Vol.2016-OS-137 No.9
2016/5/31

細粒度パワーゲーティングを用いたリアルタイムタスク向け最適省電力スケジューリング手法

佐藤未来子[†1]　榎本絵理[†1]　小柴篤史[†1]　並木美太郎[†1]

概要：細粒度パワーゲーティング(PG)を備えるプロセッサでは，未使用の演算ユニットの電源を遮断することで消費電力を削減する．しかし，電源が遮断されているスリープ期間の長さにより効果にばらつきが生じるため，ソフトウェアから PG を制御することでより省電力効果を高められる．本研究ではリアルタイムシステム特有の余裕時間を活用する PG 制御手法において，複数タスクを扱う際の最適省電力スケジューリング手法を提案する．電源の遮断時に電力的に損となる短いスリープ期間を，余裕時間を用いて電力的に得になるまで引き延ばす．本制御に必要となる複数タスク実行時の電力削減効果を最大とする PG 制御情報を最適省電力スケジューリング情報として事前に求め，OS が PG 制御に活用する．シミュレーション評価により，ハードウェア PG のみの場合に比べてエネルギー遅延積を最大で 4.0%削減できることを確認した．

キーワード：リアルタイム OS，省電力，パワーゲーティング

1. はじめに

近年，産業機器や自動車，スマートフォンなどの組み込みシステムは会社や家庭まで様々な場所で広く用いられ，普及率は年々増加の一途をたどっている．組み込みシステムにおいて搭載されるソフトウェアとしては，マルチメディア処理や通信処理など，リアルタイム性を保証する必要があるものが含まれる．このようなリアルタイムシステムでは，一定時間内での処理を保証するために高い性能が求められる．一方で，電力をバッテリでまかなうバッテリ駆動型システムが多いため，性能制約が厳しい中での省電力化が必要とされている．

近年のプロセッサは，性能向上に伴い LSI の消費電力が増加しており，プロセッサの省電力化は省エネルギーの観点から重要な課題となっている．本研究では，CPU の消費電力削減手法の一つであるパワーゲーティング(PG) を活用した省電力化の研究を進めている．PG は，未使用時の回路の電源電圧を遮断（スリープ）することで，スリープ中の回路のリーク電力を削減する技術である．特に，本研究では，WIPS(Whenever Idle Put to Sleep) と呼ばれる PG 戦略を適用可能な細粒度 PG [1] により省電力化を追求している．WIPS に基づく細粒度 PG では，マイクロプロセッサ自身が PG 対象の演算ユニットの使用状況を自律的に判断し，演算ユニットが利用される時のみ電力を供給することで，演算性能を損なうことなく高効率な電源制御を行う．しかし，電源の遮断/供給を切替える際に発生するオーバヘッド電力が存在するため，回路の電源が遮断されてから再び供給されるまでの期間（スリープ期間）が短い場合，PG のオーバヘッド電力による電力増加量が電源遮断中の電力削減量を上回る．したがって，短い間隔で電源電圧のスイッチングを頻繁に起こすようなプログラムに対しては，PG が逆に CPU の消費電力を増加させる場合がある．

本研究では，組み込みシステムにおいてリアルタイムシステム特有の余裕時間を利用して，短い間隔の PG をできる限り回避して省電力化を図る手法を提案する．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，次の CPU 割り当てまでの間に余裕時間が発生する．その余裕時間を用いて，電源の遮断/供給時に電力的な損を生じる短いスリープ期間を，電力的に得になるまで引き延ばすことでリーク電力の削減効果を得る．本研究ではリアルタイムシステムで実行する複数のタスクを対象に余裕時間を割り当てる際に，タスク実行時の電力削減効果を最大とするスリープ期間の引き延ばし戦略を事前に算出し，OS がその省電力スケジューリング情報をタスク切り替えのタイミングで活用して PG を制御することで，リアルタイムシステムの最適な省電力実行を実現する．本研究では，提案方式の効果を確認するために，シミュレーションにより，マイクロプロセッサの PG 適用時と，提案する省電力スケジューリング情報に基づく PG 制御を施す場合のエネルギー遅延積削減率を比較する．本評価により，提案手法によりエネルギー遅延積の削減可能であることを示す．

2. リアルタイムシステムにおける省電力化

2.1 余裕時間を用いた省電力化

本研究で扱うリアルタイムシステムでは，すべてのタスクの処理がデッドラインまでに終了する必要がある．そのため，リアルタイムシステムにおいて OS は，すべてのタスクがデッドラインを守れるよう適切な優先度に基づきタスクをスケジューリングする[2][3]．そして，すべてのタスクをデッドラインまでに完了させた場合に，次にタスクを実行するまでの間に余裕時間が発生する．図 1 に周期的なタスク実行と余裕時間の関係を示す．

