Vol.2012-OS-120 No.4
2012/2/28

Linuxにおける演算ユニットの電力特性を考慮した細粒度パワーゲーティング制御手法

高橋昭宏[†1] 小林弘明[†1] 坂本龍一[†1]
佐藤未来子[†1] 並木美太郎[†2]

本研究は，省電力プロセッサ “Geyser”に搭載される細粒度パワーゲーティングをOSから制御することにより，プロセッサのリーク電力削減を目指す．従来のパワーゲーティング制御手法はリーク電力の見積もり判定に用いる閾値に単一固定のものを用いていたため，誤判定が発生して省電力効果が低下するケースがあった．本研究では演算ユニットおよび温度ごとに閾値を設定して誤判定を防止することで，省電力性能を向上させる．提案した手法を評価した結果，誤判定を防止できることが確認され，リーク電力を最大15.5%削減することを達成した．また，制御前に比べてリーク電力を最大35.4%削減できた．

Linux Process Scheduler for Power Gating Control with Consideration of the Power Characteristic of Functional Units

Akihiro TAKAHASHI,[†1] Hiroaki KOBAYASHI,[†1]
Ryuichi SAKAMOTO,[†1] Mikiko SATO[†1]
and Mitaro NAMIKI [†2]

This paper describes Linux Process Scheduler to reduce leakage power of a processor by controlling a fine-grain power gating of the processor. The conventional power gating control technique cannot fully reduce electric power because threshold value of PG is fixed and not adapted to the program behavior. This paper aims at the leakage power reduction by changing the PG threshold value according to functional units and temperatures. The algorithm is implemented on Linux Process Scheduler. As the result of this method, Leakage power of functional units was reduced by a maximum of 15.5% compared with the conventional technique. And leakage power was also reduced by a maximum of 35.4% compared with previous method.

1. はじめに

LSIの集積度は年々向上し続けており，それに伴って消費電力が増加している．その一方，環境や節電あるいは発熱面などから計算機の省電力化が要請されている．LSIの消費電力は大きくダイナミック電力とリーク電力に分別できる．ダイナミック電力の削減についてはDVFSなどによる研究がなされている[1)]が，LSIの微細化に伴ってリーク電力の全体の消費電力に占める割合が年々大きくなっている[2)]ため，リーク電力の削減も必要となっている．そこで筆者らは，細粒度パワーゲーティング (Power Gating; PG) 技術を用いてリーク電力を削減する仕組みを導入したプロセッサ “Geyser”の開発を進めている．本Geyserにはソフトウェアからパワーゲーティングを制御するための機構を有している．

パワーゲーティングは，使用されていない回路への電力供給を遮断することによってリーク電力を削減する技術である．Geyserに搭載されている細粒度パワーゲーティングは，演算ユニットの使用の有無をサイクル単位で判断して電力供給・遮断を行う．しかし，短い期間に頻繁にパワーゲーティングを行うと，電力供給・遮断時に発生するオーバシュートの影響により電力が増大してしまい，結果として電力的に不利になる課題がある．

そのため筆者らは，ソフトウェア，特にOSを用いて電力的に不利となるパワーゲーティングを抑制制御する手法の研究を進めてきた．これまで，パワーゲーティングの動作を監視し，この動作状況およびコア温度情報を基にパワーゲーティングの制御を行う手法を提案し，Linuxプロセススケジューラに搭載することでGeyserの電力削減を図ってきた．この中で，さらなる省電力化を図るためのパワーゲーティング制御を行うためには，各演算ユニットの電力特性を十分に考慮しながらプロセスごとにパワーゲーティング制御を行うことが重要であることが明らかとなった．

そこで本研究では，細粒度パワーゲーティングが電力的に有利か不利かを判別するために用いる閾値を，演算ユニットおよび温度などの情報をもとに動的に変更して，実行時の演算ユニットの利用頻度や温度変化に応じた閾値を用いる制御手法を提案する．この閾値は，ベンチマークを用いて電力評価を行い，細粒度パワーゲーティングが電力的に不利になる点を

†1 東京農工大学工学府
Graduate School of Engineering, Tokyo University of Agriculture and Technology
†2 東京農工大学工学研究院
Institute of Engineering, Tokyo University of Agriculture and Technology

調べることで決定できることを示す．最後に，実チップの挙動に近い FPGA に実装された Geyser を用いて，Linux を動作させて提案するパワーゲーティング制御を行い，プロセスの消費電力の削減効果を評価する．

2. Geyser

Geyser は MIPS をベースとしたプロセッサである．アーキテクチャは R3000 に準じるが，細粒度パワーゲーティング機構およびパワーゲーティング制御のためのインタフェースが拡張されていることが特徴である．

2.1 細粒度パワーゲーティング

Geyser は，ALU・SHIFT・MULT・DIV の 4 ユニットに対して細粒度パワーゲーティングを行う．Geyser の細粒度パワーゲーティングは，それぞれのサイクルにおいて命令の実行に必要な演算ユニットのみに電力を供給し，不要な演算ユニットへの電力供給を遮断（スリープ）する．**図 1** にパワーゲーティングを行ったときの電力変化の概略を示す．演算ユニットのスリープおよび起床時には，オーバーシュートにより電力が増大する．

