費電力
P_{total}	CMP 全体の消費電力
e_i	$Core_i$ の 1 命令あたりの消費エネルギー
T_i	$Task_i$ の実行時間
L_i	$Task_i$ の実時間制約
l_B	1 回のメモリアクセスがメモリバスを占有する時間
l_M	メモリアクセスレイテンシ
I_i	$Task_i$ における実行命令数
m_i	$Task_i$ における L2 キャッシュミス回数
s_i	L2 キャッシュミスによる生じる $Core_i$ のストール時間
t_i	$Core_i$ の実効稼働時間
a, b, k	CMP に依存する定数

実効稼働時間とは主記憶からのデータ転送によるストールを除いた，プロセッサ上で実際に処理を行っている時間である．この実効稼働時間 t_i，命令数 I_i とクロック周波数 f_i との関係をプロットしたものが**図 2** である．図の横軸は I_i/t_i（単位は実行命令数/ns）で縦軸はクロック周波数（単位は GHz）である．図中の点は一定間隔ごとに L2 キャッシュミスを起こすようなプログラム（`Application A`）を用いて取得した結果であり，直線は，取得した結果を最小二乗法を用いて 1 次関数に近似したものである．データを取得した環境は 4.1 節で後述する．図 2 より，プロットした点は直線 $y = 0.52x$ 上にきれいに乗っており，f_i と I_i/t_i は線形の関係にあることが分かる．よってクロック周波数 f_i は定数 c を用いて以下のようにモデル化することができる．

$$f_i = c\frac{I_i}{t_i} \tag{2}$$

定数 c は 1 命令の実行にかかる実効サイクル数（メモリアクセスによるストールサイクル数を除いたサイクル数）である．以降 c を実効 CPI（cycles per instruction）と呼ぶ．この値は $Task_i$ の命令レベル並列度に依存するが，本モデル化においては全 $Task_i$ の命令レベル並列度は等しいと仮定しているため，c は i に依存しない定数となっている．

また，一般に電源電圧 V_i は，プロセッサに依存して決まる定数 a，b（ともに正）を用いてクロック周波数 f_i と線形の関係に近似することが可能である[16]．

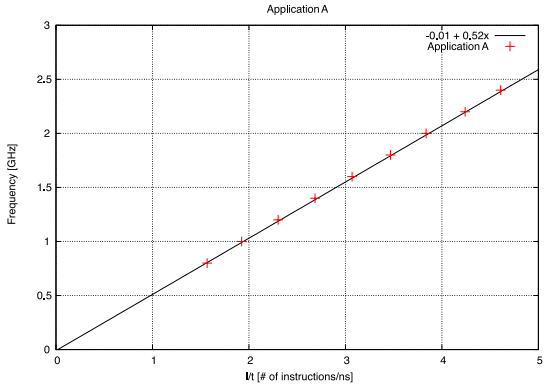


図 2　実効稼働時間・実行命令数とクロック周波数の関係
Fig. 2　Relationship between effective working time, number of instructions, and clock frequency.

$$V_i = af_i + b \tag{3}$$

1 命令あたりの消費エネルギー e_i は定数 k を用いて以下のように表すことができる[2)]．

$$e_i = kV_i^2 \tag{4}$$

以上より CMP 全体の消費電力は以下のようにモデル化できる．

$$\begin{aligned} P_{total} &= \sum_i P_i \\ &= \sum_i \frac{I_i}{T_i} e_i = \sum_i \frac{kI_iV_i^2}{T_i} \end{aligned} \tag{5}$$

性能制約のあるタスクの場合，実行時間 T_i が実時間制約 L_i 以下であることが求められる．式 (5) から分かるように，T_i が大きいほど消費電力は小さくなる．したがって，性能制約のあるタスクの実行時の消費電力は $T_i = L_i$ のとき最小である．以降，$T_i = L_i$ とする．このとき，t_i は式 (1) を用いて以下の式のように表される．

$$t_i = L_i - s_i \tag{6}$$

ここで，s_i にはメモリバスの競合により新たに生じるストール時間 $s_{bus,i}$ とメモリアクセスによるストール時間 w_i が存在する．

$$s_i = w_i + s_{bus,i} \tag{7}$$

DRAM アクセスの優先度制御は優先度を変えることでメモリアクセスの順番を入れ替えることが可能であり，$s_{bus,i}$ はそれに従って変化する．また，キャッシュパーティショニング制御はパーティションを変更することで L2 キャッシュミス回数が変化することから，s_i はそれに従って変化する．そこで以降では，DRAM アクセスの優先度およびキャッシュパーティショニングが s_i へ与える影響のモデル化を行う．

2.3　アクセス制御の影響のモデル化

キャッシュパーティショニングは前述のとおり各コアにキャッシュの領域を割り振る手法であり，割り当てられたキャッシュの大きさが変わることでキャッシュミス回数 m_i が変化する．w_i は L2 キャッシュミス回数 m_i とメモリアクセスレイテンシ（L2 キャッシュミスが生じてからデータが L1 キャッシュに届くまでの時間）l_M を用いて以下のようにモデル化できる．

$$w_i = m_i l_M \tag{8}$$

また，L_i 中の L2 キャッシュミス頻度は仮定したとおり一定であるため，1 回のメモリアクアセスがメモリバスを占有する時間を l_B とすると，メモリアクセス発生時に $Core_i$ 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．

$$p_i = \frac{m_i}{L_i} l_B \tag{9}$$

p_i を用いて，メモリバスの競合により生じる単位時間あたりの待ち時間の割合を全コアで合計した l_{total} は以下の式で表される．

$$\begin{aligned} l_{total} = \sum_i \Bigg(& \sum_{j \neq i} \frac{1}{2} p_i p_j \Big(\prod_{k \neq i,j} (1 - p_k) \Big) \\ & + \sum_{j,k \neq i, k > j} \frac{3}{2} p_i p_j p_k \prod_{l \neq i,j,k} (1 - p_l) + \cdots \\ & + \Big(\frac{1}{2} + n - 2 \Big) \prod_j p_j \Bigg) \end{aligned} \tag{10}$$

この式は，それぞれのコアごとに，そのコアが 2 コアのみ，3 コアのみ，……，n コアで競合する場合についてその単位時間あたりの待ち時間の割合の期待値（平均待ち時間の割合と競合する確率の積）を足し合わせたものを，すべてのコアについて足し合わせたものである．以下，簡単のため，3 コア (i, j, k) の場合について考える．単位時間あたりのコア i の待ち時間の割合の期待値は，(2 コアのみが競合した場合の単位時間あたりの平均待ち時間の割合) × (2 コアのみが競合する確率) + (3 コアが競合した単位時間あたりの平均待ち時間の割合) × (3 コアが競合する確率) で表すことができる．2 コアのみが競

合する確率というのは，$p_ip_j(1-p_k)+p_i(1-p_j)p_k$ で，3 コアが競合する確率というのは，$p_ip_jp_k$ でそれぞれ表される．2 コアのみが競合した場合の単位時間あたりの平均待ち時間の割合は，(0+1)/2（最短で 0，最長で 1）となり，3 コアが競合した場合の単位時間あたりの平均待ち時間の割合は，(1+2)/2（最短で 1，最長で 2）となる．よって，期待値は，$1/2(p_ip_j(1-p_k)+p_ip_k(1-p_j))+3/2(p_ip_jp_k)$ となる．これを全コアで合計すると，$\sum_i 1/2(p_ip_j(1-p_k)+p_ip_k(1-p_j))+3/2(p_ip_jp_k)$ となる．これを n コアまで拡張したものが式 (10) である．

DRAM アクセスの優先度を制御することでメモリアクセスの順番を任意に変更できるとき，待ち時間の総和の l_{total} の量は変化しないが[⋆1]，各コアの待ち時間の比 $Core_0{:}Core_1{:}\cdots{:}Core_{n-1}=r_0{:}r_1{:}\cdots{:}r_{n-1}$ を任意に制御可能である．このときストール時間 $s_{bus,i}$ は以下の式で表すことができる．

$$s_{bus,i}=r_il_{total}L_i \tag{11}$$

また，w_i，$s_{bus,i}$ ともに L2 キャッシュミス回数 m_i の関数であり，キャッシュパーティショニングの制御により s_i の値が変化することが分かる．

式 (2)，(3)，(5)，(8)，および (11) から，優先度制御・キャッシュパーティショニング制御適用時の f_i，P_{total} は定数 C_0，C_1，C_2 を用いて以下の式のように表される．

$$f_i=c\frac{I_i}{L_i-(w_i+r_il_{total}L_i)} \tag{12}$$

$$=c\frac{I_i}{L_i-(m_il_M+r_il_{total}L_i)} \tag{13}$$

$$P_{total}=\sum_i\frac{I_i}{L_i}(C_2f_i^2+C_1f_i+C_0) \tag{14}$$

2.4　電力最適点の導出

本節では，これまでに構築してきたモデルより CMP 全体の消費電力 P_{total} を最小化する優先度・キャッシュパーティショニングを導出する．

まず優先度制御のみを行い（r_i が変化），キャッシュパーティショニング（m_i）を固定したうえで，ラグランジュの未定乗数法を用い，以下の束縛条件のもとで式 (14) を最小化する DRAM アクセスの優先度を導出する．

$$g(f_0,f_1,...,f_{n-1})=\sum_i r_i-1\ \ (1\geq r_i\geq 0) \tag{15}$$

導出の結果，CMP 全体の消費電力はすべてのコアのクロック周波数が等しくなるように DRAM アクセスの優先度を制御することにより最小化されることが示され，そのときの周波数 f_{opt} および待ち時間の比 $r_{opt,i}$，CMP 全体の消費電力 P_{total} は以下のとおりである．導出の詳細な過程は付録 A.1 で述べる．

$$f_{opt}=\frac{c\sum_i\frac{I_i}{L_i}}{\sum_i\frac{L_i-w_i}{L_i}-l_{total}} \tag{16}$$

$$r_{opt,i}=\frac{1}{l_{total}L_i}\left(L_i-w_i-\frac{I_i(\sum_i\frac{L_i-w_i}{L_i}-l_{total})}{\sum_i\frac{I_i}{L_i}}\right) \tag{17}$$

$$P_{total}=(C_2f_{opt}^2+C_1f_{opt}+C_0)\sum_i\frac{I_i}{L_i} \tag{18}$$

次に，最適なキャッシュパーティショニングを導出する．キャッシュパーティショニングを変更することでキャッシュミス回数が変化することは前述のとおりであり，最適なキャッシュパーティショニングとは CMP の消費電力を最小化する m_i（$i:0\leq i\leq n-1$）を達成するキャッシュパーティショニングのことである．

