計算機アーキテクチャ 142－4
ハイパフォーマンス
コンピューティング 85－4
（2001．3．8）

低電力化のための投機的クロック供給制御

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．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，プログラムカウンタの下位8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．

Speculative Clock Control of Execution Units for Low Power

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 lower 8-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．

低電力化のためには，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，周波数や電圧を可変とする方法[9),10)]，状態遷移ループを検出して遷移を止める方法[5),7)]，パストランジスタ等低電力論理の採用[11),12)]，SOI等素子レベルの工夫[6)]など，様々なレベルの方法が提案されており，現在に至るまでに極めて多くの研究成果が報告されている．低電力化をめざ

†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

した商用マイクロプロセッサとしては，周波数と電圧をダイナミックに変化させる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．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[2),4),8)]．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する研究などが報告されている[3),14)]．

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，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，かなりの時間を要する．また，機能ブロックへの突入電流が周辺の信号線にカップリング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[2),13)]．

より多くの電力を節約するためには，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，上流のクロックから末端の論理回路に至る経路には遅延時間が存在するため，動作周波数が高いほど性能に対する影響が大きくなる．この影響をいかに小さくするかについての報告はまだなされていない．

本稿では，性能を落さずにクロック分配系統および演算ユニットの消費電力をいかに抑えるかについて，現実のJava$^{(TM)}$プロセッサJeRTy$^{(TM)1)}$をモデルに調査検討を行った．デコード結果をもとに，遅滞なく，関連する演算ユニットへのクロックを投入/停止することができれば，実行速度を低下させずに消費電力を削減することができる．しかし，クロック周波数が高い場合，命令デコードの完了を待っていたのでは上流のGated Clockの動作再開が演算開始に間に合わないため，性能低下を引き起こす．関連する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設け，クロック投入が間に合わないことによる性能低下，および，クロック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，以下に詳述する．

☆ Speedstep$^{(TM)}$, Pentium$^{(TM)}$はIntel Corp. の登録商標
☆ PowerNow!$^{(TM)}$, AMD-K6$^{(TM)}$はAMD,Inc. の登録商標
☆ LongRun$^{(TM)}$, Crusoe$^{(TM)}$はTransmeta Corp. の登録商標
☆ Java$^{(TM)}$はSUN Microsystems,Incの登録商標
☆ JeRTy$^{(TM)}$はオムロン株式会社の登録商標

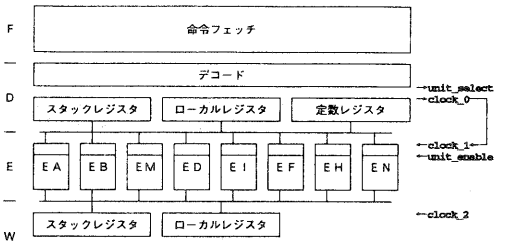

図1　演算ユニットモデル
Fig. 1　Model of functional units.

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

2. 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，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器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clock_0から遅れて動作する下流クロック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

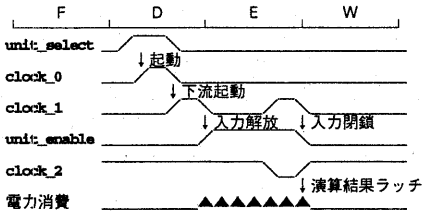

図2　起動ペナルティ=0 の場合
Fig. 2　Case of penalty=0.

る．表1の第二行に，1サイクルにつき各演算ユニット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．なお，この所要サイクル数は，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．

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 dup2 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，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，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．ただし，オブジェクト TLB ミスおよびヒープキャッシュミスは，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．

3. クロック制御方法に関する仮定

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ほど，より多くの電力を削減することができる．しかし，上流のクロック再開が末端の論理回路に届くまでの時間は長くなるため，電力削減効果と性能とはトレードオフの関係となる．**図2**および**図3**に，低電力化のための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 信号確定が次の E ステージにおける unit_enable 信号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無視できない遅延があるために1サイクルのバブルが生じる場

