Vol.2014-ARC-210 No.14
Vol.2014-OS-129 No.14
2014/5/15

Linuxスケジューラによるリークモニタを用いた細粒度パワーゲーティング制御手法と実チップにおける評価

小柴 篤史[1]　塚本 潤[1]　和田 基[1]　坂本 龍一[1]　佐藤 未来子[1]　小坂 翼[2]　宇佐美 公良[2]　天野 英晴[3]
近藤 正章[4]　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) 機能を備えた低電力プロセッサ“Geyser”を用いて，OS による PG 制御を行い，プロセッサの消費電力低減を目指す．PG 制御の省電力効果を高めるためには，実行時のチップ温度によって変化する PG の電力削減効果に対応する必要があるが，従来の制御手法では実チップの温度を OS が取得できず，省電力効果が低下する問題があった．本研究では，チップ温度をより正確に取得できるデバイスであるリークモニタを用いた，実行時のチップ温度に適した PG 制御手法を提案する．提案手法の省電力効果をより高めるためのハードウェア環境をエミュレートした場合の電力評価を実チップ上で行い，最大 17.5%の平均消費電力削減を達成した．

1. はじめに

近年のプロセッサの性能向上に伴い，LSI の消費電力が日々増加している．特に，リーク電力の増加が顕著であり，全体の消費電力のうち，リーク電力が占める割合が年々上昇している [3]．そこで筆者らは，リーク電力の削減手法である細粒度パワーゲーティング (PG) 機能を搭載した低電力プロセッサ“Geyser”を用いて，PG によるプロセッサの省電力化の研究を行っている．PG とは，未使用回路の電源電圧を遮断することで，リーク電力を削減する技術である．しかし，電源電圧の供給/遮断を切り替える時にオーバヘッド電力が発生するため，短い期間内で電源電圧のスイッチングを頻繁に行うと，電力が増加してしまう場合がある．そのため，より高効率な PG によるプロセッサの電力削減を達成するためには，オーバヘッド電力を考慮した PG の実行制御を行う必要がある．

筆者らのこれまでの研究 [2] で，PG による電力削減効果の指標である BEP(Break-even Point) を用いて，オーバヘッド電力を抑えるように Linux プロセススケジューラが PG 制御を行う制御手法を提案しており，FPGA 上でシミュレーションによる電力削減効果が得られている．更なる発展として実チップにおける省電力効果の検証が求められているが，実チップに対して本制御手法を適用する際，リーク電力のチップ温度に対する依存性が問題となる．チップ温度が変化すると発生するリーク電力量が変化し，PG 制御の指標となる BEP の値も変化する．実チップ環境において高精度な PG 制御を行うには，この BEP の変化に対応する必要があるが，先行研究では実チップの温度を取得する機能が備わっておらず，スケジューラがどのようにして温度情報を取得するかが課題となる．

[1] 東京農工大学
[2] 芝浦工業大学
[3] 慶應義塾大学
[4] 東京大学

2. 目標

本研究では，実チップ環境に適した PG 制御を実現するため，以下の二つの目標の達成を目指す．

1. リークモニタを用いた実チップ温度に適した PG 制御

本研究では，PG 制御にリークモニタを用いることで，温度変化に対応した PG 制御手法を提案する．リークモニタは Geyser チップで発生するリーク電力を監視することができるハードウェアデバイスである．提案手法では，チップ温度に応じて変化するリーク電力の特性を利用し，リークモニタを用いて実行時のチップ温度を推定する．推定したチップ温度に対応する BEP の値を用いた PG 制御により，実行時のチップ温度に適した PG 制御を実現する．

2. 実チップにおける提案手法の省電力効果の検証

提案手法の実チップにおける省電力効果を検証するため，実チップ上での電力評価を行う．実チップ環境におけるチップの製造ばらつきを考慮するため，評価用チップを

Vol.2014-ARC-210 No.14
Vol.2014-OS-129 No.14
2014/5/15

用いて BEP などの PG 制御用パラメータを計測し，PG 制御に用いることで，実チップ環境に適した PG 制御を実現する．また，現在の Geyser 実チップのハードウェア環境では PG 制御の実行が制限され，省電力効果が低減する問題がある．そこで，提案手法による PG 制御を常に可能にするための新たなハードウェア環境を提案し，本環境をエミュレートした場合の電力評価を行う．以上により，実チップにおいて提案手法による理想的な PG 制御を実現した場合の省電力効果を検証する．

3. Geyser による細粒度 PG

Geyser は MIPS-R3000 アーキテクチャをベースとして細粒度 PG 機能を実装したプロセッサである．ソフトウェア側から PG を制御するインタフェースを備えており，PG 制御用に Geyser 上で動作する Linux が開発されている．本章では PG の特徴および Geyser に搭載された細粒度 PG 機能やリークモニタについて述べる．

3.1 細粒度 PG 機能

PG とは未使用回路の電源電圧を遮断 (**スリープ**) することで，PG 対象のユニットで発生するリーク電力を低減する技術である．本研究で用いる Geyser プロセッサには，小さな時間粒度・空間粒度で PG を実行できる，細粒度 PG 機能が搭載されている．Geyser の細粒度 PG の仕組みを図 1 に示す．Geyser には PG の実行を制御するスリープコントローラが備わっており，このコントローラから四つの演算ユニット (ALU, SHIFT, MULT, DIV) に対して個別に電力遮断/供給の指示が行われる．ソフトウェアはスリープコントローラ制御用のレジスタ (PG 制御レジスタ) に値を書き込むことにより PG の実行モード (**スリープポリシ**) を設定できる．

また，Geyser はスリープコントローラの制御信号を監視するスリープカウンタを備えている．OS はスリープカウンタによって各演算ユニットのスリープ回数や長さを監視し，この情報を用いて PG 制御を行う．