式 (7)，(8)，および (11) より単位時間あたりの総ストール時間（全コアでのストール時間の合計）の割合 l_{stall} は以下の式のように表される．

$$l_{stall}=\sum_i\frac{m_i}{L_i}l_M+l_{total} \tag{19}$$

この l_{stall} の値はキャッシュミス回数の関数であり，キャッシュパーティショニングに依存する値である．式 (16) はこの式 (19) を用いて以下のように書き直すことができる．

$$f_{opt}=c\frac{\sum_i\frac{I_i}{L_i}}{n-l_{stall}} \tag{20}$$

式 (18) より CMP 全体の消費電力は f_{opt} のみの関数であり，式 (20) より最適な周波数 f_{opt} は l_{stall} の値に反比例することが分かる．以上より CMP 全体の消費電力は l_{stall} が最小値をとるときに最小となり，最適なキャッシュパーティショニングは l_{stall} を最小化する

⋆1 要求されるスループットに対してシステムのスループットが不足しているような，定常的にバスが競合している状況においては l_{total} の量が変化しないことは自明．ただし，競合が開始および解消されるときにはメモリアクセスの順番を変更することによってリクエストの到着にずれが生じ，そのため厳密には l_{total} の値が一致しない可能性はある．モデルではこの変化は無視できるものとし，l_{total} の量は変化しないものとして扱う．

キャッシュパーティショニングであることが分かる．

以上より，P_{total} を最小化するキャッシュパーティショニングと DRAM アクセスの優先度の協調制御は以下のとおりである．

(1)　キャッシュパーティショニング制御により l_{stall} を最小化

l_{stall}: 単位時間あたりの全 PU のストール時間の合計

(2)　全コアの周波数が f_{opt} に一致するように DRAM アクセスの優先度を制御

$$f_0 = f_1 = \cdots = f_{n-1} = f_{opt}$$

$$f_{opt} = c\frac{\sum_i \frac{I_i}{L_i}}{n - l_{stall}}$$

なお，2.2 節で述べたとおり，タスクの実行時間は性能制約と等しいため，上記の制御は性能が一定の場合における CMP 全体の消費電力を最小化しており，これは消費エネルギーを最小化することと等価である．

3. 協調制御手法

3.1 制御手法の概要

本節では，2 章で導いた CMP の消費電力を最小化するキャッシュパーティショニングと DRAM アクセスの優先度を，実際の CMP において実アプリケーションの実行時に実現するための制御手法を提案する．

2 章において，消費電力を最小化するキャッシュパーティショニング・優先度，またそのときのクロック周波数 f_{opt}・バス競合に起因する待ち時間の各コアでの分配比率 r_i 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システムにモデルを適用するにあたり，いくつかの考慮すべき点が存在する．まず，実アプリケーションにおいて実行中の L2 キャッシュミス率は変動することが多いため，「$Task_i$ 内では L2 キャッシュミス率は一定である」という仮定が成り立たない．これにより，構築したモデルを実アプリケーションに単純に適用することができない．そこで，アプリケーションを L2 キャッシュミス率が一定と見なすことができる *phase* という短い領域にあらかじめ分割し，各 phase に性能制約を持たせることでこの問題に対応する．phase 内の L2 キャッシュミス率は一定でありモデルの仮定を満たすため，各 phase に対しモデルを適用すればよい．

また実際の CMP の場合，一般に選択可能なクロック周波数・電源電圧の組合せは限られており任意の値を設定することはできない．加えて，モデルではバスの競合により発生する

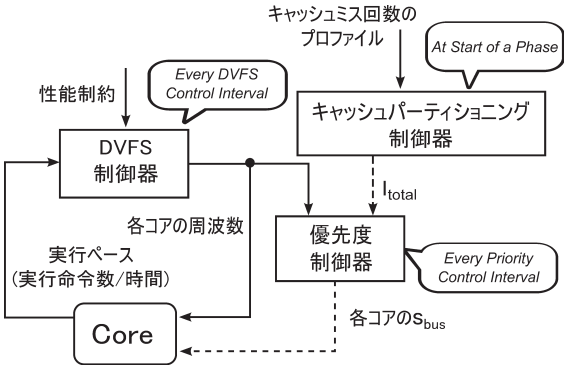


図 3　制御手法のブロック線図
Fig. 3　Block diagram of the proposed method.

待ち時間を任意に各コアに割り振れることを仮定したが，実際の CMP においては任意に割り振ることができない状況が発生する．たとえば，前のリクエストがすでにバスアクセスを開始しており，そこにリクエストが到着したような場合では，アクセスの順番を入れ替えることができないというような状況である．また，選択可能なクロック周波数の値が限られていることから，式 (20) で導出した f_{opt} の値によってはそのクロック周波数値を選択できない可能性があり，仮に選択できたとしても，待ち時間を式 (17) の値に割り振ることができず，性能制約を満たすことができなくなる可能性がある．そこで，本提案手法では周波数を f_{opt} の値に設定するのではなく，PI 制御を用いてつねに性能制約を満たす最低の周波数を選択する DVFS 制御手法を用いる．

以上の点を考慮した提案手法のブロック線図および制御の時間推移を**図 3** および **図 4** に示す．本制御手法はキャッシュパーティショニング制御器，優先度制御器，DVFS 制御器の 3 つのハードウェアと，L2 キャッシュミス回数，性能制約に関する事前のプロファイル情報を用いる．キャッシュパーティショニング制御器は，どれか 1 つのコアの phase が変化するたびに L2 キャッシュミス回数のプロファイルから新しいパーティションを計算し，各コアに割り当てる．また，図 3 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制御器である．優先度制御器は，各コアのクロック周波数の値を観測し，その値を入力として全コアのクロック周波数が等しくなるように一定周期ごとに DRAM アクセスの優先度を設定することで各コアにバス競合に起因する待ち時間 l_{total} を割り振る．DVFS 制御器は，一定周期ごとに各コアの命令の実行ペースと性能制約を比較し，前述の

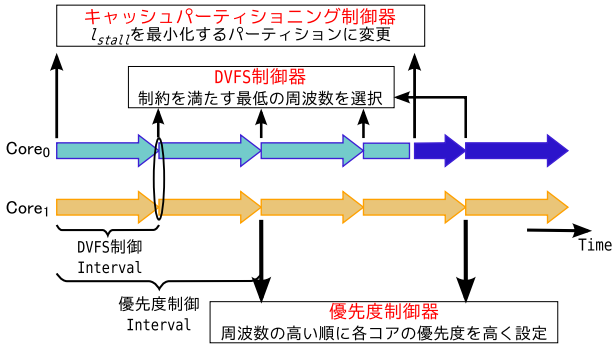


図 4　制御手法の時間推移
Fig. 4　Time flow of the proposed method.

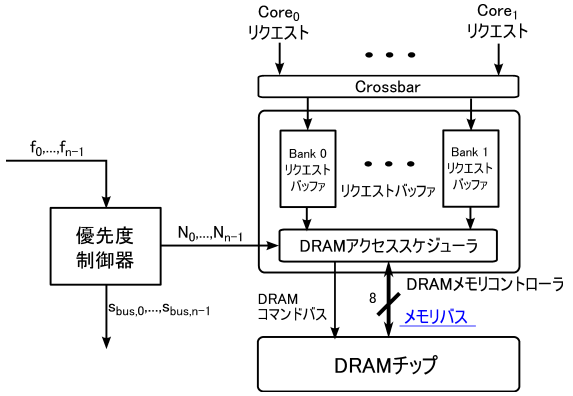


図 5　優先度制御器と DRAM メモリコントローラ
Fig. 5　Structure of the priority controller and the DRAM memory controller.

とおり各コアのクロック周波数・電源電圧をそれぞれの性能制約を満たす値に設定する．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制御器とそのアルゴリズム

キャッシュパーティショニング制御器は，事前のプロファイルの情報をもとに L2 キャッシュのパーティションを l_{stall} の値を最小化するように変更する制御器である．本提案手法ではキャッシュパーティショニングの制御単位をウェイとし，各コアが使用可能なウェイ数を制御器が動的に変更することでキャッシュのパーティショニングを行う．制御に用いるプロファイルの情報とは，実行するアプリケーションのそれぞれの phase における，対象の CMP がとりうる全ウェイ数におけるキャッシュミス回数である．たとえば対象 CMP の L2 キャッシュが 16 ウェイの場合，ウェイ数が 1 から 16 の場合のアプリケーションの各 phase におけるキャッシュミス回数である．このプロファイルは事前に対象の CMP で対象となるアプリケーションを 1 度だけ単独実行し，スタックディスタンスプロファイル[17] を取得することで得ることが可能である（L2 キャッシュのリプレースメントアルゴリズムは LRU を仮定している）．phase 内でのキャッシュミス率は一定であるため，各コア上で同時に実行されている phase の組合せが変化しない限り l_{stall} の値は変化することがない．よって本制御器による L2 キャッシュのパーティション変更は，全コアのうちどれかの 1 つのコアで実行中の phase が変化したときにのみ行えば必要十分である．

キャッシュパーティショニング制御のアルゴリズムは以下のとおりである．phase の開始時に制御器はプロファイルから式 (11) と式 (19) を用いて，全通りのパーティションに対し l_{stall} を計算する．その中から l_{stall} が最小となるパーティションを選択し各コアに割り当てる．なお，各コアには最低でも 1 ウェイは割り当てられるものとする．

キャッシュパーティショニングの実装には，Qureshi らの提案している手法[11] を用いる．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，どのコアがそのブロックを所持しているかを識別する．$Core_i$ のキャッシュミス発生時には，キャッシュパーティショニング制御器は，ミスが発生したキャッシュセットに各コア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 $Core_i$ 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，$Core_i$ 以外が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，$Core_i$ の所持するブロックの LRU ブロックを置換する．本実装手法は制御器で各コアに割り当てられたウェイ数を管理し，リプレースメントアルゴリズムを変更することでパーティショニングを実現している．既存の LRU 置換ポリシと比較すると，置換するブロックの選び方のみを変更するものであるため，性能のオーバヘッドは存在しない．

3.3　優先度制御とそのアルゴリズム

図 5 に提案する優先度制御器と DRAM メモリコントローラの構成を示す．優先度制御器は DRAM メモリコントローラに接続されており，各コアの現在のクロック周波数を入力とし，n コアの CMP における全コアのクロック周波数が等しくなるように各コアからの DRAM アクセスの優先度 N_i を設定する．N_i の値は一定周期で更新され，その周期中の平均周波数が高い順に n から 1 までの値がセットされる．たとえば，$Core_i$ の平均周波数が

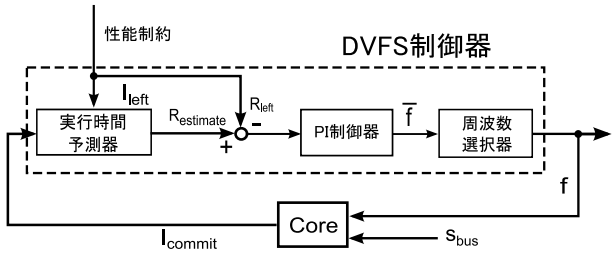


図 6　DVFS 制御手法のブロック線図
Fig. 6　Block diagram of the DVFS controller.