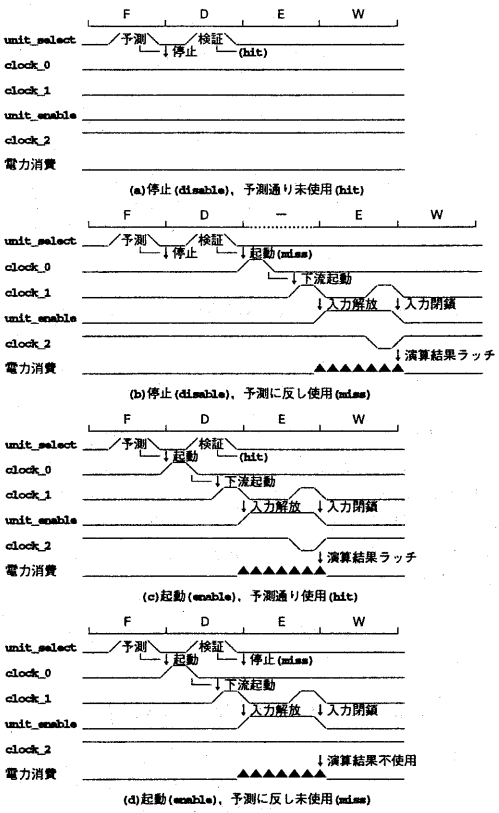

図3　起動ペナルティ=1 の場合
Fig. 3　Case of penalty=1.

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 に対応しており，命令デコードの結果を待ってから 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，性能を落すことなく，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．バブルを発生させることなく 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，clock_0 を投機的に投入する必要がある．(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，予測に反して演算ユニットが必要であった場合であり，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，電力は無駄に消費される．(b) および (d) の場合，検証時に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) のケースが発生しないよう完璧な予測を行うことにより，電力削減効果と性能は，ペナルティ=0 の場合に等しくなる．しかし，完璧な予測のための大規模なハード

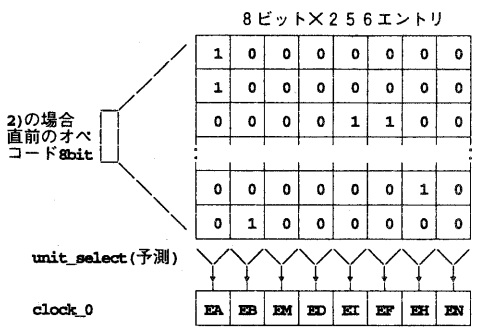

図4　演算ユニットの使用予測機構
Fig. 4　Prediction model of unit usage.

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，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．

4. 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効果が小さいと考えられる．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．

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%**

5. 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，Java仮想マシンであるKaffe1.0.6を用いて，SPECJVM98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，および，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．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における電力をカットすることにより，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．

6. 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，4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．

PC8：PCの下位8bit　「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．

OP8：オペコード8bit　「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．

PC3：PCの下位3bit　「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3bit　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3bit　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．

図5に起動ペナルティ=1の場合，**図6**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が各プログラム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．各図の上段は，起動ペナルティ=0の消費電力を1とした場合の，各予測方法における消費電力である．中段は，同様に起動ペナルティ=0の所要サイクル数を1とした場合の，各所要サイクル数である．下段は，図4に示した予測表のヒット率である．

クロックの投機的投入を行わないNPでは，消費電力比は1，サイクル数は10%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の下限およびサイクル数の上限を示している．

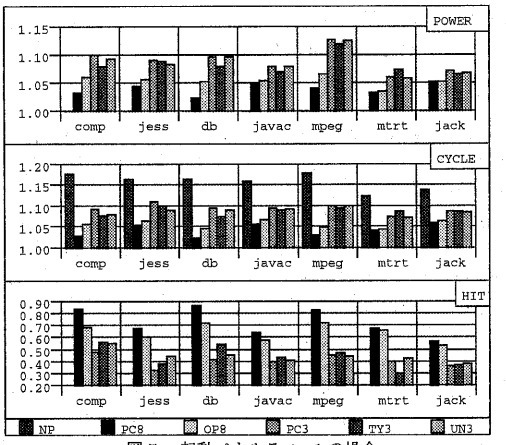

図5　起動ペナルティ=1の場合
Fig. 5　Case of penalty=1.

