主記憶アクセスの負荷情報を利用した動的周波数変更による低消費電力化

近藤正章[†,††] 中村宏[††]

近年，プロセッサの消費電力削減要求の高まりをうけ，システムのタスク処理要求の負荷などに応じて動的にプロセッサのクロック周波数と電源電圧を調節する動的電源電圧変更手法が注目されている．本論文では，既存の動的電源電圧変更手法の拡張として，演算処理と主記憶アクセスの負荷バランスに基づき，プロセッサチップの電源電圧・クロック周波数を最適化し低消費電力化を狙うマイクロプロセッサアーキテクチャを提案する．また，サイクルレベルシミュレーションによりその有効性の評価を行う．評価結果より，提案する動的電源電圧変更手法を用いることで，浮動小数点ベンチマークでは 7%の性能低下で 48%の消費エネルギーを，整数ベンチマークでは 4%の性能低下で 19%の消費エネルギーを削減できることが分かった．

Dynamic Processor Throttling for Low Power Computing

Masaaki Kondo[†,††] and Hiroshi Nakamura[††]

This paper describes a method of dynamically throttling processor speed using dynamic voltage scaling (DVS) technique for low power computing. We propose a micro-architecture level mechanism that detects performance imbalance between processor and memory and adjusts processor frequency to redress the imbalance. By the adjustment, the mechanism selects an near optimal processor speed among several voltage/frequency setting points based on the performance imbalance and thereby power consumption is reduced. We evaluate power saving and performance degradation for our proposed method by cycle level simulation. The evaluation results reveal that the method can reduce energy consumption by 48% and 19% with 7% and 4% performance degradation for floating-point and integer benchmarks respectively.

1. はじめに

近年，マイクロプロセッサの低消費電力化・低消費エネルギー化はプロセッサの設計における最も重要な課題となってきている．モバイル計算機のバッテリ駆動時間の延長という要求はもちろんのこと，商用サーバ，さらには科学技術計算用途などのハイエンドなプロセッサにおいても，放熱の問題から消費電力削減は必要不可欠な課題となっている．プロセッサ設計の際には，放熱面での消費電力の上限を定めた熱設計消費電力（*Thermal Design Power*: *TDP*）に基づいて設計する必要があるが，最近ではこの TDP がプロセッサの性能を制限する第 1 要因となることも珍しくない．TDP を満たすために電源電圧を低くすると，プロセッサのクロック周波数を低下せざるをえないためである．したがって，モバイル計算機から性能を重視するハイエンドのシステムに至るまで，あらゆる計算機システムにおいて低消費電力マイクロプロセッサの構成方式や，そのための回路技術が重要となっている．

一方，高性能および低消費電力という両要求を満たすために，動的電源電圧変更（*Dynamic Voltage Scaling*: *DVS*）機能を持つプロセッサの研究が現在広く行われており，商用プロセッサにおいても Intel Pentium M [1] などで採用されている *SpeedStep*，Transmeta Crusoe TM5800 [2] の *LongRun* など，DVS 機能を備えたプロセッサが数多く登場している．DVS は電源駆動/バッテリ駆動の別，あるいはシステムのタスク処理要求の負荷に応じて，動的にプロセッサのクロック周波数と電源電圧を調節する手法である．CMOS 半導体のスイッチングに起因する消費電力は電源電圧の 2 乗に比例するため，プロセッサを低電圧で動作させることで，大きな消費電力削減効果が期待される．

† 独立行政法人科学技術振興機構
Japan Science and Technology Agency
†† 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

表 1 Intel Pentium M プロセッサのクロック周波数と電源電圧の関係
Table 1 The relation between supply voltage and clock frequency on Intel Pentium M.

Processor Clock	1.6 GHz	1.4 GHz	1.2 GHz	1.0 GHz	800 MHz	600 MHz
FSB Clock	400 MHz	400 MHz	400 MHz	400 MHz	400 MHz	400 MHz
Memory Bus Clock	266 MHz	266 MHz	266 MHz	266 MHz	266 MHz	266 MHz
Processor Core Vdd	1.484 V	1.420 V	1.276 V	1.164 V	1.036 V	0.956 V
(%Energy)	(100%)	(92%)	(74%)	(62%)	(49%)	(41%)

従来の DVS 手法は，マルチタスク環境下におけるプロセスの負荷の監視や，組み込み分野などで必要とされる実時間処理におけるデッドラインスケジューリングをもとに電源電圧の調節を行うもの[3)~5)]がほとんどであった．近年ではプロセッサと主記憶の性能格差が非常に大きく，キャッシュミスが頻発するようなアプリケーションでは，プロセッサは多くの時間を主記憶からのデータ転送待ちに費やしている．そこで，1つのアプリケーション実行中に，プロセッサの演算処理と主記憶アクセスによるデータ転送の負荷を監視し，データ転送の負荷が大きい場合にはプロセッサチップの電源電圧・クロック周波数を下げることで，性能に影響を与えずに消費電力を削減できると考えられる．本論文では，より汎用のアプリケーションを対象に，既存の DVS 手法を拡張しプロセッサ・主記憶のチップ間の処理バランスに基づいて動的に電源電圧・クロック周波数を最適化するマイクロプロセッサアーキテクチャ手法 *Dynamic Processor Throttling*（*DPT*）を提案する．

本論文の構成は以下のとおりである．次章において DVS 手法の概要を示し，3 章で提案する DPT 手法について述べる．4 章では性能評価環境，および評価条件について説明し，5 章で評価結果を示す．6 章で関連研究を述べ，7 章でまとめと今後の課題について述べる．

2. 動的電源電圧変更手法

動的電源電圧変更（*Dynamic Voltage Scaling*: *DVS*）手法は，電源駆動/バッテリ駆動の別，あるいはプロセッサのタスク処理要求の負荷などに応じて，動的にプロセッサのクロック周波数と電源電圧を調節する手法である．バッテリ駆動時間を長くしたい，あるいは行うべきタスクが少なくプロセッサのアイドル状態が長いような場合には，プロセッサチップの電源電圧を下げ消費電力削減を狙う．

