2003－ARC－154　(17)
2003／8／6

演算部とデータ供給部の動的周波数変更による低消費電力化手法の検討

近藤正章[†,††] 藤田元信[††,†] 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．本稿では，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なう．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，わずかに性能が低下するものの，大きな消費エネルギー削減効果が得られることがわかった．

Dynamic Processor Throttling for Low Power Computing

Masaaki KONDO,[††,†] Motonobu FUJITA[††]
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 optimal processor speed among several voltage/frequency setting points 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 has potential to reduce a large portion of energy consumption with negligible performance loss.

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一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得ないためである．従って，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．

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，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繁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

† 科学技術振興事業団 JST
†† 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

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

送の負荷を監視し，データ転送の負荷が大きい場合には DVS 手法により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．本稿では，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を提案する．

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，あるいは行な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．

$$P \propto C \times Vdd^2 \times f \tag{1}$$

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[3]．

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

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 (1.6GHz 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わかる．

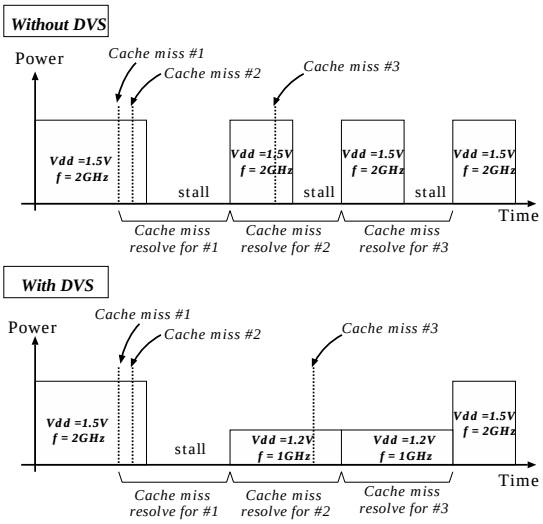

図 1 DVS の概要

3. 主記憶アクセスの負荷に基づく DVS 手法

3.1 概　　要

本稿では，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プロセッサを提案する．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繁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，DVS を用いた場合の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 DVS 手法の概要を述べるため，図 1 に通常のプロセッサ (Without DVS) と，電源電圧とクロック周波数を調節した場合 (With DVS) 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，同時には一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 DVS) 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，DVS 手法を用い (With DVS)，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，図の動作が本稿で提案する DVS 手法の意図する動作である．

次節では，本手法を実現するためのアルゴリズムとハードウェア構成について述べる．

3.2 アルゴリズムとハードウェア構成

演算処理と主記憶アクセスの負荷のバランスに基づく DVS 手法を検討するにあたり，以下のメモリ階層を持つプロセッサを前提とする．

- 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

提案する DVS 手法を実現するためには，演算処理と主記憶アクセスの負荷バランスを予測し，最適な電源電圧・クロック周波数を決定する必要がある．ノンブロッキングキャッシュの場合，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なわれるため，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．したがって，ある時点では複数のキャッシュミスが存在してい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，同時には一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，本稿では “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 DVS 手法を検討する．なお，前提とするメモリ階層においては，主記憶アクセスは L2 キャッシュミスにより生じるため，以降では L2 キャッシュミス情報に基づいた DVS 手法を検討する．

以下，負荷の監視，および負荷の予測，また電源電圧・クロック周波数の調節法について述べる．

3.2.1 負荷の監視

まず，本 DVS 手法を実現するにあたり，L2 キャッシュにおける “同時に存在するキャッシュミスの数” を監視する必要がある．そこで，ある時点で存在する

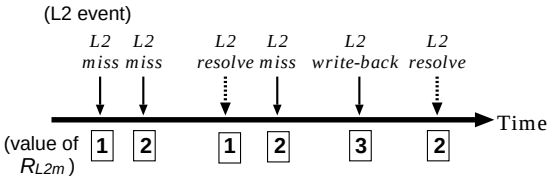

図 2 L2 キャッシュミス要求数の監視

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 (*L2 write-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なう 3 つのカウンタ $Cnt0$，$Cnt1$，$Cnt2$ を導入する．それぞれ，サイクル毎の R_{L2m} の値が 0，1，2 以上の各場合にカウントアップを行なう．この 3 つのカウンタを参照することで，ある期間の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．

3.2.2 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なうことを考える．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．また，現在までの期間の負荷の見積もりは，前述のカウンタを用いて行なう．概念的には，$Cnt0$ の数が大きければその期間の負荷は低く，$Cnt2$ の数が大きければその期間の負荷は高いと予測する．

