Vol.2012-ARC-200 No.16
Vol.2012-OS-121 No.16
2012/5/8

リアルタイムOSにおける細粒度パワーゲーティング制御の設計と実装

嶋田 裕巳[1]　小林 弘明[1]　高橋 昭宏[1]　坂本 龍一[1]　佐藤 未来子[1]　近藤 正章[2]　天野 英晴[3]
中村 宏[4]　並木 美太郎[1]

概要：本研究では，細粒度パワーゲーティング (PG) 技術を搭載したプロセッサを対象に，リアルタイムOS における PG 制御機構を提案する．PG 制御機構を搭載するリアルタイム OS は，リアルタイム性を保証しつつ省電力化を行う．PG 制御機構ではタスク実行時の余裕時間を予測することによって PG 動作モードの変更を行う．この PG 制御機構の詳細設計を行い，プロセッサ上で動作するリアルタイム OS に実装した．評価として，周期的タスクを動作させたときの平均消費リーク電力を計測した．計測の結果，従来手法に比べ，提案手法において最大で 23%のリーク電力を削減することができた．

Design and Implementation of Fine-grained Power Gating Control for Real Time OS

Shimada Yumi[1]　Kobayashi Hiroaki[1]　Takahashi Akihiro[1]　Sakamoto Ryuiti[1]　Sato Mikiko[1]
Kondo Masaaki[2]　Amano Hideharu[3]　Nakamura Hiroshi[4]　Namiki Mitaro[1]

Abstract: This paper describes a PG(power gating) control mechanism for the processor equipped with a fine-grained PG technology for RTOS. An RTOS equipped with a PG control mechanism aims to reduce the power consumption while guaranteeing real-time performance. A PG control mechanism switches the operating mode by predicting the slack time that occurs when running periodic tasks. The real-time OS running on the PG processor has been implemented to perform the detailed design of the proposed control mechanism PG. In the evaluation, the average leakage power consumption was measured when operating a periodic task. As results of the evaluation, a reduction of up to 23% in average leakage power compared with dynamic PG were achieved.

1. はじめに

近年，システム LSI は性能向上による電力の増大が問題となっている．LSI において消費される電力はダイナミック電力とリーク電力に分類される．ダイナミック電力はトランジスタのスイッチングによって発生する動的な消費電力で，DVFS やクロックゲーティングなどが電力削減の手法として挙げられる．一方，リーク電力はトランジスタの非動作時にリーク電流によって消費される静的な電力である．リーク電力は LSI の微細化に伴い回路全体に占める割合が多くなる傾向にあり，削減の必要性が高まっている．リーク電力の削減手法として，ボディバイアス，Dual-Vth があるが，特にパワーゲーティング (以下 PG:Power Gating)[1] が効果的な手法として挙げられる．PG は動的に回路への電源供給を遮断する省電力手法であ

[1] 東京農工大学
Tokyo University of Agriculture and Technology
[2] 電気通信大学
The University of Electro-Communication
[3] 慶應義塾大学
Keio University
[4] 東京大学
The University of Tokyo

り，従来ではハードウェアによる自律的な電源制御や，コアやモジュールごとの大きな単位での PG が研究されてきた．しかしハードウェアによる自律的な PG を行うと，短期間で電源のオン・オフが発生し，スイッチング時のオーバヘッド電力により逆に電力が増大してしまう場合がある．

そこで，省電力指向のプロセッサである「Geyser」[2] の研究が 5 大学間共同で進められており，本研究室では主にシステムソフトウェアの領域を担当している．Geyser は MIPS R3000 アーキテクチャをベースとしたプロセッサであり，細粒度 PG 技術を搭載している．Geyser は 4 つの演算ユニット (Alu, Shift, Mult, Div) に対して，ソフトウェアからハードウェア PG を細粒度に制御することが可能である．

また，Geyser の細粒度 PG では電力削減効果の指標となるスリープ期間である BEP(Break Even Point:電力損益分岐点)[3] を持つ．スリープ期間が BEP を超えるとき電力削減が可能となるが，スリープ期間が BEP を下回る場合，逆に電力が増加してしまう．

組み込みシステムにおいては，リアルタイムシステムの必要性が高まっている．近年では多くの組み込みシステムにおいてリアルタイムシステムが採用されており，例としてはマルチメディア処理やネットワーク処理などが挙げられる．組み込みシステムはリソースに制限があるものや，バッテリ駆動型のものなど厳しい性能制約下にあり，さらに性能向上を行うにはリアルタイム性を確保しつつ省電力を達成する必要がある．リアルタイムシステムでは処理をデッドラインまでに完了させることができた場合に余裕時間が発生する．関連研究としてその余裕時間を用いてダイナミック電力を削減する DVFS 制御 [4][5] や，デッドラインミス率を制御しながら省電力化を行う DVFS 制御 [6] がある．リアルタイムシステムにおいてはリアルタイム性を保証しつつ，省電力を実現させるために余裕時間の正確な見積もりや，デッドラインミスの発生を防ぐよう制御を行なわなければならない．