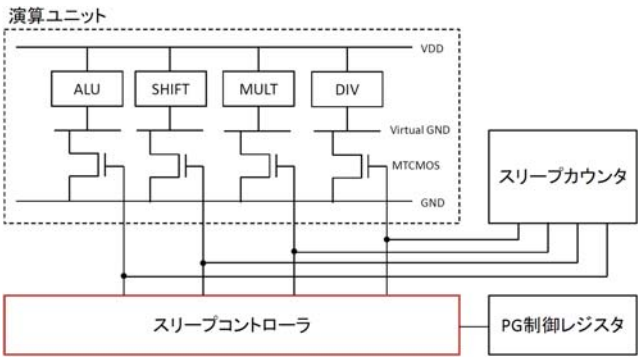

図 1 細粒度 PG の仕組み

3.2 損益分岐点 (Break-even Point)

PG によってユニットをスリープさせている間，ユニットで発生するリーク電力を低減することができる．しかし，PG によって演算ユニットの電源遮断/供給を切り替える際，電力的なオーバヘッドが生じ，一時的に消費電力が増加する．そのため，ユニットのスリープ期間が短い場合は，オーバヘッドによって増加する電力量が削減するリーク電力量を上回り，消費電力が増大する場合がある．したがって，PG の省電力効果を高めるためには，ユニットをスリープさせるタイミングが重要となる．

PG 制御における電源遮断のタイミングの指標として，**損益分岐点** (Break-even Point，以降 **BEP**) が提案されている．PG における BEP とは，「電源電圧のスイッチングによるオーバヘッド電力量と，ユニットのスリープ中に削減される電力量が等しくなるスリープ期間」を表す．図 2 のように，BEP とスリープ期間を比較することで電力削減が可能かどうか判断できる．

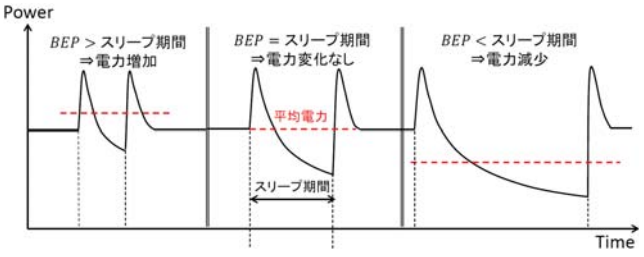

図 2 スリープ期間と平均消費電力の関係

リーク電力の発生量はチップの種類や回路規模，そしてチップ温度に依存するため，BEP の値は演算ユニットの種類やチップ温度に依存して変化する．従来の PG 制御手法では BEP の値はシミュレーション値が用いられていたが，本研究では実チップに適した PG 制御を達成するため，実チップを用いて BEP の値を計測し，制御に用いる．

3.3 リークモニタ

本研究でチップ温度の取得に用いるリークモニタについて述べる．リークモニタは，Geyser にオンチップで搭載されたハードウェアデバイス [7] であり，一定期間におけるリーク電力の発生量が設定値に達するまでの時間 (Detection Time，以降 **DT**) を計測する．リーク電力の発生量はチップ温度が高いほど多くなるため，DT の値はチップ温度に応じて図 4 のように変化する．従って，チップ温度ごとの DT の値が既知であれば，実行時の DT の値からチップ温度を推定することができる．このとき，チップの製造ばらつきの影響により，BEP と同様に DT の値もチップごとに異なるため，温度ごとの DT の値はチップごとに調べておく必要がある．リークモニタは Geyser のコア内部に搭載されているため，リークモニタをチップ温度取得に用いることで，一般的な温度センサよりも高い精度でチップ温度変化に追従できる．

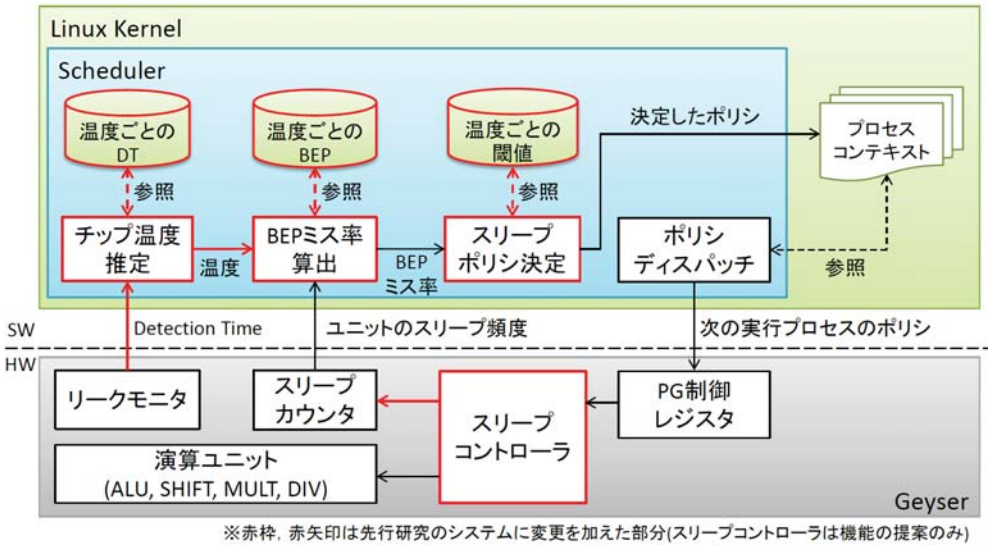

図 3　システムの全体構成

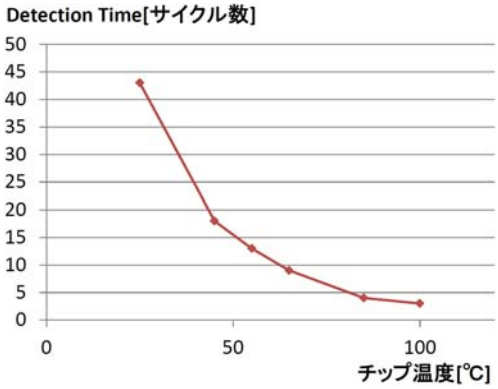

図 4　チップ温度と Detection Time の関係 (40MHz)