パワーゲーティングにより短期間のスリープが頻発すると，削減される電力よりもオーバシュート電力の方が大きくなり，平均電力は増加する．一定時間スリープ状態を継続することでオーバシュート電力を削減することができ，さらにスリープを継続することで電力を削減することができる．この電力の増減の境界となるスリープ期間を**損益分岐点 (Break-Even Point; BEP)** と呼び，リーク電力の減少を評価する際の指標とする．BEP は演算ユニットの種類および温度に依存する．Geyser 上の BEP は**表 1** に示すとおりである[3)]．

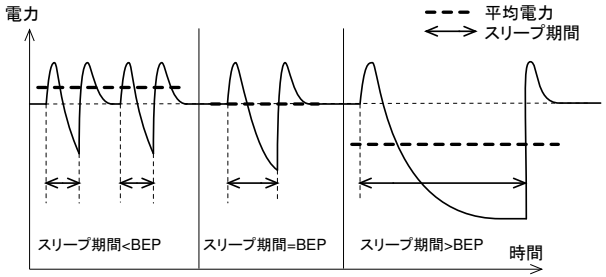

図 1 パワーゲーティングを行ったときの電力推移
Fig. 1 Electric power transition of power gating

2.2 パワーゲーティング制御インタフェース

プロセッサの電力削減には，電力的に不利となる BEP よりも短い期間のスリープを抑制すればよい．Geyser では電力的に不利なパワーゲーティングをソフトウェアから抑制制御する方法として，パワーゲーティング制御レジスタを介して制御する方法がある．パワーゲーティング制御レジスタを用いて，4 ユニットそれぞれに対して**表 2** に示すスリープポリシと呼ばれる動作方法を指定することでパワーゲーティング制御を行うことができる．

3. パワーゲーティング制御手法

本研究では，Linux プロセススケジューラにおいて，電力的に不利となるスリープを抑制するよう表 2 の Geyser のスリープポリシを制御することによって電力削減を行う．本節では，システムの構成を示し，電力見積もりに用いる BEP ミス率，および演算ユニットおよび温度ごとに応じた閾値を設定する方法について述べる．

3.1 全 体 構 成

Linux によるパワーゲーティング制御の全体構成を**図 2** に示す．パワーゲーティングの制御インタフェースとして，パワーゲーティングのスリープ頻度分布を記録するパフォーマンスカウンタ，パワーゲーティング制御レジスタ，温度デバイスが存在する．制御機構は，Linux プロセススケジューラ内の機能として，プロセススケジューリングの際に呼び出される．これにより，プロセスのタイムスライスごとのパワーゲーティング制御を実現する．

表 1 各演算ユニット・温度における BEP（単位：サイクル）
Table 1 BEPs at each units/temperatures

	25 ℃	65 ℃	100 ℃	125 ℃
ALU	124	38	18	12
SHIFT	160	50	22	14
MULT	118	44	44	34
DIV	58	14	6	2

表 2 Geyser のスリープポリシの一覧
Table 2 The list of sleep policies

名称	動作
動的パワーゲーティング	通常の細粒度 PG を行う．
キャッシュミス時スリープ	通常 PG を行わないが，キャッシュミス発生時のみ PG を行う．
常にアクティブ	PG を一切行わない

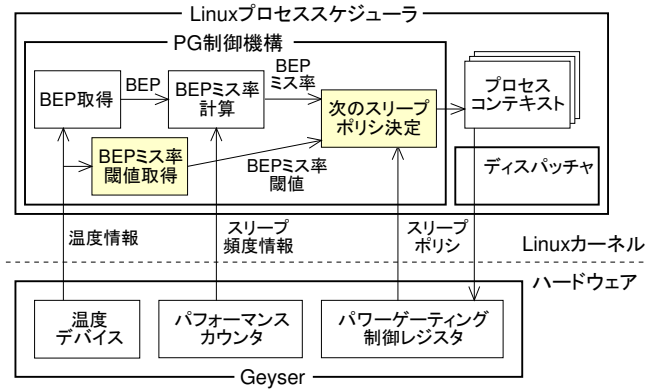

図 2 パワーゲーティング制御の構成
Fig. 2 Composition of the power gating control system

プロセススケジューラが起動されるとまずパワーゲーティング制御機構が呼び出される．パワーゲーティング制御機構では，演算ユニットごとに次に示す手続きでパワーゲーティング制御を行う．まず温度デバイスからコア温度情報を取得し，温度および演算ユニットに対応した BEP を取得する．次にパフォーマンスカウンタからスリープ頻度分布情報を取得し，3.2 節で示す計算方法により **BEP ミス率**を算出する．同時に，Linux 内に保持している閾値リストの中から，制御対象の演算ユニットおよび温度情報に対応する閾値を探索する．この閾値は，演算ユニットや温度によって変化する値であり，制御を行う度に異なるを閾値を用いる．詳しくは 3.4 節で述べる．これらの BEP ミス率と閾値を用いて，3.3 節に示すスリープポリシ決定アルゴリズムによってパワーゲーティングを抑制するようなスリープポリシを選択することで，電力的に不利なパワーゲーティングを抑制する．最後に，新たに決定されたスリープポリシをプロセスコンテキストに書き込み，制御機構は処理を終える．