リアルタイムシステム向けの省電力化では，この余裕時

†1 東京農工大学

間を利用した DVFS 制御による省電力化手法が提案されている．文献[4]では EDF スケジューリングにおけるリアルタイムタスク動作時にスケジューラで DVFS を制御する．余裕時間を近似的に見積もり，またタスク単位の動作特性に基づく性能予測を行うことで従来の予測精度を向上させ，DVFS の省電力性とリアルタイム性の改善を測っている．文献[5]では各タスクのデッドラインに基づいて一定区間における CPU 使用率を計算し，そのうちの余裕時間を用いて周波数を低下させる省電力手法を提案している．デッドライン時刻に基づき厳密に余裕時間の見積もりを行うことで，周波数の低下によるデッドラインミスを防ぎ，効果的な省電力化を可能とする．これらの研究は，余裕時間を用いてデッドラインミスを防ぎつつ省電力化を達成する点は本研究と同じであるが，本研究で対象としている細粒度 PG における省電力化を図る研究とは異なる．

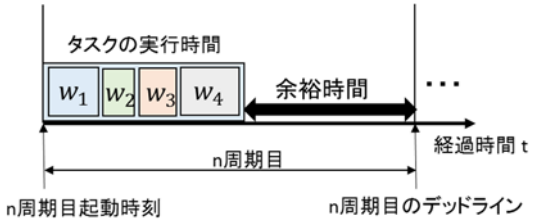

図 1　周期タスクと余裕時間の関係

2.2 リアルタイムシステムにおける PG 制御

細粒度 PG を適用した演算ユニットでは，図 2 に示すとおり PG により削減できる消費電力と電源の遮断/供給時に発生するオーバヘッド電力との損益分岐点（以下 BEP：Break Even Point）が存在する[6]．BEP 以下の間隔で電源の遮断/供給が頻繁に発生すると，PG により削減される電力よりオーバヘッド電力の方が大きくなる．逆に BEP よりもスリープ期間が長ければオーバヘッド電力よりも削減される電力が多くなる[7]．すなわち，単に PG による自律的な電源遮断/供給では BEP 以下のスリープ期間が多発して逆に電力が増加してしまう場合がある．したがって，BEP よりも長く演算ユニットがスリープする場合のみ PG を適用することが有効である．

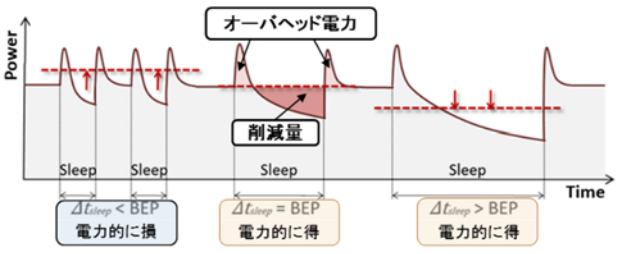

図 2　PG を適用したプロセッサの電力推移

本研究の先行研究[8][9]では，リアルタイムシステムで PG を適用する場合に，余裕時間を用いて BEP 以下のスリープ期間を BEP まで強制的に引き延ばす PG 制御方式を提案している．図 3 に PG 制御を行ったときの電力の推移とタスクの時間との関係を示す．これによってリアルタイム性を保証しつつ短いスリープを引き延ばす制御により電力を削減できることを示した．しかし，逆に電力的に損になる場合も考察しており，すべての BEP 未満のスリープを引き延ばすだけの単純な PG 制御では電力削減効果が得られないケースがあった．また，これらの先行研究はシングルタスクへ適用する方式であり，複数タスクへの余裕時間割り当ての任意性については追及されていない．複数タスクへの余裕時間の分配，タスク毎に分配した余裕時間を用いて引き延ばすスリープ対象の決定方法などが特に重要課題となる．

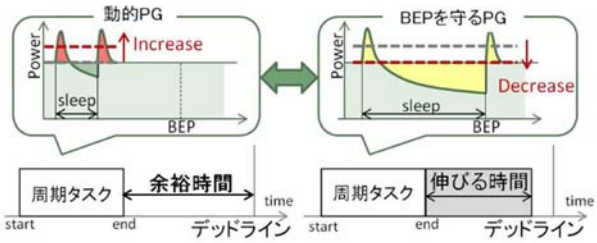

図 3　PG 制御とタスク処理の関係[9]

3. 目標

本研究では，複数タスクを扱うリアルタイムシステムにおいて，リアルタイム性を保証しながら余裕時間を利用した PG 制御を行うことで省電力性能の向上を目指す．先行研究では消費電力のみを評価指標としていたが，タスクの実行時間が引き延ばされることから，本研究ではタスク実行時の消費エネルギー削減を目標とする．