最も高く $Core_j$ 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．

DRAM メモリコントローラ内では，DRAM アクセススケジューラが設定された N_i の値を用い，以下のスケジューリングアルゴリズムに従って，DRAM メモリへのアクセスを一時的に格納するリクエストバッファの中から次に処理するリクエストを決定する．

- リクエストバッファ内の未処理のリクエストのうち，N_i の値が最も大きなコアからのリクエストを最初に処理する．
- 同じコアからのリクエストは FCFS（first-come first-served）ルールで処理する．

このように優先度を決定しリクエストを処理することで，クロック周波数の高いコアからのリクエストは早く処理される．これによりアクセスの待ち時間が少なくなり性能が上がるため，性能制約を満たしつつクロック周波数を下げることが可能となる．一方クロック周波数の低いコアからのリクエストはより優先度の高いコアのリクエストの後で処理されることとなり，待ち時間が増加する．これにより実効稼働時間が減少するため周波数を上げることが必要となる．したがって，クロック周波数の高いコアのクロック周波数は下がり，クロック周波数の低いコアのクロック周波数は上がるため，両コアのクロック周波数は近付くことになる．

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 制御器を提案する．DVFS 制御器のブロック線図を**図 6** に，アルゴリズムを**表 2** に示す．DVFS 制御器は実行時間予測器，PI 制御器，および周波数選択器で構成される．以下にアルゴリズムを示す．

phase P の k 番目の DVFS 周期の最後に，現在の周波数で phase の最後まで実行した場合の残りの実行時間 ${T_{estimate_i}}^k$ を k 番目の周期でコミットした命令数 ${I_{commit_i}}^k$，残りの命令数 ${I_{left_i}}^k$，および DVFS 周期 $T_{DVFS\_interval}$ を用いて予測する（Step 1）．次に，$k+1$ 番目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 ${T_{total_i}}^P$ に対する割合を表す．PI_control 関数は以下の式で表される．

$$\bar{f}_i^k = \bar{f}_i^{k-1} + K_P\left(e_i^k - e_i^{k-1} + \frac{1}{T_I}e_i^k\right) \tag{21}$$

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．提案手法では，**図 7** のようなシュミットトリガ関数（Step 3）を用いて $\bar{f}_i^k$ から実際に選択する周波数 f_i^k を決定する．シュミットトリガ関数とは入力値に対し

表 2　DVFS 制御のアルゴリズム
Table 2　Algorithm of DVFS control.

Initialize (At the start of phase P of $Core_i$)
  set ${I_{total_i}}^P$: total number of instructions in P
  set ${t_{deadline_i}}^P$: 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 Core i:
  /* **Step 1: Remaining time estimation** */
  ${T_{left_i}}^k = {t_{deadline_i}}^P$ - t_{now},　${I_{left_i}}^k$ -= ${I_{commit_i}}^k$
  ${T_{estimate_i}}^k = T_{DVFS\_interval}$ / ${I_{commit_i}}^k$ * ${I_{left_i}}^k$

  /* **Step 2: PI Controller** */
  ${R_{estimate_i}}^k = {T_{estimate_i}}^k / {T_{total_i}}^P$
  ${R_{left_i}}^k = {T_{left_i}}^k / {T_{total_i}}^P$
  $e_i^k = {R_{estimate_i}}^k - {R_{left_i}}^k$
  $\bar{f}_i^k$ = PI_control(e_i^{k-1}, e_i^k, $\bar{f}_i^{k-1}$)

  /* **Step 3: Frequency select** */
  f_i^k = 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
}

立ち上がりと立ち下がりの 2 つの閾値を持った関数であり，今回用いた周波数選択器では，たとえば $\bar{f}_i^k = 1.5\,[\mathrm{GHz}]$ という入力値が与えられた場合，現在のクロック周波数が $\bar{f}_i^k$ より小さい場合には立ち上がりであり（図中の → の線），$f_i^k = 1.4\,[\mathrm{GHz}]$ となる．一方，現在の周波数が $\bar{f}_i^k$ より大きい場合には立ち下がりであり（図中の ← の線），$f_i^k = 1.6\,[\mathrm{GHz}]$ となる．DVFS にはクロック周波数・電源電圧を変更するための時間的なオーバヘッドが存在するため，頻繁な周波数の遷移は性能低下を招き，逆に消費電力が増加してしまう．本関数を用いる目的は周波数の遷移回数を減少させ，消費電力の増加を抑えることである．

また，Step 2 では DVFS の時間オーバヘッドを考慮せずに周波数を求めているため，選択されたクロック周波数が現在のクロック周波数と異なる場合には，再度 DVFS の時間オーバヘッドを考慮してクロック周波数を計算する必要がある．実際の残り時間 ${T_{left}}_i^k$ から DVFS のオーバヘッド $T_{penalty}$ を引いた値を新しく ${T_{left}}_i^k$ とし（Step 4），Step 2，Step 3 を繰り返す．2 回目の Step 3 の後得られた値 f_i^k を次の周期のクロック周波数とする．

なお，DVFS 制御の制御周期は優先度制御の制御周期よりも短く設定する．これは優先度制御の制御周期内で，優先度の変化による実行ペースの変化を反映した値にクロック周波数を収束させるためである．

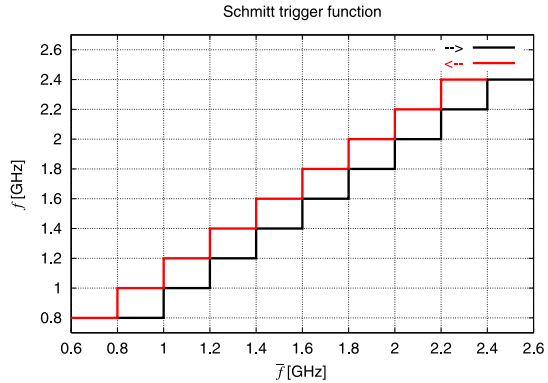


図 7　シュミットトリガ関数
Fig. 7　Schmitt trigger function.

4. 評　　価

4.1 評価環境

4.1.1 CMP の仮定

本章ではシミュレーションにより提案手法の評価を行う．シミュレータには CMP 評価用に拡張した SimpleScalar Tool Set [18] を用い，電力評価には Wattch [19] を用いる．**表 3** にシミュレーションで用いたハードウェア構成を示す．プロセッサコアはアウトオブオーダ・スーパスカラプロセッサであり，非共有の L1 命令/データキャッシュと共有の 16 ウェイ L2 キャッシュを持つことを仮定する．DVFS は全 9 段階のクロック周波数・電源電圧を選択可能とし，クロック周波数は 800 MHz から 200 MHz 刻みで 2,400 MHz までである．主記憶に関しては，2 章で提案するモデルにより近いハードウェアとなる，DRAM バンクを共有資源としない，つまりバンクコンフリクトが起こらないモデル（IDEAL）と，DRAM バンクを各コアで共有しバンクアクセスのレイテンシを詳細にモデル化したモデル（REAL）の 2 つの場合について評価する．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 4** のとおりである．

4.1.2 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 [20] と，SPEC CPU2000 ベンチマーク [21] 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

表 3　ハードウェアの構成
Table 3　Hardware configurations.

	Env: IDEAL	*Env: REAL*
Cores	2	
Clock frequency	800–2,400 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, 64 B line, 10-cycle	
Memory bank	4 bank 10-bus cycle	4 bank Row hit: 4-bus cycle Row closed: 8-bus cycle Row conflict: 16-bus cycle
Memory bus	8 B, 400 MHz	
DVFS penalty	10 us	

表 4　アルゴリズムで用いたパラメータ
Table 4　Algorithm parameters.