プロセススケジューラでは，スケジューリング処理およびプロセス切り替えが行われる．プロセスがディスパッチされる際，制御機構でコンテキスト内に格納されたスリープポリシを制御レジスタへ書き込むことによって，パワーゲーティングを制御する．

3.2 BEP ミス率

BEP ミス率とは，電力的に不利となる短期間のスリープがどの程度存在するかを判定する指標であり，全体のスリープのうち BEP を下回るスリープのサイクルの割合を示している．すなわち，BEP ミス率が大きければ，電力的に不利なスリープが頻発していることになる．パワーゲーティング制御では，3.4 節で示す閾値とともに，細粒度パワーゲーティングが電力的に有利か不利かを判別する指標として用いる．

BEP ミス率は，演算ユニットごとのスリープ頻度分布と BEP に基づいて算出する．i サイクルの長さのスリープが発生した回数を x_i とすると，BEP ミス率 $BEP_{MissRate}$ は次の式を用いて計算できる[4]．

$$BEP_{MissRate} = \frac{\sum_{i=1}^{BEP} i \times x_i}{\sum_{i=1}^{\infty} i \times x_i} \quad (1)$$

3.3 スリープポリシ決定処理

パワーゲーティング制御において，スリープポリシを選択するアルゴリズムを，**図 3** に示す．スリープポリシ決定処理では，まず BEP ミス率 $BEP_{MissRate}$ を閾値 Th と比較する．BEP ミス率が閾値以下であれば細粒度パワーゲーティングが電力的に有利であると判断し，細粒度パワーゲーティングを抑制しない「動的パワーゲーティング」ポリシを選択する．一方，BEP ミス率が閾値を上回る場合には，細粒度パワーゲーティングが電力的に不利になると判断し，パワーゲーティングを抑制するようなスリープポリシを選択する．

細粒度パワーゲーティングを抑制する際に選択するスリープポリシは，CPU ストール時間および BEP から決定する．CPU ストール時間 $CPU\_StallCycle$ が BEP の値以上である場合は，CPU ストール中に演算ユニットをスリープすることで電力の削減が期待できると判断し,「キャッシュミス時スリープ」ポリシを選択する．一方，CPU ストール時間が BEP を下回る場合は，CPU ストール中に演算ユニットをスリープしても電力の削減が期待できないと判断し，パワーゲーティングを一切行わない「常にアクティブ」ポリシを選択

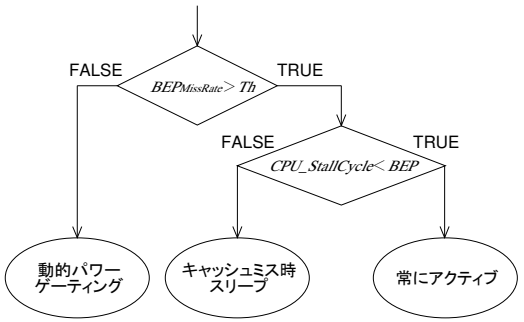

図 3 スリープポリシ決定アルゴリズム
Fig. 3 The algorithm for sleep policy decision

する．ただし，MULT および DIV に関しては「常にアクティブ」ポリシを選択する．

以上の処理をパワーゲーティング制御対象の演算ユニットに適用することによって，各演算ユニットの動作状況に応じたパワーゲーティング制御を実現する．なお BEP ミス率の概念上,「動的パワーゲーティング」ポリシ適用時以外ではパワーゲーティング制御に有効な BEP ミス率を求めることができない．しかし「常にアクティブ」ポリシ適用時については，Geyser の仕様を見直し，使用してないスリープコントローラを活用して細粒度パワーゲーティングをエミュレートさせることで，BEP ミス率を求めることができる．また，残る「キャッシュミス時スリープ」ポリシ適用時は，定期的に「動的パワーゲーティング」ポリシを適用することで BEP ミス率に基づいた制御を行わせることができる．

3.4 パワーゲーティング抑制判別の閾値

電力的に不利となるパワーゲーティングは，BEP ミス率が小さければ全体的に電力が削減できるが，BEP ミス率が大きくなるとリーク電力の増加につながる．そのため，どの程度まで BEP ミス率が大きくなったら細粒度パワーゲーティングが電力的に不利と言えるかの基準が必要である．その基準として閾値を設ける．

パワーゲーティングを行う際の電力特性は，演算ユニットの種類や温度によって異なる．したがって，電力的に不利なパワーゲーティングを判別するための閾値は，それぞれの演算ユニットや温度の電力特性に応じた値になる．閾値を定めるには，それぞれの演算ユニットおよび温度ごとに BEP ミス率とリーク電力との関係を調べ，細粒度パワーゲーティングによってリーク電力が増大し始める点を特定する必要がある．BEP ミス率およびリーク電力は，長短さまざまなスリープの発生頻度に応じて決まるが，これらの発生頻度はプログラムの処理内容によって異なる分布となるため，一般的なモデルを定めることは難しい．