前章で述べたように，本研究の課題は複数タスクへの余裕時間割り当ての任意性である．余裕時間は有限であり，BEP 未満のスリープを引き延ばすことでより高い電力削減効果があるタスクに対して多くの余裕時間を割当てることが妥当である．本研究では，組み込み向けリアルタイムシステムであることを前提に，システムで稼働させるタスクの挙動を事前解析し，省電力効果が常に最大となるように余裕時間の配分を算出する．各タスクのどの演算ユニットのどのスリープ期間を引き延ばすかをスケジュールし，その省電力スケジューリング情報に基づき OS が PG 制御することにより最適な省電力化を図る．

4. システムモデル

本研究では，WIPS に基づく細粒度 PG をサポートする演算ユニットを備えるプロセッサを対象とする．本プロセッサでは，ソフトウェアが演算ユニットごとに細粒度 PG の有効/無効を切り替えられるものとする．プロセッサに備える PG ポリシとして，細粒度 PG を有効とする「WIPS」，細粒度 PG を無効とし常に演算ユニットへ電源を供給し続ける「ACTIVE」，ある条件のスリープを指定のスリープサイクル長まで強制的に引き延ばす「強制 PG」などを設定可

能とする．OS はこれらの PG ポリシの中で適切なものを省電力性能に応じて動的に切り替える．また，事前に行うタスクの挙動解析に必要な情報として，タスクの実行サイクル数や，WIPS 適用時の各演算ユニットのスリープサイクル長ごとの分布情報（スリープサイクル頻度情報）などを計測する機能を備えるものとする．なお，2.2 節で述べた BEP やスリープ時の電力に関しては，ハードウェア固有の値であり，PG 機能が搭載された回路の規模や実行中のチップ温度によって一意に定まる値である．本研究では，文献[6][7]の細粒度 PG プロセッサモデルを前提とし，それらの研究で求められている BEP およびスリープサイクル長ごとの平均リーク電力値を用いてタスクごとの消費電力を概算する．

5. 最適省電力スケジューリング手法

リアルタイムシステムにおける PG 制御の基本は，余裕時間を活用して BEP 以下のスリープ期間を強制的に引き延ばすことで，WIPS による細粒度 PG で発生するオーバヘッド電力を削減することである．この引き延ばす時間に関しては 2.2 節で述べたように，少なくとも BEP までは引き延ばし，余裕時間の許す限りできるだけ長く引き延ばすことで WIPS 時よりも電力を削減できる．しかし，余裕時間を複数のタスクで利用するため，BEP 未満のすべてのスリープに割り当てられるとは限らず，スリープを引き延ばすことでデッドラインミスをしたり，消費エネルギーを増大させたりする恐れがある．

そこで，本研究では各タスクの BEP 未満の一部のスリープだけを BEP まで引き延ばすことを考える．タスクごとに各演算ユニットの使用頻度が異なるため，各タスクを WIPS で実行したときよりも削減できる消費エネルギー量が最大となるようなスリープの引き延ばし方の組み合わせを求める必要がある．本研究では，組み込み向けリアルタイムシステムで，実行するタスクが決められていることを前提に，タスクの挙動特性を事前に解析して最適なスリープの引き伸ばし情報を算出し，強制 PG を制御するための情報（以下，省電力スケジューリング情報）とする．OS はタスク実行時にこの情報を用いて，強制 PG による PG 制御を施しながら，タスク実行時の省電力化を図る．

5.1 省電力スケジューリング情報の生成

省電力スケジューリング情報の算出のために，本研究では事前に WIPS で実行した場合のタスク i の実行時間 w_i，その際のスリープサイクル頻度情報 Fre_i をタスクごとに得ておく．それらの情報に基づいて，各タスクへの余裕時間の算出，余裕時間を配分した時の消費エネルギーの削減量を算出しながら，最適な省電力スケジューリング情報を求める．具体的には，強制的にどこまでのスリープをどこまで引き延ばすかという閾値情報の組み合わせ th_i を，全タスク i の各演算ユニット $unit$ に対して求める．

なお，強制 PG による省電力化が可能であることを確かめるために，本研究では扱う問題を単純化した．各タスクはすべて同一の周期 T で実行する周期タスクとし，実行中に周期が変化することはなく固定長であるとした．また，4 章で述べたように，プロセッサの特性に応じて定められるリーク電力と BEP の情報を事前に得ていることを前提とした．

