社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－ARC－175 (16)
2007／11／22

共有資源の優先度と電源電圧の協調制御による チップマルチプロセッサの省電力化

椎名公康† 近藤正章† 今井雅†
中村宏† 南谷崇†

複数のプロセッサコアを1チップに搭載するチップマルチプロセッサ (CMP) は，そのスケーラビリティから近年注目されているアーキテクチャである．CMP では，複数のプロセッサコア (PU) が L2 キャッシュやチップ・メモリ間バスを共有するために PU 間にリソース競合が発生し性能の低下や消費電力の増加を招くことがある．本稿では，共有リソースアクセスの優先度を導入し，周波数・電源電圧と共に優先度を制御することにより，各 PU 間における競合を調節し消費電力を低減する手法を提案する．いくつかのベンチマークプログラムで提案手法を評価した結果，平均で 11%の消費電力削減を達成できることがわかった．

A Method of Priority and Voltage/Frequency Control for Low-Power in Chip Multiprocessors

Kimiyasu Shiina,† Masaaki Kondo,† Masashi Imai,†
Hiroshi Nakamura† and Takashi Nanya†

In a Single Chip Multiprocessor (CMP), threads on different cores share hardware resources such as cache memory and memory bus. Resource contention significantly degrades performance of each thread. In this paper, we proposes a priority control technique cooperating with DVFS. By the priority control, performance penalty caused by the resource contention for each thread is altered, and therefore, the required voltage / frequency setting is changed. The proposed technique tries to optimize the priority and voltage / frequency to minimize power consumption. We evaluate our technique and the evaluation results show that it reduces 11% power consumption on the average.

1. はじめに

近年，消費電力や発熱量の限界から，クロック周波数向上による高性能化が望めなくなりつつあり，VLSI チップにおける性能向上手段として，複数のプロセッサコア (PU) を1チップに搭載するチップマルチプロセッサ (Chip Multi-Processor: CMP) が今後の有効なアーキテクチャとして注目されている．複数の PU を用いることで，シングルタスクの並列処理，あるいは複数タスクの並行処理を行うことで，高い処理能力を得られる．したがって，周波数の向上に頼らずに高性能化が期待できることから，性能あたりの消費電力効率が向上する．

CMP ではリソース有効活用の観点から，複数の PU があるメモリ階層以下 (例えば L2 キャッシュ以下) にあるキャッシュやバスを共有するのが一般的である．この共有リソースの使用率はプログラムの性質に大きく依存し，同時に実行するプログラムの組み合わせによってはリソース競合が発生する．

リソース競合が発生すると，アクセスした PU には，共有リソースに対して既にアクセスしている PU のデータ転送を待つための待ち時間が生じる．この待ち時間は，各 PU においてプログラムを処理可能な実効稼働時間を減少させる．この場合は実時間制約を持つプログラムを仮定すると，各 PU は性能制約を満たすために，クロック周波数を上げる必要が出てくる．このことは消費電力の増加も招く．

前述したとおり，競合の起こり方はプログラムの組み合わせ，各 PU の共有リソースに対するアクセスパターンに大きく依存する．また性能制約を満たすために必要なクロック周波数は，競合の起こり方に依存する．このことは競合の起こり方を制御すれば，各 PU のクロック周波数を変化させ，消費電力を最適化することが可能であることを意味する．

本稿では，各 PU の共有リソースへのアクセスに対

† 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

して優先度を導入し，優先度制御と Dynamic Voltage/Frequency Scaling (DVFS) 手法を組み合わせることにより，各 PU の負荷を調節し CMP 全体での消費電力を削減する手法を提案する．

本稿の構成は以下の通りである．次章では関連研究について述べ，3 章では優先度制御による消費電力削減とそのモデル化について説明し，4 章では実際の制御手法について述べる．5 章でその評価を行い，6 章で本稿のまとめと，今後の課題について述べる．

2. 関連研究

現在までに，DVFS 手法を用いた消費電力削減について多くの手法が研究・提案されている．石原ら[1] は，実時間制約下で，最適なクロック周波数・電源電圧を選択する手法について述べている．