本稿では，まず 2 章で本研究の目標について述べる．3 章では提案手法において必要な機能である Geyser の PG について述べ，4 章では提案手法を実装する Geyser RTOS の概要について述べる．5 章の PG 制御機構の設計では提案手法の方針から詳細設計について述べ，6 章においては 5 章で示した設計の実装について述べる．7 章で，PG 技術を搭載した評価環境における評価実験・考察について述べ，8 章において本稿のまとめを述べる．

2. 本研究の目標

本研究では，リアルタイムシステムにおいて周期的タスク実行時に発生する余裕時間を利用し，スリープ期間が BEP を超えるような PG 制御を行うリアルタイム OS を提案する．システムにおいてリアルタイム性を保証しつつ，省電力化を実現することで電力とリアルタイム性のトレードオフを解決する．提案手法の詳細設計として，リアルタイム OS に PG 制御機構を搭載する．Geyser 上のリアルタイム OS における PG 制御機構によって周期的タスクを実行させたときに発生する余裕時間などを予測し，リアルタイム性を保証する範囲でできる限りユニットのスリープ時間を伸ばすように PG 動作モードの変更を行う．また，もしデッドラインミスを起こしてしまった場合は従来の動作モードに変更することでできる限りリアルタイム性を保証する．設計した PG 制御機構を Geyser 上で動作するリアルタイム OS に実装し，4 種類の周期的タスクをそれぞれ動作させた時の平均リーク電力の計算を行い，従来手法と提案手法との比較を行う．

3. Geyser のパワーゲーティング

Geyser 特有の機能である PG 制御は単にハードウェアに制御を任せるだけでは，逆に電力が増加する場合がある．Geyser では PG の対象となる 4 つのユニットに対してスリープコントローラで電源のオン・オフを制御する．スリープコントローラによりユニットは基本的に常にスリープにしておき，使用時にウェイクアップする．PG は主に命令パイプラインの中で動作を行い，命令サイクルごとに細かく PG 制御を行う事ができる．また，Geyser は細粒度 PG を制御するためシステム制御コプロセッサに独自の PG 制御（PGStatus）レジスタを有する．この PGStatus レジスタを OS から操作することで PG 動作モードを切り替える事ができる．実際に回路に PG を適用した時における電力の推移を図 1 に示す．電源のオン・オフが頻繁に発生するとスリープ期間が短くなる．すると削減される電力より電源のオン・オフのスイッチング時に発生するオーバヘッド電力の方が大きくなってしまい，結果的に電力が増大してしまう．PG で電力をより多く削減するためにはスリープ期間を長く確保する必要がある．つまり，Geyser が持つ電力損益分岐点である BEP より長くスリープ期間を確保することで，削減される電力がオーバヘッドより大きくなり，電力の削減効果が現れる．しかし，スリープ期間を確保する分だけ実行時間が増加してしまうという問題がある．また，BEP はユニットの種類や，さらに動作時のコア温度によって異なる．BEP を考慮して PG 制御を行うときは温度情報が必要となる．表 1 に示すように BEP はどのユニットにおいても 25 ℃のときが一番長く，高温になるにつれて短くなる．

4. Geyser RTOS の概要

提案する PG 制御機構の設計は Geyser OS にリアルタイム機能を追加した Geyser RTOS をベースに行う．Geyser OS は並木研究室で研究，開発された組み込みシステム向け OS「開聞」をベースとして，Geyser 向けに移植された

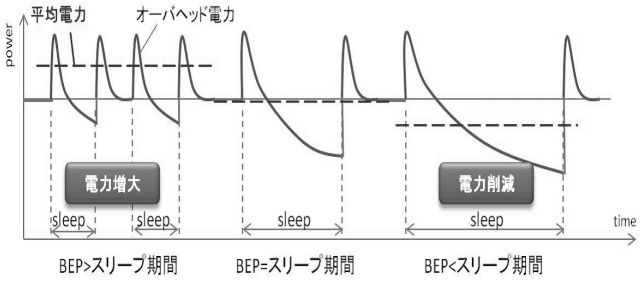

図 1 電力の推移
Fig. 1 Transition of power

表 1 BEP
Table 1 BEP

	25 ℃	65 ℃	100 ℃	125 ℃
Alu	124	38	18	12
Shift	160	50	22	14
Mult	118	44	44	34
Div	58	14	6	2

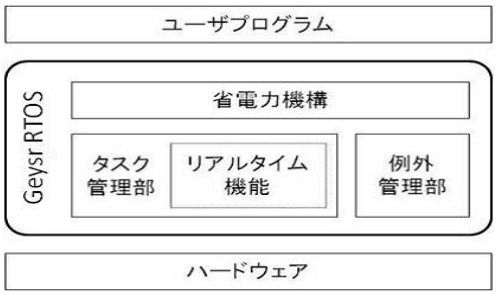

図 2 Geyser RTOS の構成
Fig. 2 Geyser RTOS configuration

OS である [7]．Geyser RTOS は，この Geyser OS へ MIPS R4000 アーキテクチャをベースとして「開聞」に実装されたリアルタイム機能 [8] を本研究において移植したものである．Geyser RTOS の構成を図 2 に示す．Geyser RTOS は Geyser 上で動作するリアルタイム OS で設計した省電力機構を備える．また，リアルタイム機能としてリアルタイムタスクを管理するリアルタイムスケジューラやデッドラインを起こした時に起動されるデッドラインミスハンドラ管理部を持つ．