以上の情報から省電力スケジューリング情報を生成する．以下，省電力スケジューリング情報の生成方法を示す．

5.1.1 余裕時間の算出

まず，省電力スケジューリング情報を生成する際に必要なシステムの余裕時間 Aff を算出する．周期 T から各タスク i の実行時間 w_i の総和を引いた値が Aff となる．周期 T，タスクの個数 M，各タスクの実行時間 w_i とした時の Aff の算出式を式(1)に示す．

$$Aff = T - \sum_{i=1}^{M} w_i \tag{1}$$

図 4 に示すように，本研究では余裕時間 Aff をスケジューリング対象の各タスクへ分配することで，余裕時間を用いた強制 PG を実現する．ただし，余裕時間は有限であることから，各タスク i に対して割当てる余裕時間 a_i の総和は常に余裕時間 Aff 以下である必要がある．式(2)にその制約条件式を示す．

$$Aff \geq \sum_{i=1}^{M} a_i \tag{2}$$

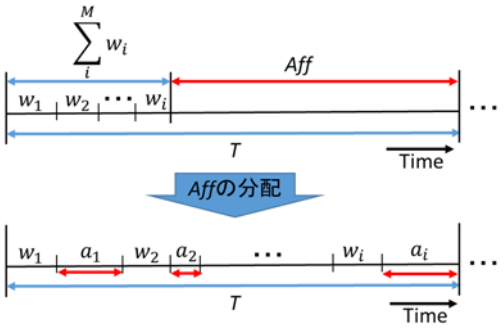

図 4　各タスクの実行時間と余裕時間の関係

5.1.2 タスクごとのスリープ引き延ばし情報の算出

5.1.1 項で求めた余裕時間 Aff を各タスクへ分配する際，式(2)の制約を満たす余裕時間 a_i の組み合わせは複数通りが考えられる．さらに，各タスクの a_i を演算ユニットごとに余裕時間 a_i^{unit} へ適切に配分する必要がある．どの組み合わせで最良の省電力化が図れるかを決定するには，各タスクの a_i^{unit} の組み合わせから消費エネルギーを見積もり，最適な余裕時間の割り当て方を選択する．

まず，消費エネルギーを見積もる前準備として，各タスク i のスリープサイクル頻度情報 Fre_i および事前に求めた

スリープサイクル長 x ごとの平均リーク電力値 L_x を用いて, 各演算ユニット $unit$ の BEP 未満のスリープサイクル長 x ごとに次の値を算出する．なおスリープサイクル頻度情報Fre_iは，各ユニットのスリープサイクル長 x の発生回数 $n_{i,x}^{unit}$で構成されたタスク実行時の情報である．

- x を BEP まで引き延ばすのに必要な余裕時間$a_{i,x}^{unit}$
- x を BEP まで引き延ばすことで得られるリーク電力削減量$z_{i,x}^{unit}$

$$a_{i,x}^{unit} = \left(BEP^{unit} - x\right) \times n_{i,x}^{unit} \quad (x = 1, 2, \ldots, BEP) \tag{3}$$

$$z_{i,x}^{unit} = \left(L_x^{unit} - L_{BEP}^{unit}\right) \times n_{i,x}^{unit} \quad \left(\begin{array}{c} x = 1, 2, \ldots, BEP \\ L_x^{unit} = x\text{時のリーク電力} \\ L_{BEP}^{unit} = BEP^{unit}\text{時のリーク電力} \end{array}\right) \tag{4}$$

タスクの特性によりスリープサイクル頻度は一定ではないため, BEP 未満のスリープのどの範囲を BEP まで引き延ばすかによって電力削減効果が異なると考えた．本研究では, 次の二つの方式で余裕時間の割り当て方式を検討した．

(a) スリープサイクル長が短いものから順に余裕時間を割当てる方式

(b) スリープサイクル長が BEP に近いものから順に余裕時間を割当てる方式

方式(a)は，スリープサイクル長 x が短いほど，オーバヘッド電力が大きいことを踏まえて，BEP 未満のスリープの中で 0 に近いものから順に余裕時間を与えて BEP まで引き延ばす方式である．図 5 にこの方式を適用した場合のスリープサイクル頻度情報の概略を示す．BEP までのスリープ引き延ばしに多くの余裕時間を要するが，スリープサイクル長が短い傾向のタスクに有効だと考える．

方式(b)は，スリープサイクル長が BEP に近いものを比較的少ない余裕時間で多くのスリープ対象を BEP まで引き延ばせることを踏まえて，BEP 未満のスリープの中で BEP に近いものから順に余裕時間を与えて BEP まで引き延ばす方式である．図 6 に示すとおり，BEP に近いスリープを BEP まで延ばすため，スリープサイクル長が BEP の近傍に多いタスクの場合に有効な方式である．

