# 低電力JAVAプロセッサのための投機的クロック制御

木村篤彦[†1] 中島康彦[†2] 宮田佳昭[†3]
中川伸二[†3] 北村俊明[†4] 五島正裕[†5]
森眞一郎[†5] 富田眞治[†5]

本稿では，オムロン株式会社が開発したJavaプロセッサJeRTyをモデルとし，演算ユニット単位のクロック投入制御を行った場合の電力削減効果と性能とのトレードオフについて述べる．クロック分配系統のより上流においてクロックを制御するほど，電力削減効果は高くなる一方，デコードサイクルに続く演算サイクルが動作可能となるまでのペナルティは増加するため性能は低下する．SPECJVM98を用いて，演算ユニットごとにクロックを投機的に投入するための予測表の構成について検討した結果，エントリ数が等しい予測表の中では，メソッド識別子の下位2ビットおよびPCの下位6ビットを用いた256エントリの表，命令に関わるデータ型の識別子3ビットを用いた8エントリの表が，それぞれ高い効果を発揮することを示す．また，8ビットの情報を用いた場合，予測による性能向上が消費電力増加を上回るものの，3ビットの情報では逆転することを明らかにする．

# A Speculative Clock Control for Low Power Java Processors

Atsuhiko Kimura,[†1] Yasuhiko Nakashima,[†2] Yoshiaki Miyata,[†3]
Shinji Nakagawa,[†3] Toshiaki Kitamura,[†4] Masahiro Goshima,[†5]
Shinichiro Mori[†5] and Shinji Tomita[†5]

This paper describes a power saving technique for a Java processor so called JeRTy developed by OMRON Corporation. We assume several execution units and related clock-trees are individually driven from higher-level gated clock distributors. The program counter or the instruction decoder predicts the associated units and speculatively starts and stops these distributors. The performance degradation derived from the clock delay and the unnecessary power consumption caused by miss predictions are tradeoffs. We evaluate these speculative methods on SPECJVM98 benchmark programs and found a prediction table that holds 256-entries indexed by the lower 2-bits of the method-ID and the lower 6-bits of the program-counter, or a smaller table that holds only 8-entries indexed by 3-bits each corresponds the data types of the stack-top show remarkable effectiveness respectively. Finally we show the 8-bit information gains higher performance than power consumption, though the 3-bit information dissipates the power.

## 1. はじめに

大量の電力を投入して実行速度を向上させてきたプロセッサ開発競争は，低電力化という新たな方向へ進路を変えつつある．理由の一つは，情報機器の小型化および可搬化にともない，バッテリー駆動時間が重視されてきたため．もう一つは，サーバ用途プロセッサにおいても，開発コストを下げるためには液冷や液浸などの高価な冷却技術ではなく空冷を採用する必要があるためである．また，超並列計算機の要素プロセッサとして使用する場合，運用時の電力コストが大きな問題となる．現実に，本来保守のために装備されている，商用並列スーパコンピュータの部分的切り離し機構は，夜間の節電のためにも用いられている．このように，低電力化は避けて通れない課題である．

本稿では，命令レベルの情報を用いて，性能を低下させることなく，演算ユニット単位の消費電力を抑え

†1 京都大学工学部情報学科
Department of Information Science, Faculty of Engineering, Kyoto University
†2 京都大学大学院経済学研究科
Graduate School of Economics, Kyoto University
†3 オムロン株式会社技術本部IT研究所
Information Technology Research Center, OMRON Corporation
†4 京都大学総合情報メディアセンター
Center for Information and Multimedia Studies, Kyoto University
†5 京都大学大学院情報学研究科
Graduate School of Informatics, Kyoto University

ることを狙う．低電力化を考慮しない場合，プロセッサの各演算ユニットには，内部レジスタから読み出されたデータが常時供給され，演算結果を使用するか否かに関わらず，演算ユニットの内部信号を変化させる．また，演算に複数サイクルを必要とする場合，同様に，演算ユニット内部のラッチが無用のデータを次段に伝播させ，次段の内部信号を変化させる．一方，本稿では，各演算ユニットの入口にゲートを設け，レジスタからの読み出しデータが，演算に関与しない演算ユニットの内部信号に影響を与えないよう工夫することにより，前者に関する電力を削減する．また，演算ユニット内部のラッチに対するクロック供給を停止させることにより，後者に関する電力を削減する．さらに，ゲートおよびラッチを開閉させるために必要なクロック分配器の動作を停止させ，クロック分配系統自身の電力削減を図る．

より多くの電力を削減するためには，クロック分配系統のより上流においてクロックを停止する必要がある．しかしながら，上流のクロック分配器から各演算ユニットの入力ゲートおよびラッチに至る経路には遅延時間が存在する．命令のデコード結果から必要な演算ユニットを特定し，クロック分配器の動作を開始させる場合，動作周波数が高くなるほど，演算ユニットの動作開始が間に合わなくなる．このような場合に，いかに演算ユニットを効率良く動作させるかについての報告はまだなされていない．

以下では，現実の Java プロセッサ JeRTy[22] をモデルとし，演算に必要な演算ユニットを予測する極力小さなハードウェア機構を装備した場合について，クロック分配器の動作開始が演算に間に合わないことによる性能低下，および，クロック分配器の動作停止が間に合わないことによる電力増加をいかに抑えるかについて詳述する．

## 2. 関連研究

プロセッサの低電力化に関しては，極めて多くの研究成果が報告されている．プロセッサの全体や一部をスリープモードに遷移させる方法[1]，周波数や電圧を可変とする方法[2)~4)]，状態遷移ループを検出して遷移を止める方法[5),6)]，非同期回路の適用[7]，パストランジスタ等低電力論理の採用[8),9)]，SOI 等素子レベルの工夫[10)]など，様々なレベルの方法が提案されている．低電力化をめざした主な商用マイクロプロセッサとしては，周波数と電圧を動的に変化させる Speedstep$^{(TM)}$ 技術を採用した Intel Corp. のモバイル Pentium$^{(TM)}$III，および，PowerNow!$^{(TM)}$ 技術を採用した AMD,Inc. の AMD-K6$^{(TM)}$-2+，同様の LongRun$^{(TM)}$ 機能に加えて VLIW の採用によりゲート数および電力を抑えた Transmeta Corp. の Crusoe$^{(TM)}$ があげられる[11)]．