この DVS 手法による消費電力および性能への影響は，以下のように定式化することができる．まず，CMOS 半導体のスイッチングに起因する消費電力 P は次式で表される．

$$P \propto C \times Vdd^2 \times f \quad (1)$$

ここで，C は CMOS の負荷容量，Vdd は電源電圧，f はクロック周波数である．また，CMOS 半導体回路の遅延時間 D は一般的に次式で表すことができる[6)]．

$$D \propto \frac{Vdd}{(V_G - V_T)^\alpha} \quad (2)$$

ここで，V_G はゲート電圧，V_T は閾値電圧である．α はトランジスタ中のキャリアの速度飽和を示す値で典型的には 1〜2 の値をとる．式 (1) に示すように，消費電力は電源電圧の 2 乗に比例するため，電源電圧を下げることで大きな消費電力削減が期待できる．しかし，式 (2) より，電源電圧を下げると回路の遅延時間が増加してしまうため，正確な動作を保証するためには同時にクロック周波数を下げる必要がある．このように CMOS 回路では電源電圧の変更にともない，消費電力と性能の間にトレードオフが存在する．

DVS による電源電圧とクロック周波数の関係を示す例として，Intel Pentium M プロセッサ（1.6 GHz 版）において設定可能なクロック周波数と，それに対応する電源電圧[1)]の関係を**表 1** に示す．また表 1 は，最高クロックで動作した場合に，あるアプリケーション実行に必要な消費エネルギーを 100%とした場合の，各電源電圧における消費エネルギーの割合（%Energy）も示している．表より，実際にクロック周波数を下げることによって，消費エネルギーを大きく削減できることが分かる．

3. Dynamic Processor Throttling: DPT

3.1 概　　要

本論文では，マルチタスク環境下におけるプロセス負荷の監視や，実時間処理におけるデッドラインスケジューリングをもとに電源電圧の調節を行う既存の DVS 手法を拡張し，演算処理と主記憶アクセスの負荷のバランスに基づいて電源電圧・クロック周波数を動的に調節するマイクロアーキテクチャ手法である *Dynamic Processor Throttling*（*DPT*）手法を提案する．近年，プロセッサと主記憶間の性能格差が深刻化しており，キャッシュミスが頻発するようなアプリケーションでは，プロセッサは多くの時間を主記憶か

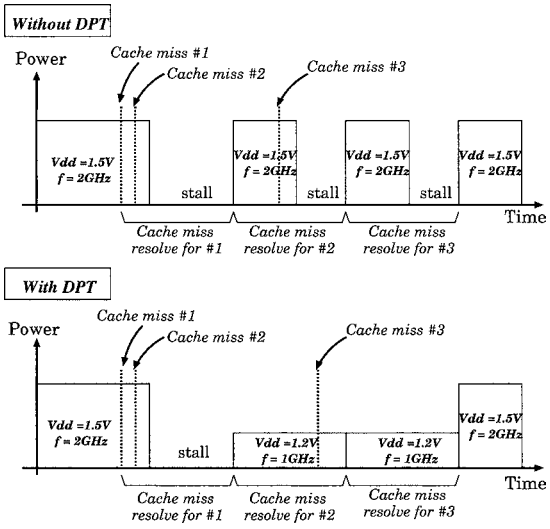

図 1 DVS の概要
Fig. 1 Illustrative example of proposed DPT method.

らのデータ転送待ちに費やしている．そこで，低電圧化によるクロック周波数低下に起因する性能のペナルティを，そのデータ転送待ち時間により隠蔽することで，性能低下を最小限に抑えつつ消費電力を削減することができると考えられる．

提案する DPT の概要を述べるため，**図 1** に通常のプロセッサ（Without DPT）と，DPT 手法を用いた場合（With DPT）の両者について，キャッシュミスが生じた際のプロセッサの消費電力の変化の様子を示す．ここではキャッシュミスが生じた場合でもプロセッサの実行を続けることができるノンブロッキングキャッシュを仮定している．また，主記憶からのデータ転送（cache miss resolve）は，同時には 1 つのリクエストしか実行できないモデルを仮定している．

通常のプロセッサ（Without DPT）では，キャッシュミスによるデータ待ちのストール時間が多く，効率的な実行ではない．一方，DPT 手法により（With DPT），主記憶間とのデータ転送処理の負荷が高い場合に，プロセッサの電源電圧・クロック周波数を下げることでストール時間が減少し，また消費電力を削減することができるため，効率的な実行が行える．ここで，図のように適切なクロック周波数を選択することができれば性能低下は生じない．したがって，図の動作が本論文で提案する DPT 手法の意図する動作である．

この DPT を実現するためには，1) 演算処理と主記憶アクセスの負荷バランスを監視し，2) 将来の負荷バランス予測することで，3) 電源電圧・クロック周波数の調整を行う必要がある．次節より，1)〜3) の各項目について以下のメモリ階層を持つプロセッサを前提として述べる．

- L1 データ/命令キャッシュ，L2 統合キャッシュを持つ．
- 全キャッシュをチップ内に搭載（L2 キャッシュはプロセッサと同一電源電圧で動作）．
- ノンブロッキングキャッシュを仮定．

3.2 負荷の監視

演算処理と主記憶アクセスの負荷バランスを考える場合，主記憶アクセスはキャッシュミスにより生じることから，キャッシュミス情報を用いることで負荷バランスを見積もることができると考えられる．前提となるノンブロッキングキャッシュでは，キャッシュミス解決のためのデータ転送中でもプロセッサの実行が継続して行われるため，転送中に再び新たなキャッシュミスが生じる可能性がある．つまり，ある時点では複数のキャッシュミスが存在することも少なくない．ここで，一般的にキャッシュ・主記憶間のデータ転送は，同時には 1 つのリクエストしか処理できないため，キャッシュミスによるデータ転送要求が積み重なるとプロセッサ（演算部）はストールする可能性が高い，すなわち演算処理に対してデータ転送要求の負荷が高いことになる．