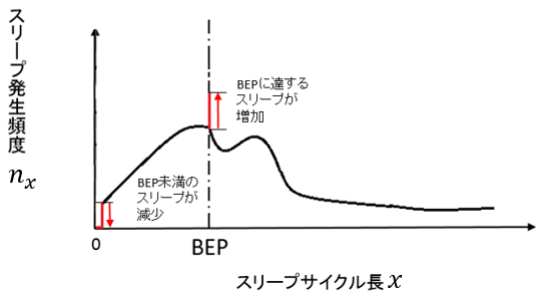

図 5　方式(a)適用時のスリープサイクル頻度情報の概略

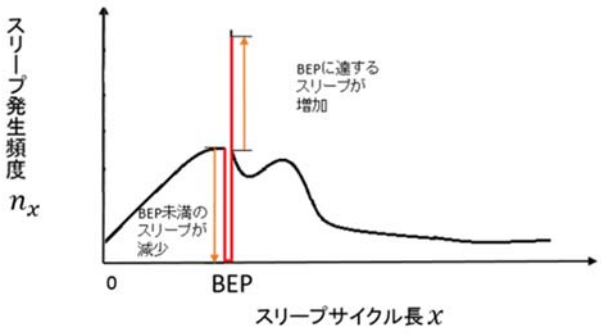

図 6　方式(b)適用時のスリープサイクル頻度情報の概略

方式(a)および(b)のいずれも BEP 未満のスリープをすべて BEP まで引き延ばせればオーバヘッド電力による消費電力増を回避可能である．しかし式(2)で示すようにシステム全体として引き延ばし可能なスリープサイクル長はAffまでである．この制約条件により，方式(a)および(b)において，どこまでのスリープサイクル長 x までを引き延ばし対象とするかという閾値情報th_iが存在する．閾値情報th_iを式(5)に示す．

$$th_i = \{x_i^{unit}\}_{unit=1,2,\ldots,m} \tag{5}$$

th_iの算出は，M 個のタスク i の m 個の演算ユニット $unit$ でスリープサイクル長 x まで引き延ばした際のリーク電力削減量の合計値が最大となるx_i^{unit}の組み合わせを求める最適化問題となる．本研究では，閾値情報th_iを省電力スケジューリング情報とする．方式(a)の場合のth_iの算出式を式(6)，方式(b)の算出式を式(7)に示す．

$$th_i = argmax \sum_{i=1}^{M} \sum_{unit=1}^{m} \sum_{k=1}^{x} z_{i,k}^{unit} \quad \left(\text{ただし，} Aff \geq \sum_{i=1}^{M} \sum_{unit=1}^{m} \sum_{k=1}^{x} a_{i,k}^{unit}\right) \tag{6}$$

$$th_i = argmax \sum_{i=1}^{M} \sum_{unit=1}^{m} \sum_{k=x}^{BEP} z_{i,k}^{unit} \quad \left(\text{ただし，} Aff \geq \sum_{i=1}^{M} \sum_{unit=1}^{m} \sum_{k=x}^{BEP} a_{i,k}^{unit}\right) \tag{7}$$

5.2 OS による省電力スケジューリング情報を用いた PG 制御

OS は 5.1 節で求めた省電力スケジューリング情報をもとに，タスクのディスパッチ時に，細粒度 PG のポリシを「強制 PG」とし，タスクコンテキストの一つとして閾値情報th_iをプロセッサへ設定することで省電力なタスク実行を図る．そのため，OS が本リアルタイムシステムで M 個タスクを実行する前に, 事前に算出しておいた M 個のタスクの閾値情報を OS へ通知しておく必要がある．本研究で

はその通知手段については言及しないが，タスクの実行時の引数，タスクと対応づけられたコンフィグレーション情報をファイルとして指定するなどが考えられる．

また，本研究で算出した余裕時間Affや閾値情報th_iは，メモリストールなど動的な条件を加味せずに算出しているため，これらを適用した場合にデッドラインをミスしてしまう可能性もある．その場合は，スリープの引き延ばしを中断するために，PG ポリシを「強制 PG」から「WIPS」へ切り替え，省電力化よりもリアルタイム性を優先する．

6. 強制 PG の機能設計

5 章で示した強制 PG 方式を実現するためには，細粒度 PG におけるハードウェアサポートが必要となる．本章では，文献[10]で提案されている細粒度 PG 機能を搭載した Geyser アーキテクチャを基盤とした機能設計を示す．