4. Linux スケジューラによる PG 制御

提案手法では，Linux スケジューラが定期的にユニットのスリープ頻度を観測し，電力増加を引き起こす BEP 未満の長さのスリープの発生率に応じて PG の実行を抑制することで，PG の省電力効果を高め，プロセッサの省電力化を達成する．本章では，提案手法を実現する PG 制御システムの設計について示す．

4.1　全体構成

PG 制御システムの全体構成を図 3 に示す．PG 制御に用いる静的なパラメータとして，チップ温度ごとの DT，BEP，閾値が存在する．これらの値はチップごとに事前に求め，カーネルが保持しておく．各パラメータの計測方法および結果については 6 章で述べる．

実行時のスケジューラの処理としては，まずリークモニタを用いたチップ温度の推定処理が行われる．処理の詳細については 4.2 節で述べる．次に，スリープカウンタから得た演算ユニットのスリープ頻度情報と BEP を用いて，一定期間中の PG の省電力効果を表す **BEP ミス率**を算出し，閾値と比較することでユニットごとのスリープポリシを決定する．処理の詳細については 4.3 節で述べる．PG 制御に用いる BEP や閾値はチップ温度によって変化するが，スケジューラはリークモニタから得たチップ温度に応じて BEP と閾値を切り替えることで，実行時のチップ温度変化に対応する．決定したスリープポリシは，プロセスコンテキストに格納される．スケジューラは通常のスケジューリング処理を行った後，プロセスコンテキストを参照し，PG 制御レジスタに次の実行プロセスのポリシを適用することで，プロセスごとに適した PG 制御を行う．

ハードウェア構成の変更点として，3.1 節で述べたスリープコントローラを演算ユニットとスリープカウンタで異なる制御信号を出力するようにする．これは現 Geyser チップには実装されていない機能であり，本論文では機能の提案と評価を行う．本機能の詳細については 4.5 節で述べる．

4.2　リークモニタを用いたチップ温度の推定

スケジューラが PG 制御に用いる実行時のチップ温度を取得する処理について示す．3.3 節で述べたように，チップ温度によって値が変化する DT の特性により，リークモニタから得た DT を用いて実行時のチップ温度を推定することができる．チップ温度を推定する手順を図 5 に示す．まずスケジューラはリークモニタに DT の計測を指示し，実行時の DT を取得する．スケジューラは取得した DT を元にチップ温度ごとの DT のリストの探索を行う．実行時の DT に一致する DT をリストの中から選択し，その DT が示すチップ温度を得る．一致する DT がリスト中に無かった場合でも，リスト内の DT のうち，計測した DT に最も近い値を選択することで，ある程度のチップ温度を推定することができる．これにより，DT の値から実行時のチップ温度を取得し，得られたチップ温度に応じて BEP などの PG 制御に使うパラメータを切り替えることで，実行時のチップ温度変化に対応した PG 制御を実現する．

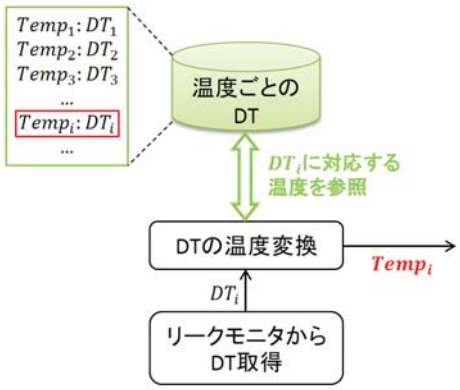

図 5　DT からチップ温度への変換方式

4.3　チップ温度と BEP ミス率に基づく PG 制御

スケジューラはチップ温度取得後，チップ温度に対応する BEP と PG 制御対象の演算ユニットのスリープ頻度情報を用いて PG 制御を行う．PG 制御の方針は，制御対象の演算ユニットのスリープ頻度を定期的に計測し，BEP より短いスリープが頻繁に発生している場合，PG を抑制して電力増加を防ぐことである．本手法では，この PG を抑制する指標として BEP ミス率を用いる．BEP ミス率とは，PG によって一定時間内に生じたスリープのうち，電力が増加するスリープ (スリープサイクルが BEP 未満) の割合を示す．BEP ミス率が大きいほど，PG が電力増加を引き起こす可能性が高いことが分かる．ある演算ユニットでの一定期間中における i サイクルのスリープの発生回数を x_i とすると，BEP ミス率 $BEP_{MissRatio}$ は，次の式で表される．

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

図 6 に，実行時の処理を示す．スケジューラは制御対象のユニットについて，スリープカウンタから得たスリープ頻度情報と，リークモニタを用いて推定したチップ温度に対応する BEP の値を式 (1) に適用し，実行時の BEP ミス率を得る．次に，得られた BEP ミス率を実行時のチップ温度に対応する閾値と比較する．閾値よりも BEP ミス率の値が大きい場合は PG を抑制するポリシを選択し，BEP ミス率が小さい場合は PG を実行するポリシを選択する．以上の処理を全ての演算ユニットに対して行う．

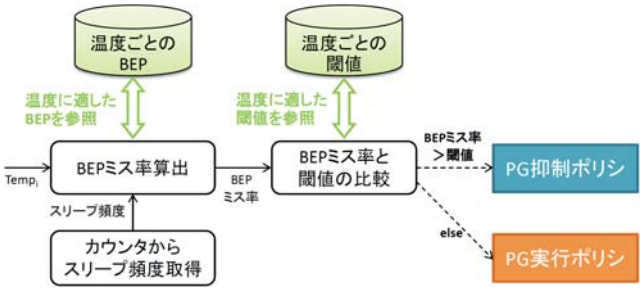

図 6　BEP ミス率の算出とスリープポリシの決定

4.4　スリープポリシ決定のための閾値

BEP ミス率の大小によって PG の省電力効果をある程度判断することができるが，消費電力の変化量は分からない．そこで本 PG 制御手法では，PG の実行/抑制を定める基準として，BEP ミス率の閾値を設定する．閾値は，BEP ミス率と電力削減量の関係を調査し，電力削減/増加の境界となる BEP ミス率を定めることで得られる．スリープポリシの決定基準に閾値を用いることで，省電力化に有効なスリープポリシをより正確に選択できると考えられる．