本稿では，実際の負荷の予測は，ある一定期間 (T_{itvl}) 毎にカウンタの値に対して，次式を用いて負荷の大きさ $Load$ を求めることにより行なう．

$$Load = (Cnt2 \times 2) + Cnt1 - Cnt0 \qquad (3)$$

なお，カウンタの値は T_{itvl} 毎にリセットする．

3.2.3 電源電圧・クロック周波数の変更

式 (3) で得られた $Load$ 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 $Load$ の値に対し，上限の閾値 Th_u と下限の閾値 Th_l を設け，$Load$ 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越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，$Load$ が 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，変更は行なわない．

以上のアルゴリズムをまとめたものを図 3 に示す．

```
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 × 2) + Cnt1 − Cnt0;
        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 3 提案する DVS 手法のアルゴリズム

4. 評　　価

4.1 評価環境

本稿で提案する DVS 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[4] 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なう．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[5]”，および消費電力を評価するための拡張がなされた “Wattch[6]” の両拡張を統合したものを用いる．さらに，DVS 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，ベクトルの内積を計算するカーネル (Vector)，および SPEC CPU2000 の浮動小数点ベンチマークから、C および Fortran77 で書かれたベンチマークを用いる．コンパイラは，SimpleScalar で用いられている ISA の一つである 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一回変換した後，gcc によりコンパイルする．

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．また，主記憶アクセスの際のレーテンシは，50ns (1.6GHz の場合 80 プロセッササイクル) として評価を行なう．

表 2 評価における仮定

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

また，電源電圧・クロック周波数を変更する際の性能へのオーバヘッド，また提案する DVS 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は無視できるものとして評価を行なう．

このような条件のもと，提案する DVS 手法による性能と消費電力について，常に最高周波数で動作する通常のプロセッサと比較評価を行なう．

5. 評 価 結 果

5.1 実 行 時 間

まず，3 節で述べた DVS 手法が性能へ及ぼす影響を見るため，図 4 にベクトル内積計算 (Vector)，および SPEC CPU2000 の浮動小数点ベンチマーク☆の各プログラムの実行時間を示す．図 4 は，DVS を行なわない通常のプロセッサ (“normal”) を基準とした場合の，DVS 手法を適用したプロセッサ (“DVS”) の相対的な実行時間を示している．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．なお，今回は 3.2 節のアルゴリズムにおける T_{itvl} の値は 10000 cycle に，Th_u および Th_l の値はそれぞれ 10000，6000 に設定して評価を行った結果を示している．

図 4 より，DVS 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わかる．また，DVS 手法により周波数を下げて動作しているにも関わらず，下げた割合ほどは性能が低下していないことがわかる．これは，実際に主記憶アクセスの負荷が高い場合にはプロセッサのストール時間が長いため，プロセッサの周波数を下げても性能に大きく影響しないことを示している．

しかし，通常のプロセッサである normal と比べると，DVS ではすべてのアプリケーションにおいて若干の性能低下が見られる．特に Vector と 188.ammp では性能低下率が大きく，それぞれ 16%および 14%ほ

☆ SPEC CPU2000 浮動小数点ベンチマークの 168.wupwise, および 200.sixtrack は，コンパイルエラーにより評価できなかった．

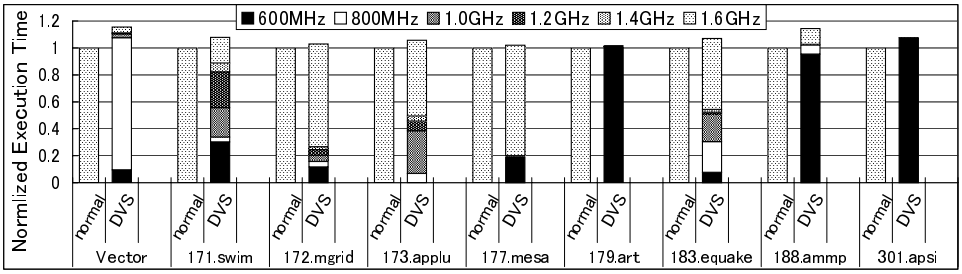

図 4 実行時間

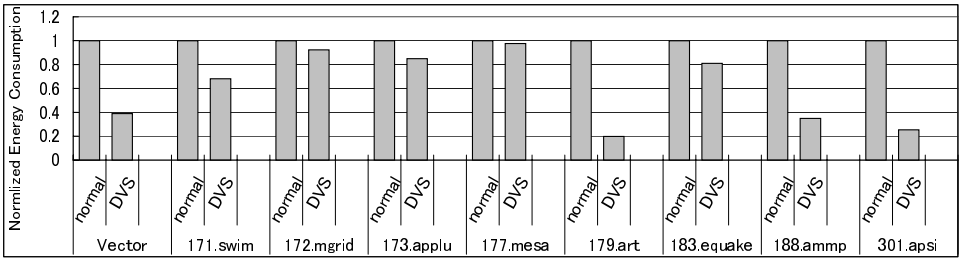

図 5 消費エネルギー

ど実行時間が増加している．これは，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，提案するアルゴリズムでは対応できなかったのが原因である．

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．178.art および 301.apsi は，ほとんどの実行時間を 600MHz の最低クロック周波数で動作しているにもかかわらず，性能低下はそれぞれ 1%，8%程度である．これは，両アプリケーションは非常に主記憶アクセスの負荷が高く，高クロック周波数で動作しても，プロセッサは多くの時間ストールしてしまっていることを表している．このようなアプリケーションでは，プロセッサを低クロック周波数で動作させることで，大きな消費エネルギー削減が期待される．次節では，提案する DVS 手法による消費エネルギー削減効果について議論する．

5.2 消費エネルギー