そこで，“同時に存在するキャッシュミスの数” の情報をもとにした負荷の監視手法を検討する．本論文では，*MSHR*（*Miss State Holding Register*）[7] ベース，およびカウンタベースの 2 つの手法を提案する．なお，前提とするメモリ階層では，主記憶アクセスは L2 キャッシュミスにより生じるため，L2 キャッシュミス情報に基づき負荷を監視する．

3.2.1 MSHR ベース

MSHR を用いたノンブロッキングキャッシュでは，MSHR は未解決のキャッシュミス要求を保持するキューとして使われる[7]．キャッシュミスが生じた際に，そのミス情報のために 1 エントリが割り当てられ，そのキャッシュミスが解決するとエントリは解放される．したがって，この MSHR のエントリ数を監視することで，ある時点で存在するキャッシュミス数を知ることができる．

MSHR ベースの監視手法では，主記憶アクセスの負荷を，ある期間における MSHR の中の平均エントリ数（$AvNum_{MSHR}$）として定義する．たとえば，ある期間につねに 1 つだけキャッシュミス要求が存在する場合には $AvNum_{MSHR}$ は 1.0 となる．したがって，1.0 を超える場合は平均して 1 つ以上のキャッシュミスが存在していたことになり，同時に 1 つのキャッシュミス要求のデータ転送しかできないメモリシステ

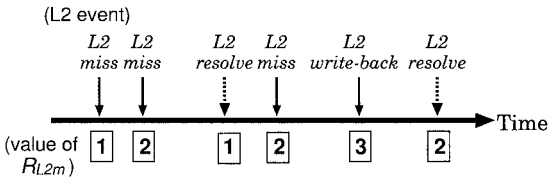

図 2 L2 キャッシュミス要求数の監視
Fig. 2 Monitoring the number of L2 cache misses.

ムでは，主記憶アクセスの負荷が大きいと考えられる．このようにして，MSHR ベースの負荷の監視手法では $AvNum_{MSHR}$ を用いて負荷の監視を行う．

3.2.2 カウンタベース

前述の MSHR ベースの負荷監視手法は直感的な手法であるが，毎サイクル何個の MSHR エントリが使用されているか，またその値のある期間における平均を求める必要があるなど，ハードウェアが複雑化する恐れがある．DPT 手法の目的は低消費電力化であるため，複雑なハードウェア機構は避けるべきである．また，MSHR ベースの手法は MSHR というハードウェア，すなわちノンブロッキングキャッシュの実装に依存している．そこで，次に MSHR を用いずに比較的簡単なハードウェア機構で実現可能と思われる別の負荷監視手法を検討する．

まず，ある時点で存在するキャッシュミス要求（ライトバック要求を含む）の数を記録するための状態レジスタ R_{L2m} を導入する．**図 2** に R_{L2m} の動作を示す．R_{L2m} は，L2 キャッシュミス（*L2 miss*），あるいは L2 キャッシュからのライトバック（*L2write-back*）が発生した際に 1 が加算され，それぞれの要求が終了した時点で 1 減算される．

次に，ある期間における負荷の状況を判断するために，上記のレジスタ R_{L2m} の値に応じて毎サイクルカウントアップを行う 3 つのカウンタ $Cnt0$，$Cnt1$，$Cnt2$ を導入する．それぞれ，サイクルごとの R_{L2m} の値が 0，1，2 以上の各場合にカウントアップを行う．この 3 つのカウンタを参照することで，ある期間において同時に存在したキャッシュミスの数の分布を知ることができる．

そして 3 つのカウンタを用い，ある一定期間ごとにカウンタの値に対して次式を適用することで負荷の大きさ $Load$ を求める．

$$Load = (Cnt2 \times w_2) + (Cnt1 \times w_1) + (Cnt0 \times w_0) \quad (3)$$

ここで，w_n はそれぞれのカウンタに対する重みを表している．主記憶アクセスの負荷は，この $Load$ の値の大小により判定する．

```
for each cycle {
   Sum += Num_miss_in_MSHR;

   /* for every T_itvl */
   if ((CycleCount % T_itvl) == 0){
      AvNum_MSHR = Sum / T_itvl;
      if (AvNum_MSHR > Th_u &&
                          C_lev > MinClockLev)
         DownClockLev(C_lev);
      elseif (AvNum_MSHR < Th_l &&
                          C_lev < MaxClockLev)
         UpClockLev(C_lev);
      else
         /* UnchangingClockLevel */;

      Sum = 0;
   }
   CycleCount++;
}
```

図 3 DPT1 のアルゴリズム（MSHR ベース）
Fig. 3 Algorithm of DPT1 (MSHR-based).

3.3 負荷の予測

未来の一定期間における主記憶アクセスの負荷を予測するにあたり，現在の負荷をもとに予測を行うことを考える．一定期間の長さを T_{itvl} サイクルとすると，現在までの T_{itvl} 期間中の主記憶アクセスの負荷が高ければ次の T_{itvl} も負荷が高いと予測し，逆に現在までの期間中の負荷が低ければ次の期間も負荷が低いと予測する．現在までの期間の負荷の見積りは，前述の $AvNum_{MSHR}$ あるいは $Load$ の値を用いて行う．

なお，$AvNum_{MSHR}$ および，3 つのカウンタの値は T_{itvl} ごとにリセットする．

3.4 電源電圧・クロック周波数の変更

予測された負荷の値から，次の T_{itvl} 期間の電源電圧・クロック周波数を以下のように変更する．得られた $AvNum_{MSHR}$ あるいは $Load$ の値に対し，上限の閾値 Th_u と下限の閾値 Th_l を設け，それが Th_u 以上であった場合，すなわちデータ転送の負荷が上限の閾値を超えた場合はプロセッサの電源電圧・クロック周波数を 1 レベル下げる．逆に，Th_l 以下であった場合，すなわちデータ転送の負荷が下限の閾値を下回った場合は，電源電圧・クロック周波数を 1 レベル上げる．それ以外であった場合は変更は行わない．