そのため本研究では，事前に予備実験を行うことでスリープ頻度分布を調べ，パワーゲーティングを抑制するべきケースの BEP ミス率を明らかにすることによって閾値を定める．その後，この閾値を用いてパワーゲーティング制御を行う．その手法は，まず細粒度パワーゲーティングを動作させた状態（「動的パワーゲーティング」ポリシ）でベンチマークを動作させ，パフォーマンスカウンタを用いてタイムスライスごとにプロセスのスリープ頻度分布を取得する．次に，各スリープ頻度分布から BEP ミス率およびリーク電力を算出し評価することによって，閾値を決定する．リーク電力の評価は，パワーゲーティングを抑制するスリープポリシ（「キャッシュミス時スリープ」および「常にアクティブ」ポリシ）で動作させた際のリーク電力と比較することにより行う．これにより「動的パワーゲーティング」ポリシが，パワーゲーティングを抑制するポリシに比べて電力的に不利となるケースの BEP ミス率の値を調べることによって，閾値を定めることができる．

3.5 パワーゲーティング抑制時に選択するスリープポリシ

BEP ミス率および閾値からパワーゲーティングを抑制すべきだと判定した場合に，どのようなスリープポリシを適用すればよいかについて述べる．

Geyser 上のパワーゲーティングを抑制するスリープポリシには,「キャッシュミス時スリープ」および「常にアクティブ」の 2 種類がある．これらのスリープポリシの差異は，キャッシュミスによる CPU ストールの際に演算ユニットをスリープさせるか否かである．したがって CPU のストール時間と BEP と比較することによってストール中に演算ユニットをスリープさせるべきかを判別することができる．すなわち CPU のストール時間が BEP より長ければ演算ユニットをスリープすることによって電力の削減が期待できるので「キャッシュミス時スリープ」とし，BEP より短ければ「常にアクティブ」とする．

CPU のストール時間に基づいてスリープポリシを決定するためには，CPU のストール時間が明らかである必要があるが，CPU のストール時間はメモリの読み書き速度に依存する．OS の動作環境が固定であれば，CPU ストール時間を事前に調査しておけば十分である．また OS の動作環境が可変である場合でもキャッシュミス発生時のパワーゲーティングの挙動を調べることによって，CPU ストール時間を求めることも可能である．本研究では動作環境が既知であり，またストール時間を動的に求めることが目的ではないので，CPU ストール時間を事前に求める．ただし，演算ユニット MULT および DIV に関してはハードウェア上の制約により「キャッシュミス時スリープ」ポリシが設定できないため，この 2 ユニットについては，常に「常にアクティブ」ポリシを適用する．

4. 予 備 実 験

本節では，予備実験としてベンチマークを Geyser 上で動作させ，電力的に不利なパワーゲーティングとなるケースを判別するための閾値，および CPU ストール時間を求める．

4.1 評 価 環 境

予備実験および本実験における評価環境を**表 3** に示す．実行基盤として木村らが Xilinx ML501 上に構築した Geyser 評価基盤[5] を用いた．評価に用いた Geyser は,「常にアクティブ」ポリシ適用時にも細粒度パワーゲーティングの動作をエミュレートして BEP ミス率を求められるよう修正した．本 Geyser 上でパワーゲーティング制御機構を搭載した Linux を動作させて評価を行った．なお，本研究では Geyser を温度デバイスが存在しない FPGA 環境上で動作させるため，温度については固定値としてエミュレーションを行った．

表 3 評価環境の一覧
Table 3 The list of the evaluation environment

名称	製品名など
プラットフォーム	Xilinx FPGA 評価ボード Virtex-5 LX FPGA ML501
評価用 CPU	Geyser-3 (v1.703)
評価 OS	Linux 2.6.35.11
クロスコンパイラ	GNU Compiler Collection (gcc) 4.5.1

評価用ベンチマークとして，クイックソート・行列演算・Dhrystone・ダイクストラ・Blowfish・ビットカウント・FFT・Whetstone の 8 種類のプログラムを用いた．これらのベンチマークを Linux 上でプロセスとして動作させ，パフォーマンスカウンタを用いて細粒度パワーゲーティングの動作を計測し，電力評価を行った．

4.2 電力見積もりの計算方法

見積もりに用いた電力の計算方法について述べる．本評価基盤ではリーク電力は直接計測することができないので，本研究では Geyser のパフォーマンスカウンタのスリープ頻度分布から以下のように計算する．

スリープ時の平均リーク電力は，スリープする期間によってリーク電力が異なるため，まずそれぞれの長さのスリープごとに集計する．スリープ期間 N サイクルの平均電力 $\overline{P_N}$ は，先行研究[6] により求められている．これにスリープを行った頻度（サイクル数）に応じて加重平均を取ることで，スリープ時の平均リーク電力を求めることができる．すなわちスリープ時平均リーク電力 $\overline{P_{sleep}}$ は，スリープ状態の総サイクル数を T_{sleep}，N サイクルスリープの出現頻度を R_N とすると，次式により求められる．

$$\overline{P_{sleep}} = \frac{\sum_N (\overline{P_N} \times R_N \times N)}{T_{sleep}} \tag{2}$$