Partitioning l_M	22.5 ns
l_B	11.25 ns
Priority feedback interval	400 us
DVFS interval	200 us
PI control gain	K_P: 0.1, T_I: 0.1（2 core）
	K_P: 0.3, T_I: 0.2（4 core）

ず，その後 `H.264` が 2 億命令の実行を完了するまでを評価した．**図 8** は，評価に用いた各アプリケーションのそれぞれの phase について，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 1,000 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，縦軸は MPKI を表す．図 8 より，`H.264` の Phase $B_{H.264}$，`art` の Phase B_{art}，`mcf` の Phase A_{mcf}，B_{mcf} は MPKI が 50 を超えており，非常にキャッシュミス率の高いメモリバウンドな phase であることが分かる．逆に `H.264` の Phase $A_{H.264}$，C $_{H.264}$，`mesa` の Phase B_{mesa}，`bzip2` の Phase A_{bzip2} はすべて MPKI が 10 以下のキャッシュミス率の低い phase である．また，`art` の Phase A_{art}，`mcf` の Phase A_{mcf}，`twolf` の Phase A_{twolf}，`vpr` の Phase A_{vpr}，`bzip2` の Phase B_{bzip2} ではキャッシュサイズの増加にともない，大きく MPKI が減少している．以降これらの phase を *way-sensitive* な phase と呼ぶ．

各アプリケーションの各 phase に対する性能制約は**表 5** のとおりである．単位は IPS（instructions per second）であり，1 秒あたりに実行すべき命令数を表している．`H.264` は Phase $A_{H.264}$，Phase $B_{H.264}$，Phase $C_{H.264}$ で 1 フレームのデコードを行っており，一般的に 30 fps（flames per second）という制約が課される．これは 1 秒間に 30 フレームを処理しなければならないことを意味し，表 5 の性能制約は 30 fps を IPS に変換した値である．SPEC ベンチマークはリアルタイムアプリケーションではないため性能制約を持たない．そこで IPS を性能制約として妥当だと考えられる値（競合がない状況では余裕を持って満たせるが，競合が起こると周波数を高くしないと守れないような制約）を付加した．

また，提案手法の有効性を調べるために以下の 4 つの手法における消費電力を 2 コアおよび 4 コアの CMP で評価した．

- *DVFS only*：**既存手法**．DVFS 制御のみ適用．
- *priority*：DVFS 制御と優先度制御を適用．
- *partition*：DVFS 制御とキャッシュパーティショニング制御を適用．

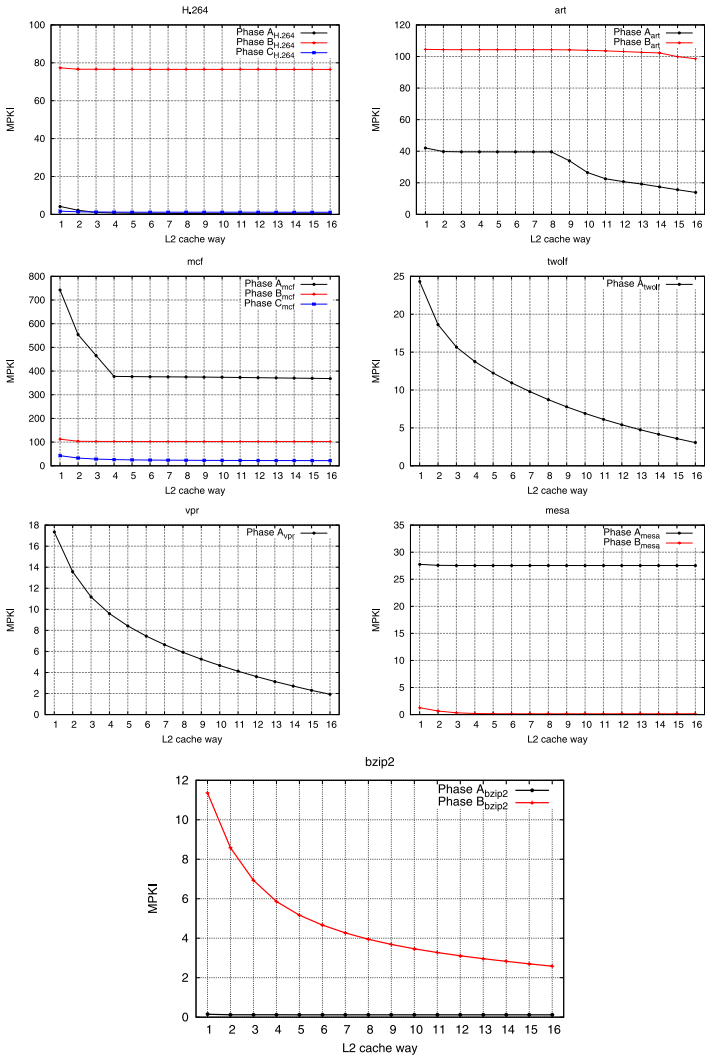


図 8　各アプリケーションの MPKI
Fig. 8　MPKI of the evaluated applications.

表 5　性能制約
Table 5　Performance constraints of the evaluated applications.

	Application	IPS
H.264	Phase $A_{H.264}$	1,589 M
	Phase $B_{H.264}$	199 M
	Phase $C_{H.264}$	1,681 M
art	Phase A_{art}	840 M
	Phase B_{art}	412 M
mcf	Phase A_{mcf}	59 M
	Phase B_{mcf}	271 M
	Phase C_{mcf}	575 M
twolf	Phase A_{twolf}	1,504 M
vpr	Phase A_{vpr}	1,631 M
mesa	Phase A_{mesa}	792 M
	Phase B_{mesa}	2,144 M
bzip2	Phase A_{bzip2}	2,363 M
	Phase B_{bzip2}	1,701 M

- *priority+partition*：**提案手法**．DVFS 制御，優先度制御，およびキャッシュパーティショニング制御を適用．

2 コアの CMP における評価では `H.264` と SPEC CPU2000 ベンチマークから選択した 6 種類のアプリケーションを用いた 6 通りを行い，4 コアの CMP における評価では `H.264`，`art`，`twolf`，および `mesa` の組合せと `H.264`，`twolf`，`bzip2`，および `mesa` の組合せの 2 通りを行った．

4.2　評 価 結 果

4.2.1　2 コアでの評価

図 9 および **図 10** に 2 コアの CMP の *IDEAL* と *REAL* における消費電力の評価結果を示す．図の縦軸は消費電力であり，上記 4 つの手法における各コアの消費電力および CMP 全体の消費電力を *only DVFS* 時の全消費電力の値で正規化した値である．

評価の結果 *IDEAL* では全アプリケーションの組合せにおいて提案手法（*priority+partition*）が，アクセス制御を行わず DVFS 制御のみを行う場合（*DVFS only*）に比べ消費電力を削減できていることが分かる．削減率は `H.264-art` のとき最大となり 28%，平均で 12%の削減を達成している．また全組合せにおいて，提案手法を用いた場合が最も消費電力を削減できている．組合せ別に結果を見ると，`H.264-art` と `H.264-mcf` の場合では，優先度制御・キャッシュパーティショニング制御両方により消費電力を削減できており，

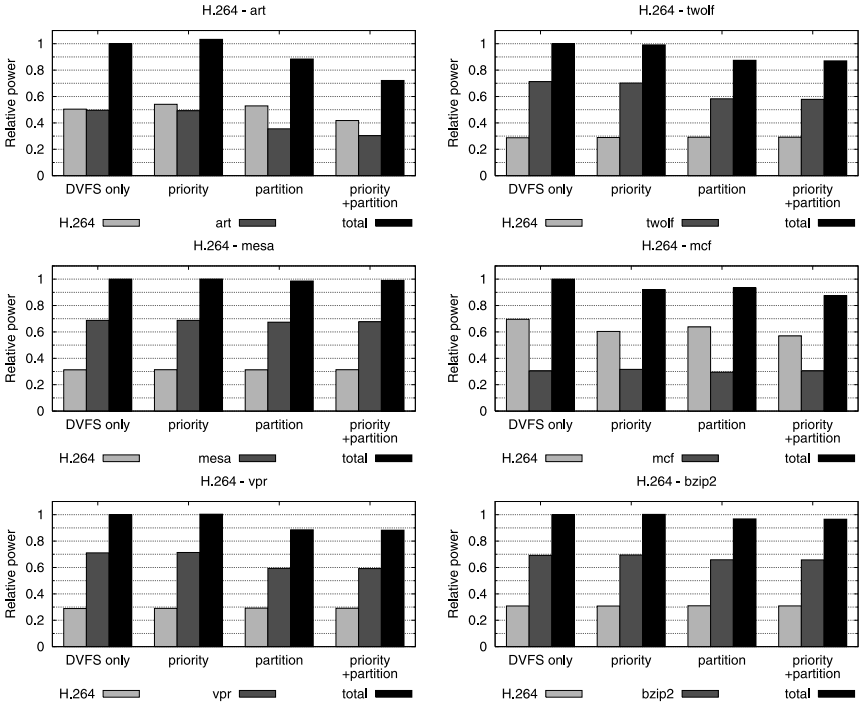


図 9　消費電力（*IDEAL*，2 コア）
Fig. 9　Relative power consumption (*IDEAL*, dual core).

`H.264-vpr` では，優先度制御による削減率が非常に小さく，キャッシュパーティショニング制御のみの場合（*partition*）の削減率が提案手法とほぼ等しい．

REAL においてもすべての組合せで消費電力が削減できており，削減率は `H.264-twolf` のとき最大となり 19%，平均で 10%の削減を達成している．*IDEAL* と比べ優先度制御による削減率は小さいが，逆にキャッシュパーティショニング制御による削減率が大きくなっており，*IDEAL* では提案手法により消費電力を削減できない `H.264-bzip2` において 10%の削減率を達成している．

4.2.2　4 コアでの評価

図 11 および **図 12** に 4 コアの CMP の *IDEAL* と *REAL* における消費電力の評価結果

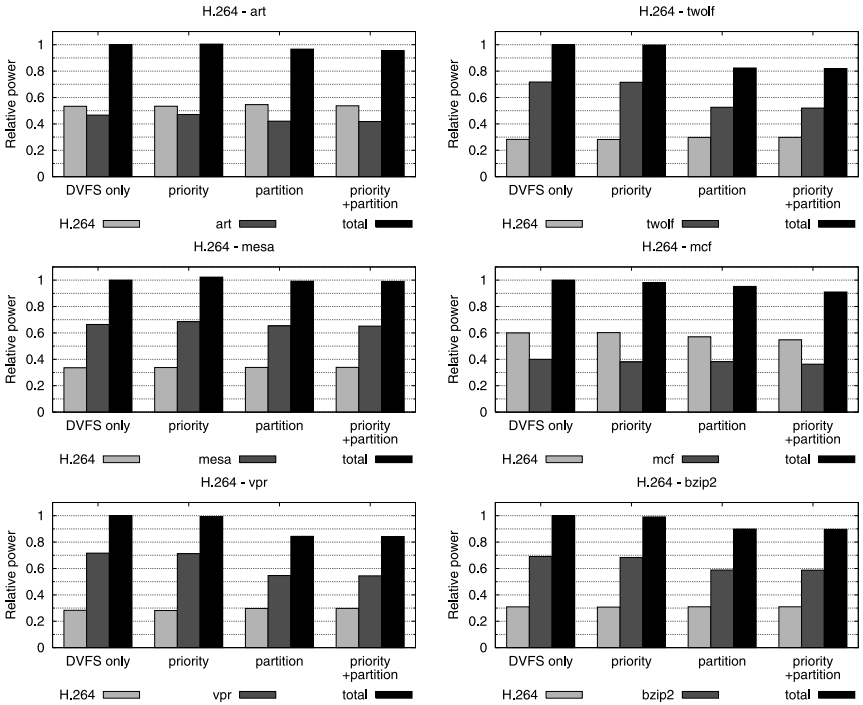


図 10　消費電力（*REAL*，2 コア）
Fig. 10　Relative power consumption (*REAL*, dual core).