最近ではアーキテクチャレベルでの低電力化に関する研究が活発である．電力消費モデルを付加したサイクルシミュレータにより，SPEC95 などのベンチマークプログラムの消費電力をモデル化した研究[12),13)]，コンパイラが消費電力を考慮した最適化を行うことにより，レジスタファイルなどの消費電力を引き下げる方法[14),15)]，Direct Rambus DRAM（RDRAM）の電力モードと OS の実ページ割り付けとの連携による省電力化手法[16)]，パイプライン・ゲーティングと呼ぶ方法により，性能を低下させずに投機的実行による電力消費を抑制する手法[17)]などが報告されている．Java$^{(TM)}$ 関連では，ポケットコンピュータにおいて Java 仮想マシンを動作させた場合の消費電力に関する報告がある[18)]．

さて，N 型と P 型トランジスタを対称に配置する CMOS 回路は，スイッチング時にのみ電荷が移動するため，本来消費電力が小さい性質がある．しかし，最近では，プリチャージが必要なダイナミック回路を多用したり，周波数を極限まで高めることにより，特にクロック分配に関わる消費電力が極めて多くを占めるようになってきている．プロセッサを構成する機能ブロックのうち，未使用ブロックへの電源供給そのものを遮断することにより，電力を削減する方法が考えられる．しかし，最近の CMOS 回路は全体が巨大なコンデンサであるため，電源供給を再開してからそのブロックの電圧が安定するまでには，かなりのサイクル数を要する．また，機能ブロックへの突入電流が周辺の信号線にインダクティブノイズを引き起こす恐れがあることから，演算サイクル程度の時間間隔で機能ブロックの電力をこまめに節約する方法としては不向きである．このような状況では，クロックを停止することのできるゲート付きクロック（Gated Clock）を導入することにより，機能ブロック単位の電力の大幅な削減が期待できる[19)~21)]．

## 3. 演算ユニットと所要サイクル数の仮定

本稿において仮定する，Fetch，Decode，Execute，

☆ JeRTy はオムロン株式会社の登録商標

☆ Speedstep，Pentium は Intel Corp. の登録商標
☆ PowerNow!，AMD-K6 は AMD,Inc. の登録商標
☆ LongRun，Crusoe は Transmeta Corp. の登録商標
☆ Java は SUN Microsystems,Inc の登録商標

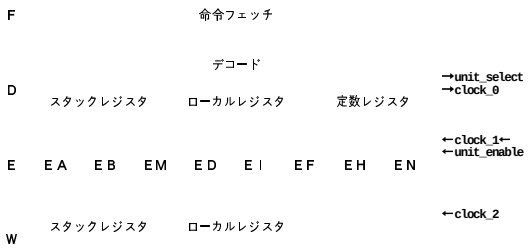


**図 1** 演算ユニットモデル
Fig. 1 Model of functional units.

**表 1** 所要サイクル数の仮定
Table 1 Model of execution cycles.

| 演算ユニット名 | EA | EB | EM | ED | EI | EF | EH | EN |
|---|---|---|---|---|---|---|---|---|
| 消費電力比率(%) | 10 | 10 | 22 | 2 | 10 | 24 | 12 | 10 |
| nop,popX,dupX | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Xconst,Xipush,Xstore | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| IALU,SFT,I2I,Fneg | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| Xload | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| if,Icmp,goto,jsr,ret | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| Xswitch | 0 | 8 | 0 | 0 | 0 | 0 | 0 | 0 |
| Imul | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 0 |
| Idiv,Irem | 0 | 0 | 16 | 0 | 0 | 0 | 0 | 0 |
| swap,dup_X | 0 | 0 | 0 | 10 | 0 | 0 | 0 | 0 |
| dup2_X | 0 | 0 | 0 | 20 | 0 | 0 | 0 | 0 |
| invokeX,Xreturn | 0 | 0 | 0 | 0 | 18 | 18 | 0 | 0 |
| Xaload,Xastore | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 |
| ldcX_X,astore | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 |
| Xstatic,field | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 |
| arraylength | 0 | 0 | 0 | 0 | 0 | 0 | 10 | 0 |
| new,Xnewarray | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 20 |
| Fadd,Fsub,Fmul,Fcmp | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 |
| I2F,F2I,F2F | 0 | 0 | 0 | 0 | 0 | 4 | 0 | 0 |
| Fdiv,Frem | 0 | 0 | 0 | 0 | 0 | 16 | 0 | 0 |
| Ddiv,Drem | 0 | 0 | 0 | 0 | 0 | 32 | 0 | 0 |
| invokeinterface | 0 | 0 | 0 | 0 | 0 | 18 | 0 | 0 |
| athrow | 0 | 0 | 0 | 0 | 0 | 18 | 0 | 0 |
| checkcast,instanceof | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 |
| multianewarray | 0 | 0 | 0 | 0 | 0 | 10 | 0 | 0 |
| monitorX | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 0 |

Write の 4 段パイプラインからなる，JeRTy の簡略化モデルを図 1 に示す．スタック上に配置される変数およびローカル変数のために，それぞれスタックレジスタとローカルレジスタを備え，命令中に含まれる定数を保持する定数レジスタとともに，D ステージにおける演算ユニットへの入力部を構成している．E ステージに対応する演算ユニットは 8 つの部分にわかれており，入力部の信号変化がユニット内部に伝搬しないよう，unit_enable 信号によって開閉するゲートが設けられている．unit_enable 信号は，E ステージ各ユニットの上流クロック分配器 clock_0 から遅れて動作する下流クロック分配器 clock_1 から生成され，clock_0 は，命令のデコード結果から得られる unit_select 信号によりゲートされる．表 1 の第二行に，1 サイクルにつき各演算ユニットおよび関連するクロック分配器が消費する電力を演算ユニット全体の電力に占める比率として示す．消費電力比は，各ユニットにおける論理素子の動作率が全て等しいと仮定し，JeRTy における論理素子数から概算したものである．また，第三行以降に，各命令の実行において E ステージに要するサイクル数を示す．X，I，ALU，SFT，F，D は，それぞれ任意の英数字，整数，整数加減/論理，シフト，単精度，倍精度浮動小数点数をそれぞれ代表して表記したものである．なお，各所要サイクル数は，簡略化モデルにおける最悪条件下での平均値であり，JeRTy の実際の動作とは若干異なる．