一方，アクティブ時の平均リーク電力 $\overline{P_{active}}$ は，直接計算することができない．しかし BEP を求める先行研究[3] ではアクティブ時電力を基準に BEP を算出していたことから，BEP と同じサイクル数だけスリープする際の平均リーク電力がアクティブ時の平均リーク電力と等しくなることが期待できる．そのため，次の仮定が成り立つ．

$$\overline{P_{active}} = \overline{P_{i=bep}} \tag{3}$$

また，全体の平均リーク電力 $\overline{P_{all}}$ は，スリープ時の平均リーク電力とアクティブ時の平均リーク電力の加重平均であり，アクティブ状態の総サイクル数を T_{active} とすると，次式により求められる．

$$\overline{P_{all}} = \frac{\overline{P_{sleep}} \times T_{sleep} + \overline{P_{active}} \times T_{active}}{T_{sleep} + T_{active}} \tag{4}$$

4.3 パワーゲーティング抑制判別に用いる閾値の導出

電力的に不利なパワーゲーティングを判別するための閾値を求めた．細粒度パワーゲーティングを動作させた状態（「動的パワーゲーティング」ポリシ）で 4.1 節に挙げた 8 種類のベンチマークを実行し，各ベンチマークのタイムスライスごとにスリープ頻度分布を取得して BEP ミス率および平均リーク電力を算出した．次にそれぞれのタイムスライスについて平均リーク電力をパワーゲーティング抑制時の平均リーク電力と比較することでパワーゲーティングが電力的に有利か不利かを分別し，それぞれの境界となる BEP ミス率を調べた．

25 ℃環境下における ALU・SHIFT・MULT・DIV の各ユニットの BEP ミス率・平均リーク電力の関係をそれぞれ**図 4**，**図 5**，**図 6**，**図 7** に示す．各サンプル点は 8 種類のベンチマークのタイムスライスごとのリーク電力およびそのときの BEP ミス率に対応した座標に打点されている．

ALU については，細粒度パワーゲーティングが電力的に有利となるサンプルが BEP ミス率 0-26.0%の範囲に分布していた一方，不利となるサンプルが 39.3-100%の範囲に分布していたため，32.6%を境とすることでそれぞれのサンプルを区別することができた．SHIFT については，細粒度パワーゲーティングが電力的に有利となるサンプルが BEP ミス率 0-22.7%の範囲に分布していた一方，不利となるサンプルが 24.1-100%の範囲に分布していたため，23.5%を境とすることでそれぞれのサンプルを区別することができた．したがってこれらの値を閾値として採用することで，細粒度パワーゲーティングの有利・不利を判別できると期待できる．

MULT および DIV については，サンプルの分布が偏っておりデータのない中間部分があった．サンプルはおおむね線形に分布していることから直線を用いて近似できると考えられるため，データのない部分については近似直線を用いて推定することとした．この近似直線上において，細粒度パワーゲーティングの有利・不利の境界となるリーク電力（すなわちアクティブ時のリーク電力）に対応する BEP ミス率を求め，この BEP ミス率を閾値とした．この閾値を用いることでサンプルを区別することができたため，この閾値によって細粒度パワーゲーティングの有利・不利を判別できると期待できる．

65 ℃，100 ℃および 125 ℃の環境下においても同様に計算を行いそれぞれ閾値を求めた．その結果を**表 4** に示す．これらの閾値を用いてパワーゲーティング制御を行うことで，電力的不利なパワーゲーティングを正確に判別し，電力を削減することが期待できる．

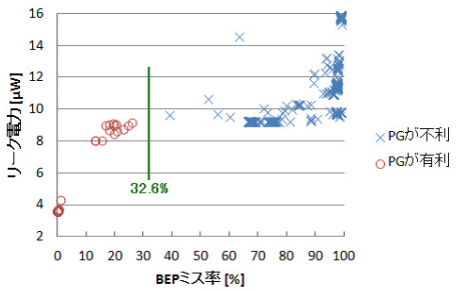

図 4 ALU の BEP ミス率/リーク電力分布
Fig. 4 The BEP-miss-rate/leakage-power distribution of ALU unit

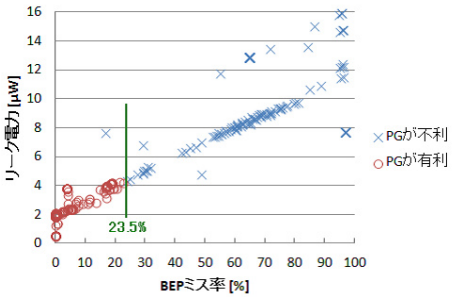

図 5 SHIFT の BEP ミス率/リーク電力分布
Fig. 5 The BEP-miss-rate/leakage-power distribution of SHIFT unit

