Vol.2011-ARC-195 No.31
Vol.2011-OS-117 No.31
2011/4/14

細粒度パワーゲーティング制御による省電力化を行うLinuxプロセススケジューラの試作

高橋 昭宏[†1] 茂木 勇[†1]
佐藤 未来子[†1] 並木 美太郎[†1]

本研究では，汎用のオペレーティングシステムであるLinuxを用いてプロセッサに搭載されている細粒度パワーゲーティングを制御することで，プロセッサのリーク電力の削減を行った．Linuxによるパワーゲーティング制御は，Linuxのプロセススケジューラに先行研究で省電力効果を確認したパワーゲーティング制御アルゴリズムを組み込むことで実現した．パワーゲーティング制御アルゴリズムは，コア温度およびこれまでのパワーゲーティング情報を基に制御を行うアルゴリズムである．評価の結果，パワーゲーティングを制御しない場合に比べて，スリープ時平均リーク電力を平均8.1%削減することができた．また，プロセスのエネルギー遅延積についても常温において平均8.7%削減することができた．

A Prototype of Linux Process Scheduler for the Power Gating Control

Akihiro TAKAHASHI,[†1] Isamu MOGI,[†1]
Mikiko SATO[†1] and Mitaro NAMIKI [†1]

This paper describes a reduction of leakage power of a processor by controlling a fine-grain power gating of the processor by Linux. Power gating control on Linux was realized by including a power gating control algorithm, whose power reduce effect was confirmed by a preceding study, in Linux process scheduler. This power gating control algorithm is an algorithm controlling the power gating based on a temperature in the core and current power gating information. As a result of an experiment, leakage leak power at sleep was able to to be reduced by 8.1% on average compared to no power gating control. And energy delay product of processes was also able to be reduced by 8.7% on average at the normal temperature.

1. はじめに

LSIの性能は年を追うごとに向上し続けており，それに伴って消費電力や発熱量も増大している．しかし一方で，近年は地球環境の面から省エネルギーが叫ばれており，LSIも性能の向上だけでなく消費電力の低電力化が必要とされている．LSIで消費される電力には，大きくダイナミック電力とリーク電力に分けられるが，とくにリーク電力がLSIの微細化に伴って増大し，年々LSI全体の消費電力に占める割合が大きくなってきている．

筆者らは，このLSIのリーク電力に着目し，リーク電力の削減をすることによって省電力化を実現するプロセッサ『Geyser』の開発を進めている．Geyserは，細粒度パワーゲーティング技術によりプロセッサの省電力化を行う機構を有している．また，ソフトウェアから細粒度パワーゲーティングを制御する機構を有している．

細粒度パワーゲーティングは，短期間でユニットのオン・オフを繰り返すと，スイッチング時に発生する電力のオーバーシュートが多くなり電力的に不利になる．そのため筆者らは，この短い期間のスイッチングをソフトウェアで検出して，ソフトウェアにより制御を行う手法を提案している．これまで，筆者らの研究室では，コア温度およびこれまでのパワーゲーティング情報を基にパワーゲーティングの制御を行うアルゴリズムを提案し，組み込みオペレーティングシステムに搭載することでVerilogシミュレーションによってGeyserの電力削減を実現した．しかし，Linuxなどの汎用のオペレーティングシステムを用いて制御を行った場合の電力削減効果は明らかでなく，OSアーキテクチャの違いによる削減効果について検証する必要がある．また，先行研究での電力評価はシミュレーション環境にとどまっており，ハードウェアのコアでの電力削減効果についても明らかにする必要がある．

2. 本研究の目的

本研究では，汎用のオペレーティングシステムであるLinuxを用いてGeyserのパワーゲーティング制御を実現して，Linux環境下における消費電力の削減を目指す．また，実チップに近いFPGAに実装されたGeyserを用いて，ハードウェア上で消費電力の削減ができることを示す．これらの実現のために，Linux上でのアプリケーションによる電力評価を行う基盤を確立する．これによりFPGA上でLinuxおよびLinuxアプリケーションを

†1 東京農工大学
Tokyo University of Agriculture and Technology

Vol.2011-ARC-195 No.31
Vol.2011-OS-117 No.31
2011/4/14

動作させて，ユニットのスリープ時の消費電力や消費エネルギーの削減効果を評価する．

3. Geyser の細粒度パワーゲーティング

Geyser には，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機構を搭載している．Geyser のパワーゲーティングは，命令パイプラインにある命令を調べ，必要な演算ユニットだけ電力を供給し，ユニットを使用しないサイクルではユニットへの電力供給を遮断することにより省電力化を図る．**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要を示す．パワーゲーティングによりユニットがスリープしてもただちに消費電力が低下するわけではなく，一旦上昇したのちに過渡的に減少する．またユニットのウエイクアップの際にも，一時的に大きな電流が流れて消費電力が増加する．