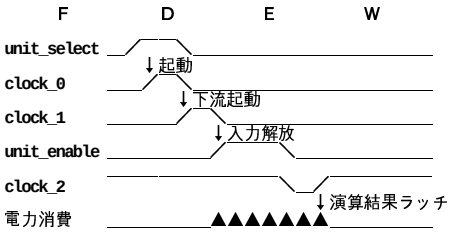


**図 2** 起動ペナルティ=0 の場合
Fig. 2 Case of penalty=0.

EA は基本的に 1 サイクルで終了する単純な命令，EB は分岐，EM は整数乗除算，ED は swap および複雑な dup_命令，EI はメソッド呼び出しおよびリターン，EF はフレーム生成および浮動小数点演算，EH はヒープ参照，EN は new 等オブジェクト生成に関する命令をそれぞれ担当するものとする．一般的な RISC プロセッサの TLB やデータキャッシュタグに相当する，ヒープ参照を高速化するためのオブジェクト TLB（4 ウェイ，1024 エントリ，ミスペナルティ=20 サイクル）やヒープキャッシュタグ機構（ダイレクトマップ，1024 エントリ，ラインサイズ=64 バイト，ミスペナルティ=20 サイクル）は全て EH に含まれるとする．また，E ステージを構成する演算ユニット全体の消費電力は，プロセッサ全体の 80%を占めると仮定する．

システム全体には，上記プロセッサの外に，キャッシュ本体，主記憶，入出力制御が存在するが，本稿では簡単のために，プロセッサ部分のうち，特に E ステージに関わる演算ユニットのみを消費電力の評価対象とする．なお，オブジェクト TLB ミスおよびヒープキャッシュミスが発生した場合，表 1 に示す所要サイクル数に加算し，この間，演算ユニットにおける消費電力は 0 であるとする．

## 4. クロック制御方法に関する仮定

前述したように，より上流のクロック分配器においてクロックを停止するほど，より多くの電力を削減す

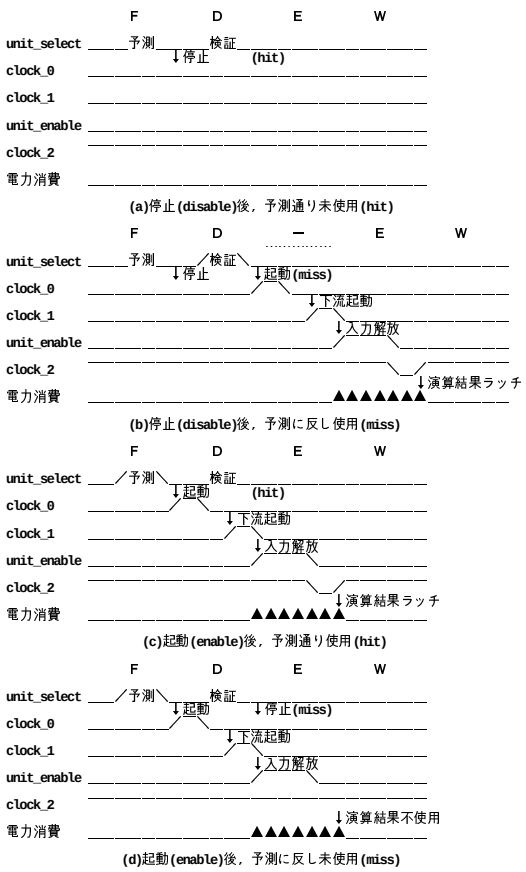


図 3　起動ペナルティ=1 の場合
Fig. 3　Case of penalty=1.

ることができる．しかし，上流のクロック再開が末端のゲートおよびラッチに届くまでの時間は長くなるため，電力削減効果と性能はトレードオフの関係となる．**図 2** および**図 3** に，クロック制御の基本的な考え方を示す．D ステージにおける unit_select 信号の確定が次の E ステージにおける unit_enable 信号の確定に間に合う場合を起動ペナルティ=0，クロック分配系統に大きな遅延があるために 1 サイクルのバブルが生じる場合を起動ペナルティ=1 と呼ぶことにする．

図 2 は起動ペナルティ=0 に対応している．命令デコードの結果，使用する演算ユニットに対する unit_select 信号が HIGH となる．これを受けて上流のクロック分配器からクロックパルス clock_0 が出力される．さらに下流のクロック分配器から clock_1 が出力され，E ステージの開始と同時に unit_enable 信号が一定期間だけ HIGH となる．演算ユニットが複数サイクルを要する場合には，unit_select 信号が複数回 HIGH となる．clock_2 は，特定のレジスタに対する書き込み信号であり，clock_0 および clock_1 とは独立して動作する．命令デコードの完了を待ってから，E ステージにおいて必要な各演算ユニットを起動できることから，性能を低下させることなく，E ステージにおける消費電力を最小とすることが可能である．

一方，図 3 は起動ペナルティ=1 に対応している．駆動能力の大きい上流のクロック分配器を制御するためには，多段のインバータを用いて unit_select 信号の駆動力を上げる必要がある．また，クロックスキューを抑えるために分配器間の等長配線を行う場合には，目的とするクロック分配器までの距離が短いとは限らない．このような場合，unit_select 信号が HIGH となってから，clock_1 が出力されるまでに多くの時間を要する．パイプラインをストールさせることなく E ステージを開始するためには，D ステージにおいて unit_select 信号が確定する前に，F ステージにおいて unit_select 信号を投機的に HIGH とする必要がある．(a) は予測通り未使用であった場合であり，対応する演算ユニットの消費電力は 0 となる．(b) は unit_select 信号を LOW としたものの，予測に反して演算ユニットが必要であった場合である．D ステージにおいて unit_select 信号が HIGH となることから，1 サイクルのバブルが発生する．(c) は予測通り使用した場合であり，バブルは発生しない．(d) は予測に反して演算ユニットが不要であった場合であり，バブルは発生しないものの電力は無駄に消費される．なお，演算ユニットが複数サイクルを要する場合でも，無駄な電力消費は 1 サイクル分に留まる．(b) および (d) の場合，D ステージにおける検証時に，後述する予測表の更新を行うとする．