リーク電力の特性により，PG の省電力効果は実行時のチップ温度や回路，チップの種類によって変化するため，閾値もチップごと，温度ごとに異なる．本研究では，実チップに適した PG 制御を実現するため，予備実験によって実チップに適した閾値を定め，PG 制御に用いる．実チップにおける閾値の算出方法および算出結果は 6.3 節で述べる．

4.5　高効率な PG 制御のためのハードウェア機能

4.3 節で述べたように，本 PG 制御手法では，スリープカウンタから得られたスリープ頻度情報を元にプロセス実行時の BEP ミス率をランタイムで算出し，その値に基づいて適切なポリシを決定する．現在の Geyser チップのハードウェア構成では，制御対象のユニットに PG を抑制するポリシが適用されている場合はスリープが発生しないためにスリープ頻度情報が得られず，BEP ミス率を算出できない．そこで，図 7 のようにスリープコントローラの仕様を変更し，制御信号を 2 種類としたハードウェア構成を提案する．本構成では，スリープカウンタには適用するポリシに関わらず PG 実行時の制御信号が送られるため，常に BEP ミス率の算出が可能となる．これにより，PG 抑制時でも BEP ミス率に基づく PG 制御を行うことができ，提案手法の省電力効果を高めることができると考えられる．

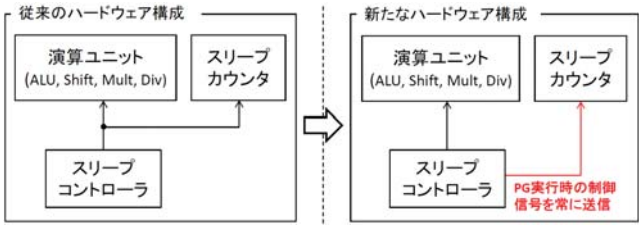

図 7　PG 抑制時もスリープ頻度情報を取得するための新たなハードウェア機能

新ハードウェア機能の省電力効果を現 Geyser 実チップを用いて検証するため，新ハードウェア構成における PG 制御のエミュレート機能を Linux に実装し，電力評価の際に利用する．エミュレート機能については 7.1 節で述べる．

5.　実チップにおける電力評価環境

実チップにおける提案手法の省電力効果を検証するため，

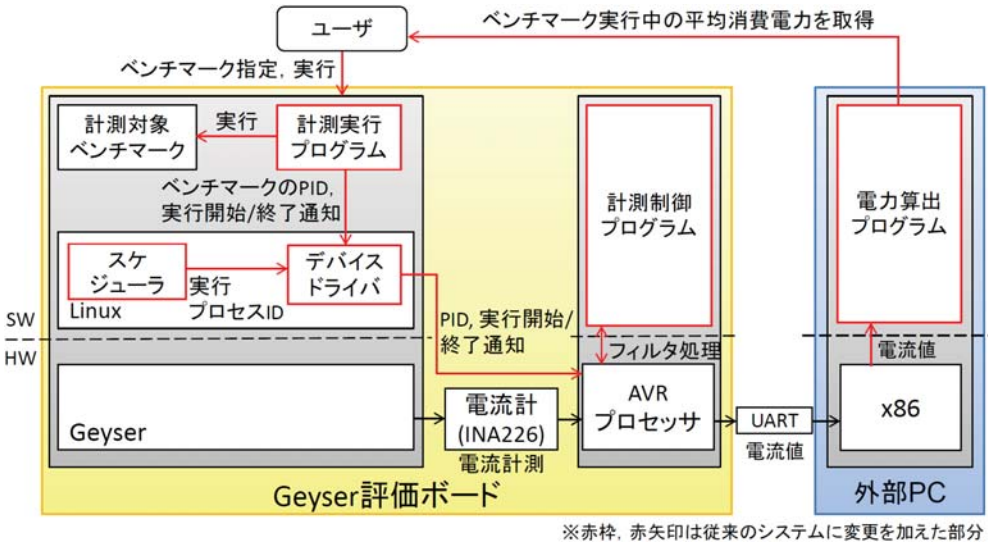

図 8　Geyser 実チップの電力計測システム

4 章で述べた PG 制御システムの Linux への実装，Geyser 実チップの電力評価環境の構築を行った．システム開発に用いた環境を表 1 に，実チップの電力評価環境として用いた Geyser 評価ボードを図 9 に示す．評価ボードには Geyser 実チップのほか，Xilinx の ML605 ボードや電流計 (INA226) が搭載されており，Geyser 上での Linux の起動やチップの電流計測が可能である．更に，本研究ではチップ温度を変化させた場合の評価を行う必要があるため，Geyser チップ上に温度制御装置を取り付け，外部から温度を与えてチップを任意の温度に保つことで，異なるチップ温度における電力計測を行う．

表 1　開発・評価環境一覧

名称	製品名など
カーネルビルド用 OS	Linux/Fedora release 7
クロスコンパイラ	GNU Compiler Collection 4.5.1
対象 CPU	Geyser-3 ver.1.0
OS カーネル	Linux 2.6.35.11