短期間でユニットのオン・オフを繰り返すと，スリープにより削減できる電力よりもオーバシュート電力の方が多くなり，平均電力は増加する．一方でスリープ期間が長ければ，オーバシュート以上の電力を削減できる．消費電力を削減できるかできないかを判断するための指標として損益分岐点（Break-Even Point, BEP）がある．BEP は，スイッチング時に発生するオーバシュートによる電力を回収するために必要なスリープ期間であり，ユニットの種類およびや温度に依存する値である．

プロセッサの省電力化を実現するためには，BEP よりも短い期間しかスリープできないときはスリープを行わず，BEP よりも長い期間スリープできるときにスリープを行うようにすればよい．しかし，これらをすべてハードウェア単体で行うには限界があるため，より効果的にパワーゲーティングを行うため，オペレーティングシステムによる制御を行う．

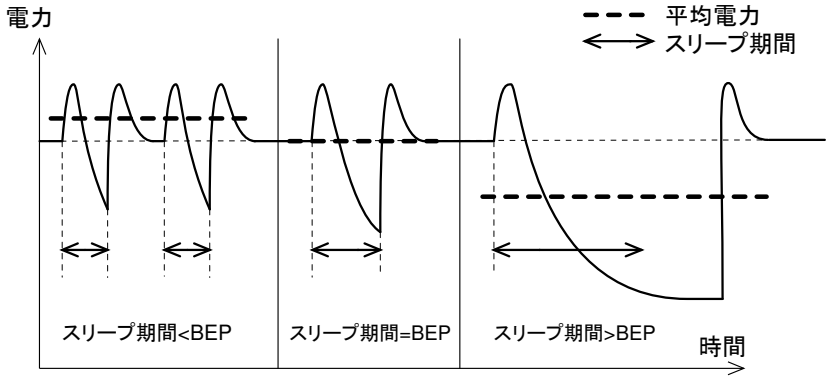

図 1 パワーゲーティングを行ったときの電力推移

3.1 パワーゲーティング制御インタフェース

オペレーティングシステムからパワーゲーティングを制御するためのインタフェースとして，パワーゲーティング制御レジスタがある．パワーゲーティング制御レジスタを用いて，それぞれのユニットに対して次の動作モードを指定することで制御を行う．

(1) **動的パワーゲーティング**：通常スリープにしておき，演算器を使用するサイクルのみ電力を供給する．

(2) **キャッシュミス時スリープ**：キャッシュミスが発生して主記憶にデータをとりに行く操作が発生する場合にスリープを行う．

(3) **常にアクティブ**：を常にアクティブ状態にし，パワーゲーティングを行わない．

なお，4 ユニットそれぞれの動作モードの設定ひとまとまりを「スリープポリシ」と定義する．オペレーティングシステムは，各ユニットの動作モードを決定するとき以外は，スリープポリシを単位として扱う．

4. パワーゲーティング制御手法

ソフトウェアによるパワーゲーティング制御は，電力的に不利となる短い期間スリープするようなパワーゲーティングを抑制し，これにより電力削減を行う．本節では，制御の指標に用いる BEP ミス率を定義し，これを用いた制御アルゴリズムを説明する．

4.1 BEP ミス率

ソフトウェアによるパワーゲーティング制御では，複数のパワーゲーティングに対してまとめて制御を行う．この際，電力的に不利となる短い期間のスリープがどの程度存在するのかの指標として，BEP ミス率を定義する．

BEP ミス率とは，総スリープサイクル数のうち，BEP を越えないスリープ期間のサイクル数の割合であり，この値は温度によって異なる．BEP ミス率は，ユニットごとのスリープ頻度情報と BEP に基づいて算出する．総スリープサイクルを $t_{sleepAll}$，i サイクルのスリープサイクルを t_i とすると，BEP ミス率 $BEP_{MissRate}$ は次の式を用いて計算できる[1)]．

$$T = \{t_i | i < BEP\}$$
$$BEP_{MissRate} = \frac{\sum_{x \in T} x}{t_{sleepAll}} \quad (1)$$

4.2 パワーゲーティング制御アルゴリズム

本研究で搭載するパワーゲーティング制御に用いるアルゴリズムについて述べる[1)]．新し

Vol.2011-ARC-195 No.31
Vol.2011-OS-117 No.31
2011/4/14

いスリープポリシは，BEP ミス率 $BEP_{MissRate}$ および現在のスリープポリシを入力として用いる．BEP ミス率が大きくなると電力的に不利となるパワーゲーティングが多くなったと判断し，今までより粗粒度でパワーゲーティングを行うよう動作モードを変更して，短いスリープ期間のパワーゲーティングを抑制するよう制御する．

具体的には，BEP ミス率を 3 種類の閾値 Th_{CMiss}，Th_{PG}，Th_{NoPG} と比較して，**図 2** のように現在の動作モードを維持するか別の動作モードに遷移するかを決定する．