性能が低下する (b) および電力が増加する (d) の場合が発生しないよう完全な予測を行うことにより，電力削減効果と性能は，起動ペナルティ=0 の場合に等しくなる．しかし，完全な予測のための大規模なハードウェア機構を設けることにより，かえって電力が増加することは本末転倒である．また，(b) の発生を抑えるためにより多くの演算ユニットを起動しておくことは (d) の増加となる．逆に，(d) の発生を抑えるためになるべく演算ユニットを起動しない方針とすると，(b) が増加する．

以下では，次に実行する演算を予測し，必要最小限の演算ユニットのみを起動することとし，いかに小さなハードウェアおよび予測機構により，(b) および (d) を削減することができるかについて，議論を進める．

## 5. 演算ユニット使用予測方法の提案

起動ペナルティ=1 である場合，すなわち，F ステー

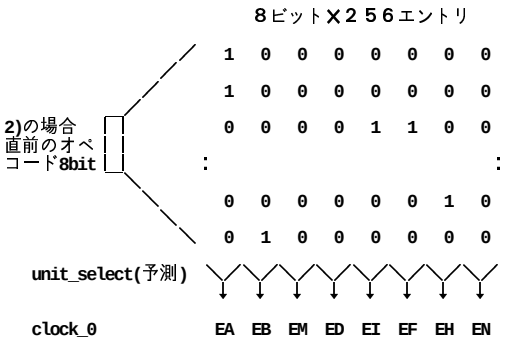


図 4 演算ユニットの使用予測機構
Fig. 4 Prediction model of unit usage.

ジ完了時に演算ユニットの使用を予測するための情報としては，1）F ステージ開始時におけるプログラムカウンタ（PC）の下位数ビット；2）直前にデコードした命令のオペコード；3）命令に関連するデータ型（int，long，float，double，reference）により直前命令のオペコードをグループ化したもの；4）使用する演算ユニットによりグループ化したもの；の 4 つが考えられる．1）は比較的小さなメソッドに対して有効であるものの，各メソッドの先頭は 0 番地であることから，あまりに小さくループ構造もないようなメソッドが多数実行される場合には，メソッドを識別する情報を加える必要があると考えられる．2）は連続する命令のオペコードに強い相関がある場合に有効である．Java 仮想マシンはスタックマシンであることから，一連のスタック操作を繰り返すバイトコードは，一般の RISC 命令よりもオペコードの相関が強いと考えられる．3）は命令間において授受されるデータ型の連続性を利用して，2）よりもハードウェアを削減する試みである．4）は使用する演算ユニットに着目して，同様にハードウェアを削減する試みである．予測機構の概略を図 **4** に示す．予測表の各エントリは，表 1 の各行と同様に，8 個の演算ユニットそれぞれを起動（1）または停止（0）するための 8 ビットの値を保持している．F ステージにおいて，前述の情報をインデックスとして予測表を参照する．そして，D ステージにおける命令デコード結果に従い，演算ユニットの実際の使用状況を予測表に反映する．

さらに，起動ペナルティ=2 である場合についても，同様の方法により予測を試みる．ただし，1）は 1 サイクル前の PC となるため，条件分岐による撹乱が生じる．2）は 2 命令前のオペコードとなるため，相関が弱くなるはずである．3）および 4）についても，同様に相関が弱くなると推測できる．

表 **2** 演算ユニット使用率（上段%）および電力比（下段%）
Table 2 Activity ratio(upper%) and Power consumption(lower%).