を示す．図の見方は図 9 および図 10 と同様である．評価の結果，*IDEAL* では提案手法（*priority+partition*）が，アクセス制御を行わず DVFS 制御のみを行う場合（*DVFS only*）に比べ `H.264-art-twolf-mesa` において 9%，`H.264-twolf-bzip2-mesa` において 7%と平均して 8%消費電力を削減できていることが分かる．また，*REAL* においても結果は同様で提案手法によって消費電力は削減されており，その削減率はそれぞれ 5%と 12%で，平均して 8.5%である．この結果から，4 コアの場合でも 2 コアの場合と同様に消費電力の削減が達成されていることが分かる．削減率が 2 コアの場合と比較して若干少なくなっているが，これは性能制約の与え方を統一したために性能制約を守るために周波数を高く保つ必要があり，待ち時間を割り振っても十分に周波数を低下させることができなかったからだと考

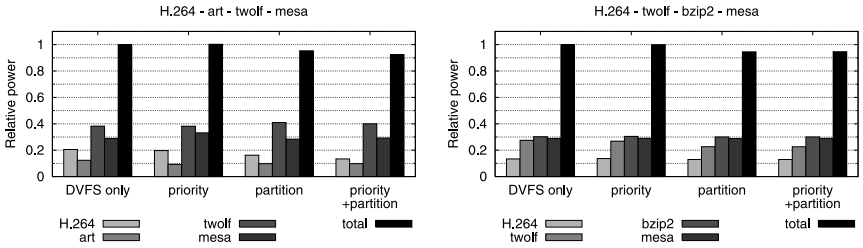


図 11　消費電力（*IDEAL*，4 コア）
Fig. 11　Relative power consumption (*IDEAL*, quad core).

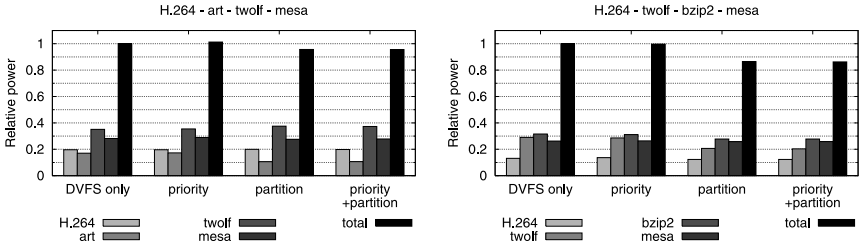


図 12　消費電力（*REAL*，4 コア）
Fig. 12　Relative power consumption (*REAL*, quad core).

えられる．次節で，提案手法の制御が期待どおりに振る舞っているかどうかを，2 コアの場合の詳細な解析を行うことによって考察する．

4.3 考　　察

本節では 2 コアでの評価を解析し，消費電力の評価結果について考察を行う．各組合せにおける，バスの競合による待ち時間の比（r_i），メモリバスの競合により生じる単位時間あたりの待ち時間の全コアでの総和（l_{total}），単位時間あたりの総ストール時間（l_{stall}）を**表 6** に示す．

4.3.1 理想的なモデル（*IDEAL*）における評価結果の考察

`H.264-art` と `H.264-mesa` におけるクロック周波数の推移を**図 13** と**図 14** に示す．図は各コアの，10 ms ごとのクロック周波数の平均値をプロットしたものであり，縦軸はクロック周波数 [MHz]，横軸はプログラムの実行を開始してからの時間 [ms] を表す．これらの図

表 6　ストール時間の分析結果
Table 6　Analysis of the stall time.

		H.264-art			H.264-mcf		
		$r_{H.264}$:r_{art}	l_{total}	l_{stall}	$r_{H.264}$:r_{art}	l_{total}	l_{stall}
IDEAL	DVFS only	0.62:0.38	0.11	1.20	0.55:0.45	0.13	1.24
	priority	0.61:0.39	0.12	1.21	0.37:0.63	0.12	1.20
	partition	0.65:0.35	0.13	1.16	0.56:0.44	0.13	1.21
	priority+partition	0.54:0.46	0.12	1.08	0.42:0.58	0.13	1.18
REAL	DVFS only	0.64:0.36	0.10	1.21	0.60:0.40	0.12	1.34
	priority	0.66:0.34	0.11	1.24	0.49:0.51	0.13	1.33
	partition	0.67:0.33	0.10	1.19	0.60:0.40	0.14	1.29
	priority+partition	0.55:0.45	0.12	1.18	0.51:0.49	0.15	1.27
		H.264-twolf			H.264-vpr		
		$r_{H.264}$:r_{twolf}	l_{total}	l_{stall}	$r_{H.264}$:r_{vpr}	l_{total}	l_{stall}
IDEAL	DVFS only	0.53:0.47	0.04	0.72	0.49:0.51	0.03	0.67
	priority	0.70:0.30	0.05	0.73	0.58:0.42	0.03	0.67
	partition	0.52:0.48	0.02	0.58	0.52:0.48	0.01	0.52
	priority+partition	0.62:0.38	0.02	0.58	0.58:0.42	0.01	0.52
REAL	DVFS only	0.51:0.49	0.05	0.70	0.50:0.50	0.05	0.65
	priority	0.71:0.29	0.06	0.71	0.63:0.37	0.05	0.65
	partition	0.53:0.47	0.03	0.48	0.51:0.49	0.02	0.43
	priority+partition	0.64:0.36	0.03	0.48	0.57:0.43	0.02	0.43
		H.264-mesa			H.264-bzip2		
		$r_{H.264}$:r_{mesa}	l_{total}	l_{stall}	$r_{H.264}$:r_{bzip2}	l_{total}	l_{stall}
IDEAL	DVFS only	0.48:0.52	0.00	0.42	0.55:0.54	0.02	0.37
	priority	0.58:0.42	0.00	0.42	0.68:0.32	0.02	0.37
	partition	0.48:0.52	0.00	0.40	0.55:0.45	0.01	0.37
	priority+partition	0.54:0.46	0.00	0.41	0.66:0.34	0.01	0.37
REAL	DVFS only	0.54:0.46	0.01	0.33	0.54:0.46	0.02	0.31
	priority	0.45:0.55	0.02	0.34	0.72:0.28	0.02	0.30
	partition	0.49:0.51	0.01	0.31	0.54:0.46	0.01	0.29
	priority+partition	0.55:0.45	0.01	0.31	0.68:0.32	0.01	0.29

と表 6 から *IDEAL* の評価結果を考察し，各アクセス制御および協調制御の効果の確認と，提案した制御手法がモデルから導出した最適な協調制御を達成していることを明らかにする．

キャッシュパーティショニング制御について：表 6 より，`H.264-mesa`，`H.264-bzip2` を除き，キャッシュパーティショニングを行うことで単位時間あたりの総ストール時間 l_{stall} が減少していることが分かる．制御手法において述べたとおりキャッシュパーティショニング制御器は，l_{stall} の値を計算し，その値が最小となるパーティションを選択しており，この結果からキャッシュパーティショニング制御器が期待どおりの動作をしていることが分かる．

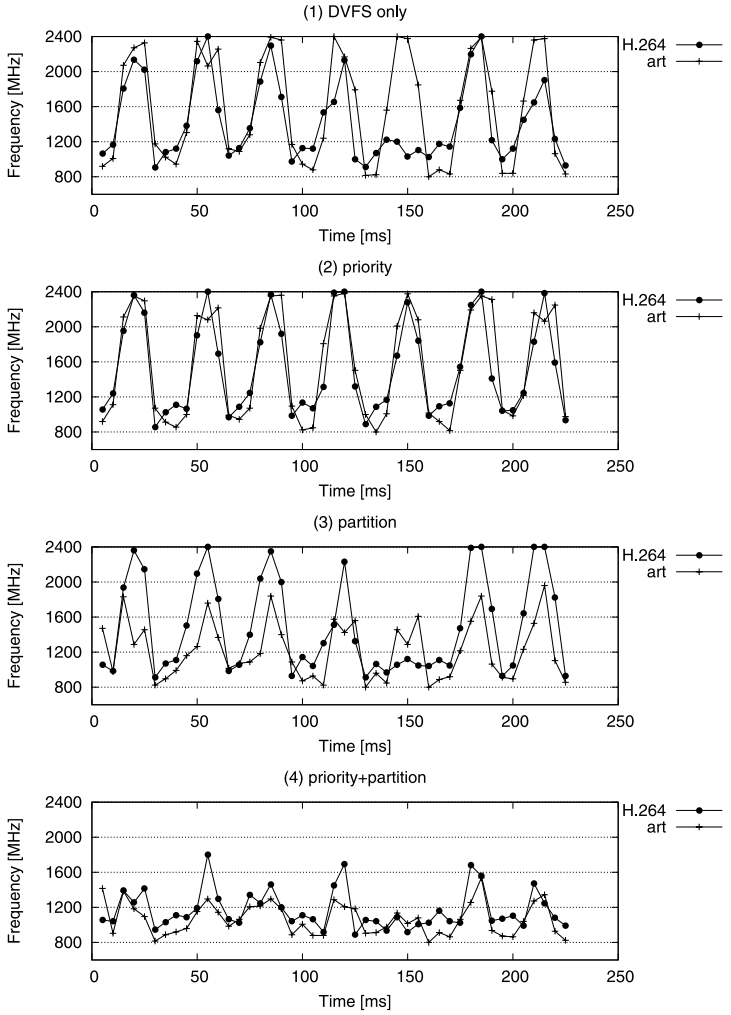


図 13　H.264-art におけるクロック周波数の推移（*IDEAL*，2 コア）
Fig. 13　Transition of the clock frequency when executing H.264-art (*IDEAL*, dual core).