なお，粗粒度に動作モードへ制御するとスリープ回数が減少するため，スリープ頻度情報の情報量が減少する特徴がある．そのため，上述のアルゴリズムでは，判断ミスが発生して複数回制御を行ってもひとつの動作モードに収束しない場合がある．これを回避する方法として，直近に決定した動作モードを記憶しておき，ひとつの動作モードに収束しないと判断した場合に動作モードに固定するよう制御する．具体的には以下の場合である．

- 動作モードが「動的パワーゲーティング」→「キャッシュミス時スリープ」→「動的パワーゲーティング」→…のように振動する場合,「キャッシュミス時スリープ」に固定.
- 動作モードが「動的パワーゲーティング」→「キャッシュミス時スリープ」→「常にアクティブ」→「動的パワーゲーティング」→…のように振動する場合,「常にアクティブ」に固定.

5. Linux でのパワーゲーティング制御方法

本研究で実現するパワーゲーティング制御は，前に述べた BEP ミス率，およびスリープポリシ決定アルゴリズムを用いて行う．Linux によるパワーゲーティング制御を実現するため，パワーゲーティング制御機構を Linux のプロセススケジューラに組み込む．これにより，スケジューリングが実行される度にパワーゲーティング制御を行うことを実現する．

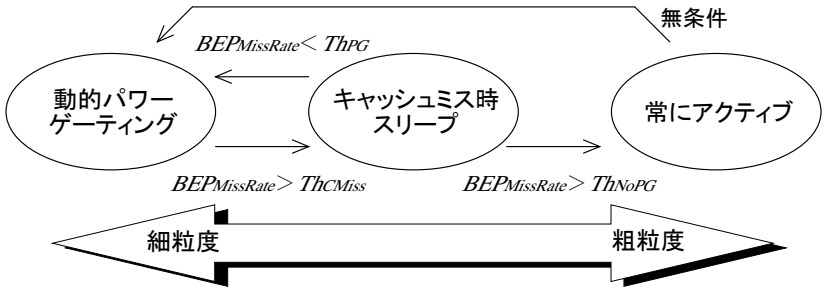

図 2 スリープポリシ決定アルゴリズムの概要

パワーゲーティング制御機構を Linux のプロセススケジューラに組み込む理由として，プロセスごとにパワーゲーティングを制御することが挙げられる．プロセスはその処理内容によって使用する命令や特徴に偏りがあり，ユニットの使用頻度はプロセスによって変化する．したがって，最適となるスリープポリシはプロセスにより異なる．そのため，プロセスごとにパワーゲーティング制御を適用して，それぞれのプロセスの特徴に応じたパワーゲーティングを行うことで，プロセッサ全体の電力を削減することが実現できる．

5.1 構　　成

Linux によるパワーゲーティング制御の構成を**図 3** に示す．パワーゲーティング制御では，入力として直近のパワーゲーティングの動作状況であるスリープ頻度情報，そのときのスリープポリシ，およびコアの温度情報を用いる．

Geyser にはパワーゲーティング機構とこれを制御するためのパワーゲーティング制御レジスタ，パワーゲーティングの動作状況を調べるパフォーマンスカウンタや，温度デバイスが備わっている．これらのデバイスの制御や情報取得処理などは，オペレーティングシステムにより行われる．なお，本研究では Geyser は FPGA 環境で実現するため温度デバイスが存在しないので，温度デバイスについてはエミュレーションを行う．

Linux 側では，カーネル内のプロセススケジューラにパワーゲーティング制御機構を搭載する．このパワーゲーティング制御機構は，パフォーマンスカウンタからスリープ頻度情報

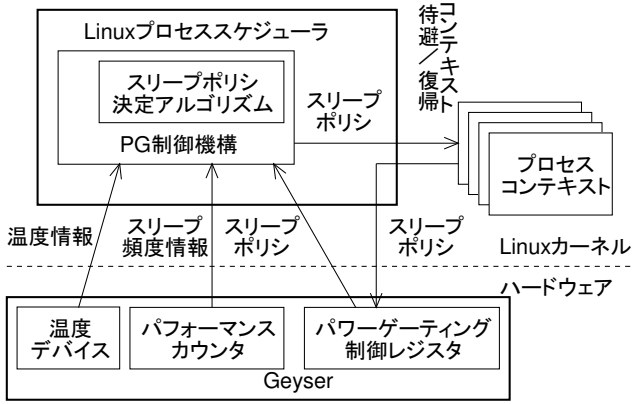

図 3 Linux でのパワーゲーティング制御の構成

Vol.2011-ARC-195 No.31
Vol.2011-OS-117 No.31
2011/4/14

を，温度デバイスから温度情報を読み込み，これらの情報を基にスリープポリシを決定することで，パワーゲーティング制御を行う．また，プロセスごとにパワーゲーティングを行うため，それぞれのプロセスコンテキストにスリープポリシを待避させる領域を確保し，実行中でないプロセスのスリープポリシをコンテキストに待避させる．