| | EA | EB | EM | ED | EI | EF | EH | EN | 全体 |
|---|---|---|---|---|---|---|---|---|---|
| compress | | | | | | | | | |
| s1 | 16.3 | 2.2 | 0.0 | 2.6 | ○11.4 | ○11.4 | 46.7 | 0.0 | **79.2%** |
| | 1.6 | 0.2 | 0.0 | 0.1 | 1.1 | 2.7 | 5.6 | 0.0 | **11.4%** |
| s10 | 17.0 | 2.5 | 0.0 | 2.5 | ○10.2 | ○10.2 | 46.4 | 0.0 | **78.6%** |
| | 1.7 | 0.2 | 0.0 | 0.1 | 1.0 | 2.5 | 5.6 | 0.0 | **11.1%** |
| s100 | 15.7 | 2.2 | 0.0 | 2.5 | ○10.9 | ○10.9 | 44.9 | 0.0 | **76.1%** |
| | 1.6 | 0.2 | 0.0 | 0.1 | 1.1 | 2.6 | 5.4 | 0.0 | **10.9%** |
| jess | | | | | | | | | |
| s1 | 12.6 | 3.5 | 0.3 | 0.7 | 35.3 | 37.2 | 35.3 | 1.5 | **91.1%** |
| | 1.3 | 0.4 | 0.1 | 0.0 | 3.5 | 8.9 | 4.2 | 0.1 | **18.5%** |
| s10 | 14.4 | 4.9 | 0.1 | 0.1 | 29.5 | 30.9 | 40.0 | 0.3 | **90.7%** |
| | 1.4 | 0.5 | 0.0 | 0.0 | 2.9 | 7.4 | 4.8 | 0.0 | **17.2%** |
| s100 | 12.4 | 3.6 | 0.4 | 0.2 | 36.0 | 38.4 | 33.6 | 1.2 | **89.7%** |
| | 1.2 | 0.4 | 0.1 | 0.0 | 3.6 | 9.2 | 4.0 | 0.1 | **18.6%** |
| db | | | | | | | | | |
| s1 | 15.0 | 4.3 | 0.4 | 1.4 | 35.4 | 36.2 | 36.4 | 1.2 | **94.9%** |
| | 1.5 | 0.4 | 0.1 | 0.0 | 3.5 | 8.7 | 4.4 | 0.1 | **18.8%** |
| s10 | 16.5 | 3.3 | 0.0 | 0.2 | 17.2 | 19.3 | 38.3 | 0.6 | **78.2%** |
| | 1.6 | 0.3 | 0.0 | 0.0 | 1.7 | 4.6 | 4.6 | 0.1 | **13.0%** |
| s100 | 14.2 | 2.7 | 0.0 | 0.3 | 16.1 | 19.4 | 38.2 | 0.2 | **75.1%** |
| | 1.4 | 0.3 | 0.0 | 0.0 | 1.6 | 4.7 | 4.6 | 0.0 | **12.6%** |
| javac | | | | | | | | | |
| s1 | 15.9 | 3.6 | 0.7 | 2.1 | 32.0 | 33.9 | 37.5 | 1.0 | **94.7%** |
| | 1.6 | 0.4 | 0.2 | 0.0 | 3.2 | 8.1 | 4.5 | 0.1 | **18.1%** |
| s10 | 14.0 | 4.9 | 0.8 | 1.4 | 32.0 | 33.8 | 35.7 | 0.9 | **91.5%** |
| | 1.4 | 0.5 | 0.2 | 0.0 | 3.2 | 8.1 | 4.3 | 0.1 | **17.8%** |
| s100 | 12.8 | 5.1 | 0.6 | 1.4 | 31.8 | 34.0 | 35.3 | 0.8 | **89.9%** |
| | 1.3 | 0.5 | 0.1 | 0.0 | 3.2 | 8.2 | 4.2 | 0.1 | **17.6%** |
| mpegaudio | | | | | | | | | |
| s1 | 16.6 | 1.4 | 0.1 | 0.3 | ○ 6.4 | ○13.1 | 44.2 | 0.0 | **75.7%** |
| | 1.7 | 0.1 | 0.0 | 0.0 | 0.6 | 3.1 | 5.3 | 0.0 | **10.9%** |
| s10 | 16.5 | 1.2 | 0.0 | 0.3 | ○ 5.0 | ○12.0 | 44.7 | 0.0 | **74.9%** |
| | 1.7 | 0.1 | 0.0 | 0.0 | 0.5 | 2.9 | 5.4 | 0.0 | **10.5%** |
| s100 | 16.7 | 1.3 | 0.1 | 0.3 | ○ 6.1 | ○12.8 | 44.2 | 0.0 | **75.5%** |
| | 1.7 | 0.1 | 0.0 | 0.0 | 0.6 | 3.1 | 5.3 | 0.0 | **10.8%** |
| mtrt | | | | | | | | | |
| s1 | 11.7 | 1.9 | 0.0 | 1.3 | ● 44.3 | ● 45.7 | 29.8 | 1.2 | **91.7%** |
| | 1.2 | 0.2 | 0.0 | 0.0 | 4.4 | 11.0 | 3.6 | 0.1 | **20.5%** |
| s10 | 9.9 | 1.4 | 0.0 | 0.7 | ● 46.4 | ● 48.8 | 26.4 | 0.9 | **88.1%** |
| | 1.0 | 0.1 | 0.0 | 0.0 | 4.6 | 11.7 | 3.2 | 0.1 | **20.7%** |
| s100 | 8.0 | 0.8 | 0.0 | 0.1 | ● 49.7 | ● 53.0 | 21.6 | 0.6 | **84.2%** |
| | 0.8 | 0.1 | 0.0 | 0.0 | 5.0 | 12.7 | 2.6 | 0.1 | **21.2%** |
| jack | | | | | | | | | |
| s1 | 10.8 | 2.8 | 0.4 | 1.7 | 35.0 | 38.2 | 35.9 | 2.9 | **92.7%** |
| | 1.1 | 0.3 | 0.1 | 0.0 | 3.5 | 9.2 | 4.3 | 0.3 | **18.8%** |
| s10 | 10.8 | 2.8 | 0.4 | 1.7 | 35.1 | 38.2 | 35.9 | 2.9 | **92.7%** |
| | 1.1 | 0.3 | 0.1 | 0.0 | 3.5 | 9.2 | 4.3 | 0.3 | **18.8%** |
| s100 | 10.8 | 2.8 | 0.4 | 1.7 | 35.2 | 38.4 | 36.0 | 3.0 | **92.9%** |
| | 1.1 | 0.3 | 0.1 | 0.0 | 3.5 | 9.2 | 4.3 | 0.3 | **18.8%** |

## 6. 起動ペナルティ=0 の場合の理想的電力

演算ユニットの使用率および消費電力のシミュレーションには，表 1 に示した各演算ユニットの所要サイクル数と消費電力比率，および，Java 仮想マシンである Kaffe1.0.6[23]を用いて，SPECJVM98[24]の各ベンチマークプログラムを実行サイズ s1，s10，s100 により走行して得た命令トレースを用いた．**表 2** に，起動ペナルティ=0 の場合の各演算ユニット使用率（上段），使用率に表 1 の消費電力比率を乗じて得られる電力比（下段）を示す．右端列の上段は，全体の所要サイクル数のうち，少なくとも 1 つの演算ユニットが

動作した割合を表している．100%との差分，すなわち，演算ユニットが1つも動作していないサイクルは，前述したオブジェクト TLB ミスまたはヒープキャッシュミスが発生している期間に対応する．

右端列の下段は，電力比の合計，すなわち，クロックを常時投入した場合の演算ユニット全体の消費電力に対する，本方式における消費電力の比率を表しており，起動ペナルティ=1，2 の場合において予測が全て的中した場合の消費電力の下限値に相当する．未使用の演算ユニットおよびクロック分配器における消費電力を抑えることにより，約 80%から 90%の電力を削減できることがわかる．

●は EI および EF ユニットにおいて使用率が40%を越えること，また，○は 15%未満であることを示す．電力比の合計が 20%を越えるものは，いずれも●を，また，電力比の合計が 10%程度のものは，いずれも○を含んでおり，invoke 命令が多数出現する場合に消費電力が 20%程度に倍増することを示している．これは，表 1 の第二行に示した EI および EF ユニットの単位時間あたりの消費電力が 10+24=34%を占め，かつ，他の演算ユニットに比べて 1 命令あたりの所要サイクル数が多いためと考えられる．

## 7. 起動ペナルティ>0 の場合の予測の効果

我々の目標は，起動ペナルティ>0 の場合について，前述した電力下限値に近く，かつ，起動ペナルティによる性能低下が小さくなるような，極力小さい演算ユニット使用予測機構を提案することである．本章では，5章において述べた予測方法を次のように具体化して，消費電力および所要サイクル数の測定を行った．