3.5 アルゴリム全体の流れ

本論文では以降，MSHR ベースの監視手法を用いた DPT 手法を DPT1，カウンタベースの監視手法を用いた DPT 手法を DPT2 と呼ぶ．前節までの DPT 手法のまとめとして，DPT1 および DPT2 のアルゴ

```
for each cycle {
   if (R_L2m == 0) Cnt0++;
   elseif (R_L2m == 1) Cnt1++;
   else Cnt2++;

   /* for every T_itvl */
   if ((CycleCount % T_itvl) == 0){
      Load = (Cnt2 × w_2) + (Cnt1 × w_1)
                          +(Cnt0 × w_0)
      if (Load > Th_u && C_lev > MinClockLev)
         DownClockLev(C_lev);
      elseif (Load < Th_l && C_lev < MaxClockLev)
         UpClockLev(C_lev);
      else
         /* UnchangingClockLevel */;

      Cnt0 = Cnt1 = Cnt2 = 0;
   }
   CycleCount++;
}
```

図 4 DPT2 のアルゴリズム（カウンタベース）
Fig. 4 Algorithm of DPT2 (counter-based).

リズムをそれぞれ**図 3** および**図 4** に示す．

4. 評　　価

4.1 評価環境

本論文で提案する DPT 手法による消費電力削減の効果と性能への影響を調べるため，SimpleScalar Tool Set [8] を用いたサイクルレベルシミュレーションにより評価を行う．本評価では，主記憶アクセスの負荷など，メモリ階層の振舞いが重要であるため，SimpleScalar に対し メモリ階層を正確にシミュレーションするための拡張がなされた “SimpleScalar with Memory Extention” [9]，および消費電力を評価するための拡張がなされた “Wattch” [10] の両拡張を統合したものを用いる．さらに，DPT 手法を評価するための拡張も加える．

評価に用いるプログラムは，SPEC CPU2000 の整数および浮動小数点ベンチマークから，C および Fortran77 で書かれたプログラム☆と，ベクトルの内積を計算するカーネル（Vector）を用いる．コンパイラは，SimpleScalar で用いられている ISA の 1 つである PISA 用のコードを生成する gcc を用い，コンパイラオプションは “-O2” とした．なお，Fortran77 のコードは `f2c` を用いて C 言語のプログラムに 1 回変換した後，gcc によりコンパイルする．

☆ SPEC CPU2000 中のいくつかのベンチマークでは，コンパイルエラーにより評価できなかった．

表 2 評価における仮定
Table 2 Processor configuration.

Fetch Width	4
Branch Prediction	bimodal 2 Ktable
BTB	512sets, 4way
Mis-Prediction penalty	3cycles
RUU size	64
LSQ size	32
L1 I-Cache	32 KB, 32 B line, 2way 1 cycle latency
L1 D-Cache	32 KB, 32 B line, 4way 1 cycle latency
L2 Cache	128 KB, 64 B line, 4way 10 cycle latency
Bus width	8 B

表 3 DPT アルゴリズムにおけるパラメータ
Table 3 Parameters for DPT algorithm.

Parameter	DPT1	DPT2
T_{itvl}	10000 cycle	10000 cycle
Th_u	1.2	10000
Th_l	0.8	6000
w	—	$w_2 = 2, w_1 = 1, w_0 = -1$

4.2 評価の仮定

表 2 に評価におけるプロセッサの仮定を示す．なお，クロック周波数と電源電圧の仮定については，表 1 に示した Intel Pentium M プロセッサにおけるプロセッサとバスのクロック，および電源電圧にならい，6 通りのクロックレベルに変更可能なものとする．ここで，主記憶アクセスの際のレーテンシは，50 ns（1.6 GHz の場合 80 プロセッササイクル）として評価を行う．また，電源電圧・クロック周波数を変更する際にかかる時間的なオーバヘッドは，無視できるものとして評価を行う．なお，このオーバヘッドに関しては 5.2 節において考察する．また提案する DPT 手法を実現するために必要なハードウェアで消費される電力なども無視できるものとして評価を行う．

DPT のアルゴリズム中で用いられる閾値などのパラメータに関しては，**表 3** に示す値を用いる．なお，評価では T_{itvl} と閾値を変化させた場合の評価も行う．ここで，DPT2 アルゴリズムで用いられる重みの値について，主記憶アクセスがなく演算部の負荷が高い場合にカウントアップされる $Cnt0$ の重みの値 w_0 を “-1” としている．これは，主記憶アクセスがない場合は，演算部ができるだけ高いクロック周波数で動作することが望ましく，負の値の重みを用いることで $Load$ を低く見積もるようにするためである．

このような条件のもと，提案する DPT1 と DPT2 における性能と消費エネルギーについて，つねに一定

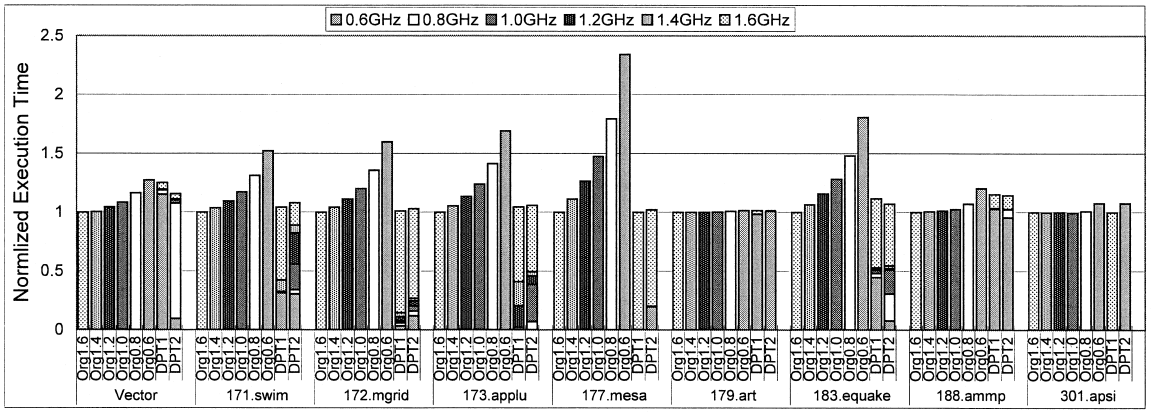

図 5 実行時間（浮動小数点プログラム）
Fig. 5 Execution time (floating-point programs).