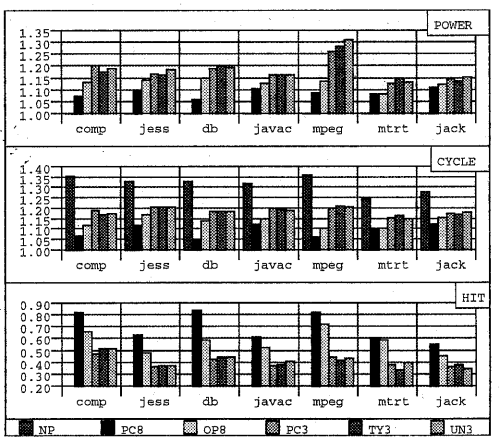

図6　起動ペナルティ=2の場合
Fig. 6　Case of penalty=2.

8ビットの情報を用いる方法の中では，PC8のヒット率が60%から90%と高く，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とがわかる．

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ことがわかる．

7. 考　　察

測定の結果，8 ビットまたは3 ビットの情報を用いて次命令のオペコードを予測するためには，次命令のPC の下位8 ビットまたは現命令のデータ型3 ビットを用いる方法が，それぞれ高い効果を示すことが明らかになった．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，PC8 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．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，メソッドの種類を8 ビットにより表現し，これにPC の下位8 ビットを加えた合計16 ビットを用いることにより，ヒット率を格段に上げられると推測できる．しかし，はじめに述べたように，大量のハードウェアを投入することは，省電力化の目的に反する．さらに実験を行った結果，合計が8 ビットとなる範囲においてメソッド識別子を0，1，2，3，4 ビット，PC を8，7，6，5，4 ビットと変化させた場合，起動ペナルティ=1 における jess-s10 のヒット率は，67%，69%，81%，74%，52%と，2+6 ビットの組み合わせが最も良いことがわかっている．他のプログラムも含めた詳細な評価については，今後の課題である．

ところで，mpeg の消費電力が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ことが，消費電力を押し上げている主な要因であると考えられる．

8. 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．上流クロックを再開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，上流クロック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，PC の下位8 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．さらに，一部をメソッド識別子に置き換えることにより，高い効果を得られる可能性があるものの，詳細な評価については今後の課題である．

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) OMRON Corporation: JeRTy, http://www.jerty.com (2000).
2) D.Brooks, et al.: Power-Aware Microarchitecture: Design and Modeling Challenges for Next-Generation Microprocessors, IEEE MICRO, Nov/Dec, pp.26–44 (2000).
3) D.Brooks, et al.: Wattch: A Framework for Architectural-Level Power Analysis and Optimizations, ISCA'00 (2000).
4) N.Vijaykrishnan, et al.: Energy-Driven Integrated Hardware-Software Optimizations Using SimplePower, ISCA'00 (2000).
5) M.Koegst, et al.: Low Power Design of FSMs by State Assignment and Disabling Self-Loops, EUROMICRO'97 (1997).
6) R.V.Joshi, et al.: Low Power 900 MHz Register File (8 Ports, 32 Words x 64 Bits) in 1.8V, 0.25 μ SOI Technology, VLSI Design'00 (2000).
7) N.Raghavan, et al.: Automatic Insertion of Gated Clocks at Register Transfer Level, VLSI Design'99 (1998).
8) L.Benini, et al.: System-Level Dynamic Power Management, VOLTA'99 (1999).
9) W.Athas: Low-Power VLSI Techniques for Applications in Embedded Computing, VOLTA'99 (1999).
10) T.D.Burd and R.W.Brodersen: Design issues for dynamic voltage scaling, ISLPED'00 (2000).
11) A.G.M.Strollo, et al.: New clock-gating techniques for low-power flip-flops, ISLPED'00 (2000).
12) T.Vinereanu and S.Lidholm: An improved pass transistor synthesis method for low power, high speed CMOS circuits, ISLPED'00 (2000).
13) D.Garrett, et al.: Challenges in clockgating for a low power ASIC methodology, ISLPED'99 (1999).
14) 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).