**NP：予測なし**　予測を行わず，必要になるまでクロックを止める方法．消費電力の下限（最良）値および所要サイクル数の上限（最悪）値が得られる．

**PC08：PC8 ビット**　「起動ペナルティ−1」だけ手前の PC の下位 8 ビットを用いる．

**PC26：メソッド識別子 2 ビット＋ PC6 ビット**
「起動ペナルティ−1」だけ手前の，メソッド識別子（メソッドに固有の値）下位 2 ビットおよび PC の下位 6 ビットからなる合計 8 ビットを用いる．

**PC44：メソッド識別子 4 ビット＋ PC4 ビット**
「起動ペナルティ−1」だけ手前の，メソッド識別子下位 4 ビットおよび PC の下位 4 ビットからなる合計 8 ビットを用いる．

**OP8：オペコード 8 ビット**　「起動ペナルティ」だけ手前の命令のオペコード 8 ビットを用いる．ただし WIDE 命令は引き続くオペコードを使用する．

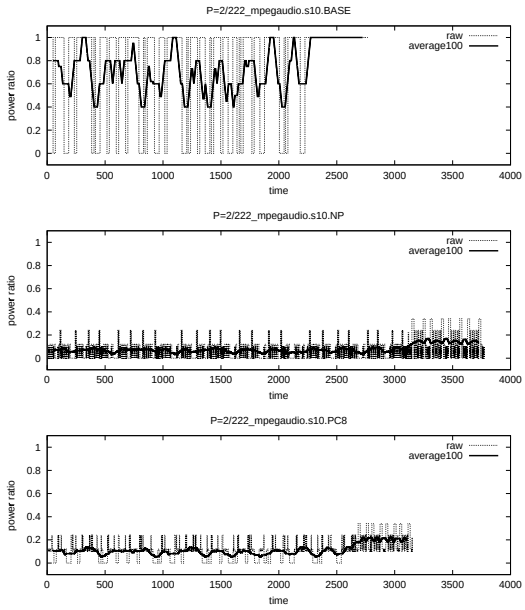


図 5　mpeg の電力消費
Fig. 5　Power consumption of mpeg.

**PC3：PC3 ビット**　「起動ペナルティ−1」だけ手前の PC の下位 3 ビットを用いる．

**TY3：データ型 3 ビット**　スタック上に出力するデータ型（前述の 5 種類）により分類し，3 ビットにより識別する．出力がない場合は入力データ型を用いる．

**UN3：演算ユニット 3 ビット**　表 1 に示したように，使用する演算ユニットに応じてオペコードを 8 つのグループに分類し，3 ビットにより識別する．

まず，演算ユニット全体における電力消費の様子を**図 5** および**図 6** に示す．それぞれ，表 1 において最も電力比の小さい mpeg および電力比の大きい mtrt を実行サイズ s10 により走行させ，途中の同じ 500 命令分の消費電力をサイクルごとに抽出したものである．なお太線は，前後合わせて 100 サイクル分の移動平均値である．上段は起動ペナルティ=0 かつクロックを常時投入した場合すなわち低電力化対策を何も施さない場合，中段は起動ペナルティ=2 かつ NP の場合，下段は起動ペナルティ=2 かつ PC08 の場合を表している．

浮動小数点演算が多い mpeg では，演算ユニットの使用率を示す表 1 の右端列上段の値が 100%に大きく届かないことからわかるように，オブジェクト TLB ミスまたはヒープキャッシュミスが多発する．このため，低電力化対策を施さない上段のグラフにおいて消費電力が激しく変化することがわかる．ただし，横軸は 2800 サイクルまでとなっている．一方，NP の場合，

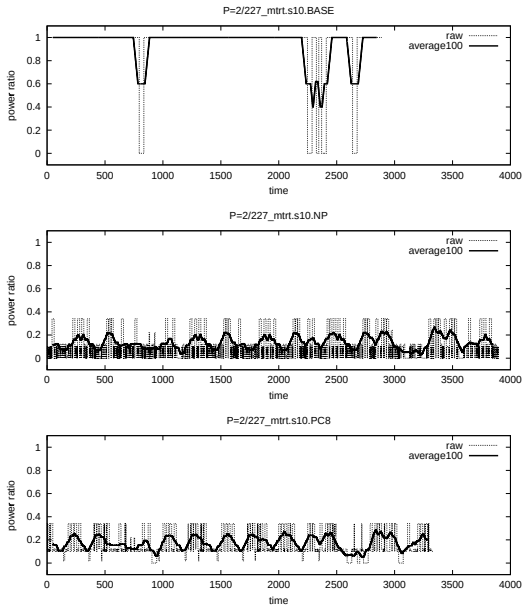


図 6 mtrt の電力消費
Fig. 6 Power consumption of mtrt.

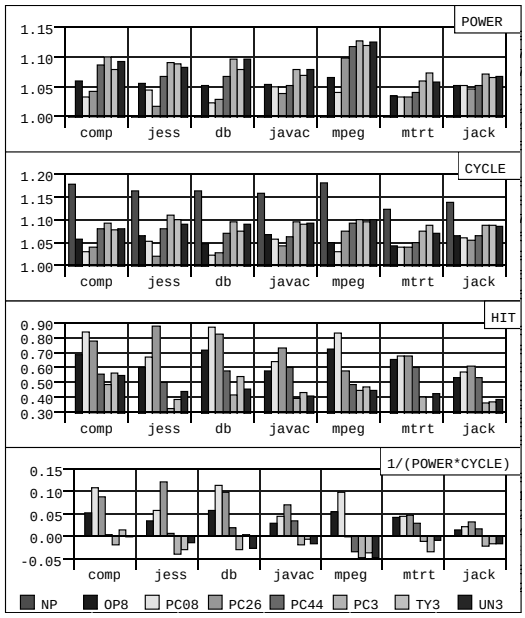


図 7 起動ペナルティ=1 の場合
Fig. 7 Case of penalty=1.