本プロセッサは 1 クロック単位で四つの演算ユニット（Alu, Shift, Mult, Div）の電源を制御することができ，ソフトウェアから PG を制御するためのインタフェースを備えている．ソフトウェアから演算ユニットごとに PG 制御を行えるよう PG ステータスレジスタを備え，ソフトウェアで PG ポリシを指定することで提案する強制 PG を適用する．本研究では，5 章で示した二つの強制 PG 方式を指定するために二つの PG ポリシを設ける．図 7 に PG ステータスレジスタの形式を示す．「強制 PG（モード１）」は方式(a)を指定する PG ポリシを，「強制 PG（モード２）」は方式(b)を指定する PG ポリシを指定可能とする．

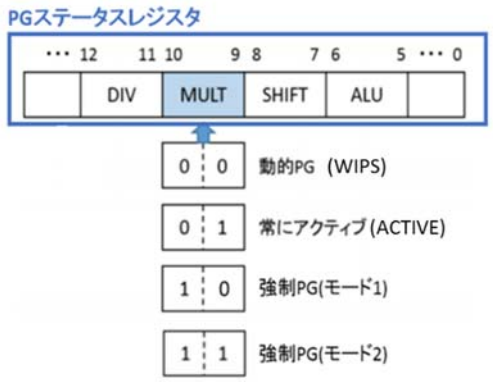

図 7　強制 PG を含む PG ステータスレジスタ

強制 PG では，5 章で求めた閾値情報th_iを用いて対象スリープを BEP まで引き延ばす．そのため，スリープをどれだけ引き延ばすかをプロセッサへ指定するための「スリープ制御レジスタ」と「閾値設定レジスタ」を Geyser アーキテクチャに設ける．図 8 に二つのレジスタの形式を示す．

スリープ制御レジスタは，演算ユニットごとに指定したクロックサイクル数までスリープを引き延ばすためのレジスタである．本研究の場合，スリープ制御レジスタには BEP を設定する．この機能はユニットが強制 PG で動作しているときにのみ有効であり，指定されたクロックサイクル数が 0 の場合 WIPS による PG 制御に準じる．

閾値設定レジスタは，スリープ制御レジスタで指定したスリープサイクル長まで引き延ばす対象となるスリープを演算ユニットごとに指定するためのレジスタである．本研究の場合，5 章に示す方式で求めた閾値情報のスリープサイクル長を設定する．この機能もユニットが強制 PG で動作しているときにのみ有効であり，モード 1 の場合は，本レジスタに指定した値以下のスリープをスリープ制御レジスタに設定したスリープサイクル長まで強制的にスリープさせ，モード 2 の場合は，本レジスタに指定した値以上のスリープをスリープ制御レジスタに設定したスリープサイクル長まで強制的にスリープさせる．

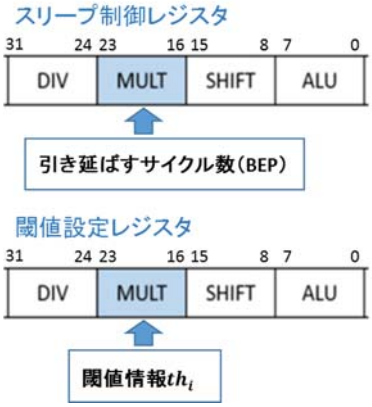

図 8　強制 PG 向け制御レジスタ

7. 評価

提案する二つの方式により省電力スケジューリング情報を算出し，その情報に基づいて複数タスクを PG 制御しながら実行させたときの省電力効果をシミュレーションにより評価した．その結果を示し，考察する．

7.1 評価方法

本研究では，単純なケースでの基礎評価とするために，タスクの個数は Matrix，DhryStone，Dijkstra の三種とし，扱うタスクの周期 T はすべて同一とした．また，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 を用いた．Geyser on FPGA はパフォーマンスカウンタを備えており，WIPS 時のタスク実行サイクル数やスリープサイクル頻度情報などを取得できる．しかし強制 PG に関する機能は実装されておらず，実際の消費電力を計測することができない．そこで，本評価では，WIPS でのタスク実行時における実行サイクル数 w_i とスリープサイクル頻度情報 Fre_i をパフォーマンスカウンタから取得し，本研究で提案する強制 PG でタスク実行した場合のリーク電力のエネルギー遅延積削減率を算出して評価指標とした．

エネルギー遅延積 E は，スリープサイクル長xにおけるリーク電力L_x^{unit}と，WIPS あるいは強制 PG 適用時のスリープの発生回数$n_{i,x}^{unit}$と，スリープサイクル長xを，スリープサイクル長の最小値１から最大値x_{max}までの全スリープを