5.2 パワーゲーティング制御の流れ

Linux によるパワーゲーティング制御の流れを**図 4** に示す．パワーゲーティング制御機構を Linux のプロセススケジューラに組み込むことでパワーゲーティング制御を実現するために，スケジューラ内にパワーゲーティング制御機構を実行する部分を追加する．図 4 中に示した薄灰色の部分が本件研究において追加した処理である．なお，パワーゲーティング制御の実現にあたって，割り込みハンドラ，Linux プロセススケジューラのスケジューリングアルゴリズム，および既存のコンテキストスイッチ処理などには変更を加えない．

まず，タイマ割り込みやシステムコールなどを契機に再スケジューリングが行われる際，

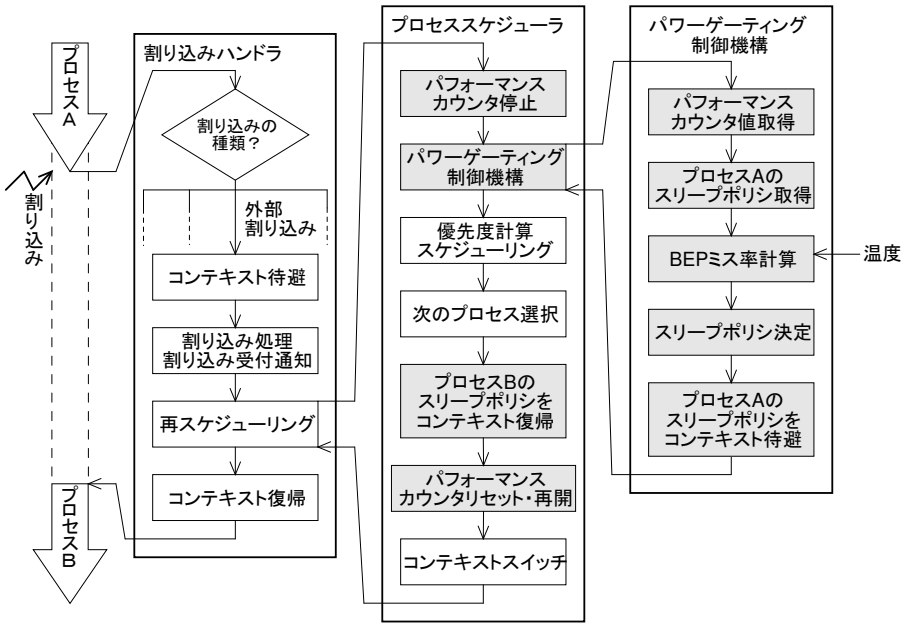

図 4 Linux でのパワーゲーティング制御の流れ

プロセススケジューラが起動される．スケジューラは，起動されると最初にパフォーマンスカウンタを停止させ，パワーゲーティング制御機構を呼び出す．パワーゲーティング制御機構では，今まで実行してきたプロセスに対してパワーゲーティング制御を行う．パワーゲーティング制御機構は，パワーゲーティングのスリープ頻度情報および温度情報を取得し，これらの情報から新しいスリープポリシを決定する．決定されたスリープポリシは，後にスケジューリングを行ってプロセスを切り替えることから，直接パワーゲーティング制御レジスタには適用せず，今まで実行してきたプロセスのコンテキストに待避させる．

パワーゲーティング制御機構による制御が終了すると，スケジューラはプロセスのスケジューリングを実行し，次に割り当てるプロセスを決定する．スケジューラは，汎用レジスタ類のコンテキストスイッチの直前で，次に割り当てるプロセスのコンテキスト内に待避させていたスリープポリシを読み込み，パワーゲーティング制御レジスタに適用する．最後に，パフォーマンスカウンタのリセットを行って，次のプロセスに対するパワーゲーティング動作状況の記録を開始させ，汎用レジスタ類のコンテキストスイッチを行って処理を終える．

以上がプロセススケジューラによるパワーゲーティング制御の流れである．パワーゲーティング制御機構をプロセススケジューラ内で行うことで，プロセスとして扱われるアプリケーションやデーモンなどはすべてパワーゲーティング制御の対象となり，各アプリケーションやデーモンに対し定期的にパワーゲーティング制御を行うことができる．

5.3 パワーゲーティング制御機構

パワーゲーティング制御機構は，パフォーマンスカウンタのスリープ頻度情報，温度情報，および各演算ユニットの動作モード（スリープポリシ）を入力として各ユニットの新しい動作モードの計算を行い，新しいスリープポリシを決定する．パフォーマンスカウンタの制御処理やスリープポリシの適用処理のタイミングは，本機構の外で記述する必要がある