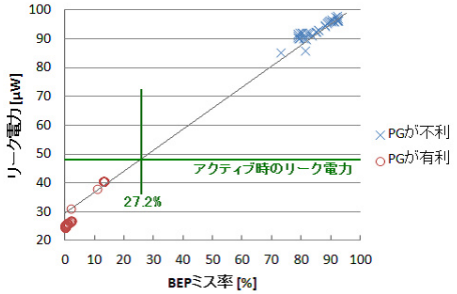

図 6 MULT の BEP ミス率/リーク電力分布
Fig. 6 The BEP-miss-rate/leakage-power distribution of MULT unit

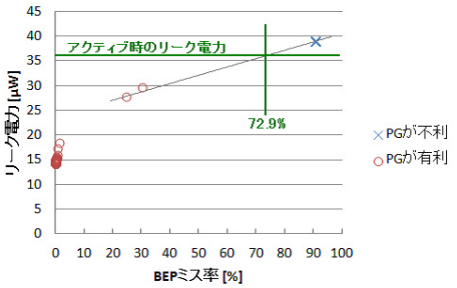

図 7 DIV の BEP ミス率/リーク電力分布
Fig. 7 The BEP-miss-rate/leakage-power distribution of DIV unit

4.4 CPU のストール時間の導出

キャッシュミス時に演算ユニットをスリープするかどうかの判断に用いる CPU のストール時間は，CPU のストール時間をパフォーマンスカウンタを用いて事前に求めた．これは動作環境が固定環境であるため，動的に求める必要がないためである．ただし，本節で示した方法を Linux 上に実現することで，動的に求めることも可能である．

スリープポリシを「キャッシュミス時スリープ」に固定して，キャッシュミス時のパワーゲーティングの挙動を調べた．このときのスリープ頻度分布を**図 8** に示す．スリープ時間

表 4 演算ユニットおよび温度ごとの閾値 [%]
Table 4 Thresholds at each units/temperatures

	25 ℃	65 ℃	100 ℃	125 ℃
ALU	32.6	2.2	2.0	1.9
SHIFT	23.5	31.8	49.7	35.8
MULT	27.2	35.2	60.5	76.7
DIV	72.9	9.6	7.2	–

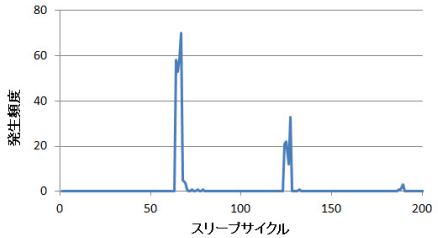

図 8 キャッシュミス時スリープでのスリープ頻度分布
Fig. 8 A sleep distribution by Sleep-by-Cache-Mistake Policy

が 64 サイクル，125 サイクル，190 サイクルのスリープが頻出しており，これが CPU のストール時間に相当する．このうち，パワーゲーティングによって最も影響を受けるものが，スリープ時間が短く出現頻度の大きい 64 サイクルの CPU ストール時であると考えられる．そのため本環境での CPU ストール時間は，64 サイクルとするのが適当である．

5. 評　　価

予備実験の結果を受けて，電力的に不利なパワーゲーティングを判別するための閾値に表 4 を用いて検証し，パワーゲーティング制御無しおよび従来手法に比べ，提案手法がどの程度電力を削減することができるか評価した．従来手法はどの演算ユニットおよび温度の場合でも閾値が変化しない固定値を用いたもので，その閾値は ALU の動作に合わせた 8%のものであったが，提案手法は演算ユニットおよび温度に応じて閾値を変更する．評価環境およびベンチマークは，4.1 節で示したものと同様のものを用いた．

5.1 選択されるスリープポリシ

まず，BEP ミス率と閾値を用いて細粒度パワーゲーティングを抑制できるか検証を行った．例として SHIFT での判定結果を**表 5** に示す．各ベンチマークにおいて正しい判定が行

表 5 SHIFT のパワーゲーティング判定の成否
Table 5 Power gating judgements of SHIFT unit based on thresholds

	従来手法による判定				提案手法による判定			
	25 ℃	65 ℃	100 ℃	125 ℃	25 ℃	65 ℃	100 ℃	125 ℃
クイックソート	**miss**	hit	hit	hit	hit	hit	hit	hit
行列演算	hit	hit	hit	hit	hit	hit	hit	hit
Dhrystone	hit	hit	hit	hit	hit	hit	hit	hit
ダイクストラ	hit	**miss**	**miss**	hit	hit	**miss**	**miss**	hit
Blowfish	hit	hit	hit	**miss**	hit	hit	hit	**miss**
ビットカウント	hit	hit	hit	hit	hit	hit	hit	hit
FFT	**miss**	**miss**	hit	hit	hit	hit	hit	hit
Whetstone	hit	hit	**miss**	**miss**	hit	hit	hit	hit