5. PG 制御機構の設計

5.1 設計方針

先に述べたように Geyser の PG 制御では 4 つのユニットごとに細かく電源をオン・オフを行う．従来では使用していないユニットを常にスリープさせ，使用するたびにウェイクアップ状態にしていたため，オーバヘッド電力により電力的に損になる場合も多く存在した．しかし常に BEP を超えるようにスリープをさせると実行時間が伸びてしまうという問題があった．そこで，設計する PG 制御機構では二つの PG 動作モードを OS によって切り替えることでリアルタイム性を保証しつつできる限りスリープ時間を確保する．図 3 に Geyser RTOS と PG 制御機構の関係を示す．デッドラインミスを起こしたとき OS はデッドラインミスハンドラを起動し，PG 動作モードを変更することでできる限りリアルタイム性を保証する．

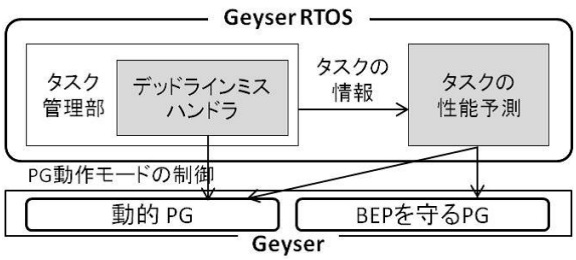

図 3 Geyser RTOS と PG 制御
Fig. 3 Geyser RTOS and PG control

本システムで扱うタスクは以下のように定義される．タスクには開始時間と起動周期，周期的処理を何回繰り返すかをあらかじめ設計者が設定し，周期は途中で変更されることはなく固定優先度のタスクとなる．また，組み込みシステムでの利用を想定するため，タスクの実行時間はあらかじめ分かっているものとする．

5.2 PG 制御機構概要

Geyser RTOS は PG 制御の動作モードとして次に示す二つの動作モードを切り替えて用いる．

- 動的 PG(高性能モード)：ハードウェアの自律的によってユニット使用時だけウェイクアップ状態にする電源制御
- BEP を守る PG(低電力モード)：スリープ期間が BEP を上回るまで必ずユニットをスリープする電源制御

PG 制御機構では性能予測に基づき PG 制御動作モードを「動的 PG」から「BEP を守る PG」へ変更する．PG 制御機構の流れを図 4 に示す．PG 制御機構は初回タスクの処理終了後に呼び出されるリアルタイムスケジューラにおいて実行される．まず性能予測では BEP までユニットをスリープさせたときの伸びる実行時間を見積もり，余裕時間との比較を行う．余裕時間と伸びる実行時間の関係を図 5 に示す．性能予測の後に行われる動作モード変更処理では，リアルタイム性を保証する範囲内で動作モードの変更を行う．また，もし「BEP を守る PG」で処理を行っているときにデッドラインミスを起こしてしまった場合は起動されるデッドラインミスハンドラで動作モードを「動的 PG」へ戻すことによってできる限りリアルタイム性を保証する．時節に PG 制御機構の詳細について述べる．