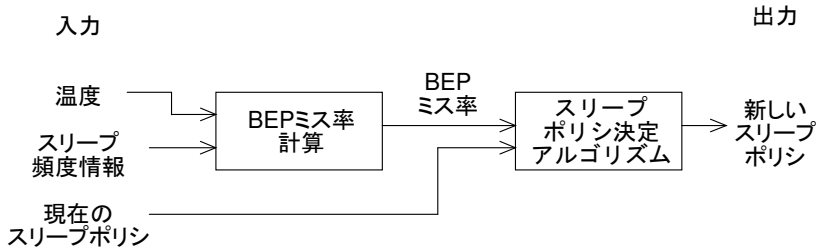

図 5 パワーゲーティング制御機構の流れ

Vol.2011-ARC-195 No.31
Vol.2011-OS-117 No.31
2011/4/14

ものの，制御処理自体は機構内で完結しており，パワーゲーティング制御機構はひとつのモジュールとして機能している．パワーゲーティング制御機構の処理の流れを**図 5** に示す．

パワーゲーティング制御機構は，パフォーマンスカウンタからのスリープ頻度情報，および温度デバイスからのコアの温度情報から，4.1 で示した計算式に基づき，BEP ミス率を計算する．この BEP ミス率と閾値を比較して，4.2 で示したアルゴリズムに基づき現在の各ユニットの動作モードから新しい動作モードを決定し，新しいスリープポリシを求める．

6. 実　　装

実装では，Linux の既存のコードを変更する箇所と新たに追加する箇所に分かれた．パワーゲーティング制御機構を実装するにあたって変更・追加したコードの箇所を**表 1** に示す．

既存の Linux のコード量は 600 万行ほどあるが，この中からスケジューラ部を調べたところ，パワーゲーティング制御機構を呼び出す記述を追加すれば実現可能であることが確認できたので，パワーゲーティング制御の追加を合計 12 行程度の呼び出し処理の追記により実現した．一方，パワーゲーティング制御やパフォーマンスカウンタの制御などの独自の処理については，合計 371 行のコード量で実現した．

7. 評　　価

本節ではパワーゲーティング制御機構を搭載した Linux を FPGA 上で動作させ，ベンチマーク・アプリケーションを実行してスリープ頻度情報を計測し，スリープ頻度情報から電力を見積もり，実ハードウェア上におけるスリープ時平均リーク電力およびエネルギー遅延積を評価した．また，プロセスごとのパワーゲーティング制御を実現するにあたり，複数のプロセスがマルチプロセスで動作している場合でも各々のプロセスの制御に影響しないことを示した．

表 1　パワーゲーティング制御機構を搭載した際の変更箇所一覧

変更内容概略	変更量
スケジューラ本体へ PG 制御機構呼び出し	5 行挿入
Geyser 依存部へ割り込み処理追加	5 行挿入
プロセスコンテキストの拡張	2 行挿入
パワーゲーティング制御機構コード	144 行新規追加
パフォーマンスカウンタ制御コード	95 行新規追加
パワーゲーティング 制御機構などの定義ファイル	132 行新規追加

7.1 評 価 環 境

評価に用いた環境を**表 2** に示す．パワーゲーティングを制御するプロセッサには Geyser-2 を用いた．この Geyser-2 を Xilinx 社の『Virtex-5 LX FPGA ML501 評価プラットフォーム』の FPGA 上に実装し，この環境を用いてパワーゲーティング制御機構を搭載した評価用 Linux 2.6.32.26 を動作させて評価を行った．この Linux カーネルのビルドには，x86 で動作する Linux 2.6.26/Debian 5.0.7 上に構築した gcc クロスコンパイル環境を用いた．

評価用ベンチマークとして，クイックソート，行列計算，Dhrystone，ダイクストラ，Blowfish，ビットカウント，高速フーリエ変換，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それぞれひとつの Linux アプリケーションとして作成し，Linux 上で実行させた．温度については，25 ℃および 100 ℃に固定して行った．スリープポリシ計算アルゴリズムで用いる閾値は，$Th_{CMiss} = 8\%$，$Th_{PG} = 10\%$，$Th_{NoPG} = 60\%$ とした．

8. 電力見積もり計算方法

スリープ時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情報から計算することにより求める．スリープ期間ごとの平均リーク電力は，先行研究[4] により求められており，これを用いて計算する．

スリープ時平均リーク電力は，スリープ期間 i のサイクル数 T_i の平均電力 $\overline{P_i}$ にそのスリープ期間の出現頻度 R_i および T_i を乗じた値を，全スリープ期間の各スリープについて算出して合計し，これをスリープ期間の総サイクル数 T_{all} で割ることで求められる．すなわち，スリープ時平均リーク電力 $\overline{P_{sleep}}$ は，

$$\overline{P_{sleep}} = \frac{\sum_i (\overline{P_i} \times R_i \times T_i)}{T_{all}} \tag{2}$$

で求められる．

アクティブ時の平均リーク電力 $\overline{P_{active}}$ については，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，

表 2　評価環境の一覧

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-2　(a9.5)
評価 OS	Linux 2.6.32.26
クロスコンパイラ	GNU Compiler Collection (gcc) 4.5.1