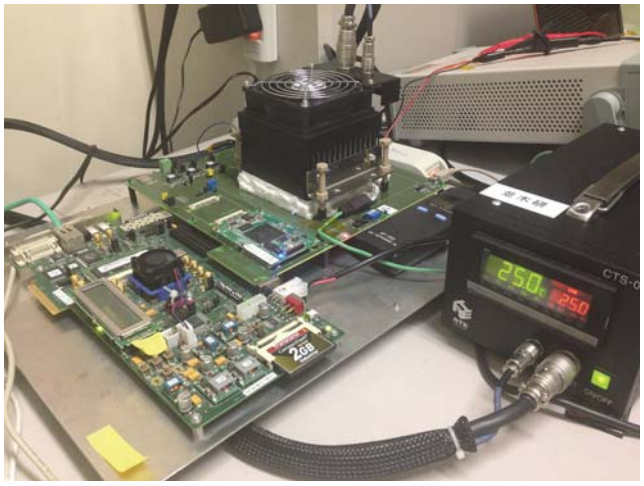

図 9　温度制御装置を取り付けた Geyser 評価ボード

本評価環境を用いて提案手法の電力評価を行うにあたって，実チップの消費電力をより正確に計測するため，評価ボードに搭載されている FPGA 上に ATmega32 の AVR プロセッサを実装し，電流計の計測制御を行う電力計測システムを構築した．電力計測システムの全体構成を図 8 に示す．本システムは，図 9 に示した Geyser 評価ボードと計測データを受け取るための外部 PC で成り立つ．電力計測の流れとしては，まずユーザが外部 PC 上で電力算出プログラムを，Geyser の Linux 上で計測実行プログラムを動作させる．計測実行プログラムはベンチマークを実行したのち，ベンチマークの PID および実行開始をデバイスドライバを介して AVR へ通知する．AVR 上で動作する計測制御プログラムは電流計測を開始し，計測した電流値に対してノイズを軽減するフィルタ処理を行い，UART を介して外部 PC へ送信する．計測中，スケジューラから送られる実行プロセス ID がベンチマークの PID と異なる場合，AVR は電流計測を中断する．ベンチマークが終了すると，計測実行プログラムは AVR に終了を通知し，AVR は電流計測を停止して外部 PC に通知する．外部 PC 上の電力算出プログラムは，ベンチマーク実行中に AVR から取得した電流値から平均消費電力を算出し，ユーザに提供する．このように，実行プロセスに応じた電流計測の実行制御や，電流値のノイズを低減するフィルタ処理を AVR が行うことで，実行プロセスごとの正確な電力計測が可能となる．

本評価環境を用いて，PG 制御に用いるパラメータの計測および提案手法の電力評価を行う．評価を行うチップ温度は 25°C，55°C，85°C の 3 種類とした．また，Geyser の動作周波数は 40MHz で，タイムスライスは約 80ms とした．

6. PG 制御用パラメータの計測

本章では，PG 制御に用いる 3 種類の PG 制御用パラメー

タ (DT，BEP，閾値) の計測方法，および評価用実チップを用いて計測した結果について示す．各パラメータの計測には，5 章で示した評価環境を用いる．

6.1 Detection Time の計測

4.2 節で述べたチップ温度の推定に用いる，チップ温度ごとの DT を計測する．温度制御装置を用いて Geyser チップを一定の温度に保持しておき，Geyser 上で Linux を動作させ，デバイスドライバを介してリークモニタから DT を読み取った．計測結果を表 2 に示す．この値を元にスケジューラが実行時の DT をチップ温度に変換し，得られたチップ温度に適した PG 制御を行う．今回の計測では，評価対象となる 3 種類のチップ温度でのみ計測を行ったが，提案手法を実用する際には，更に細かい粒度での DT の計測が必要となる．

表 2　チップ温度ごとの Detection Time 実測値 (40MHz)

Detection Time[サイクル数]		
25°C	55°C	85°C
43	13	4

6.2 BEP の計測

4.3 節で述べた実行時の BEP ミス率の算出に用いる，BEP を計測する．計測方法としては，算出対象の演算ユニットにおいて，Geyser チップ上で任意の長さのスリープを発生させるプログラムを動作させ，PG 実行/不実行の 2 種類の実行環境における電流値を計測する．スリープの長さを変更しながら電流計測を行っていき，PG 実行/不実行の電流値が一致するスリープの長さを定め，BEP とする．25°C の ALU ユニットにおける，スリープサイクルと電流値の関係を図 10 に示す．スリープサイクルが BEP 未満のときは PG を実行しない方が電流値が小さく，BEP 以上のときは PG を実行した方が電流値が小さくなっていることが確認できる．

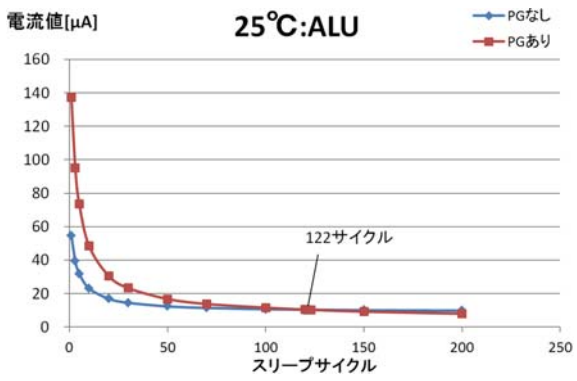

図 10　25°C におけるスリープサイクルと電流値の関係: ALU

25°C，55°C，85°C における BEP の計測結果を表 3 に示す．スケジューラはこの値を用いて PG 制御を行う．

表 3　チップ温度ごとの BEP 実測値 (40MHz)

	BEP[サイクル数]		
	25°C	55°C	85°C
alu	122	44	19
shift	92	33	18
mult	18	7	3
div	1	1	1

6.3 スリープポリシ決定に用いる閾値の算出

4.4 節で述べたスリープポリシの決定に用いる閾値を算出する．算出方法としては，まず算出対象の演算ユニットにおいて，PG 実行/不実行の 2 種類の実行環境において，アプリケーション実行中の Geyser の平均消費電力と BEP ミス率を一定時間ごとに計測する．同じ期間における PG 実行/不実行の消費電力を比較し，BEP ミス率ごとの電力削減量を調査する．電力が増加する BEP ミス率と電力が減少する BEP ミス率の境界を定め，閾値とする．計測対象のアプリケーションには，Mibench から basicmath, dijkstra, fft, lame, patricia, adpcm, gsm の 7 種類，SPEC から libquantum, mcf の 2 種類の計 9 種類を使用し，電力計測はタイムスライス単位 (約 80ms) で行った．