電力最適化のための様々な手法が研究される中で，将来の負荷予測に対する関心が高まりつつある．性能制約下において，動的に DVFS 手法を適用するためには将来の負荷変動を正確に見積もる必要があるからである．Zhu らは[2]，タスクを複数に分割する PID フィードバック手法を提案し，周波数・電源電圧変更の際に停止させる必要のないプロセッサとの協調による利点を示している[3]．これらの分野における多くの研究は，画像・音響分野などの，信号処理に関するアプリケーションを対象としている．

本稿における提案手法が関連研究と異なる点は，DVFS 手法を単独で用いるのではなく，優先度制御との協調により消費電力効率を改善する点である．

3. 優先度制御の導入

3.1 優先度制御による電力削減

以下，2 個の同一 PU (PU_0, PU_1: PU_1) がメモリを共有しているシステムを例に取って，優先度制御による消費電力削減を説明する．なお，PU_1 上で実行されるプログラムの性能制約が PU_0 より厳しいと仮定する．

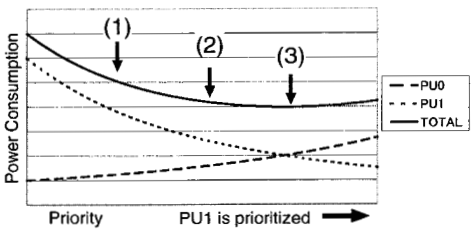

図 1　優先度の変化による各 PU の消費電力の遷移

図 1 は各 PU の優先度の変化と消費電力の遷移を示した概念図である．右に行くほど PU_1 の優先度が高くなるため，PU_1 の周波数・消費電力は減少する．逆に PU_0 の優先度は低くなるので，PU_0 の周波数・消費電力は増加する．ここで，PU_1 の性能制約の方が PU_0 のそれよりも厳しいため，優先度に対する消費電力の遷移曲線はより急峻になっている．

今，2 個の PU の優先度が等しい図中の (2) では，PU_1 の性能制約の厳しさから，PU_1 の消費電力の方が大きくなっている．ここで優先度を偏らせ，PU_0 を優先すると (1)，更に PU_1 の消費電力は増加し，CMP 全体の消費電力も増加する．

しかし PU_1 を優先させると (3)，PU_1 の消費電力は減少し，PU_0 における消費電力の増加は性能制約の緩さから小さく，CMP 全体での消費電力が減少する．すなわち，各 PU の優先度を偏らせることで，消費電力削減が可能となる．

3.2 優先度制御による性能・電力変化に関するモデリング

3.2.1 問題設定

優先度制御による電力最適条件を明らかにするため，まず優先度制御による各 PU の性能・電力変化に関するモデリングを行った．

本モデルでは，対象とする CMP は n 個の同一 PU から構成されており，各 PU は L1・L2 キャッシュを内蔵しているものとする．また各 PU は独立なクロック周波数 f_i・電源電圧 V_i で稼動する．ここで i は各 PU の番号を表している．各 PU は，全ての PU で共有してるメモリバスを通してメインメモリにアクセスするものとする．

PU_i はアプリケーションタスク T_i を実行する．前述したとおり，各タスクは異なる PU 上で独立に実行され，T_i はそれぞれ独立な実時間制約 L_i を課せられている．これは，T_i は時間 L_i 内で実行される必要があることを意味する．

3.2.2 変数設定

先述した変数，f_i・V_i・L_i に加えて，タスクと共有メモリバスに関する以下の変数を追加する．

- l_B: バス転送レイテンシ (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)
- I_i: T_i における命令数
- m_i: T_i における L2 ミス回数
- s_i: L2 ミスによる PU_i のストール時間

上記の仮定をふまえて，各 PU の実効稼働時間は以下の式で与えられる．

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

実効稼働時間は PU が演算を行う実際の時間を意味しており，PU_i は時間 t_i 内で I_i の命令を完了すればよい．PU_i の演算速度は，I_i と t_i に依存するデッドラインを満たす必要がある．また，電力最適化において，無限に選択可能な周波数・電源電圧があるとき，最適な周波数・電源電圧は単一に定まるという研究結果もある[1]．そこで最適なクロック周波数は比例定数 c を用いて以下の式で表すことができる．