Vol.2011-ARC-195 No.31
Vol.2011-OS-117 No.31
2011/4/14

BEP と同じサイクル数だけスリープしたとき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と期待できるため，

$$\overline{P_{active}} = \overline{P_{i=bep}} \quad (3)$$

と仮定できる．

また，プロセス全体のエネルギー E_{all} は，スリープ時平均電力とスリープ時間 T_{sleep} の積と，アクティブ時平均電力とアクティブ時間 T_{active} の積の和から求められ，

$$E_{all} = \overline{P_{sleep}} \times T_{sleep} + \overline{P_{active}} \times T_{active} \quad (4)$$

で計算できる．

8.1 スリープ時平均リーク電力

スリープ時平均リーク電力を，各ベンチマークプログラムを単独で実行させて計測した．それぞれのベンチマークについてタイムスライスごとにパワーゲーティング制御を行った場合と行わなかった場合における全ユニットのスリープ時平均リーク電力を**図 6** に示す．

その結果，Blowfish の 100 ℃の場合を除き，すべてのベンチマークについて，パワーゲーティング制御を行った場合のスリープ時平均リーク電力が，制御を行わなかった場合の電力を下回った．これは，パワーゲーティング制御を行うことで，制御を行わない場合に比べてスリープ時平均リーク電力を削減できたことを示している．

スリープ時平均リーク電力の削減割合はベンチマークの種類や温度により変動するが，3.0%から最大 23.2%であった．全体では平均 8.1%のリーク電力を削減することができた．とくに行列計算では電力削減効果が大きく，最大で 23.2%の電力を削減できた．これは，行列計算が比較的大電力を要する MULT ユニット（乗算）を多用するベンチマークであり，短い期間のスリープが頻繁に発生したためと考えられる．

先行研究におけるシミュレーションによる電力削減効果は 2〜25%程度であった[1)]．先行研究では制御対象が ALU ユニットだけであったという点が本研究と異なるが，本研究で 3.0〜23.2%の電力削減が実現できたことから，FPGA 上においても，先行研究のシミュレーションと同程度のスリープ時平均リーク電力の削減効果を得ることができたといえる．

一方で，Blowfish ベンチマークでは 100 ℃の場合に電力削減効果が得られなかった．これは，制御を行うことで MULT のスリープ時平均リーク電力が増大してしまったためである．この原因は，細粒度にパワーゲーティングを行った方がよいケースで，スリープポリシ決定アルゴリズムが誤って粗粒度のパワーゲーティングを行うよう制御したためであると考えられる．解決にはスリープポリシ決定アルゴリズムで用いる閾値を調整することであるが，他のプロセスに対して悪影響を及ぼすことがないよう，どのような閾値にするかを今後

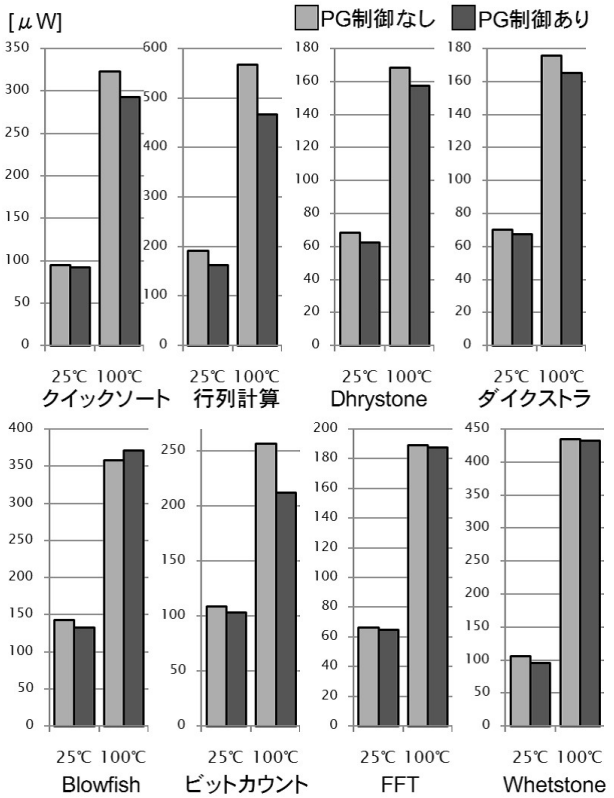

図 6 ベンチマークのスリープ時平均リーク電力

検討していく必要がある．しかしながら，本評価で電力削減効果がなかったのはわずかなケースであるため，様々なアプリケーションを実行するような実際の運用環境では，現段階の状態でもトータルとしてスリープ時平均リーク電力の削減効果は望めると考えられる．

8.2 エネルギー遅延積

プロセス全体の消費エネルギーを，各ベンチマークプログラムを単独で実行させて計測した．それぞれのベンチマークについてタイムスライスごとにパワーゲーティング制御を行った場合と行わなかった場合での消費エネルギーを**図 7** に示す．