5.3 性能予測手法

5.3.1 余裕時間の取得

性能予測手法ではタスクの性能を予測し，PG 制御の動

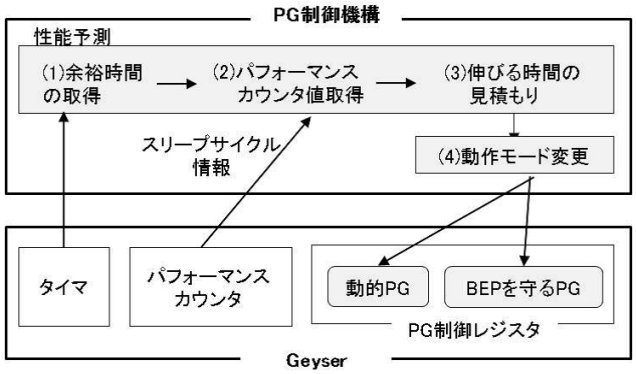

図 4 PG 制御機構の概要
Fig. 4 Overview of the control mechanism PG

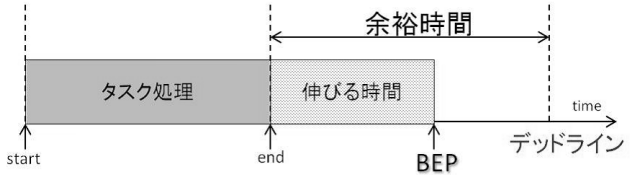

図 5 余裕時間と伸びる時間の関係
Fig. 5 Relationship between slack time and extend time

作モードを選択する．まず余裕時間の取得としてタスク処理終了後，デッドラインまでの時間を取得する．本システムにおいてリアルタイムタスクは 1 周期分の周期的処理が終了すると次の起動時刻とデッドラインを定め，その起動時刻になるまで待機する．よって余裕時間を「次回起動時刻設定終了後」から「次回起動時刻までの時間」とする．「次回起動時刻設定終了後」時点での時間はシステム時間から取得し，「次回起動時刻までの時間」はその次回起動時刻設定で決められた値となる．よって余裕時間は次回起動時刻と次回起動時刻の設定終了時間の差で求められる．PG 制御によるユニットのスリープに関して，BEP を下回ると消費電力が増大し，BEP を超えると消費電力は削減されるが実行時間が伸びるという問題を述べたが，実行時間を伸ばしても余裕時間以内に収まり，デッドラインミスをしないということが分かれば動作モードを BEP を守る PG に変更すればよいと判断できる．そこでタスクに対し BEP を超えるようにスリープさせたときの伸びる処理時間を見積もり，余裕時間と比較することで適用する PG 制御の動作モードを判断する．伸びる処理時間の取得方法を次に述べる．

5.3.2 伸びる処理時間の見積もり

性能予測における伸びる処理時間の見積もりとして，BEP までユニットをスリープさせた時の伸びる処理時間をスリープサイクル情報より計算を行う．伸びるサイクルの例を図 6 に示す．この性能予測の結果，伸びる処理時間が余裕時間より短ければ動作モードを変更し，伸びる処理時間がデッドラインを超えてしまうなら動作モードは変更

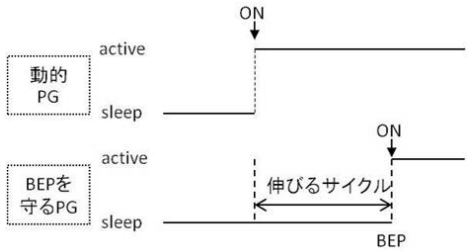

図 6 伸びるサイクル
Fig. 6 Extend cyclen

を行わない．

タスクの性能予測において，伸びる処理時間の見積もりを行う際にはユニットごとの BEP を用いる．BEP はコア温度によって変動するため，BEP を取得するにはタスク動作時に予測されるコア温度の情報が必要となるため，事前に設定する必要がある．見積もりは温度ごとに，またユニットごとに行う．見積もりを行うためにはパフォーマンスカウンタからユニットごとにスリープサイクルの情報を取得する必要がある．伸びる処理時間を見積もるための計算式としてまずユニットごとに伸びるスリープサイクル数を計算する．取得したスリープサイクル数 Ti，そのスリープサイクルの出現回数 Ri の値を用いて，伸びるスリープサイクルを求める計算式は以下のようになる．

$$\text{伸びるサイクル} = \sum_{i=1}^{BEP} \{R_i \cdot (BEP - T_i)\} \qquad (1)$$

複数のユニットに BEP を守る PG を適用する場合，計算によりユニットごとに得られたサイクルを足し合わせ，伸びるサイクルとする．あるユニットのスリープサイクルが長くなると，他ユニットもその影響を受け，他ユニットのスリープサイクル分だけスリープサイクルが長くなる．つまりどのユニットも少なくとも自分がスリープした分と，他のユニットがスリープした分合計でスリープすることになる．よって，それぞれのユニットごとで見積もったスリープサイクルを足し合わせることで実際に BEP を守る PG 制御を行った時の伸びるスリープサイクルに近づけることとした．この求めた伸びるスリープサイクルを時間に変換し，先に求めた余裕時間と比べることで動作モードの変更を行う．動作モードの変更については次節で詳細を述べる．

5.4 動作モード変更

PG 制御の動作モードは Geyser が持つ動作モードのうち動的 PG と BEP を守る PG の 2 つの動作モードを用いる．動作モードの遷移状態を図 7 に示す．タスクにおける周期的処理の初回実行時に性能予測によりデッドラインミスを起こさないような動作モードへ変更する．図 7 のように見積もった伸びる処理時間が余裕時間より小さければ BEP を守る PG に変更することで消費電力の削減に努め