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

また電源電圧とクロック周波数の関係は，比例定数 $a, b\ (>0)$ を用いて

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

のように表される．このような周波数・電源電圧の線形関係は多くの商用プロセッサで見られるものであり，妥当なものであると言える．

また命令あたりの PU_i の消費エネルギー e_i は電源電圧の二乗に比例するので[4]，k を比例定数とし次式のように表すことができる．

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

以上から，PU_i における平均消費電力は

$$P_i = \frac{I_i}{L_i}e_i = \frac{kI_iV_i^2}{L_i} \tag{5}$$

のように表される．

次に，競合が性能・電力に与える影響についてモデリングする．T_i を実行中に L2 ミスが発生した場合，PU_i は時間 m_il_b の間，メモリバスを占有する．ここで，L2 ミスの分布が一定だと仮定すると，PU_i がバスリクエストをしたときに競合が起きる確率は，他のいずれかの PU がメモリバスを占有している確率として

$$\begin{aligned} p_i = &\sum_{j\neq i}\frac{m_jl_B}{L_j} \\ &- \sum_{j,k\neq i;j<k}\frac{m_jl_B}{L_j}\frac{m_kl_B}{L_k} \\ &+ \sum_{j,k,l\neq i;j<k<l}\frac{m_jl_B}{L_j}\frac{m_kl_B}{L_k}\frac{m_ll_B}{L_l} \\ &- \cdots \\ &\pm\prod_{j\neq i}\frac{m_jl_B}{L_j} \end{aligned}$$

と与えられる．また競合が発生したときに，PU_i の転送が待たされる時間の期待値は

$$\begin{aligned} E_i = \frac{1}{p_i}\Bigg(&\sum_{j\neq i}\frac{m_jl_B}{L_j}\frac{l_B}{2} + \sum_{j,k\neq i;j<k}\frac{m_jl_B}{L_j}\frac{m_kl_B}{L_k}\frac{3l_B}{2} \\ &+\cdots+\prod_{j\neq i}\frac{m_jl_B}{L_j}\left(\frac{1}{2}+n-2\right)l_B\Bigg) \end{aligned}$$

と表される．以上より競合に伴う，各 PU の単位時間当たりの待ち時間の総和 l_{total} は以下の式で示される．

$$l_{total} = \sum_i \frac{m_i}{L_i}p_iE_i$$

ここで優先度制御をモデルに導入する．今，メモリバスにおいて優先度制御を行うと仮定する．競合が発生したとき，優先度制御器は複数の PU のバスリクエストの中から優先度に依存してリクエストを選択する．

この優先度制御では各 PU の待ち時間の比が，$\mathrm{PU}_0{:}\mathrm{PU}_1{:}\cdots{:}\mathrm{PU}_{n-1} = r_0 : r_1 : \cdots : r_{n-1}$ $(r_0 + r_1 + \cdots + r_{n-1} = 1)$ となるよう制御を行う．このとき各 PU の実効稼動時間は r_i の関数として以下のように表される．

$$t'_0 = t_0 - r_0l_{total}L_0 \tag{6}$$

$$t'_1 = t_1 - r_1l_{total}L_1 \tag{7}$$

$$\cdots$$

$$t'_{n-2} = t_{n-2} - r_{n-2}l_{total}L_{n-2} \tag{8}$$

$$t'_{n-1} = t_{n-1} - (1 - r_0 - \cdots - r_{n-2}) \times l_{total}L_{n-1} \tag{9}$$