25°C の SHIFT ユニットにおける，BEP ミス率と PG の電力削減量の分布図を図 11 に示す．同じ BEP ミス率のサンプルでも電力削減量にばらつきがあるが，これは Geyser の電源電圧の誤差や，電流計測に用いる抵抗の誤差によるものだと考えられる．サンプルはおおむね線形に分布していることから，近似直線を引き，直線と x 軸の交点すなわち PG による電力削減量が ±0 となるときの BEP ミス率を閾値として定めた．

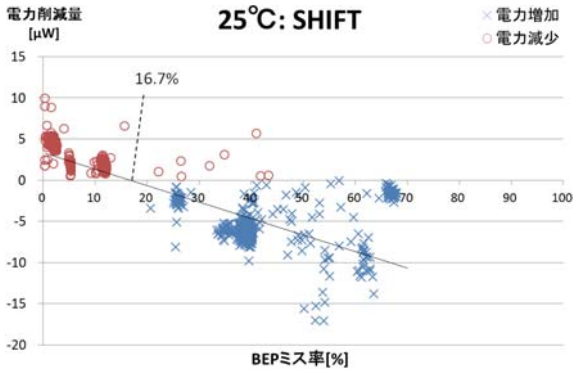

図 11　25°C における BEP ミス率-電力削減量分布: SHIFT

チップ温度 55°C，85°C における閾値も同様の方法で算出し，各チップ温度における閾値のデータを得た．閾値の算出結果を表 4 に示す．高温時の MULT ユニットと DIV ユニットについては，表 3 に示したように BEP が非常に小さく，BEP ミスがほぼ起きないため，閾値は設定しなかった．得られた閾値を PG 制御に用いることで，より正確に PG の実行/抑制を判断でき，実チップの省電力効果が向上すると考えられる．

表 4　チップ温度ごとの BEP ミス率閾値

	25°C	55°C	85°C
alu	9.4%	9.7%	27.3%
shift	16.7%	17.8%	54.2%
mult	28.0%	-	-
div	-	-	-

7. 評価

提案手法の電力評価を行うため，実チップ上でベンチマーク実行中の消費電力を計測した．また，PG 制御の処理時間オーバヘッドを計測した．

7.1 実チップにおける PG 制御のエミュレート機能

現 Geyser チップにおいても 4.5 節で示したハードウェア機能を用いた場合の電力評価を行うため，PG 制御のエミュレート機能を実装し，電力評価に用いる．エミュレートの手順を図 12 に示す．エミュレート機能は，ポリシ決定モード (図中左) とポリシ適用モード (図中右) の 2 種類の実行モードによって成り立つ．初めに，Linux カーネルをポリシ決定モードに遷移させ，制御対象のアプリケーションを実行する．ポリシ決定モードでは，全ての演算ユニットに PG を実行するポリシを適用してプロセスを実行するため，常に BEP ミス率の算出が可能となる．アプリケーション実行中，PG 制御機構は BEP ミス率を元に，タイムスライスごとのスリープポリシを決定し，決定したポリシを Linux カーネル内に保持する．次に，Linux カーネルをポリシ適用モードに遷移させ，ポリシ決定モードで実行したものと同一のアプリケーションを実行する．ポリシ適用モードでは，ポリシ決定モードで得られたポリシを適用しながらアプリケーションを実行する．これにより，あたかも実行時に PG 制御を行っているようにアプリケーションを実行できる．ポリシ適用モードで実行中のアプリケーションの消費電力を計測することで，4.5 節のハードウェア機能を用いた場合の評価を実現する．次期に製作する実チップではこのハードウェア機能を実装する．

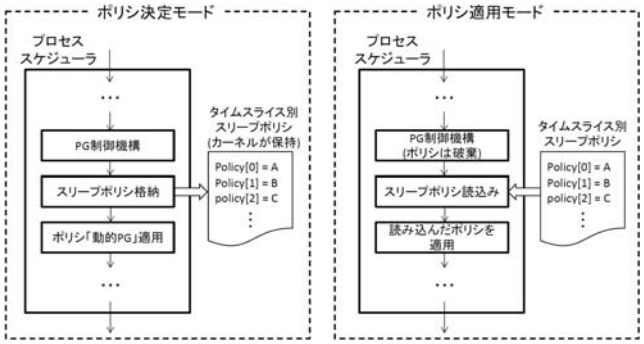

図 12　PG 制御のエミュレート機能

7.2 エミュレート機能を用いた電力評価

5 節で示した評価環境を用いて，提案手法の電力評価を行った．電力評価は，CPU の性能評価用のベンチマークを Linux 上で動作させ，ベンチマーク実行中の平均消費電力を計測することで行った．本実験では，計測対象のベンチマークとして，6.3 節の BEP ミス率閾値算出の際に用いた 9 種のベンチマークを使用した．提案方式の効果を検証するため，次に示す 3 種類の実行方式における消費電力を計測し，比較を行った．

(1) OS による PG 制御なし (常に PG 実行)
(2) 実温度と異なる温度向けの PG 制御
(3) 実行時のチップ温度に適した PG 制御 (提案手法)

PG 制御を行う (2) と (3) については，7.1 節で示したシミュレート機能を用いて，常時 PG 制御が可能な環境をエミュレートした状態で電力計測を行った．

上記の 3 種類の実行方式をそれぞれ適用した場合の，各ベンチマークの平均消費電力を図 13，図 14，図 15 に示す．OS による PG 制御を行わない方式 (1) と比較して，提案手法 (3) では，チップ温度 25°C のとき平均 10.5%, 最大 17.5%の消費電力削減，55°C のとき平均 5.2%, 最大 11.3%の消費電力削減，85°C のとき平均 2.2%, 最大 5.2%の消費電力削減に成功した．これにより，提案手法が実チップの省電力化に有効であることが確かめられた．評価結果より，チップ温度が低いほど提案手法の電力削減量が大きくなっていることが確認できる．これは，チップ温度が低いと BEP の値が大きく，高温時と比較して BEP ミス率が大きくなり，PG 制御によって抑制できるオーバヘッド電力が多くなったためである．