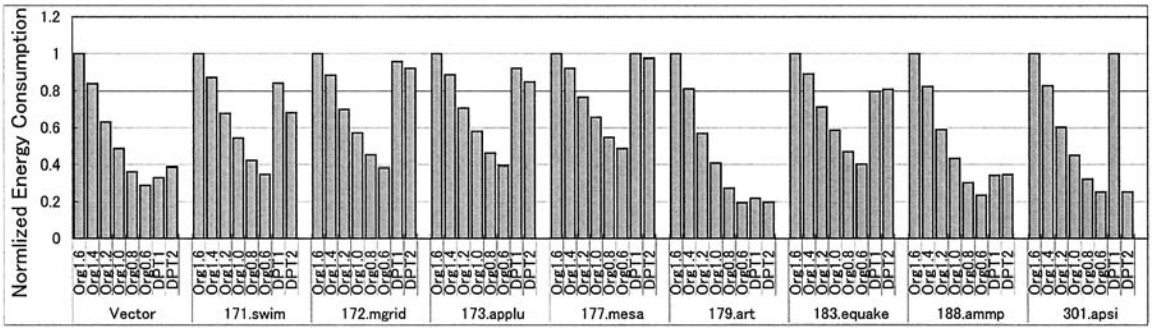

図 6 消費エネルギー（浮動小数点プログラム）
Fig. 6 Energy consumption (floating-point programs).

の周波数で動作する通常のプロセッサと比較評価を行う．なお，評価では DPT 手法による消費エネルギー削減効果と性能への影響を評価することを目的とし，本論文では熱設計消費電力の観点からの評価は行わない．

5. 評価結果

5.1 実行時間および消費エネルギー

本論文で提案する DPT 手法が性能へ及ぼす影響と消費エネルギー削減の効果を見るため，**図 5** および **図 6** に浮動小数点プログラムの実行時間と消費エネルギーを示す．また，**図 7** および **図 8** に整数プログラムの実行時間と消費エネルギーを示す．

図中，“Org” は DPT を行わずつねに一定の周波数で動作する通常のプロセッサを表し，後に続く数字がその周波数（GHz）を表す．また，DPT1 と DPT2 はそれぞれ 3.5 節で述べた 2 種類の DPT アルゴリズムを示している．すべての図では，各アプリケーションの “Org1.6” を基準とした場合の相対的な値を示している．なお各棒グラフは，それぞれのクロック周波数で動作していた時間の内訳も示している．

実行時間

まず，実行時間の評価結果について議論する．図 5 や図 7 において，つねに一定の周波数で動作する Org どうしを比べると，アプリケーションによって，周波数が低くなると性能が大きく低下するもの（173.applu や 177.mesa，175.vpr，300.twolf など）とほとんど性能が変化しないもの（179.art や 188.ammp，301.apsi，181.mcf など）とに分けられる．前者は演算バウンドなアプリケーションであり，主記憶アクセスの負荷が低く演算部の処理能力が実行時間を支配するため，プロセッサの周波数を下げると性能が低下する．一方，後者はメモリバウンドなアプリケーションであり，プログラムの実行を通して主記憶アクセスの負荷が高く，実行時間がメモリシステムの能力に支配されるため，プロセッサの周波数を下げても性能に大きく影響を与えない．

次に DPT 手法を用いた場合の実行時間を見ると，DPT1 と DPT2 ではアプリケーションに応じて様々なクロック周波数で動作していることが分かる．また，

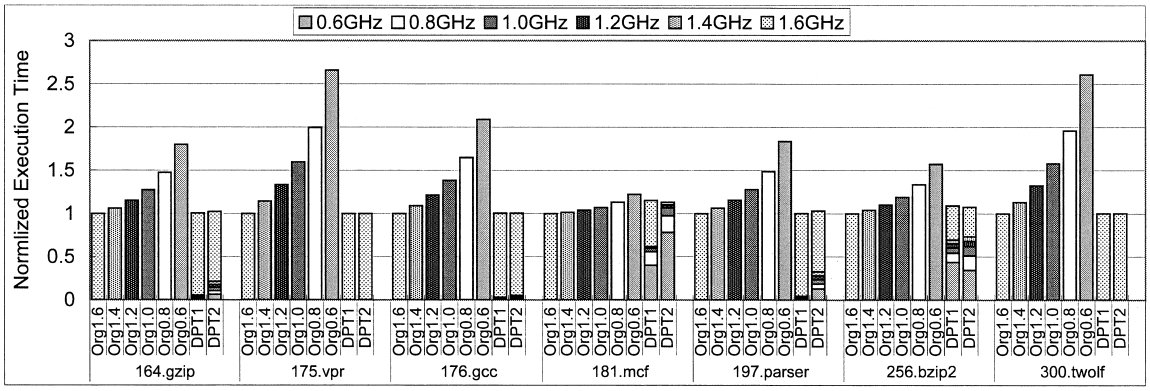

図 7 実行時間（整数プログラム）
Fig. 7 Execution time (integer programs).