優先度制御後の各 PU の周波数・電源電圧・消費エネルギー・消費電力は，(2)〜(5)，(6)〜(9) 式より以下のように表される．

$$f'_i = c\frac{I_i}{t_i - r_il_{total}L_i}\ (0 \le i \le n-2) \tag{10}$$

$$f'_{n-1} = c\frac{I_{n-1}}{t_{n-1} - (1 - \cdots - r_{n-2})l_{total}L_{n-1}} \tag{11}$$

$$V'_i = af'_i + b \tag{12}$$

$$e'_i = kV'^2_i \tag{13}$$

$$P_{total} = \sum_i \frac{I_i}{L_i}e'_i \tag{14}$$

また e'_i は (12)，(13) 式より以下のようにも表される．

$$e'_i = c_2f'^2_i + c_1f'_i + c_0\ (c_2, c_1, c_0 > 0) \tag{15}$$

次に消費電力最小となる最適点の解析について述べる．(12) 式より，V'^2_i は下に凸な f'_i の関数である．また V'^2_i に比例する e'_i，そして P_{total} も下に凸な f'_i の関数である．

ここで P_{total} の，f'_i についての偏微分を考えると，(10)〜(15) 式より

$$\begin{aligned} \frac{\partial}{\partial f'_i}P_{total} = &\frac{2c_2I_i}{L_if'^3_i}(f'^3_i - f'^3_{n-1}) \\ &+ \frac{c_1I_L}{L_if'^2_i}(f'^2_i - f'^2_{n-1}) \end{aligned}$$

となる．

P_{total} は f'_i に関して下に凸な関数なので

$$\begin{cases} f'^3_i - f'^3_{n-1} = 0 \\ f'^2_i - f'^2_{n-1} = 0 \end{cases}$$

を満たす

$$f'_i = f_{n-1}$$

となる点で P_{total} は最小値を取る．

以下同様に全ての i について同様に P_{total} の偏微分を取ることで，共通の消費電力最小条件，

$$f_0 = f_1 = \cdots = f_{n-1} \tag{16}$$

を得る．また $f_0 = f_1 = \cdots = f_{n-1}$ を満たすような点，$r_{min} = (r_0, r_1, \cdots, r_{n-2})$ は P_{total} の大域的な電力最小点を与える．この r_{min} は，(10)，(11)，(16) 式を連立させて解くことにより求めることができる．

まとめると全 PU の周波数が等しくなるよう優先度を制御したとき，CMP 全体の消費電力は最小値を取

るということが言える．

4. 制御手法

本章では，3章のモデリングから得られた電力最適点を実現するための，優先度と周波数に関する制御手法について述べる．

なおここでは，2個の同一PU (以下PU_0，PU_1) を持つCMPにおいて，L2キャッシュまでは各PUに内蔵しメモリバスを共有，各PUは独立に性能制約を持つアプリケーションタスクを実行することを仮定して制御手法を考える．

4.1 概　　略

本稿では2つの制御手法を提案する．図2，3に，その2つの制御手法のブロック図を示す．以下，それぞれについて説明する．

4.1.1 制御手法 1

図2の左側にある優先度制御器は，3章のモデルから求められた電力最適点を実現するr_iの値を使用し，競合を制御する．右側にあるDVFS制御器は，各PUが性能制約を満たすように周波数・電源電圧を選択する．性能制約の厳しさは優先度制御の結果に依存して決定される．

4.1.2 制御手法 2

モデリングから得られた帰結である，「消費電力最小となるのは各PUの周波数が等しいとき」を実現する制御手法である．電力最適点を実現するrの値は求めず，優先度制御器では各PUの周波数を入力とし，これが等しくなるように制御を行う．

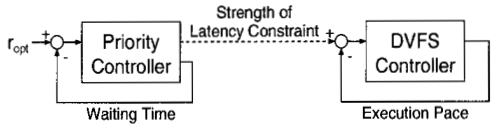

図2　制御手法 1

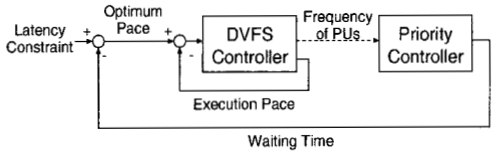

図3　制御手法 2

4.2 優先度制御

本稿では優先度制御をメインメモリに対するアクセスキュー (図4) において実現する．アクセスキューは共有リソースへのアクセスを制御する技術として一般的なハードウェアである．全てのPUからの，L2ミスによるメモリアクセスリクエストはキューに蓄積され，1個ずつ発行される．