対象に積算した値の総和となる．なお 4 章で述べたとおり，リーク電力L_x^{unit}と BEP はチップ温度により一意に定まる値であるため，エネルギー遅延積 E はチップ温度ごとに式(8)で算出する．

$$E_{temp} = \sum_{i=1}^{M} \sum_{unit=1}^{m} \sum_{k=1}^{x_{max}} \left(L_{i,k}^{unit} \times n_{i,k}^{unit} \times k \right)_{temp} \quad (8)$$

$$\left(\begin{array}{c} \text{ただし，} M = \text{タスク数:} 3 \\ m = \text{ユニット数:} 4 \end{array} \right)$$

そして，WIPS でタスク実行時のエネルギー遅延積をE_{temp}^{wips}，強制 PG でタスク実行時のエネルギー遅延積をE_{temp}^{sch}とした時のエネルギー遅延積削減率 Y は式(9)で算出する．

$$Y_{temp} = \frac{E_{temp}^{wips} - E_{temp}^{sch}}{E_{temp}^{wips}} \quad (9)$$

本評価のために取得した各タスク実行サイクル数 w_i を表 1 に示す．また，本評価では周期 T の長さに応じた省電力性能を比較するために，いくつかの周期 T を用意し，周期中の余裕時間の割合が異なる条件下において提案する PG 制御を行う場合のエネルギー遅延積を算出した． 表 2 に，シミュレーションのために用意した周期 T，周期中の余裕時間Aff，周期中の余裕時間の割合を示す．

表 1　タスクの実行時間（Mega cycle）

Matrix	DhryStone	Dijkstra
23.4	40.5	23.4

表 2　タスクの周期と余裕時間

#	周期 T (Mega cycle)	余裕時間 Aff (Mega cycle)	周期中の余裕時間の割合 (%)
1	87.6	0.39	0.4
2	88.0	0.78	0.9
3	88.8	1.56	1.8
4	90.3	3.13	3.5
5	93.5	6.25	6.7
6	99.7	12.5	12.5
7	112.2	25	22.3
8	137.2	50	36.4
9	187.2	100	53.4
10	287.2	200	69.6
11	387.2	300	77.5
12	487.2	400	82.1
13	587.2	500	85.1

7.2 評価結果

本評価を，表 2 における周期 T で方式(a)と(b)で生成した省電力スケジューリング情報に基づく PG 制御により得られるエネルギー遅延積削減率 Y の推移を，チップ温度 25℃，65℃，100℃，125℃ごとに図 9 に示す．