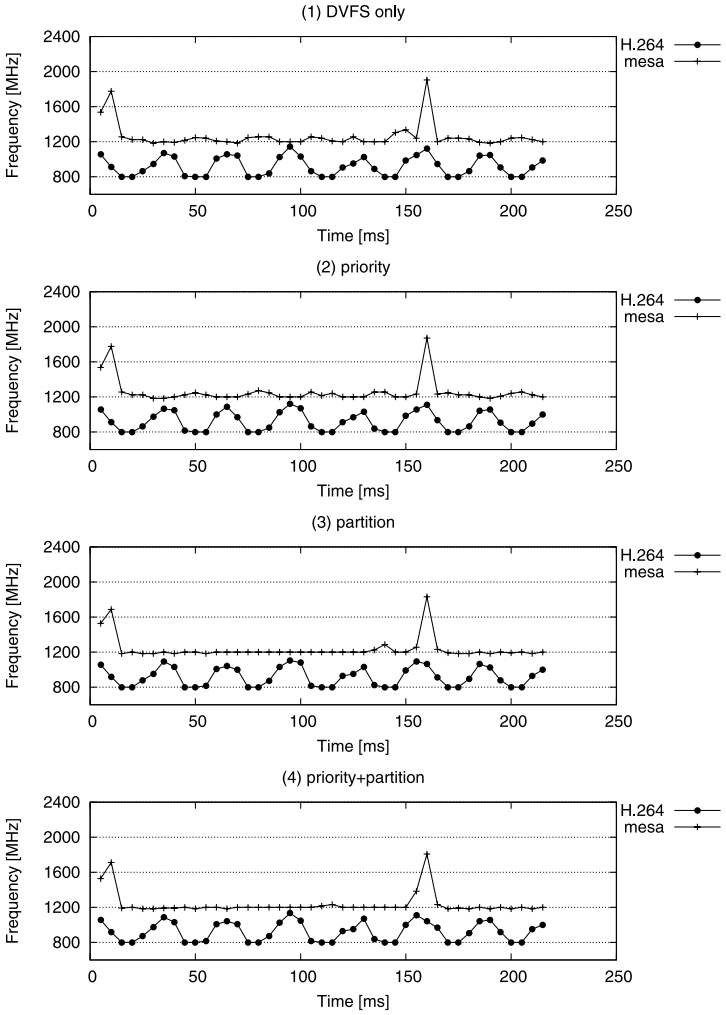


図 14　H.264-mesa におけるクロック周波数の推移（*IDEAL*，2 コア）
Fig. 14　Transition of the clock frequency when executing H.264-mesa (*IDEAL*, dual core).

アプリケーション別で見ると，*partition* による消費電力削減量が大きかった `H.264-twolf`，`H.264-vpr` における l_{stall} の削減量が非常に大きい．また，これらの組合せでは，図 9 のとおり，`H.264` の消費電力はアクセス制御により変化せず，`twolf`，`vpr` の消費電力が大きく削減されている．`twolf`，`vpr` は *way-sensitive* な phase のみで構成されており，キャッシュパーティショニングにより L2 キャッシュの使用率が上がることでキャッシュミス回数が大幅に減少する．一方，`H.264` はどの phase においても L2 キャッシュの使用率の変化により劇的にキャッシュミス回数が変化することはない．したがって，これらのアプリケーションを同時に実行するとキャッシュパーティショニング制御器は分配可能な L2 キャッシュのウェイをすべて `twolf`，`vpr` に割り振ることとなる．その結果，`H.264` のクロック周波数を上昇させることなく，`twolf`，`vpr` のクロック周波数が大きく低下し，消費電力が大きく削減された．

また，`H.264-art`，`H.264-mcf` においても，キャッシュパーティショニング制御を行うことで消費電力が 10%程度削減されている．`H.264-art` の場合，`art` の消費電力が大きく削減されており，図 13 からも `art` のクロック周波数が低下していることが分かる．`art` には *way-sensitive* な Phase A_{art} が存在し，キャッシュパーティショニングを行うことでその phase のストール時間が削減されたことでクロック周波数を低下することができている．

`H.264-mesa`，`H.264-bzip2` ではキャッシュパーティショニングにより l_{stall} の値が変化せず，図 14 のとおり，クロック周波数にも変化が見られない．その結果として消費電力が削減されないか削減分が非常に小さくなっている．`mesa`，`bzip2` は元々L2 キャッシュミス回数の少ないアプリケーションであり，アクセス制御を行わない *DVFS only* の時点で l_{stall} の値が小さく，キャッシュパーティショニング制御が消費電力に影響を与えなかったといえる．

以上の考察より，キャッシュパーティショニング制御器はモデルから導出した制御を達成していることと，アプリケーションが *way-sensitive* な場合に効果的に消費電力を削減できることが分かる．

優先度制御について：3 章で述べたとおり，優先度制御器はすべてのコアのクロック周波数を一致させるように DRAM アクセスの優先度を設定する．図 13，図 14 のクロック周波数の推移を見ると，優先度制御を用いている *priority*，*priority+partition* に関し，`H.264-art` においてはクロック周波数が一致しているが，`H.264-twolf`，`H.264-mesa` においては一致せず，優先度制御導入により周波数が変化している様子は見られない．しかしながら表 6 のバスの競合による待ち時間の比 r を見ると，優先度制御により待ち時間の比が変化し，クロック周波数の低い `H.264` の待ち時間が増加していることが分かる．つまり，待ち時間の

割り振りは正常に行われており，優先度制御器による DRAM アクセスの優先度の設定は期待したとおりに行われていることが分かる．

H.264-art の場合，*priority* では消費電力は削減されないが，*priority+partition* では *partition* に比べ 20%程度削減率が向上している．特に H.264 の消費電力削減が大きい．図 13 のとおり，*DVFS only* の時点でクロック周波数がほぼ一致しているため，*priority* では待ち時間の比が変化しない．そのためクロック周波数も *DVFS only* 時とほぼ変わらないが，提案手法の *priority+partition* では，優先度制御により待ち時間が art に割り振られることで H.264 のクロック周波数が大きく低下している．クロック周波数の低下が大きかった部分は，前述のとおり，Phase $B_{H.264}$ が実行されている部分である．この phase は非常にメモリバウンドな phase であるため，少しの待ち時間の変化が大きく性能を変化させる．そのため，優先度制御により待ち時間の比が 0.65 : 0.35 から 0.54 : 0.46 へと少量変化しただけで大きな消費電力削減へとつながった．

H.264-twolf，H.264-mesa の場合，待ち時間は適切に割り振られているにもかかわらず消費電力が削減されなかったのは，これらの組合せの場合，待ち時間の合計が小さく，待ち時間の比の変化が性能へ影響を与えなかったことが原因である．表 6 の l_{total} は，メモリバスの競合により生じる単位時間あたりの待ち時間を全コアで合計した値であり，優先度制御器はこの l_{total} を各コアに割り振る．l_{total} の値は，式 (10) のとおり，時間あたりのキャッシュミス回数に依存して決まる値であり，H.264-twolf の場合 0.02 であり，H.264-mesa では 0.01 以下となっている．これは優先度制御の影響が大きかった H.264-art の約 5 分の 1 以下の値である．

以上の考察より，優先度制御器はモデルから求めた条件を達成するように優先度を設定しているが，待ち時間の合計 l_{total} の値によっては，消費電力を削減できず，アプリケーションの単位時間あたりのキャッシュミス率が高い組合せの場合に優先度制御により効果的に消費電力を削減できることが分かる．

キャッシュパーティショニングと優先度の協調制御の効果について：前述のとおり，H.264-art および H.264-mcf では，提案手法である *priority+partition* の場合に最も消費電力が削減できており，そのときの削減率は *priority*，*partition* の両方の削減率を大きく上回っている．さらに注目すべき点は，H.264-art において *priority* では消費電力を削減できなかったが，キャッシュパーティショニングの制御を行うことで *priority+partition* では大きく消費電力を削減できている点である．図 13 から分かるように，*partition* ではキャッシュパーティショニング制御を行うことで art のクロック周波数が下がり，H.264 と art のクロック周波数に差が生じている．それにより優先度制御が有効に働くことで，*DVFS only* と比較して両アプリケーションのクロック周波数が大幅に下がり，28%の消費電力削減となっている．以上より，優先度とキャッシュパーティショニングの協調制御は，独立にアクセス制御を用いた場合に比べ CMP の消費電力の削減に対し効果的であるといえる．

4.3.2　現実的な環境（*REAL*）における評価結果の考察

以上の *IDEAL* における考察より，提案する制御手法が想定するハードウェアモデルにおいてはモデルから導出した最適な協調制御を達成できており，協調制御を行うことで消費電力が効率的に削減できることが分かった．しかしながら，実際の CMP は主記憶 DRAM バンクも共有しており，DRAM アクセスのレイテンシも一定ではない．そこで本項では実際の CMP により近い環境である *REAL* における消費電力の評価結果を考察し，本提案手法の実際の CMP 環境における有効性について述べる．

REAL における評価の結果，前述のとおり提案手法により消費電力削減できている．しかしながら，アプリケーションによっては *IDEAL* と削減率で大きな差が生じている．H.264-art の場合，削減率が大きく減少し約 5%となっている．特に *partition*，*priority+partition* の差が大きい．これは主記憶 DRAM バンクが競合により，特にメモリバウンドな phase において性能が大きく低下し，提案手法による制御の余地がなくなってしまったためである．

一方で，H.264-twolf，H.264-vpr，H.264-bzip2 では，提案手法による消費電力の削減率が *IDEAL* の場合よりも増加している．表 6 を見ると，いずれの場合においても l_{stall} の *DVFS only* に対する減少率が *IDEAL* を上回っており，これにより削減率が向上している．twolf，vpr，bzip2 はキャッシュパーティショニングを行うと全 DRAM アクセスにおける Row hit の割合が増加する傾向にある．*IDEAL* では DRAM アクセスのレイテンシが Row hit と Row conflict で異なるため，そのレイテンシの差がさらなるストール時間の減少につながったものである．同様の傾向は，上記アプリケーションのうち twolf，bzip2 を含む，図 12 の右側のグラフの H.264-twolf-bzip2-mesa でも見られており，この場合でも *IDEAL* よりも *REAL* の方が高い消費電力の削減率を達成している．