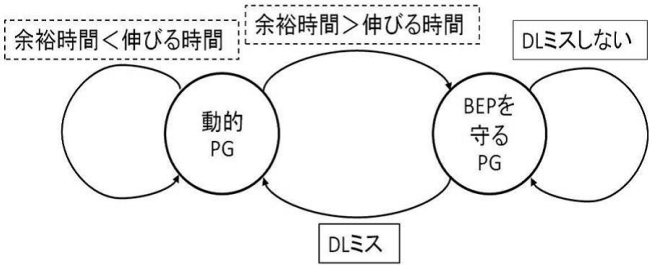

図 7 動作モード変更アルゴリズム
Fig. 7 Algorithm to change the operating mode

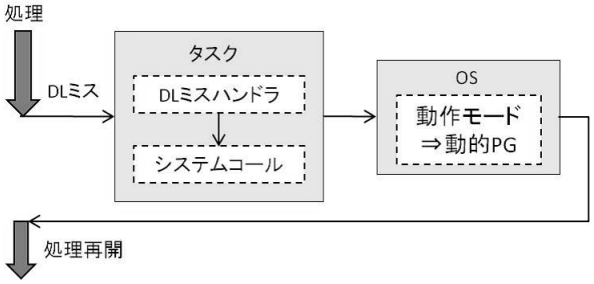

図 8 デッドラインミスハンドラの処理
Fig. 8 Processing during the deadline miss

る．逆に伸びる処理時間が余裕時間より大きくなるときは BEP を守る PG を適用するとデッドラインミスをすると判断し，動作モードは動的 PG のままタスク処理を続ける．また，BEP を守る PG で動作しているときにデッドラインミスを起こすと動的 PG へ変更することでソフトリアルタイム性を保証する．タスクがデッドラインミスを起こさない場合は BEP を守る PG で動作を続けることで省電力化を図る．動作モード変更のアルゴリズムより，動作モードを BEP を守る PG に変更することになった場合，スリープ制御レジスタにタスクの温度による BEP を指定する．

5.5 デッドラインミス時の処理

タスクが BEP を守るモードで実行しているときにデッドラインミスが生じた場合の動作モード変更の方針について述べる．性能予測により BEP を守る PG に変更するが，実際に BEP を守る PG で動作させたときの実行時間が見積もりよりも大きくなり，デッドラインミスを引き起こす可能性がある．基本的にはデッドラインミスが生じたときは動作モードを動的 PG に変更することで実行時間が伸びることを防ぎ，リアルタイム性を保証する．デッドラインミス時の動作モード変更処理の流れを図 8 に示す．スケジューラではタスクが処理の途中でデッドラインミスを起こすと，デッドラインミスハンドラを起動するように設定することができる．そのためデッドラインミスハンドラの処理においてデッドラインミス後に動作モードの変更を行う．タスクがデッドラインミスをするとデッドラインミスハンドラが起動した後，デッドラインミスハンドラの処理が終了したことを通知するシステムコールが必ず呼び出される．そのシステムコールが終了した後に動作モード変更処理を行い，変更が完了したら中断されていた処理を再開する．

6. 実装

本研究を行うにあたり，Geyser 上で動作する OS にリアルタイム機能と，PG 制御機構を実装した．また，BEP を守る PG を実現するため，プロセッサにスリープ制御レジスタが追加された．実装の概要を図 9 に示す．リアルタイムシステムへの対応のため，Geyser 上に本研究室で開発された組込み用軽量リアルタイム OS であるリアルタイム開聞を移植した．この移植において，Geyser OS にリアルタイムタスクとして周期的タスクを扱うリアルタイムタスク管理部，デッドラインミスハンドラの起動を管理するためのデッドラインミスハンドラ管理部を追加した．

表 2 コードの変更量
Table 2 The number of code changes

	追加/挿入行数
リアルタイム機能	540
PG 制御機構	115

また，設計した PG 制御機構の実装として，まず Geyser OS におけるスケジューラの修正を行った．PG 制御機構における性能予測を行うためにはスリープサイクル数が必要となるため，スケジューラにおいて周期的タスクをディスパッチする際に，スリープサイクル数取得を行うためのパフォーマンスカウンタ制御を追加した．また，性能予測，動作モード変更処理をリアルタイムタスク管理部に追加した．性能予測ではタイマから時間を取得し余裕時間を計算する部分，パフォーマンスカウンタから得られたスリープサイクル数から伸びる実行時間を見積もる部分を追加した．動作モード変更処理では性能予測で得られた余裕時間と伸びる実行時間からリアルタイム性を損なわない場合に動的 PG から BEP を守る PG へ変更するようなレジスタの制御を追加した．デッドラインミスハンドラ管理部には，BEP を守る PG で処理が行われてる場合にデッドラインミスを起こしたときに，デッドラインミスハンドラ終了後動作モードを動的 PG へ変更する処理を追加した．

プロセッサに追加されたスリープ制御レジスタは本研究の実現にあたり，ハードウェア要件として新たに追加された BEP を守る PG 制御のためのレジスタである．このレジスタにユニットごとにスリープさせたいクロック数として BEP を 8 ビットで指定することで，その分だけユニットをスリープさせることができるように機能が追加された．

Geyser OS に追加・挿入したコードを表 2 に示す．PG 制御機構における追加・挿入行の合計は 115 行となり，リアルタイム機能の追加・挿入行と合わせると Geyser OS に追加・挿入されたコードは合計で 655 行となった．

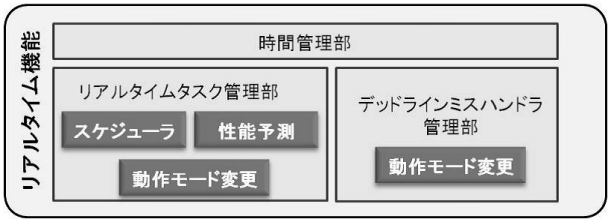

図 9 実装の概要
Fig. 9 Implementation Overview

表 3 開発環境
Table 3 Development environment

項目	製品名，バージョンなど
ターゲットデバイス	Geyser on FPGA(Xilinx ML501)
ホストデバイス	DELL Optiplex 980
ホスト OS	Windows 7 Professional(Cygwin-1.7.9)
コンパイラ	gcc4.3.4
ターミナルエミュレータ	Tera Term4.69
統合開発環境	Xilinx ISE Design Suite