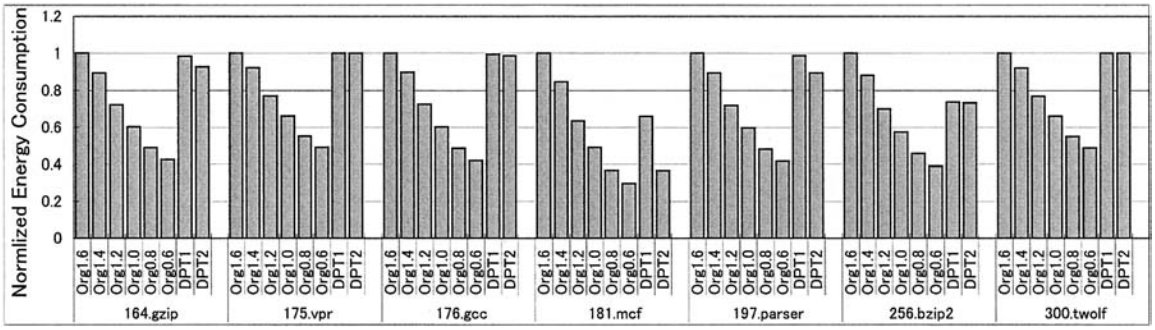

図 8 消費エネルギー（整数プログラム）
Fig. 8 Energy consumption (integer programs).

DPT 手法では周波数を下げて動作している部分があるにもかかわらず，すべてのアプリケーションで大きな性能低下は見られないことが分かる．DPT 手法では演算部と主記憶アクセスの負荷バランスに応じて動的に周波数を変更するため，性能低下をできる限り抑えつつ周波数を低下させることが可能なためである．

また，演算バウンドなアプリケーションでも，DPT 手法では低い周波数で動作できる領域を予測して性能にあまり影響を与えずに周波数を下げることができている（171.swim や 183.equake，256.bzip2 など）．これは，1 つのプログラム実行中，つねに演算バウンドなわけではなく，途中にメモリバウンドな部分があることを示しており，その場合に動的に主記憶アクセスの負荷バランスに基づいて周波数を調整する DPT 手法が有効であることを示している．

しかし，DPT 手法においても，Org1.6 と比べると若干性能が低下してしまい，特に Vector と 188.ammp，また 177.mcf では性能低下率が大きく，DPT1 ではそれぞれ 25%，15%，15%，DPT2 では 16%，14%，13%ほど実行時間が増加している．これらのアプリケーションでは，今回のアルゴリズム，およびそれに用いた閾値などのパラメータのもとでは，ある時点での最適なクロック周波数を選択することができなかったためである．また，188.ammp では最低クロックで動作している（主記憶アクセスの負荷が高い）状態から，瞬間的に高いプロセッサの処理能力が必要になることが多く，その瞬間的な負荷バランスの変化に対し，提案する DPT 手法では対応できなかったのも原因である．一方，その他のアプリケーションでは性能低下は数%程度とそれほど大きくなく，提案するアルゴリズムによって，性能に大きく影響を及ぼさない範囲でクロック周波数を下げることができている．

ここで，DPT1 と DPT2 を比較した場合，アプリケーション依存ではあるが，DPT2 の方がより低い周波数で動作する時間が長くなっている．特に，301.apsi の結果を見ると，DPT1 ではほとんどの時間 1.6 GHz で動作しているが，DPT2 では 600 MHz で動作しており，顕著に異なる結果を示している．これはライトバック要求の扱い方の違いによるものである．MSHR にはライトバック要求の情報は保持されないため，DPT1

ではライトバックによるデータ転送の負荷が考慮されていないが，DPT2 ではライトバック要求も含めて R_{L2m} でカウントすることで，プロセッサ・メモリ間のデータ転送の負荷を正確に見積もることができるためである．

消費エネルギー

次に，消費エネルギー削減効果について議論する．図 6 および図 8 において Org どうしの消費エネルギーをを比較すると，周波数が低くなるに従い，消費エネルギーも大きく削減される．2 章で述べたように，消費電力は電源電圧の 2 乗に比例するため，低いクロック周波数を用い低電圧で動作させることで，大きな消費エネルギー削減効果が得られるためである．

Org1.6 と DPT 手法を比較した場合，DPT を用いることですべてのプログラムで消費エネルギーが削減されていることが分かる．また，低クロック周波数で動作する時間が長いプログラムほど，消費エネルギーの削減率は大きい．178.art や DPT2 における 301.apsi ではほとんどの実行時間を 600 MHz の最低周波数で動作しているため，それぞれ 80%，75%と多くの消費エネルギーが削減されている．また，その他のアプリケーションでも，消費エネルギーが 20%から 60%と大きく削減されているものが多い．

つねに一定の周波数で動作する Org では，低い周波数を採用することで多くの消費エネルギーを削減することができるが，性能が大きく低下してしまうアプリケーションが多い．特に，175.vpr や 300.twolf では DPT 手法を用いても 1.6 GHz でずっと動作しており消費エネルギーはまったく削減されていないが，このようなアプリケーションで周波数を下げると，図 7 からも分かるとおり実行時間が非常に長くなってしまう．本論文で提案する DPT 手法の目的は，最高周波数で動作させた場合（Org1.6）に比べ，性能低下を最小限に抑えつつ最大限の消費エネルギー削減効果を得ることであり，この点において提案する DPT 手法は非常に有効であると考えられる．

5.2 パラメータによる影響

前節の評価より，DPT1 を用いることで浮動小数点ベンチマークでは平均 7%の性能低下で 37%の消費エネルギーを，整数ベンチマークでは 4%の性能低下で 10%の消費エネルギーを削減できることが分かった．また DPT2 の場合，浮動小数点ベンチマークでは平均 7%の性能低下で 48%の消費エネルギーを，整数ベンチマークでは 4%の性能低下で 19%の消費エネルギーを削減できることが分かった．本節では，DPT のアルゴリズムで用いられる閾値と，電源電圧・クロック周波数を調整する間隔である T_{itvl} のパラメータにおいて，異なるパラメータの値が実行時間や消費エネルギーにどのような影響を及ぼすかについて評価を行う．なお，DPT1 手法に比べて DPT2 手法の有効性が高かったため，DPT2 手法のみを評価する．

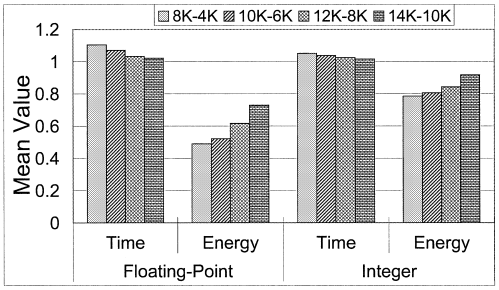

図 9 DPT2 において閾値（Th_u-Th_l）を変化させた場合
Fig. 9 Varying threshold (Th_u-Th_l) in DPT2.