また，実際のチップ温度と異なる温度向けの PG 制御を行う方式 (2) と比較した場合も，提案手法の電力削減量は，それぞれのチップ温度において最も大きくなっていることが確認できる．特に，チップ温度 85°C においては，25°C や 55°C のときの BEP および BEP ミス率閾値を使った PG 制御によって，gsm, mcf 以外の 7 種のベンチマークの消費電力の増加が確認された．このように，実行時のチップ温度を取得することができず，誤った温度向けの PG 制御を行ってしまうと逆に電力が増加する場合があったが，提案手法では実行時のチップ温度を追従することで電力増加を防ぎ，PG 制御の省電力効果を改善できた．

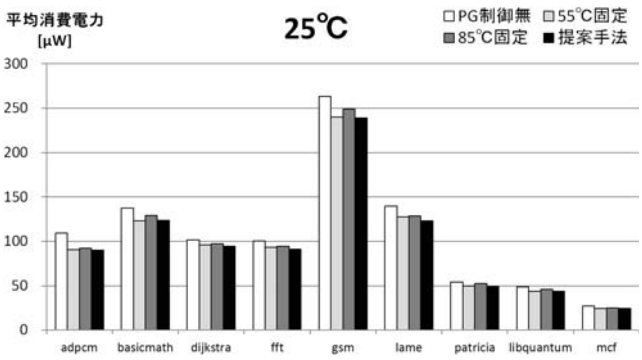

図 13　25°C における平均消費電力

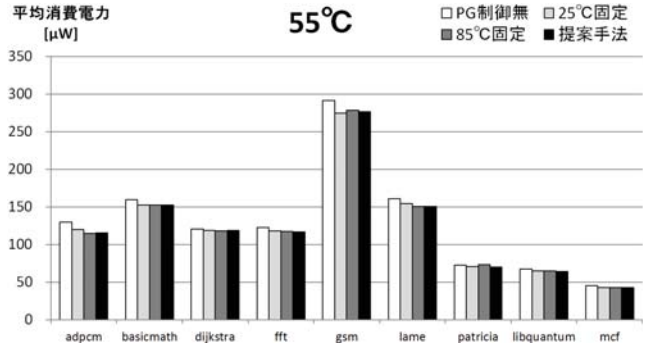

図 14　55°C における平均消費電力

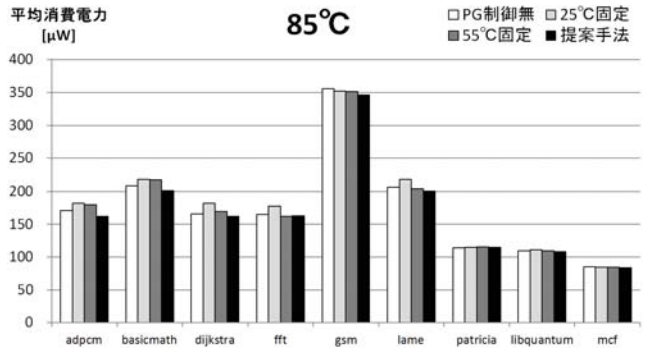

図 15　85°C における平均消費電力

7.3　エミュレート機能を使わない場合の電力評価

次に，4.5 節で示したハードウェア機能の有効性を検証するため，現 Geyser 環境において 7.1 節のエミュレート機能を使わずに本 PG 制御手法を実行した場合の評価を行い，7.2 節の結果と比較する．まず，エミュレート機能を使わない場合の PG 制御手法について述べる．4.5 節に示したように，現在の Geyser 実チップでは，PG を行わない場合はスリープ頻度情報を得られず，PG 制御を行うことができない．したがって，現 Geyser チップで提案手法を行うために制御方式を変更し，PG を抑制するポリシを適用した場合，次の PG 制御時は無条件に PG を実行するポリシを適用することで，断続的に PG 制御を行うものとする．

上記の PG 制御手法を適用した場合のベンチマーク実行中の平均消費電力を，チップ温度 25°C，55°C，85°C 環境において計測した．25°C における計測結果を図 16 に示す．評価結果には比較対象として，7.2 節で計測した OS による PG 制御を行わない方式 (1)，提案手法 (3) の消費電力も示している．現環境での PG 制御手法は，PG 制御を行わない場合と比較して，チップ温度 25°C のとき平均 5.8%，55°C のとき平均 2.5%，85°C のとき平均 0.8%の消費電力削減となった．これは，常に PG 制御が可能な場合 (提案手法) の電力削減量と比較すると，各チップ温度において，4〜5 割程度の電力削減量にとどまった．

現 Geyser チップでの PG 制御手法の省電力効果が低い理由は，PG を抑制するポリシを維持することができないためであると考えられる．PG を抑制するポリシを適用した場合，次の制御時は BEP ミス率の大小に関わらず PG を実行しなければならないため，適切なスリープポリシを適用できる時間は本来の約半分になり，電力削減効果が半減したと考えられる．以上の結果から，4.5 節で示したハードウェア機能によって，本 PG 制御手法の電力削減効果が向上することが確かめられた．

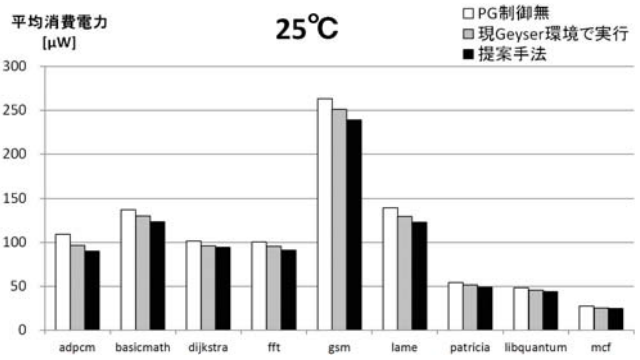

図 16　25°C におけるエミュレート機能あり/なしの平均消費電力

7.4　処理時間オーバヘッドの評価

提案手法の処理時間オーバヘッドによってスケジューラの処理時間が増加すると，アプリケーション実行に必要な消費エネルギーが増加し，提案手法の省電力効果が低下すると考えられる．そこで，PG 制御の処理時間オーバヘッドが消費エネルギーに与える影響を調査するため，提案手法の処理時間オーバヘッドを計測し，適用時の実行時間とエネルギー増加量の検証を行った．