図 5 に，各ベンチマークプログラムにおける normal を基準とした場合の DVS の相対的な消費エネルギーを示す．normal と DVS を比較すると，DVS により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わかる．また，低クロック周波数で長く動作しているプログラムほど，消費エネルギーの削減率は大きい．前述のように 178.art および 301.apsi はほとんどの実行時間を 600MHz の最低周波数で動作しているため (図 4 参照)，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，このことから，提案する DVS 手法により，プロセッサの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プロセッサチップの電源電圧・クロック最適化することで，大きな消費エネルギー削減効果が得られることがわかる．

5.3 消費エネルギー・遅延積

性能と消費エネルギーの両者を統合して評価するために，図 6 に相対的な消費エネルギー・遅延積 (以下 ED 積と呼ぶ) の値を示す．図より，すべてのプログラムにおいて normal に比べ DVS では ED 積が改善されており，最大で 80%，平均でも 44%程度，ED 積の値が削減されている．本 DVS 手法は性能面でのペナルティーが多少あるものの，大きく消費エネルギーが削減できるため，ED 積で評価した場合にも通常のプロセッサに対し優れていることがわかる．

以上より，提案する DVS 手法はプロセッサの消費電力削減という要求に対し，非常に有効な手段となり得ると結論付けることができる．

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なわれている．

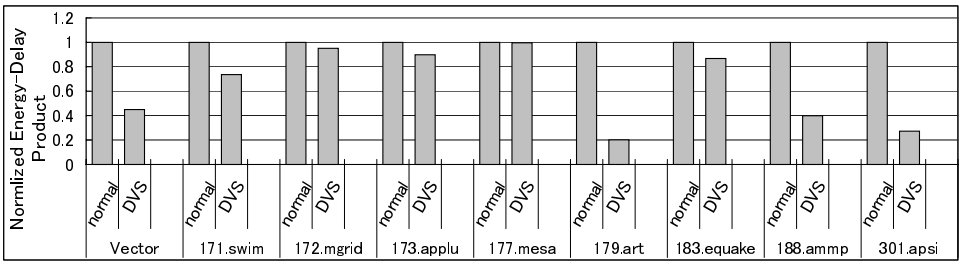

図 6 消費エネルギー・遅延積

文献[7] 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ーとして，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[8] 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[9),10)]．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，プログラムの振る舞いを完全に予測するのは難しく，例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得るため，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

本稿では，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常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，性能低下がわずかに生じるものの，大きく消費エネルギー削減できることがわかった．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他，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．

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) K. Krewell, “Pentium M Hits the Street”, Microprocessor Report, Vol.17, No.3, March 2003.
2) Transmeta, “Crusoe Processor Product Brief Model TM5800”, Feb. 2003.
3) 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”, 信学技報 ICD98-46 FTS98-46, Vol. 98, No. 68, pp.69–76, May 1998.
4) T. Austin, et al., “SimpleScalar: An Infrastructure for Computer System Modeling”, IEEE Computer, Vol. 35, No. 2, pp.59–67, Feb. 2002.
5) D. Burger, et al., “Memory Hierarchy Extensions to the SimpleScalar Tool Set”, Technical Report TR99-25, Department of Computer Science, University of Texas at Austin, April 1999.
6) D. Brooks, et al. , “Wattch: A Framework for Architectural-Level Power Analysis and Optimizations”, In Proc. 27th ISCA, pp.83–94, June 2000.
7) D. Marculescu, “On the Use of Microarchitecture-Driven Dynamic Voltage Scaling”, Workshop on Complexity-Effective Design in conjunction with the 27th ISCA, June 2000.
8) G. Semeraro, et al. “Dynamic Frequency and Voltage Control for a Multiple Clock Domain Microarchitecture”, In Proc. MICOR-35, pp.356–367, Dec. 2002.
9) C.-H. Hsu, et al., “Compiler-Directed Dynamic Frequency and Voltage Scheduling”, Workshop on PACS2000, Nov. 2000.
10) H. Saputra, et al., “Energy-Conscious Compilation Based on Voltage Scaling”, In Proc. LCTES’02-SCOPES’02, pp.2–11, June 2002.