ここで優先度N (Nは整数) を導入する．このNの値を使用しリクエストのアクセス順序を制御する．例えばNの値が正であるならば，キューの先頭N個までにあるPU_0からのリクエストをPU_1からのリクエストに先行してアクセスさせる．Nの値が負である場合は，逆にキューの先頭$|N|$個までにあるPU_1からのリクエストをPU_0からのリクエストよりも先にアクセスさせる．このアルゴリズムを図5に示す．

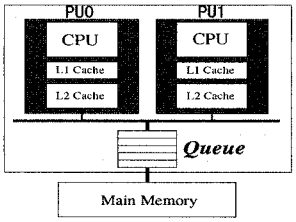

図4　メモリアクセスキュー

提案手法では待ち時間の比がモデルと等しくなるよう制御 (制御手法1)，または各PUの周波数が等しくなるよう制御 (制御手法2) するものであるが，待ち時間の値を直接制御はできないため，フィードバック制御機構を設けている．

制御手法1では (図6)，モデルから求めた最適点r_{min}と一定周期毎に観測した各PUの待ち時間W_iを使用する．$W_0/W_1 > r_{min}/(1-r_{min})$となる場合は$PU_0$が待たされすぎということであるから，$N$を1上げる．逆に$W_0/W_1 < r_{min}/(1-r_{min})$となる場合は$PU_1$が待たされすぎということであるから，$N$を1下げる．

制御手法2では (図7)，一定周期毎に観測した各PUの周波数f_iのみを使用する．制御手法1と同様に，$f_0 > f_1$となる場合はPU_0が待たされすぎということであるから，Nを1上げる．逆に$f_0 < f_1$となる場合はPU_1が待たされすぎということであるから，Nを1下げる．

```
if ( N > 0 ) {
   allow requests from PU0 to pass
   up to N requests from PU1
} else if ( N < 0 ) {
   allow requests from PU1 to pass
   up to |N| requests from PU0
} else{
   select a request at the head of the queue
}
```

図5　キューにおけるリクエスト選択アルゴリズム

4.3 DVFS制御

実際のCMPにおいては，仮定したCMPのモデルと異なる点がある．例えば，実際のCMPでは選択可能な周波数・電源電圧の段階は限られているので，モデルから求めた電力最適点rを実現できない可能性がある．そのため本稿で提案する手法では，フィードバック制御を用いたDVFS手法を付加している．

```
Parameter
  THp: threshold for changing the priority
  Nmax : the maximum value of N
Counters
  Ci : the number of requests from PUi in the queue
  Wi : the length of the waiting time due to the conflict
Register
  R : the index of PU which uses the bus at this moment
Counting Routine (called every bus cycle)
  for ( i = 0 ; i ≤ 1 ; i++ ) {
    /* Is PUi waiting due to access from any other PUs? */
    if ( Ci > 0 and R ≠ i ) Wi++ ;
  }
Feedback Routine (called at regular intervals)
  if ( (1 − rmin)W0 > rmin W1 × THp ) {
    if ( N < Nmax ) N++ ;  /* PU0 waits too much */
  } else if ((1 − rmin)W0 × THp < rmin W1)
    if ( |N| < Nmax ) N−− ;  /* PU1 waits too much */
  }
```

図 6 手法 1 におけるフィードバック制御

```
Parameter
  THp: threshold for changing the priority
  Nmax : the maximum value of N
Counters
  Ci : the number of requests from PUi in the queue
  fi : the average frequency of PUi
Register
  R : the index of PU which uses the bus at this moment
Counting Routine (called every bus cycle)
  for ( i = 0 ; i ≤ 1 ; i++ ) {
    /* Is PUi waiting due to access from any other PUs? */
    if ( Ci > 0 and R ≠ i ) Wi++ ;
  }
Feedback Routine (called at regular intervals)
  if ( f0 > f1 × THp ) {
    if ( N < Nmax ) N++ ;  /* PU0 waits too much */
  } else if (f0 × THp < f1)
    if ( |N| < Nmax ) N−− ;  /* PU1 waits too much */
  }
```