7. 評価

評価として 4 種類のベンチマークを従来手法と提案手法を用いてどう平均リーク電力の計測を行った．スリープ・アクティブサイクルの頻度情報を取得し，電力を計算した．電力を動的 PG と提案手法の平均リーク電力の場合において取得し，比較を行った．

評価環境として FPGA 上に Geyser アーキテクチャを構築した Geyser on FPGA を用いる．Geyser on FPGA では本研究のために指定されたクロックサイクルまで必ずスリープするよう本研究室でハードウェアに機能が追加された．また，Geyser on FPGA はパフォーマンスカウンタを備えており，電力計算のためのスリープ・アクティブサイクル値の取得を行うことができる．評価環境の詳細を表 3 に示す．

7.1 電力評価方法

評価では周期タスク実行時の平均リーク電力を算出し，省電力効果について評価を行う．評価環境では実際の消費電力を計測することができないため，タスク実行時におけるスリープ・アクティブサイクル数の頻度情報をパフォーマンスカウンタから取得し，文献 [9] で提案されている手法により計算を行う．平均リーク電力を求めるにはスリープ時平均リーク電力とアクティブ時平均リーク電力が必要となる．まず，スリープ時平均リーク電力 $\overline{P_{sleep}}$ はスリープサイクル数 T_i と，スリープサイクル数に対する平均リーク電力 $\overline{P_i}$，スリープサイクル数の出現回数 R_i を乗じた値をスリープサイクルごとに合計し，総スリープサイクル T_{total} で割ることで求められる．

表 4 評価タスク
Table 4 Periodic task

処理内容	処理時間 (ms)
matrix	2717
dijkstra	1845
dhrystone	2752
whetstone	2658

$$\overline{P_{sleep}} = \frac{\sum_{i=1}(\overline{P_i} \cdot T_i \cdot R_i)}{T_{total}} \tag{2}$$

アクティブ時の平均リーク電力 $\overline{P_{active}}$ は文献 [3] により BEP と同じサイクル数だけスリープした時の平均リーク電力と等しくなるとしていることから次式で求められる．

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

このスリープ時平均リーク電力とアクティブ時平均リーク電力を用いて平均リーク電力を算出する．平均リーク電力 $\overline{P}$ は $\overline{P_{sleep}}$，$\overline{P_{active}}$ をスリープ/アクティブ時平均リーク電力，T_{sleep}，T_{active} を総スリープ/アクティブサイクル数，T_{total} を総サイクル数として次式によって求められる．

$$\overline{P} = \frac{\overline{P_{sleep}} \cdot T_{sleep} + \overline{P_{active}} \cdot T_{active}}{T_{total}} \tag{4}$$

7.2 評価タスク

評価タスクとしては周期 5000ms で 30s 間実行される周期的タスクを用いる．評価タスクは 4 種類あり，それぞれ行列演算，整数演算などの簡単な計算を行う．評価タスクの処理内容とタスクを動作させたときの実質の処理時間を表 4 に示す．また，タスクを動作させている際のコアの温度について，Geyser は温度計測ができないため，開始から終了まで固定（25，65，100 ℃）とする．

7.3 評価結果

評価結果として各温度においてひとつのユニットに BEP を守る PG を適用する場合と，複数のユニットに BEP を守る PG を適用する場合についてリーク電力を測定した．ひとつのユニットに BEP を守る PG を適用した場合 48 パターンのうち 2 パターンでデッドラインミスを起こし，複数のユニットに BEP を守る PG を適用した場合はデッドラインミスを起こすことなく処理が行われた．次節からそれぞれの場合における電力評価結果の詳細について示す．

7.3.1 ひとつのユニットに BEP を守る PG を用いる場合の電力評価

まず，ひとつのユニットに BEP を守る PG を適用する．適用ユニットは Div，Mult，Shift，Alu のうちひとつのユニットに BEP を守る PG，それ以外に動的 PG を適用する．

タスクを各パターンで動作させたとき，「25 ℃，matrix，Alu のみ」，「25 ℃，whetstone，Alu のみ」では周期的処理中にデッドラインミスを起こし，動作モードが BEP を守る

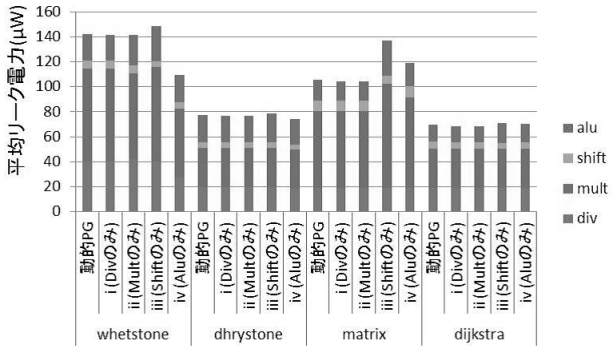

図 10 平均リーク電力
Fig. 10 Average leakage power

PG から動的 PG に変更となった．また，「25 ℃，matrix，Shift のみ」，「25 ℃，whetstone，Mult のみ・Shift のみ」では性能予測において BEP を守る PG を適用するとデッドラインミスをすると判断されたため，BEP を守る PG には変更せず動的 PG のまま処理を続けることとなった．それ以外のパターンについては 30ms 間でデッドラインミスをすることなく処理が終了した．