表 5　PG 制御とスケジューラ全体の実行サイクル数

処理内容			サイクル	割合
PG 制御	温度推定	DT 計測	504	3.93%
		DT・温度変換	843	6.58%
	ポリシ決定	BEP ミス算出	1477	11.5%
		ポリシ決定	2743	21.4%
		ポリシ適用	26	0.20%
PG 制御全体			5567	43.5%
PG 制御適用前のスケジューラ			7484	58.4%
PG 制御適用後のスケジューラ			12810	100%

リークモニタを用いたチップ温度推定処理と BEP ミス率に基づく PG 制御処理の実行サイクル数，および各処理を実装したスケジューラ全体の実行サイクル数を表 5 に示す．なお，表内のサイクル数は処理ごとに個別に複数回計測し，平均化した値を掲載しているため，合計値は全体の値と一致しない場合がある．チップ温度推定処理と PG 制御の処理を合わせた処理時間は，スケジューラ全体の処理時間のうちの 43%を占め，非適用時のスケジューラの処理時間と比較して 77%程度となった．本研究の評価環境における Linux のタイムスライスは約 80ms としているため，本実行環境において提案手法を適用した際のアプリケーションの実行時間は，非適用時と比較して 0.18%増加する．この実行時間の増加に伴う消費エネルギーの増加量は，提

案手法による電力削減量に対して十分に小さく，処理時間オーバヘッドを考慮しても提案手法はプロセッサの省電力化に有効であると言える．

8. おわりに

本研究では，最粒度 PG 機能を備えた低電力プロセッサである Geyser を用いた，OS による PG 制御に取り組んだ．チップ温度によって変化するリーク電力の特性に対応した PG 制御を実現するため，リークモニタから得られる Detection Time を用いて実行時のチップ温度を推定し，得られたチップ温度に応じた PG 制御手法を提案した．提案手法を実現する制御システムを Linux プロセススケジューラに実装し，Geyser 実チップを用いて提案手法の電力評価を行った．また，現在の Geyser チップには実装されていない，提案手法の省電力効果を高めるハードウェア機能を提案し，評価実験の際にハードウェア機能を搭載した環境をエミュレートすることで，提案手法によって最大 17.5%の平均消費電力削減が可能であることを確かめた．以上により，提案した PG 制御手法がプロセッサの省電力化に有効であることを示した．今後の課題としては，リークモニタと一般的な温度センサの温度追従性の比較，本研究で提案したハードウェア機能を搭載した Geyser 実チップの開発およびそれを用いた提案手法の評価が挙げられる．

謝辞 本研究は, 日本学術振興会の科学研究費補助金 (基盤研究 (S) 25220002,「誘導結合を用いたビルディングブロック型計算システムの研究」) の助成を受けたものである．

参考文献

[1] 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・天野英晴著『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)』芝浦工業大学・慶応義塾大学 / 情報処理学会研究報告. SLDM, [システム LSI 設計技術] 2008(2), 43-48, 2008-01-16.

[2] 高橋弘明・小林弘明・坂本龍一・佐藤未来子・並木美太郎著『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』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] 2012-OS-120(4), 1-8, 2012-02-21

[3] Yan Meng, Timothy Sherwood, Ryan Kastner. Exploring the limits of leakage power reduction in caches, *ACM Transactions on Architecture and Code Optimization(TACO)*. Vol.2, Issue 3, pp. 221-246, September 2005.

[4] Stijn Eyerman and Lieven Eeckhout. Fine-grained dvfs using on-chip regulators. *ACM Trans. Archit. Code Optim.*, Vol. 8, No. 1, pp. 1:1-1:24, February 2011.

[5] Gu-Yeon Wei, David Brooks, Ali Durlov Khan, and Xiaoyao Liang. Instruction-driven clock scheduling with glitch mitigation. In *Proceedings of the 13th international symposium on Low power electronics and design*, ISLPED '08, pp. 357-362, 2008.

[6] F. Fallah and M. Pedram, "Standby and Active Leakage Current Control and Minimization in CMOS VLSI Circuits.", *IEICE Trans. on Electronics, Special Section on Low-Power LSI and Low-Power IP*, Vol. E88-C, No.4 Apr.2005, pp. 509-519.

[7] Kimimyoshi Usami and Yuya Goto and Kensaku Matsunaga and Satoshi Koyama and Daisuke Ikebuchi and Hideharu Amano and Hiroshi Nakamura, "On-Chip Detection Methodology for Break-Even Time of Power Gated Function Units", *Proceedings of the 17th IEEE/ACM International Symposium on Low-Power electronics*, pp. 241-246, 2011.

[8] Zhigang Hu, Alper Buyuktosunoglu, Viji Srinivasan, Victor Zyuban, Hans Jacobson, and Pradip Bose. Microarchitectural techniques for power gating of execution units. In *Proceedings of the 2004 International Symposium on Low Power Electronics and design*, pp. 32-37, 2004.

[9] Aviral Shrivastava, Deepa Kannan, Sarvesh Bhardwaj, and Sarma Vrudhula. Reducing functional unit power consumption and its variation using leakage sensors. *IEEE Transactions on VLSI Systems*, Vol. 18, No. 6, pp. 988-997, 2010.

[10] Xiuyi Zhou and Jun Yang. Performance-aware thermal management via task scheduling. *ACM Transactions on Architecture and Code Optimization*, Vol. 7, pp. 1-31, 2010.