われていたケースを『hit』，誤判定が発生していたケースを『miss』で示す．

従来手法では，クイックソート・ダイクストラ・Blowfish・FFT・Whetstone において誤判定が発生していた．一方，提案手法では，クイックソート・FFT・Whetstone の誤判定を防止し，正しい判定を行えることを実現した．これにより，誤判定が発生するケースを 62.5%削減できた．しかし，ダイクストラおよび Blowfish では誤判定が残った．例えば Blowfish（125 ℃）については BEP ミス率が 62%前後となっており，本来パワーゲーティングを抑制すべきでないにもかかわらず抑制すべきとする誤判定がなされていた．

従来手法の閾値では，ALU の動作に合わせて 8%の固定値を用いていた．これは表 4 の ALU の閾値に近い値であり，そのため従来手法を用いても ALU においては誤判定はなかった．しかし一方で SHIFT の閾値は大きく異なっており，SHIFT の電力特性に対応した閾値を用いたことで，表 5 に示したように従来手法に比べて誤判定を削減することにつながったものと考えられる．したがって，閾値を演算ユニットおよび温度に応じて変更する本提案手法は有効であったと言える．

一方，誤判定が残った問題については，リーク電力を BEP ミス率から求める都合上，見積もりに多少の誤差が発生するのはやむを得ない．Blowfish の例で言えば BEP ミス率の閾値を 70%程度にすることで誤判定を防止することができるが，仮に閾値を 70%とした場合，リーク電力の割に BEP ミス率が小さめに出るビットカウントにおいて電力の増加が懸念される．BEP ミス率の閾値の決定にあたっては，判定精度の向上のために今後も更に検討する必要があると考えられる．

続いて，BEP と CPU のストール時間から判断する細粒度パワーゲーティングを抑制した際に適用すべきスリープポリシの判定は，全てのベンチマークにおいて，25 ℃の環境で「常にアクティブ」ポリシ，65 ℃以上の環境で「キャッシュミス時スリープ」ポリシが選択された．各温度環境下での「キャッシュミス時スリープ」ポリシ適用時の電力は,「常にアクティブ」ポリシ時に比べて，25 ℃の環境で 3.3%増加し，65 ℃以上の環境で 2.9-11.7%減少した．したがって，25 ℃環境下のみ「常にアクティブ」ポリシを適用するのは適切な判断であり，CPU のストール時間と BEP を用いて判定する本手法は有効であったと考えられる．

5.2 ベンチマークの平均リーク電力

次に，パワーゲーティング制御を行うことによってどの程度リーク電力が削減できるか検証を行った．25 ℃および 65 ℃環境下での SHIFT のリーク電力を**図 9** に，全演算ユニットのリーク電力を**図 10** に示す．いずれもパワーゲーティング制御無し時（常にパワーゲーティングを行った状態）を 100%とした電力比を示している．

SHIFT のリーク電力は，パワーゲーティング制御無しに比べて平均で 17.4%削減できた．クイックソート・行列演算・Dhrystone・FFT については最大 5.6%の削減にとどまったが，これらのベンチマークは細粒度パワーゲーティングを行うことが適切であったためである．これ以外のベンチマークでは電力を大きく削減することができ，とくにビットカウント（25 ℃）では 71.7%の削減を達成した．また，従来手法との比較では，リーク電力を平均 9.2%削減できた．とくに従来手法で誤判定が発生していたクイックソートおよび FFT については，3.4-65.6%の削減を達成した．これは，従来手法で誤判定により電力が 3.5-65.7%増加していたものを，提案手法によって正しい判定ができるようになったことで電力の増加が防止できたためである．またこれ以外でも，従来手法では行列演算（25 ℃）および Whetstone（65 ℃）で電力が増加していたものを，提案手法により電力増加を抑制することができた．

全演算ユニットのリーク電力では，パワーゲーティング制御無しに比べて平均 7.3%，行列演算（25 ℃）については 35.4%のリーク電力を削減することができた．行列演算で電力が大きく削減できたのは，MULT に対して制御が正しく行われ，電力的に不利なパワーゲーティングを抑制されたためである．また，従来手法と比較しても 25 ℃下の全てのベンチマークで平均 2.2%，最大 8.1%のリーク電力を削減することができ，65 ℃の環境でも行列演算および FFT で 4.1-15.5%の電力削減を達成した．従来手法では電力が増加していたクイックソート（25 ℃）および FFT（65 ℃）についても，従来手法より電力を削減することができた．

提案手法によって，従来手法にあった「制御を行うことで電力が増加する問題」を解決することができた．これは提案手法により誤判定を防止することができたためであると考えられる．また，それ以外のベンチマークでも，とりわけ 25 ℃環境下において軒並み電力を