図 7 手法 2 におけるフィードバック制御

この DVFS 手法はある一定周期で各タスクの演算速度 (命令数/経過時間) を観測する．現在の演算速度と，デッドラインまでに全タスクを完了するのに必要な速度を比較し，周波数段階を上下させるかどうかを決定する．

5. 評　　価

この章では 3 章におけるモデルの評価と，4 章で述べた制御手法の実アプリケーションによる評価について述べる．

評価には Simple Scalar Tool Set[5] を，CMP が評価できるよう拡張したものを使用した．また電力の見積りには Wattch[6] を使用した．評価に用いたプログラムは，Alpha AXP ISA 用の DEC C コンパイラを用いてコンパイルしている．

表 1 に設定したパラメタの一覧を示す．クロック周波数と電源電圧の関係は，Pentium M における線形予想を使用している[7]．

表 1 ハードウェアパラメタ

PU 数	2
PU クロック周波数	200〜1600 MHz
PU 電源電圧	0.721〜1.502 V
Memory latency	100ns
Bus width	8B
Bus clock	200MHz
Priority feedback interval	1.87us
DVFS feedback interval	187us

5.1 モデルの評価

3 章で述べたモデルを評価するために，L2 ミス率が一定 (PU_0・PU_1 共に 1.76%) となるよう作成した synthetic プログラムを用いて評価を行った．この synthetic プログラムにおける L2 ミス分布はポアソン分布に従う．また選択可能な周波数・電源電圧段階は，モデルの仮定を満たすために非常に細かい設定が可能となっている．

図 8 は制御手法は用いず，各 PU の優先度種々の値に固定して synthetic プログラムを実行したときの消費電力の推移である．

モデルから求めた r_{min} の値は 0.837 であり，図 8 における消費電力最小点における r_{min} の値とほぼ一致している．このことから，3 章で述べたモデルは妥当性のあるものだと言える．

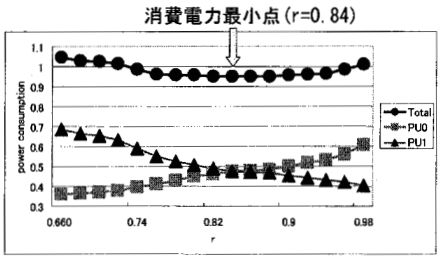

図 8 優先度と消費電力の関係

5.2 実アプリケーションによる評価

手法を適用した際の電力削減効果を調べるために，実アプリケーションにより評価した．評価で使用したプログラムは，H.264/AVC デコーダ[8] と SPEC CPU2000 ベンチマーク[9] から選択したプログラム (mcf, art, amp) である．実アプリケーションの挙動は一定ではなく，L2 ミス率も時間と共に変化する．そこでモデルの仮定を満たすために，プログラムをミス率一定と見なせるフェーズに分割し，そのフェーズ内において一定周期で制御を行った．また制御手法 1 では，各 PU 間のタスクの組み合わせが変更される毎に，r の値を再計算している．また PU_0 上で H.264 を，PU_1 上で SPEC から選択したプログラムを実行している．

評価結果を図 9〜12 に示す．図 9 は，手法適用なし，手法 1 適用，手法 2 適用の場合における消費電力の比較である．手法 1 または 2 を適用した場合，適用なしと比較して平均で 11% の消費電力削減を達成できているのがわかる．ここで mcf だけ他の 2 つのベンチマークとして比較して削減率が低いのは，mcf は L2 ミス率の変動が大きく，フェーズ分割数も多いため，優先度・DVFS 制御のためのオーバヘッド電力が増大しているためである．

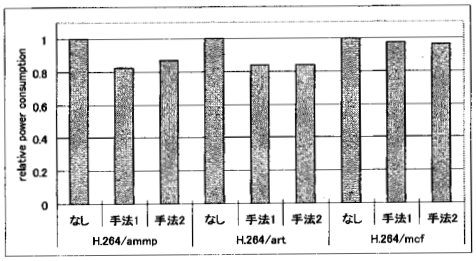

図 9 評価結果 (消費電力)