65 ℃におけるリーク電力計測の評価結果を評価結果を図 10 に示す．グラフはそれぞれのパターンに関して各ユニットの平均リーク電力を積み上げて表示したものとなっている．この提案手法では「65 ℃，whetstone，Alu のみ」において最大 23.14%の平均リーク電力を削減することができた．これは他の評価タスクに比べて 65 ℃の whetstone において，Div，Mult の使用頻度が高く，その分スリープ期間が長くなるためであると考えられる．また，Shift にのみ BEP を守る PG を適用した場合，すべてのタスクでリーク電力が増加してしまった．電力が増加する理由としてはあるユニットをスリープさせたときに，他のユニットがウェイクアップ状態のままになってしまうことが挙げられる．これらのことより，BEP を守る PG を用いるユニットや処理内容によって電力削減の動向が変化することがわかった．

7.3.2 複数のユニットに BEP を守る PG を用いる場合の電力評価

次に複数のユニットに BEP を守る PG を適用する．適用ユニットはタスクによって Div，Mult と Alu もしくは Shift の 3 ユニットに適用する．評価タスクごとに BEP を守る PG を適用したユニットを表 5 に示す．Alu と Shift の両方に適用しない理由としてはひとつに両方のユニットに BEP を守る PG を適用すると実行時間が伸びすぎてしまい，デッドラインミスが多発してしまう可能性があるからである．Alu，Shift どちらを選ぶかに関してはひとつのユニットに BEP を守る PG を適用した際の電力の結果をもとに電力削減が見込まれるユニットを選択した．

65 ℃における評価結果を図 11 に示す．各パターンおいてデッドラインミスをしたものはなかった．電力の測定結果は動的 PG に比べ電力削減できたものはなく，平均で 4.01%電力が増加してしまった．電力が増加する理由としては，ひとつのユニットに対し BEP を守る PG を適用した時と同様に，あるユニットをスリープさせたときに，他のユニットがウェイクアップ状態のままになってしまうことが挙げられ，それがさらに複数ユニットになったため影響が大きくなってしまっていることが考えられる．これらことから複数ユニットに BEP を守る PG を用いても削減効果が向上するわけではなく，BEP を守る PG を適用するユニットの組み合わせが重要であることがわかる．

表 5 BEP を守る PG を適用するユニット
Table 5 Unit to apply the BEP-preserving PG

評価タスク	ユニット
whetstone	Div,Mult,Alu
dhryston	Div,Mult,Alu
matrixe	Div,Mult,Shift
dijkstra	Div,Mult,Shift

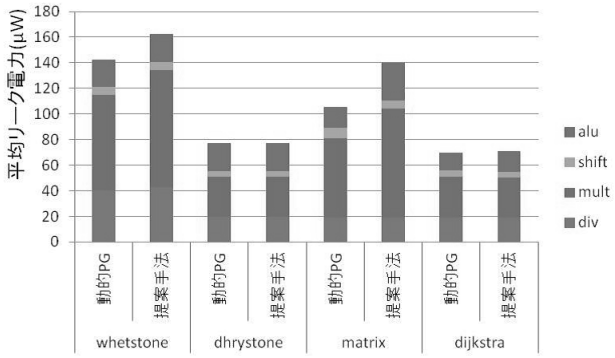

図 11 平均リーク電力
Fig. 11 Average leakage power

7.4 アクティブ・スリープ状態の分析

全節において電力が増加してしまった原因について分析を行う．先に述べたように電力が増加する原因としては，あるユニットをスリープ状態にさせているときに，他のユニットがウェイクアップ状態のままになってしまうことが挙げられる．図 12 では whetstone 動作時において BEP を守る PG を shift にのみ適用した場合の，shift と Alu のスリープ (0)・ウェイクアップ (1) 状況を示している．BEP を守る PG のときの shift は動的 PG に比べてスリープ期間が伸びていることがわかる．しかし Shift のスリープ期間が伸びている間，Alu はスリープせずにアクティブ状態のままになってしまっている．図 12 の例ではデッドラインまでの間における Alu のアクティブサイクル数がおよそ 4 倍となる．このときスリープ時よりアクティブ時の方が電力が大きいため，Alu は結果的に動的 PG に比べ BEP