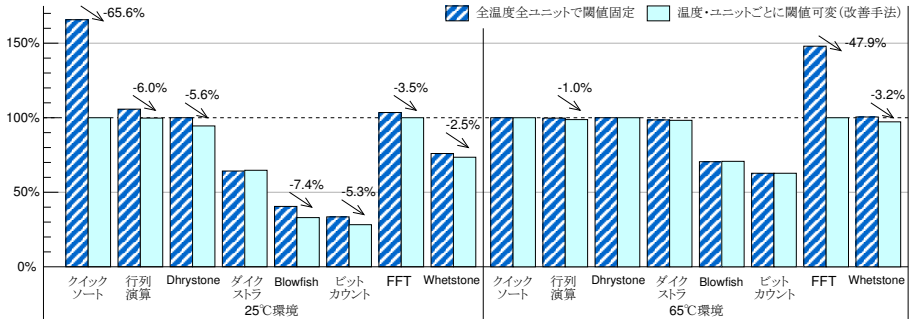

図 9 SHIFT の平均リーク電力
Fig. 9 Mean leakage power of SHIFT unit

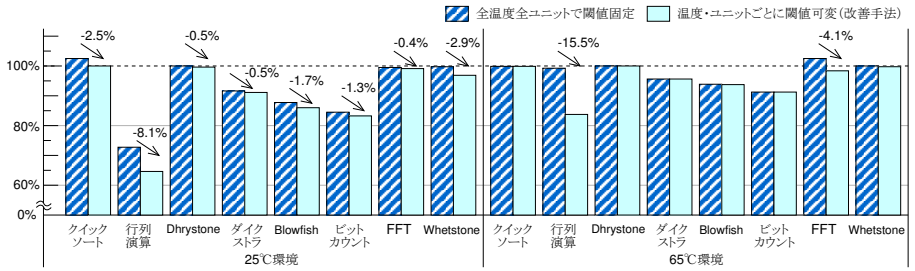

図 10 全演算ユニットの平均リーク電力
Fig. 10 Mean leakage power of all functional units

削減できた．これは従来手法ではパワーゲーティングを抑制した際にプロセスの挙動が計測できず，一時的にフィードバック制御が行えなくなる問題があったためである．提案手法では，細粒度パワーゲーティングを行っていない場合でもプロセスの挙動を計測できるようになったことにより，BEP ミス率に基づいた制御を行えるようになったためである．以上より本提案手法では，従来手法に比べて演算ユニットのリーク電力を削減することができ，リーク電力削減に有効な手法であると言える．

6. おわりに

本研究では，Geyser の細粒度パワーゲーティング制御手法を提案し，電力見積もり時の誤判定を削減することを実現した．提案手法である電力的に不利なパワーゲーティングを判別するための閾値を演算ユニットおよび温度ごとに変更して，演算ユニットおよび温度ごとの電力特性に応じた閾値とすることは有効であった．また本提案手法により実際にベンチマークにより計測を行ったところ，リーク電力をパワーゲーティング制御無しと比べて最大 35.4%，従来手法に比べて 15.5%削減することを達成した．以上より，このパワーゲーティング制御手法を提案することによって電力的に不利なパワーゲーティングを抑制し，プロセッサのリーク電力削減を達成することができた．

今後の課題として，電力的に不利なパワーゲーティングを判別する精度を向上させることがある．現行の手法でも一定の成果があったが，誤判定があるケースがあった．また,「キャッシュミス時スリープ」適用時は依然としてフィードバック制御ができない問題がある．今後はこれらの問題を，閾値による誤判定を防止するよう判定精度を向上させること，および「キャッシュミス時スリープ」ポリシ時でもプロセスの動作状況を計測できるように例えば演算命令の出現頻度などの指標を用いることで改善する必要がある．

謝辞 本研究は, 科学技術振興機構「JST」の戦略的創造研究推進事業「CRSET」における研究領域「情報システムの超低電力化を目指した技術革新と統合化技術」の研究課題「革新的電源制御による次世代超低電力高性能システム LSI の研究」によるものである．

参 考 文 献

1) 宮川大輔，石川裕：プロセス単位電力制御機構の設計と実装，情報処理学会研究報告, 2005-OS-99, No.29, pp.167-168 (2005).
2) Yan Meng, Timothy Sherwood, Ryan Kastner: Exploring the limits of leakage power reduction in caches, ACM Transactions on Architecture and Code Optimization, Vol.2, Issue 3 (2005).
3) 白井利明，香嶋俊裕，武田清大，中田光貴，宇佐美公良，長谷川揚平，関直臣，天野英晴：ランタイムパワーゲーティングを適用した MIPS R3000 プロセッサの実装設計と評価，情報処理学会研究報告, 2008-SLDM-133, No.8, pp.43-48 (2008).
4) 砂田徹也，木村一樹，近藤正章，天野英晴，宇佐美公良，中村宏，並木美太郎：細粒度パワーゲーティングを制御する OS の資源管理方式，情報処理学会研究報告, 2010-OS-114, No.8, pp.1-8 (2010).
5) 木村一樹，砂田徹也，長井智英，関直臣，近藤正章，天野英晴，宇佐美公良，中村宏，並木美太郎：省電力 MIPS プロセッサコア評価のための計算機システムの FPGA による試作，情報処理学会研究報告, 2009-OS-111, No.34, pp.1-8 (2009).
6) 中田光貴，白井利明，香嶋俊裕，武田清大，宇佐美公良，関直臣，長谷川揚平，天野英晴：ランタイムパワーゲーティングを適用した回路での検証環境と電力見積もり手法の構築，情報処理学会研究報告, 2008-SLDM-133, No.7, pp.37-42 (2008).