Dhrystone，ダイクストラ，Blowfish などのベンチマークは高温では軒並みエネルギー遅

Vol.2011-ARC-195 No.31
Vol.2011-OS-117 No.31
2011/4/14

延積が増大するケースがあった．一方で，常温ではすべてのベンチマークについて，パワーゲーティング制御時のエネルギー遅延積が，制御を行わなかった場合のエネルギーを下回った．したがって，常温ではパワーゲーティング制御によりプロセスのエネルギーを削減することができたことがいえる．常温の削減割合は，スリープ時リーク電力と同様にベンチマークにより変動するが，4.3%から最大 36.0%であった．全体では平均 8.7%のエネルギーを削減できた．行列計算の 25 ℃の場合では，36.0%のエネルギーを削減できた．

一方で，高温では平均 5.7%エネルギーが増大しており，エネルギーの削減効果が得られなかったことがわかる．また，スリープ時リーク電力が削減できた Blowfish 以外のベンチマークについてもエネルギーが増大しているものがある．その理由として次のことが考えられる．スリープ時平均リーク電力は，パワーゲーティングを粗粒度に行うよう制御することによって削減を実現した．しかし，パワーゲーティングを粗粒度に制御することは同時にパワーげーティングを行わないアクティブ時間の増大を引き起こす．また，粗粒度にパワーゲーティング制御すると，本来電力削減効果のあったパワーゲーティングも動作しないよう制御してしまうことがある．電力削減効果のあるパワーゲーティングの削減量が，電力増大を引き起こす短いサイクル期間のパワーゲーティングの削減量を上回ると，全体として電力削減効果を得ることができずにエネルギーが増大する．本評価でエネルギーが増大した原因はこの 2 点が理由であると考えられる．

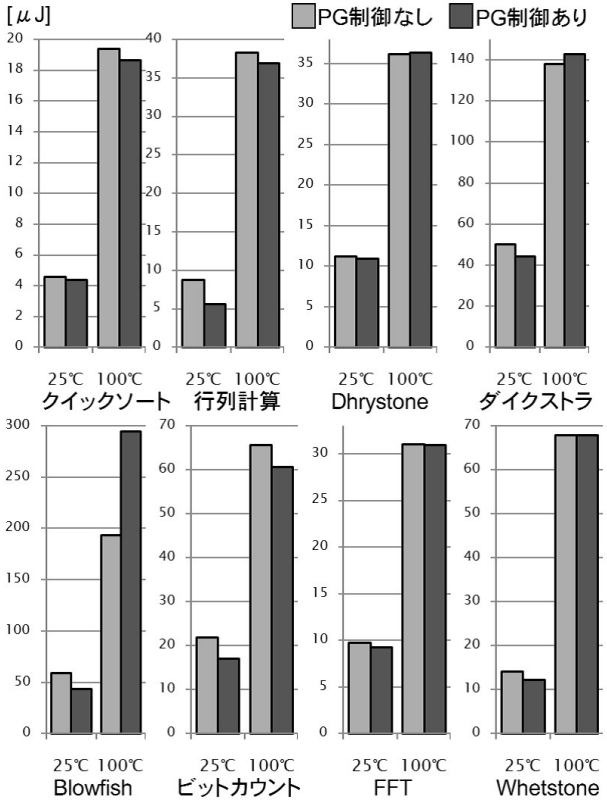

図 7 ベンチマークプロセスのエネルギー遅延積

これらの問題はパワーゲーティングを粗粒度に制御する際の閾値を引き上げることで解決できる．しかしながら，閾値を引き上げすぎると常に細粒度でパワーゲーティングを行うようになる．細粒度でのパワーゲーティングが最もエネルギーを削減できるものであればよいが，現段階ではどの程度の粒度でパワーゲーティングを行えばよいかは明らかでない．そのため，閾値をどのような値にするかは，8.1 節の問題とあわせて今後検討する必要がある．

このように常温で 4.3〜36.0%のエネルギー削減効果を得ることができたことから，常温でプロセッサを稼働させれば全体としてエネルギー消費を削減できることが期待できる．しかし，プロセッサは使用すると温度が上昇するため，常温で稼働し続けることは実際には難しい．したがって，高温においてもエネルギーを削減しなければ，実際の運用でパワーゲーティング制御による効果を得ることができない．本評価では高温においてエネルギーを削減することはできなかった．したがって，実際の運用で削減効果が得られるために，高温の環境においても効果が出るよう閾値の見直しなど制御方式を改良する必要がある．

8.3 マルチプロセス環境下でのパワーゲーティング制御

スリープ時平均リーク電力を例に，クイックソート，行列計算，Dhrystone，ダイクストラの 4 つのベンチマークプロセスについて，それぞれ個別に実行させた場合と，マルチプロセスで実行させた場合における電力削減割合の比較を**図 8** に示す．複数のプロセスをマルチプロセスで実行した場合のスリープ時平均リーク電力の削減割合は，それぞれ個別に実行した場合に比べて，ベンチマークにより 2%程度の誤差がみられるものの，単独実行時の削減効果と動揺の傾向を示した．