次に mcf において，200ms 毎の各 PU における周波数遷移を観測したものを図 10〜12 に示す．モデルの帰結は「消費電力最小となるのは全 PU の周波数が等しいとき」であったが，手法 1・2 共にこの帰結を満たすように，各 PU の周波数が他の PU の周波数を追随できているのが図 11，12 から読み取れる．

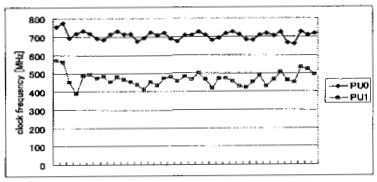

図 10 手法なしにおける各 PU の平均周波数の遷移

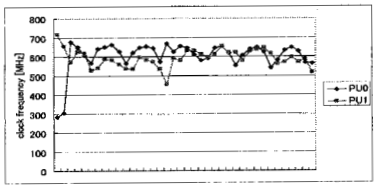

図 11 手法 1 を適用した場合の各 PU の平均周波数の遷移

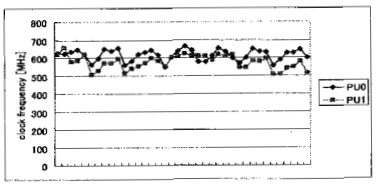

図 12 手法 2 を適用した場合の各 PU の平均周波数の遷移

6. まとめと今後の課題

本稿では，CMP におけるプロセッサコアの，共有リソースへのアクセスに優先度制御を導入し，DVFS 制御と協調することにより CMP 全体の消費電力を削減する手法を提案した．また優先度制御が性能・消費電力に与える影響をモデリングし，電力最適となる条件を導出した．提案手法を評価した結果，手法非適用と比較して，平均で 11%の消費電力削減を達成できた．

今後の課題としては，PU 数が 2 より大きい場合 ($n > 2$) の優先度制御アルゴリズムの考案が挙げられる．5 章の評価で用いたアルゴリズムは，2PU の 2 個の待ち時間の単純比較により優先度を決定してるが，$n > 2$ の場合は観測された n 個の待ち時間から各 PU の優先度を決定するため，今回用いた単純比較を適用することはできない．

今後は $n > 2$ の場合における優先度制御アルゴリズムを考案し，提案手法の $n > 2$ における効果について評価・検討していく予定である．

謝辞 本研究の一部は，(株) 半導体理工学研究センター，および科学技術振興機構・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」の支援によって行われた．

参 考 文 献

1) T. Ishihara, and H. Yasuura, "Voltage scheduling problem for dynamically variable voltage processors," In *ISLPED*, pages 197-202, 1998.
2) Y. Zhu and F. Mueller, "Feedback edf scheduling exploiting dynamic voltage scaling," In *IEEE Real-Time and Embedded Technology and Applications Symposium*, pages 84-93, 2004.
3) Y. Zhu and F. Mueller, "Feedback edf scheduling exploiting hardware-assisted asynchronous dynamic voltage scaling," *In LCTES*, pages 203-212, 2005.
4) A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Lowpower cmos digital design," IEEE J. of Solid-State Circuits, Apr. 1992.
5) T. M. Austin, E. Larson, and D. Ernst, "Simplescalar: An infrastructure for computer system modeling," IEEE Computer, 35(2):59-67, 2002.
6) D. Brooks, V. Tiwari, and M. Martonosi, "Wattch: a framework for architectural-level power analysis and optimizations," In ISCA, pages 83-94, 2000.
7) K. Krewell. Pentium m hits the street. Microprocessor Report, Mar. 2003.
8) H.264/AVC Software Coordination. H.264/AVC reference software. http://iphome.hhi.de/suehring/tml.
9) Standard Performance Evaluation Corporation (SPEC). SPEC CPU2000. http://www.specbench.org.
10) R. Watanabe, M. Kondo, H. Nakamura, and T. Nanya, "Power Reduction using Shared Resource Control Cooperating with DVFS," In ICCD, 2007.