図 9 は，DPT2 において，閾値 Th_u，および Th_l を変化させた場合の実行時間と消費エネルギーについて，“Org1.6” を基準とした場合の相対的な値を示している．なお，すべての浮動小数点プログラムおよびすべての整数プログラムにおける幾何平均の値を示している．また，図中の凡例は “上限の閾値 Th_u–下限限の閾値 Th_l” を示しており，たとえば 10 K-6 K は Th_u が 10000，Th_l が 6000 の場合を示している．その他の仮定は 4.2 節で述べたとおりである．

図 9 より，閾値の値を小さくすることで，実行時間は長くなり，消費エネルギー削減率は大きくなる傾向にあることが分かる．これは，小さい閾値を用いると低いクロック周波数で動作しやすくなるためである．また，整数プログラムに比べて浮動小数点プログラムの方が閾値の値による変化が顕著であることが分かる．浮動小数点プログラムに比べ整数プログラムでは主記憶アクセスの負荷が低いものが多く，そのような負荷の低いアプリケーションではこの範囲の閾値のとり方には依存せずに最高クロック周波数で動作してしまうためである．

DVS 手法では一般的に，性能と消費エネルギーの間にトレードオフが存在するが，DPT 手法においてはこの閾値をどのように選択するかにより，トレードオフのポイントを調整することができる．したがって，許容される性能低下の範囲内で最大限に消費エネルギーを削減できるような閾値の値を選ぶことが重要となる．

次に，**図 10** に T_{itvl} の値を変化させた場合の評価結果を示す．図 9 と同様に，“Org1.6” を基準とした場合の相対値について，すべての浮動小数点プログラム

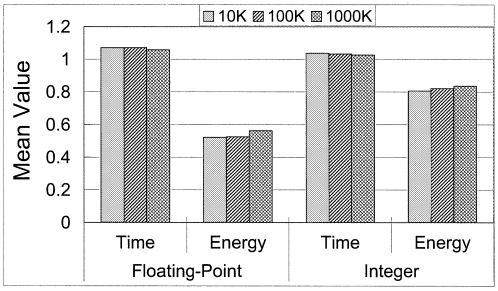

図 10 DPT2 において T_{itvl} を変化させた場合
Fig. 10 Varying time interval (T_{itvl}) in DPT2.

表 4 各 T_{itvl} における閾値
Table 4 Threshold values for each time interval.

T_{itvl}	Th_u	Th_l
10 K (10000 cycle)	10000	6000
100 K (100000 cycle)	100000	60000
1000 K (1000000 cycle)	1000000	600000

およびすべての整数プログラムにおける幾何平均を示している．図中の凡例が T_{itvl} の値を表している（たとえば 10 K は T_{itvl} が 10000 となる）．また，T_{itvl} の値に応じて，**表 4** のように閾値も変化させている．

図 10 を見ると，T_{itvl} を変化させても実行時間や消費エネルギーにあまり影響がないことが分かる．これは，今回評価に用いたベンチマークの多くのプログラムで，短い期間で負荷バランスが大きく変化するアプリケーションが少なかったためと思われる．

一般的には，短い時間間隔を用いて電源電圧とクロック周波数を調整する方が細粒度で最適化を行うことができるため望ましい．しかし，電源電圧・クロック周波数変更の際に性能と消費エネルギーに無視できないペナルティがある場合には，細粒度で調整を行うとペナルティが大きくなってしまう．このように，時間間隔の設定には細粒度な最適化による利点と，性能・消費エネルギーのペナルティとの間にトレードオフがある．本論文ではここまで，このペナルティを無視して評価を行ってきた．

ここで，Intel Xscale プロセッサでは周波数の切替えに要する時間は 20 μ 秒以下とされている[11]．そこで，1 回の電源電圧/クロック周波数切替えに 20μ 秒かかると仮定した場合の，各 T_{itvl} における実行時間の増加率（ベンチマークの平均値）を**表 5** に示す．表 5 から，T_{itvl} が 10 K の場合，実際には DPT 手法では大きな時間的オーバヘッドがかかることが分かる．しかし，T_{itvl} が 100 K，あるいは 1000 K となるとオーバヘッドは非常に小さくなり，実行時間にはほとんど影響を与えない．図 10 の結果から，多くのアプリケーションでは T_{itvl} を 100 K 以上に長くしても DPT 手法の有効性は変わらないため，T_{itvl} をある程度長くすることで，時間的ペナルティの影響は抑えられると考えられる．

表 5 周波数切替えによる時間的オーバヘッド
Table 5 Time overhead for changing clock frequency.

	Floating-Point	Integer
10 K	13%	22%
100 K	1%	3%
1000 K	0.5%	0.2%

6. 関連研究

本研究のベースとなる動的電源電圧変更手法は，いくつかの商用プロセッサにも採用され，また多くの研究も行われている．

文献 12) は，本研究と同じ視点からマイクロアーキテクチャレベルの DVS 手法を提案しており，すべてのキャッシュミスをトリガとして，キャッシュミスが解決されるまでの間，低消費電力モードに移行する手法を検討している．しかし，この研究では演算処理と主記憶アクセスの負荷のバランスについてあまり考慮されていないほか，2 段階の周波数を持つプロセッサを対象としており，本研究のように複数の電源電圧・クロック周波数レベルから最適なものを選択することは考えられていない．

文献 13) はプロセッサチップ内を複数のクロック周波数領域に分割し，各領域の負荷のバランスに基づき，それぞれのクロック周波数を最適化するプロセッサを提案している．しかしながら，主記憶間とのデータ転送の負荷バランスについては考慮されていない．