以上より，現実的な環境下においても提案手法の効果があることが確認できた．しかしながら，考察の結果，主記憶 DRAM バンクの競合および DRAM アクセスレイテンシの違いが消費電力に大きく影響を与えることが分かり，さらなる低消費電力化のためには主記憶 DRAM バンクの競合も考慮した協調制御手法が必要であるといえる．

5. 関 連 研 究

共有資源の競合の関連研究：CMP における共有資源の競合の問題に対し，従来より多くの研究がなされている[4)–14),22),23)]．共有キャッシュの競合に対して，Chandra ら[4)] は同時実行されるアプリケーションの静的な情報から各コアのキャッシュミス回数を予測するモデルを提案し，Stone ら[12)] はキャッシュサイズが変化したときのキャッシュミス回数のプロファイルが与えられている場合の最適なキャッシュパーティショニング手法を提案した．これらの手法がプロファイル情報を用いることを前提とした静的な手法であるのに対し，Suh ら[6),7)] は，スタックディスタンスカウンタを利用した動的なキャッシュパーティショニング手法を提案した．この手法はカウンタの実装コストが大きいため，これに対して Qureshi ら[11)] は，ハードウェアオーバヘッドの小さい動的なキャッシュパーティショニング手法である UCP の提案を行った．また，キャッシュパーティショニングを行う際の指標についても多くの検討がなされている[5),8),10)]．

競合による性能低下を回避するキャッシュパーティショニング以外の手法として，Qureshi ら[23)] は，キャッシュのリプレースメントの際に，主記憶から取得されたデータを LRU ポジションに挿入する新しいキャッシュリプレースメントポリシである DIP を提案し，さらに Jaleel ら[9)] は，実行中のアプリケーションの性質も考慮した TADIP を提案した．また，メモリバスや主記憶 DRAM バンクの競合に対しては，QoS を保つメモリアクセススケジューラ[13),14)] が提案されている．

DVFS 制御手法の関連研究：DVFS を用いた低消費電力化技術としてこれまでに，ユニプロセッサを対象としたリアルタイム DVFS 制御手法[22),24),25)]，シングルコアマルチクロックドメイン SoC および CMP を対象とした DVFS 制御手法[26)]，CMP を対象としたリアルタイム DVFS 制御手法[16),27),28)] が提案されている．

Ishihara ら[24)] は，実行時の電源電圧のスケジューリングを線形計画問題として定式化し，実行されているアプリケーションが実時間制約を持つ場合にプロセッサの消費電力を最小化するスケジューリングが存在することを示した．Pillai ら[22)] は，システム使用率を 100%まで利用可能な実時間スケジューリングアルゴリズムである EDF を拡張した静的な DVFS 制御手法を提案した．Poellabauer ら[25)] は，メモリバウンドなリアルタイムアプリケーションを対象とし，実行時の情報をフィードバックすることで実行時間をより正確に予測する DVFS 制御手法を提案した．これらの研究はユニプロセッサを対象にした DVFS 制御手法である点で本研究とは異なる．

Wu ら[26)] は，シングルコアマルチクロックドメイン SoC および CMP を対象とし，同期用のキューおよびタスクキューの占有率をフィードバックに用いた PID 制御による DVFS 制御手法を提案した．しかしながらこの手法は，性能制約は仮定せず，1 つの並列アプリケーションが複数のクロックドメインやコアで実行される状況を想定している点で本研究と異なる．

Funaoka ら[27)] は，性能制約を持つ CMP に対し，実時間スケジューリングと DVFS 制御を組み合わせた実時間電圧周波数制御（RT-VFS）手法を提案した．しかしながら Funaoka らの手法では，CMP における共有資源の競合の問題は考えられていない．また，Watanabe ら[16)] と椎名ら[28)] は，性能制約を持つ CMP を対象にし，共有資源の競合を考慮した DVFS 制御手法を提案した．これらの研究は LLC が非共有の CMP を想定しており，LLC における競合を考慮していない点で本研究とは異なる．

6. まとめと今後の課題

本論文では，性能制約を持つ CMP の低消費電力化手法として，従来の DVFS に加え共有資源へのアクセスを協調制御することにより CMP の消費電力を削減する手法を提案した．具体的には，最下位共有キャッシュにおけるキャッシュパーティショニングと DRAM アクセスの優先度を協調制御することで，性能制約を満たし，かつ消費電力を最小にする最適なクロック周波数・電源電圧を選択する手法である．まず，協調制御時の消費電力をモデル化することにより，消費電力を最小化する最適なキャッシュパーティショニングと優先度を導出した．その後，キャッシュパーティショニング制御器，優先度制御器，DVFS 制御器を用い，低消費電力化のためのモデルに基づく協調制御手法を提案した．

評価の結果，提案手法はモデル化の対象である主記憶 DRAM バンクが競合しない環境では，2 コアの CMP において最大で 28%，平均で 12%，また 4 コアの CMP において平均 8%の消費電力を削減できることが分かった．また，主記憶 DRAM バンクが競合する環境においても 2 コアの CMP において最大 19%，平均で 10%，4 コアの CMP において平均 8.5%の消費電力を削減できた．

今後の課題としては，まず主記憶 DRAM バンクの競合も考慮したモデルへの拡張があげられる．本論文では評価において，主記憶 DRAM バンクを共有した場合にも消費電力が削減できることが示されたが，モデルを拡張することによりさらなる低消費電力化の可能性がある．主記憶 DRAM バンク競合のモデル化は，アクセスレイテンシが DRAM へのアクセスパターンによって変化するため，より複雑で興味深い課題である．

次に，プロファイル情報を用いた現在の協調制御手法の改良も課題としてあげられる．今回の手法では，プロファイル情報として実行命令数，キャッシュサイズを変更した場合のキャッシュミス回数が必要である．しかしながら，これらの値はプログラムの入力データセットにより大きく異なる値である．そのため，各データセットに対応したプロファイルが必要があるが，全データセットのプロファイルを事前に取得すことは現実的ではない場合には妥当な条件とはいえない．制御手法を改良するにあたり，現在プロファイルを用いている，キャッシュパーティショニング制御器と DVFS 制御器に変更を加える必要がある．キャッシュパーティショニング制御器に関しては，関連研究で述べたとおり，実行時にスタックディスタンスを取得でき，ハードウェアコストも小さい手法が提案されており[11]，その手法を拡張することで対応可能である．DVFS 制御器に関しては，各データセットごとの実行命令数を用いるのではなく，最悪実行時間となる場合の実行命令数を使用することが考えられる．最悪実行時間とはプログラムが最長の実行パスを通った場合の時間のことであり，リアルタイムプログラムは最悪実行時間の値を事前に得ていることが一般的である．通常，実行を繰り返すことで最悪実行時間は見積もられており，そのときの実行命令数を所持していることは妥当であるといえる．今後は，最悪実行時間となる場合の実行命令数を用いた制御手法の有効性を静的な手法を用いた場合と比較し評価していくことが必要である．

将来的な課題として，メニーコアプロセッサへの拡張があげられる．本論文の「全コアで L2 キャッシュとメモリバスを共有する」という仮定はメニーコアプロセッサでは成立しないと思われるが，一方でコア間で共有される資源が多くなることは必然であり，本論文が明らかにした「共有資源の競合に対する協調制御が消費電力削減に効果的である」という知見の適用範囲と有効性はさらに広がるものと考えられる．この点についても検討を行いたい．

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」，科学研究費補助金（挑戦的萌芽研究 No.22650008），および科学研究費補助金（若手研究（B）No.21700054）の支援によって行われた．

参 考 文 献

1) Burd, T.D. and Brodersen, R.W.: Energy Efficient CMOS Microprocessor Design, *HICSS '95: Proc. 28th Hawaii International Conference on System Sciences*, pp.288–297 (1995).
2) Chandrakasan, A.P., Sheng, S. and Brodersen, R.W.: Low-Power CMOS Digital Design, *IEEE Journal of Solid-State Circuits*, Vol.27, No.4, pp.473–484 (1992).
3) Pering, T., Burd, T. and Brodersen, R.: The simulation and evaluation of dynamic voltage scaling algorithms, *ISLPED '98: Proc. 1998 International Symposium on Low Power Electronics and Design*, pp.76–81 (1998).
4) Chandra, D., Guo, F., Kim, S. and Solihin, Y.: Predicting Inter-Thread Cache Contention on a Chip Multi-Processor Architecture, *HPCA '05: Proc. 11th International Symposium on High Performance Computer Architecture*, pp.340–351 (2005).
5) Chang, J. and Sohi, G.S.: Cooperative Cache Partitioning for Chip Multiprocessors, *SC '07: Proc. 21st Annual International Conference on Supercomputing*, pp.242–252 (2007).
6) G. Edward Suh, Srinivas Devadas, L.R.: A New Memory Monitoring Scheme for Memory-Aware Scheduling and Partitioning, *HPCA '02: Proc. 8th International Symposium on High-Performance Computer Architecture*, pp.117–128 (2002).
7) Suh, G.E., Rudolph, L. and Devadas, S.: Dynamic Partitioning of Shared Cache Memory, *The Journal of Supercomputing*, Vol.28, No.1, pp.7–26 (2004).
8) Hsu, L.R., Reinhardt, S.K., Iyer, R. and Makineni, S.: Communist, Utilitarian, and Capitalist Cache Policies on CMPs: Caches as a Shared Resource, *PACT '06: Proc. 15th International Conference on Parallel Architectures and Compilation Techniques*, pp.13–22 (2006).
9) Jaleel, A., Hasenplaugh, W., Qureshi, M., Sebot, J., Steely, S., Jr. and Emer, J.: Adaptive Insertion Policies for Managing Shared Caches, *PACT '08: Proc. 17th International Conference on Parallel Architectures and Compilation Techniques*, pp.208–219 (2008).
10) Kim, S., Chandra, D. and Solihin, Y.: Fair Cache Sharing and Partitioning in a Chip Multiprocessor Architecture, *PACT '04: Proc. 13th International Conference on Parallel Architectures and Compilation Techniques*, pp.111–122 (2004).
11) Qureshi, M.K. and Patt, Y.N.: Utility-Based Cache Partitioning: A Low-Overhead, High-Performance, Runtime Mechanism to Partition Shared Caches, *MICRO '06: Proc. 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pp.423–432 (2006).
12) Stone, H.S., Turek, J. and Wolf, J.L.: Optimal Partitioning of Cache Memory, *IEEE Trans. Computers*, Vol.41, No.9, pp.1054–1068 (1992).
13) Mutlu, O. and Moscibroda, T.: Stall-Time Fair Memory Access Scheduling for Chip Multiprocessors, *MICRO '07: Proc. 40th Annual IEEE/ACM International Symposium on Microarchitecture*, pp.146–160 (2007).
14) Nesbit, K.J., Aggarwal, N., Laudon, J. and Smith, J.E.: Fair Queuing Memory Systems, *MICRO '06: Proc. 39th Annual IEEE/ACM International Symposium on Microarchitecture*, pp.208–222 (2006).