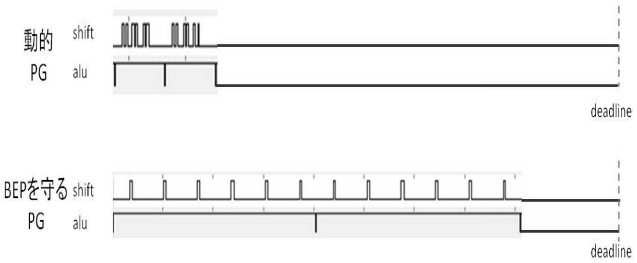

図 12 各動作モードにおけるアクティブ・スリープ状態
Fig. 12 Active/sleep state in each operating mode

を守る PG の方が平均リーク電力が増加してしまう．実際に Alu は全てアクティブ状態のままである必要はなく，スリープ状態になれる期間があるが，現在のハードウェアの仕様によりアクティブ状態のままになってしまう．もし BEP を守る PG でスリープできる期間スリープした場合，アクティブ期間は動的 PG に近い値のスリープ期間が確保できると考えられる．リーク電力をさらに削減するためにはこのようにスリープ・アクティブ期間の分析を行い，もし Alu をスリープさせることができた場合のリーク電力を想定した上で，ハードウェアの改良を行うなどが必要となると考えられる．

8. おわりに

8.1 本研究の成果

本研究では余裕時間の中で BEP を上回るまでスリープさせることでリアルタイム性を保証しつつ，省電力化を行う新たな PG 制御を搭載したリアルタイム OS を提案した．提案手法の詳細設計を行い Geyser 上で動作する Geyser RTOS に提案手法を実装し，評価を行った．

評価として Geyser on FPGA 環境で 4 つの周期的タスクを動作させた，ひとつのユニットに BEP を守る PG を適用した場合，従来手法に比べて最大 23.14%の電力削減効果を得ることができた．結果から，適用するユニットや評価タスクの挙動によって削減効果が変化することが分かった．また，複数ユニットに BEP を守る PG を適用した場合，削減効果が表れなかった．このことから複数ユニットに提案手法を用いても削減効果が向上するわけではなく，どのユニットに適用するかが重要となることがわかった．これらのことよりどのユニットに対しても電力を悪化させることなく削減効果が出るように，更なる PG 制御機構の改善の必要性も明らかとなった．

8.2 今度の課題

今後の課題としては得られた結果のスリープ挙動のさらなる詳細分析が挙げられる．詳細分析としてスリープによる他ユニットへの影響を調査する必要がある．また，温度に対する PG 制御効果を考慮する必要が挙げられる．ユニットをスリープ状態にさせることでコアの温度上昇を防げる可能性があるためこれについて考慮する必要がある．また，BEP は温度によって変化するため温度変化を考慮し，適切な BEP を選択する必要がある．これらのことを踏まえて，動作モード変更アルゴリズムやスケジューラの改良を行うことでさらなる消費電力の削減が実現できると考えられる．

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「CREST」における研究領域「革新的電源制御による次世代超低消費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] N. Seki, L. Zhao, J. Kei, D. Ikebuchi, Y. Kojima, Y. Hasegawa, H. Amano, T. Kashima, S. Takeda, T. Shirai, M. Nakata, K. Usami, T. Sunata, J. Kanai, M. Namiki, M. Kondo, and H. Nakamura, “ A Fine-grain Dynamic Sleep Control Scheme in MIPS R3000 ”, Proceeding of the 26th IEEE International Conference on Computer Design, pp.612-617, 2008.

[2] 中村宏, 天野英晴, 宇佐美公良, 並木美太郎, 今井雅, 近藤正章, “ 革新的電源制御による超低消費電力高性能システム LSI の構想 ”, 情報処理学会研究報告 ARC-173, pp.79-84, Jun 2007.

[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) ”, 信学技報, vol.107, no.414, VLD2007-111, pp.43-48, Jan 2008.

[4] Pillai, P. and Shin, K. G., “ Real-Time Dynamic Voltage Scaling for Low-Power Embedded Operating Systems ”, 18th ACM Symposium on Operating Systems Principles, pp.89-102, 2001.

[5] Kim, W., Kim, J. and Min, S. L., “ A Dynamic Voltage Scaling Algorithm for Dynamicb Priority Hard RealTime Systems Using Slack Time Analysis ”, in Proc. Design Automation and Test in Europe, pp.788-794, 2002.

[6] 中村哲朗, 加藤真平, 小林秀典, 山崎信行, “ リアルタイム性を考慮したフィードバック制御による動的周波数制御手法 ”, 組込技術とネットワークに関するワークショップ (ETNET2006), Vol. 105, No. 670, pp.7-12, 2006.

[7] 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 ”, IPSJ SIG Technical Report 2010-ARC-189 pp.1-8, Apr 2010.

[8] 堀口努，並木美太郎, “ 組み込み用オペレーティングシステム『開聞』におけるリアルタイム機能の開発 ”, 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, pp.91-98, Feb 2003.

[9] 中田充貴, 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 ”, 信学技報, vol.107, no.414, VLD2007-111, pp.37-42, Jan 2008.

[10] David Duarte, Yuh-Fang Tsai, Narayanan Vijaykrishnan, and Mary Jane Irwin, “ Evaluation Run-Time Techniques for Leakage Power Reduction ”, ASPDAC 2002, pp. 31-38, Aug (2002).

[11] James T. Kao and Anantha P. Chandrakasan “ Dual-Threshold Voltage Techniques for Low-Power Digital Circuits ”, IEEE Journal of Solid-State Circuits, vol. 35, pp. 1009-1018, Jul (2000).