また，コンパイラによる静的な解析やプロファイル情報により，クロック周波数を下げても性能に影響しないプログラム中のコード領域を抽出し，各コード領域のクロック周波数を静的に最適化する研究も提案されている[14),15)]．しかし，これらの研究のようにコンパイル時のプログラムの解析情報に基づく DVS 手法では，プログラムの振舞いを完全に予測するのは難しく，たとえばデータセットが変わった場合にはキャッシュミスの頻度も変わりうるため，コンパイル時に決定したクロック周波数が最適でなくなってしまうことも考えられる．

本研究は，演算処理と主記憶アクセスの負荷バランスをハードウェアにより検出し，動的に電源電圧・クロック周波数を最適化することで，従来の計算機システムに比べ飛躍的に消費電力の削減を狙う点で新規性が高いと考えられる．

7. まとめと今後の課題

本論文では，既存の DVS 手法の拡張として，プロセッサ・主記憶間の処理の負荷バランスに基づき，プロセッサチップの電源電圧・クロック周波数を動的に調節するマイクロプロセッサアーキテクチャ*Dynamic Processor Throttling: DPT* を提案した．近年のプロセッサ・主記憶間の性能格差を背景に，キャッシュミスが生じるとプロセッサは多くの時間を主記憶からのデータ転送待ちに費やしている．そこで本手法では，主記憶アクセスの負荷が高い場合にはプロセッサチップの電源電圧・クロック周波数を下げることで，性能へのペナルティを最小限に抑えつつ消費電力削減を狙う．

クロックレベルシミュレーションによる評価では，提案する手法によりつねに最高クロック周波数で動作する通常のプロセッサに比べ，浮動小数点ベンチマークでは 7% の性能低下で 48%の消費エネルギーを，整数ベンチマークでは 4%の性能低下で 19% の消費エネルギーを削減できることが分かった．今後，提案するアルゴリズムにおける最適な閾値などのパラメータ設定法などについて検討するほか，性能低下を抑えつつさらに消費電力を削減できる手法についても検討していく予定である．また，今回は消費エネルギーの観点からのみ評価を行ったが，放熱の問題も含め，消費電力の観点から評価を行うことも今後の課題である．

謝辞 本研究の一部は，文部科学省科学研究費補助金（基盤研究（B）No. 14380136）によるものである．

参 考 文 献

1) Krewell, K.: Pentium M Hits the Street, *MICROPROCESSOR REPORT*, Vol.17, No.3 (March 2003).
2) Transmeta: *Crusoe Processor Product Brief Model TM5800* (2003).
3) Qu, G.: What is the Limit of Energy Saving by Dynamic Voltage Scaling, *Proc. 2001 Intl. Conf. on Computer Aided Design*, pp.560–563 (2001).
4) Gruian, F.: Hard Real-Time Scheduling for Low-Energy Using Stochastic Data and DVS Processors, *Proc. 2001 Intl. Sym. on Low Power Electronics and Design*, pp.46–51 (2001).
5) Shin, D. and Kim, J.: A Profile-Based Energy-Efficient Intra-Task Voltage Scheduling Algorithm for Hard Real-Time Applications, *Proc. 2001 Intl. Sym. on Low Power Electronics and Design*, pp.271–274 (2001).
6) 石原 亨，安浦寛人：可変電圧プロセッサを用いた省エネルギー化のための基本定理，電子情報通信学会技術研究報告，ICD98-46 FTS98-46, Vol.98, No.68, pp.69–76 (1998).
7) Farkas, K.I. and Jouppi, N.P.: Complexity/Performance Tradeoffs with Non-Blocking Loads, *Proc. 21st Intl. Sym. on Computer Architecture*, pp.211–222 (1994).
8) Austin, T., Larson, E. and Ernst, D.: SimpleScalar: An Infrastructure for Computer System Modeling, *IEEE Computer*, Vol.35, No.2, pp.59–67 (2002).
9) Burger, D., Kagi, A. and Hrishikesh, M.S: Memory Hierarchy Extensions to the SimpleScalar Tool Set, Technical Report TR99-25, Department of Computer Science, University of Texas at Austin (1999).
10) Brooks, D., Tiwari, V. and Martonoshi, M.: Wattch: A Framework for Architectural-Level Power Analysis and Optimizations, *Proc. 27th Intl. Sym. on Computer Architecture*, pp.83–94 (2000).
11) Mudge, T.: Power: A First-Class Architectural Design Constraint, *IEEE Computer*, Vol.34, No.4, pp.52–58 (2001).
12) Marculescu, D.: On the Use of Microarchitecture-Driven Dynamic Voltage Scaling, *Workshop on Complexity-Effective Design in conjunction with the 27th Intl. Sym. on Computer Architecture* (2000).
13) Semeraro, G., et al.: Dynamic Frequency and Voltage Control for a Multiple Clock Domain Microarchitecture, *Proc. 35th Intl. Sym. on Microarchitecture*, pp.356–367 (2002).
14) Hsu, C.-H., Kremer, U. and Hsiao, M.: Compiler-Directed Dynamic Voltage/Frequency Scheduling for Energy Reduction in Microprocessors, *Proc. 2001 Intl. Sym. on Low Power Electronics and Design*, pp.275–278 (2001).
15) Saputra, H., et al.: Energy-Conscious Compilation Based on Voltage Scaling, *Proc. Joint Conf. on Languages, Copilers and Tools for Embedded Systems: Software and Compilers for Embedded Systems*, pp.2–11 (2002).

(平成 15 年 10 月 10 日受付)
(平成 16 年 2 月 2 日採録)

近藤 正章（正会員）

平成 10 年筑波大学第三学群情報学類卒業．平成 12 年同大学大学院工学研究科博士前期課程修了．平成 15 年東京大学大学院工学系研究科先端学際工学専攻修了．工学博士．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 研究員を経て，現在東京大学先端科学技術研究センター特任助手．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 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 年より東京大学先端科学技術研究センター助教授．この間，1996 年〜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・低消費電力プロセッサのアーキテクチャ，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，ディジタルシステムの設計支援の研究に従事．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．