15) Takagi, N., Sasaki, H., Kondo, M. and Nakamura, H.: Cooperative Shared Resource Access Control for Low-power Chip Multiprocessors, *ISLPED '09: Proc. 14th ACM/IEEE international symposium on Low power electronics and design*, pp.177–182 (2009).
16) Watanabe, R., Kondo, M., Nakamura, H. and Nanya, T.: Power Reduction of Chip Multi-Processors using Shared Resource Control Cooperation with DVFS, *ICCD '07: Proc. 25th International Conference on Conputer Design*, pp.615–622 (2007).
17) Cascaval, C., Rose, L.D., Padua, D.A. and Reed, D.A.: Compile-Time Based Performance Prediction, *LCPC '99: Proc. 12th International Workshop on Languages and Compilers for Parallel Computing*, pp.365–379 (1999).
18) Austin, T., Larson, E. and Ernst, D.: SimpleScalar: An Infrastructure for Computer System Modeling, *Computer*, Vol.35, No.2, pp.59–67 (2002).
19) Brooks, D., Tiwari, V. and Martonosi, M.: Wattch: A Framework for Architectural-Level Power Analysis and Optimizations, *ISCA '00: Proc. 27th Annual International Symposium on Computer Architecture*, pp.83–94 (2000).
20) Sühring, K., H.264/AVC Software Coordination: H.264/AVC JM reference software. http://iphome.hhi.de/suehring/tml/
21) Standard Performance Evaluation Corporation (SPEC): SPEC CPU2000. http://www.specbench.org
22) Pillai, P. and Shin, K.G.: Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operationg Systems, *SOSP '01: Proc. 18th ACM Symposium on Operating Systems Principles*, pp.89–102 (2001).
23) Qureshi, M.K., Jaleel, A., Patt, Y.N., Steely, S.C. and Emer, J.: Adaptive insertion policies for high performance caching, *ISCA '07: Proc. 34th Annual International Symposium on Computer Architecture*, pp.381–391 (2007).
24) Ishihara, T. and Yasuura, H.: Voltage Scheduling Problem for Dynamically Variable Voltage Processors, *ISLPED '98: Proc. 1998 International Symposium on Low Power Electronics and Design*, pp.197–202 (1998).
25) Poellabauer, C., Singleton, L. and Schwan, K.: Feedback-Based Dynamic Voltage and Frequency Scaling for Memory-Bound Real-Time Applications, *RTAS '05: Proc. 11th IEEE Real Time on Embedded Technology and Applications Symposium*, pp.234–243 (2005).
26) Wu, Q., Juang, P., Martonosi, M., Peh, L.-S. and Clark, D.W.: Formal Control Techniques for Power-Performance Management, *IEEE Micro*, Vol.25, No.5, pp.52–62 (2005).
27) Funaoka, K., Takeda, A., Kato, S. and Yamasaki, N.: Dynamic Voltage and Frequency Scaling for Optimal Real-Time Scheduling on Multiprocessors, *SIES '08: Proc. 2008 International Symposium on Industrial Embedded Systems*, pp.27–33 (2008).
28) 椎名公康, 近藤正章, 今井　雅, 中村　宏, 南谷　崇：共有資源の優先度制御によるチップ・マルチプロセッサの省電力化手法，先進的計算基盤システムシンポジウム SACSIS 2008，pp.317–324 (2008).

付　　録

A.1　消費電力を最小化する DRAM アクセスの優先度の導出

2 章において，CMP 全体の消費電力を最小化するためには，すべてのコアのクロック周波数を一致させるように DRAM アクセスの優先度を制御すればよいと述べた．本節では，ラグランジュの未定乗数法を用いた上記条件の詳細な導出過程について述べる．

r_i は式 (12) を用いて f_i の関数として以下のように書き表せる．

$$r_i = \frac{1}{l_{total} L_i}\left(L_i - w_i - \frac{cI_i}{f_i}\right) \tag{22}$$

よって，式 (15) の束縛条件は以下のように書き表せる．

$$g = \sum_i \left(\frac{1}{l_{total} L_i}\left(L_i - w_i - \frac{cI_i}{f_i}\right)\right) - 1 \tag{23}$$

式 (14)，式 (23) を用いて以下の $n+1$ 本の方程式を連立させる．

$$\frac{\partial}{\partial f_i}(P_{total} + \lambda g) = 0$$

$$\Leftrightarrow \frac{\partial}{\partial f_i}\left(\sum_i \left(\frac{I_i}{L_i}(C_2 {f_i}^2 + C_1 f_i + C_0)\right) + \lambda\left(\sum_i \left(\frac{1}{l_{total} L_i}(L_i - w_i - \frac{cI_i}{f_i})\right) - 1\right)\right) = 0 \tag{24}$$

$$\frac{\partial}{\partial \lambda}(P_{total} + \lambda g) = 0 \tag{25}$$

式 (24) より以下の n 本の式が得られる．

$$\frac{I_i}{L_i}(2C_2 f_i + C_1) + \lambda\left(\frac{c}{l_{total}}\frac{I_i}{L_i}\frac{1}{{f_i}^2}\right) = 0 \tag{26}$$

式 (26) は以下のように i に依存する項と依存しない項に分けることができる．

$$2C_2{f_i}^3 + C_1{f_i}^2 = -\frac{c\lambda}{l_{total}} \tag{27}$$

これにより，任意の j，k について以下の式が成り立つことが分かる．

$$\begin{aligned}
&2C_2f_j^3 + C_1f_j^2 = 2C_2f_k^3 + C_1f_k^2\\
&\Leftrightarrow 2C_2(f_j^3 - f_k^3) + C_1(f_j^2 - f_k^2) = 0\\
&\Leftrightarrow (f_j - f_k)(2C_2(f_j^2 + f_jf_k + f_k^2) + C_1(f_j + f_k)) = 0
\end{aligned} \tag{28}$$

式 (28) の左辺第 2 項はつねに正である．式 (28) を満たすための条件は $f_j = f_k$ となり，これより各コアのクロック周波数が等しくなるように DRAM アクセスの優先度を制御することで全体の消費電力が最小化されることが示される．

またこの条件より，一致したときのクロック周波数 f_{opt} を用いて式 (25) は以下のように書き直すことができる．

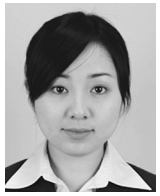
$$\begin{aligned}
&\sum_i\left(\frac{1}{l_{total}L_i}\left(L_i - w_i - \frac{cI_i}{f_{opt}}\right)\right) = 1\\
&\Leftrightarrow \frac{1}{l_{total}}\left(\sum_i\frac{L_i - w_i}{L_i} - \frac{c}{f_{opt}}\sum_i\frac{I_i}{L_i}\right) = 1
\end{aligned} \tag{29}$$

この式 (29) より，f_{opt}（式 (16)）および待ち時間の比 r_i（式 (17)）が導出される．

(平成 22 年 7 月 26 日受付)
(平成 22 年 10 月 28 日採録)

佐々木　広（正会員）

2003 年東京大学工学部計数工学科卒業．2005 年同大学大学院情報理工学系研究科修士課程修了．2008 年同大学院工学系研究科博士課程修了．博士（工学）．同年東京大学先端科学技術研究センター特任助教．2010 年東京大学大学院情報理工学系研究科特任助教．計算機アーキテクチャ，オペレーティングシステムの研究に従事．IEEE，ACM，USENIX 各会員．



高木　紀子（正会員）

2008 年東京大学工学部計数工学科卒業．2010 年同大学大学院情報理工学系研究科修士課程修了．同年富士通株式会社入社．次世代テクニカルコンピューティング開発本部にてスーパコンピュータ・サーバ向け CPU の開発に従事．

近藤　正章（正会員）

1998 年筑波大学第三学群情報学類卒業．2000 年同大学大学院工学研究科博士前期課程修了．2003 年東京大学大学院工学系研究科先端学際工学専攻修了．博士（工学）．独立行政法人科学技術振興機構戦略的創造研究推進事業 CREST 研究員，2004 年東京大学先端科学技術研究センター特任助手，2007 年同特任准教授を経て，現在，電気通信大学大学院情報システム学研究科准教授．計算機アーキテクチャ，ハイパフォーマンスコンピューティング，ディペンダブルコンピューティングの研究に従事．電子情報通信学会，IEEE，ACM 各会員．

中村　　宏（正会員）

1985 年東京大学工学部電子工学科卒業．1990 年同大学大学院工学系研究科電気工学専攻博士課程修了．工学博士．同年筑波大学電子・情報工学系助手．同講師，同助教授を経て，1996 年東京大学先端科学技術研究センター助教授，2010 年東京大学大学院情報理工学系研究科教授．この間，1996〜1997 年カリフォルニア大学アーバイン校客員助教授．高性能・低消費電力 VLSI システム，省電力コンピューティング，ハイパフォーマンスコンピューティングの研究に従事．情報処理学会より論文賞（平成 5 年度），山下記念研究賞（平成 6 年度），坂井記念特別賞（平成 13 年度）各受賞．IEICE，IEEE，ACM 各会員．