本研究で搭載したパワーゲーティング制御機構は，プロセス単位で制御を行っており，プロセスを複数起動してもそれぞれ独立に制御を行う．したがって起動している別のプロセスによって，あるプロセスのパワーゲーティング制御や電力削減効果が影響を受けることはな

Vol.2011-ARC-195 No.31
Vol.2011-OS-117 No.31
2011/4/14

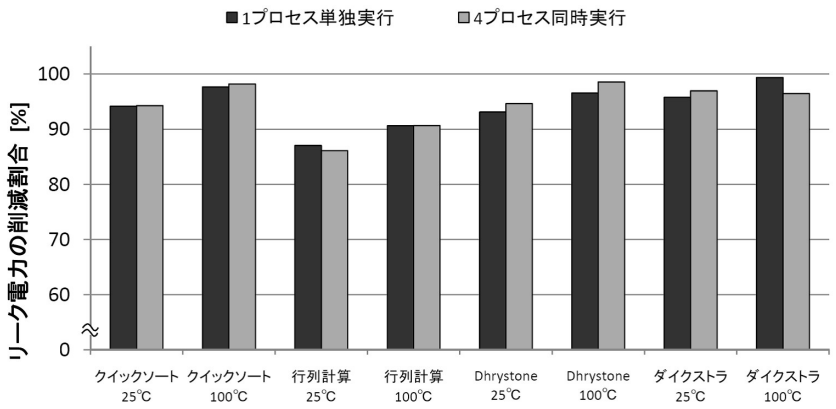

図 8 プロセスをマルチプロセスで実行した場合のスリープ時平均リーク電力の削減割合の比較

い．図 8 の結果からも，それぞれのプロセスの電力削減効果に大きな差が見られないことから，パワーゲーティング制御はプロセスごとに個別に行えていることが確かめられた．

9. おわりに

9.1 本研究の成果

本研究では，FPGA 上に実装されたプロセッサ「Geyser」に対してパワーゲーティング制御を行う Linux を実現し，消費電力について評価を行った．また，シミュレーションではなく実チップに近い FPGA 環境で評価を行うことで，ハードウェア上で電力を削減できることを，8 種類のベンチマークを用いて示した．

評価の結果，汎用のオペレーティングシステムである Linux にパワーゲーティング制御機構を組み込むことで，パワーゲーティングを制御できることを示し，スリープ時平均リーク電力を平均 8.1%，エネルギー遅延積を常温の場合で平均 8.7%削減することを達成した．また，パワーゲーティング制御機構を Linux のプロセススケジューラに搭載することで，プロセスごとに独立してパワーゲーティング制御を行うことを実現した．

9.2 今後の課題

今後の課題として，パワーゲーティング制御アルゴリズムの改良が挙げられる．例えば 8.1 節では，パワーゲーティング制御を行うことで，逆に平均リーク電力が増大する場合がみられた．また，8.2 節では高温の場合にエネルギーの削減効果がみられなかった．これらの原因として，閾値が適切でないということが考えられるので，今後は閾値を変更して評価を行い，エネルギーが最小となる閾値を求める．また，現在はすべての状況下でも同じ閾値を用いているが，これをユニットや温度ごとに閾値を可変にすることで，それぞれの状況に応じた制御を行うよう改良することが挙げられる．

この他，粗粒度にパワーゲーティングした場合では，スリープ回数が少なくなるため，スリープ頻度情報から得られる情報が減少してしまうという問題がある．そのため，ユニットの使用頻度情報などスリープ頻度情報以外の情報を考慮してより正確な制御を行うようにすることなどが挙げられる．

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) 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. [システムソフトウェアとオペレーティング・システム] 2010-OS-114(8), pp.1-8 (2010).
2) 茂木勇，木村一樹，砂田徹也，並木美太郎 : 省電力 MIPS プロセッサ Geyser の FPGA 版評価ボードへの Linux の移植，情報処理学会研究報告. [システムソフトウェアとオペレーティング・システム] 2010-OS-114(9), pp.1-8 (2010).
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 (低消費電力化技術)，情報処理学会研究報告 SLDM [システム LSI 設計技術] 2008(2), pp.43-48 (2008).
4) 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 SLDM [システム LSI 設計技術] 2008(2), pp.37-42 (2008).
5) 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 [システムソフトウェアとオペレーティング・システム] 2005(48), pp.167-168 (2005).
6) Wanghong Yuan, Klara Nahrstedt: Energy-Efficient Soft Real-Time CPU Scheduling for Mobile Multimedia Systems, Proceedings of the nineteenth ACM symposium on Operating systems principles, SOSP '03, pp.149-163 (2003).
7) Ashwini H.S., Amit Thawani, Y. N. Srikant: Middleware for Efficient Power Management in Mobile Devices, Proceedings of the 3rd international conference on Mobile technology, applications & systems, Mobility '06 (2006).