消費電力を抑えることができるものの，横軸は 3800 サイクルまで増加しており，消費電力と引き替えに性能が低下していることがわかる．これに対し，PC08 の場合，消費電力が若干増加するものの，横軸が 3200 サイクルに短縮されている．ところで，PC08 は，必要になるまで全ての演算ユニットを止める NP よりも消費電力の変動が緩やかであることがわかる．これは，冒頭において述べたように，電源ノイズ対策の観点から重要な特長であると言える．

一方，メソッド呼び出しが多い mtrt では，オブジェクト TLB ミスやヒープキャッシュミスが少なく，上段のグラフにおける変化は少ないものの，消費電力はむしろ mpeg よりも多い．また，mpeg と同様，PC08 の場合には，NP よりも消費電力が若干増加するものの，性能低下や消費電力の変動が抑えられていることがわかる．

次に，各予測方法について，**図 7** に起動ペナルティ=1 の場合，**図 8** に起動ペナルティ=2 の場合の測定結果を示す．なお，プログラムサイズ s1，s10，s100 の全てについて測定した結果，s10 が全体を最もよく代表していると判断した．見やすさのため，s10 の測定結果のみをグラフ表示している．各図の上段（POWER）は，各プログラムを完走させるために必要であった総電力である．起動ペナルティ=0 の場合の総電力を 1 としたときの，各予測方法における総電力を表している．二段目（CYCLE）は，同様に，各プログラムを

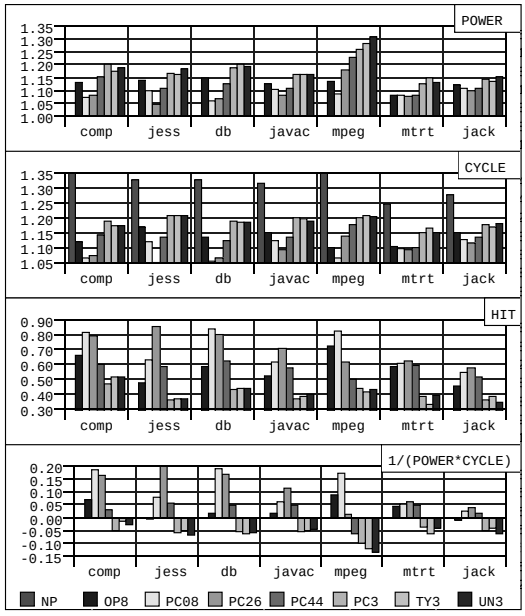


図 8 起動ペナルティ=2 の場合
Fig. 8 Case of penalty=2.

完走させるために必要であった総サイクル数である．起動ペナルティ=0 の場合の総サイクル数を 1 としたときの，各予測方法における総サイクル数を表している．三段目（HIT）は，図 4 に示した予測表のヒット率である．

ところで，一般的に用いられる評価尺度 $Mips/Watt$

は，一定の電力により実行可能な命令数を表しており，実行速度が考慮されない．このため本稿では，$Mips^2/Watt$ に相当する次式を評価尺度として用いることにする．

$$\frac{(\text{総実行命令数}/\text{総サイクル数})^2}{\text{総電力}/\text{総サイクル数}} \quad (1)$$

ただし，各プログラムごとに総実行命令数は一定であるため，予測方法の比較には，次式を用いることができる．

$$1/(\text{総電力} * \text{総サイクル数}) \quad (2)$$

以上の考えに基づき，下段に，各方法における1/（総電力✻総サイクル数）の値と，NP における同様の値との比を求め，1 を減じたものを表示した．すなわち，NP を基準とし，＋は $Mips^2/Watt$ が優れていること，－は劣っていることを意味する．

さて，クロックの投機的投入を行わない NP では，電力比は 1，サイクル数は，起動ペナルティ=1 の場合 10%から 20%，起動ペナルティ=2 の場合 25%から 35%の増加となっており，それぞれ，各予測方法における電力比の下限およびサイクル数の上限を示している．NP では予測表のヒット率を 0 と見なしており，グラフ上には現れない．

8 ビットの情報を用いる方法の中では，起動ペナルティ=1，2 のいずれの場合にも，comp，db，mpeg においては PC08，jess，javac，mtrt，jack においては PC26 が最もヒット率が高く，ほぼ 60%から 90%となっている．起動ペナルティ=1 の場合，消費電力およびサイクル数ともに 5%程度の増加，また，起動ペナルティ=2 の場合，ともに 10% 程度の増加に留まっている．NP に対するサイクル数の比では，起動ペナルティ=1 の場合約10%，起動ペナルティ=2 の場合約 20%の減少となっており，予測による性能向上が電力増加を上回っている．これは，下段のグラフに示すように，PC08 や PC26 における $Mips^2/Watt$ の値が，起動ペナルティ=1 の場合は多いもので約 10%，起動ペナルティ=2 の場合は多いもので約 20%改善されていることからも確認できる．

一方，3 ビットの情報では，全般的にヒット率が 50%未満と低くなる．起動ペナルティ=1 の場合，TY3 が最も消費電力およびサイクル数が少ないものの，ともに 10%程度増加している．また，起動ペナルティ=2 の場合，PC3 と TY3 とが同程度であり，消費電力では最大 30%，また，サイクル数では 20%程度増加していることがわかる．NP に対するサイクル数の比では，起動ペナルティ=1 の場合約 5%，起動ペナルティ=2 の場合約 10%の減少に留まっており，予測による性能向上よりも電力増加のほうが大きい．下段のグラフでも，TY3 における $Mips^2/Watt$ の値の大部分が負の値となっていることがわかる．

## 8. 考　　察

測定の結果，起動ペナルティ分だけ先の命令が使用する演算ユニットを予測するためには，8 ビットの情報では，メソッド識別子の下位 2 ビットおよび PC の下位 6 ビット，また 3 ビットの情報では，データ型 3 ビットが，それぞれ最も有効であることが明らかになった．ただし，予測を全く行わない方法に対する $Mips^2/Watt$ の改善という点では，3 ビットの情報は不十分であることがわかった．

また，以上に示した測定結果とは別に，予測表のエントリ数を $2^{16}$ とし，PC の下位 16 ビットを用いて同様に測定したところ，PC08 よりも若干ヒット率が向上したものの，ほぼ同様の効果しか得られないことがわかっている．多くのビット数を用いても効果が上がらないのは，各メソッドの先頭 PC が常に 0 であること，また，SPECJVM98 の各ベンチマークプログラムでは，メソッドあたりの命令数が比較的小さく，PC の下位 8 ビットを除く上位ビットが 1 になる可能性が極めて小さいためと考えている．実際，PC08 ではヒット率が 60%に届かない jack について，予測表のエントリ数を $2^{16}$ とし，PC08 にメソッド識別子の下位 8 ビットを加えた合計 16 ビットをインデックスとして用いたところ，ヒット率が 70%を越えることを確認した．各プログラムにおいて動作するメソッドの種類は，たかだか 300 程度であることから，エントリ数さえ増加させれば，全てのプログラムについてヒット率を格段に上げられると推測できる．ただし，予測表は全ての命令が参照することから，予測表自身の消費電力が問題となる．最適な予測表のサイズを決定するには，より詳細な電力消費モデルの構築が必要であり，今後の課題である．

ところで，表 1 では最も小さかった mpeg の電力比が，図 7 や図 8 では他のプログラムに比べて際立って大きい．表 2 から，mpeg における EI の使用率が他に比べて極めて低いこと，また，EI と EF の使用率の差分が 6%程度と，他に比べて極めて大きいことが読みとれる．表 1 からわかるように，EI と EF の使用率差分とは，主に浮動小数点演算が占めるサイクル数の比率である．浮動小数点演算が多いために，予測表のヒット率の若干の低下が，浮動小数点演算ユニットにおける消費電力を大幅に増加していると考えられる．

## 9. おわりに

本稿では，JeRTy をモデルとし，演算ユニットごとにクロックを投入および停止して消費電力を抑える方法について評価を行った．上流のクロック分配器を起動してから演算ユニットが使用可能となるまでに 1 または 2 サイクルを要する場合でも，クロック分配器を投機的に制御することにより，性能低下を抑えつつ電力を削減できることを示した．特に，予測に用いる情報としては，メソッド識別子の下位 2 ビットおよび PC の下位 6 ビット，または，データ型を表す 3 ビットが，同じビット数を用いる方法の中ではそれぞれ最も高い効果が得られること，また，8 ビットの情報を用いる場合，予測による性能向上が消費電力増加を上回るものの，3 ビットでは逆転することを明らかにした．

**謝辞**

本研究の一部は文部省科学研究費補助金（基盤研究 (B)(2) 課題番号 12480072 ならびに 12558027）による．

## 参考文献

1) Benini L., et al.: System-level Dynamic Power Management, Proc. 1999 IEEE Alessandro Volta Memorial Workshop on Low-Power (VOLTA'99) (1999).
2) Usami K. and Horowitz M.: Clustered voltage scaling technique for low-power design, Proc. Int'l Symp. on Low Power Electronics and Design (ISLPED'95), pp.3–8 (1995).
3) Athas W.: Low-Power VLSI Techniques for Applications in Embedded Computing, Proc. VOLTA'99 (1999).
4) Burd T.D. and Brodersen R.W.: Design Issues for Dynamic Voltage Scaling, Proc. ISLPED'00, pp.9–14 (2000).
5) Koegst M., et al.: Low Power Design of FSMs by State Assignment and Disabling Self-Loops, Proc. 23rd Euromicro Conf. on New Frontiers of Information Technology (1997).
6) Raghavan N., et al.: Automatic Insertion of Gated Clocks at Register Transfer Level, Proc. 12th Int'l Conf. on VLSI Design (1998).
7) Schuster S, et al.: Asynchronous Interlocked Pipelined CMOS Circuits Operating at 3.3-4.5GHz, Proc. IEEE Int'l SolidState Circuits Conf. (ISSCC'00), pp.292–293 (2000).
8) Strollo A.G.M., et al.: New Clock-Gating Techniques for Low-Power Flip-flops, Proc. ISLPED'00, pp.114–119 (2000).
9) Vinereanu T. and Lidholm S.: An Improved Pass Transistor Synthesis Method for Low Power, High Speed CMOS Circuits, Proc. ISLPED'00, pp.120–124 (2000).
10) Joshi R.V., et al.: Low Power 900 MHz Register File (8 Ports, 32 Words x 64 Bits) in 1.8V, 0.25 $\mu$ SOI Technology, Proc. 13th Int'l Conf. on VLSI Design (2000).
11) Halfhill T.: Transmeta breaks x86 low-power barrier, Microprocessor Report, Feb. (2000).
12) Brooks D., et al.: Wattch: A Framework for Architectural-Level Power Analysis and Optimizations, Proc. 27th Int'l Symp. on Computer Architecture (ISCA'00), pp.83–94 (2000).
13) 井上弘士, 村上和彰: 実行履歴に基づいた低電力命令キャッシュ向けタグ比較回数削減手法, 情報処理学会研究報告 (2000).
14) Mehta H., et al.: Techniques for low energy software, Proc. ISLPED'97, pp.72–75 (1997).
15) Vijaykrishnan N., et al.: Energy-Driven Integrated Hardware-Software Optimizations Using SimplePower, Proc. ISCA'00, pp.95–106 (2000).
16) Lebeck A.R., et al.: Power Aware Page Allocation, Proc. Architectural Support for Programming Languages and Operating Systems (ASPLOS IX), pp.105–116 (2000).
17) Manne S., et al.: Pipeline Gating: Speculation Control for Energy Reduction, Proc. ISCA'98, (1998).
18) Farkas K.I., et al.: Quantifying the energy consumption of a pocket computer and a Java virtual machine, Proc. Joint Int'l Conf. on Measurement and Modeling of Computer Systems (SIGMETRICS'00), pp.252–263 (2000).
19) Brooks D., et al.: Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors, IEEE MICRO, Nov/Dec, pp.26–44 (2000).
20) Garrett D., et al.: Challenges in Clockgating for a Low Power ASIC Methodology, Proc. ISLPED'99, pp.176–181 (1999).
21) M.D.Pant et al.: An Architectural Solution for the Inductive Noise Problem due to Clock-Gating, Proc. ISLPED'99, pp.255–257 (1999).
22) OMRON Corporation: JeRTy, http://www.jerty.com (2000).
23) Kaffe.org: Welcome to Kaffe, http://www.kaffe.org/
24) SPECJVM98 VERSION1.03, the Standard Performance Evaluation Corporation (1998).