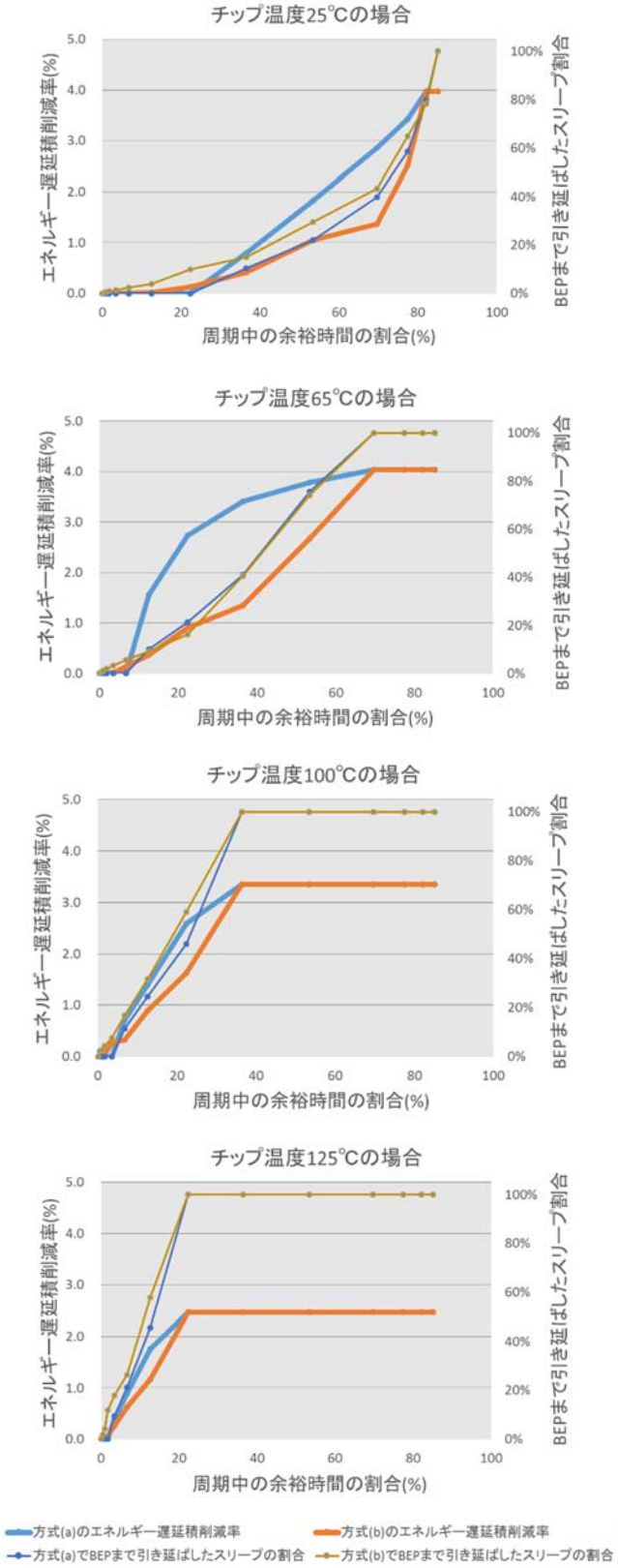

図 9　チップ温度に応じたエネルギー遅延積削減率

本評価結果から，周期中の余裕時間の割合が十分なケースについては，方式(a)と方式(b)ともにエネルギー遅延積を削減することができた．しかし，いずれの温度でも周期中

の余裕時間の割合を増やしていくとエネルギー遅延積削減率が飽和し，それ以上に余裕時間を増やした条件においても削減率が向上することはなかった．本評価で得られた最大のエネルギー遅延積削減率は，方式(a)・方式(b)両者に共通し，チップ温度 25℃の場合に 4.0%, 65℃の場合に 4.0%, 100℃の場合に 3.3%，125℃の場合に 2.5%であった．

飽和した要因を確認するために，BEP 未満のスリープ総数に対する BEP まで引き延ばしたスリープ総数の割合を調べ，図 9 に併記した．どの温度においても，BEP まで引き延ばしたスリープ割合が 100%となる場合には，リーク電力のエネルギー遅延積削減率が飽和していた．この結果より，提案方式によりエネルギー遅延積削減率が飽和してしまうのは，周期中の余裕時間Affを割り当てて引き延ばす対象となる BEP 未満のスリープがすべて BEP まで引き延ばされた状態となり，更に多くの余裕時間を実行時条件として与えた場合でも，引き延ばすことができるスリープが存在しないためであることが判明した．なお，チップ温度が高いほど早く飽和した理由としては，チップ温度が高いほど BEP が小さいことに起因すると考える．参考までに，本評価で用いた BEP の値を表 3 に示す．スリープサイクル頻度情報 Fre_i はチップ温度には関係ないタスク特性の情報であるため，同一タスクにおいて BEP が小さくなれば，本提案方式で引き延ばし対象のスリープが減少することは定性的に明らかである．

表 3　各ユニットの温度ごとの BEP[6]

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

7.3 考察

提案した「BEP 未満のスリープがすべて BEP まで引き延ばす」という強制 PG の制御は WIPS 適用時に比べて省電力化可能である一方で，エネルギー遅延積削減率が飽和するという結果を得た．このことを踏まえ，本節では余裕時間の許す限りさらにスリープを引き延ばすことでさらなる省電力効果を得られるかを考察する．

7.3.1 残余裕時間

エネルギー遅延積削減率が飽和した時点で，スリープの引き延ばしには適用できなかった残りの余裕時間を「残余裕時間」とし，引き延ばすスリープの対象を広げて残余裕時間を割り当てることで更なるエネルギー遅延積削減効果を図る．そのためにまず，残余裕時間がどれほど残されている状態なのかを算出した．システムの余裕時間Affから M 個のタスク i に割り当てた余裕時間a_iの総和を減ずることで残余裕時間Aff'を算出したものを表 4 に示す．この結果では，チップ温度が高温であるほど残余裕時間Aff'が多く残されることが示されている．したがって，この残余裕時間Aff'を利用して更にスリープを引き延ばすことでエネルギー遅延積削減効果を得られると考える．

表 4　エネルギー遅延積削減率飽和時のチップ温度別の残余裕時間[Mega cycle]

周期中の余裕時間[%]	25℃	65℃	100℃	125℃
22.3	N/A	N/A	N/A	1.2
36.4	N/A	N/A	4.3	26.2
53.4	N/A	N/A	54.3	76.2
69.6	N/A	75.7	154.3	176.2
77.5	N/A	175.7	254.3	276.2
82.1	39.6	275.7	354.3	376.2
85.1	139.6	375.7	454.3	476.2

7.3.2 残余裕時間を優先的に適用する演算ユニットの選定

残余裕時間も有限であるため，より効果の高い演算ユニットへ優先的に残余裕時間を割り当てることを検討する．そのために本評価で得られた結果においてエネルギー遅延積削減率の演算ユニットごとの内訳を調査し，BEP 未満のスリープではエネルギー遅延積削減効果が乏しかった演算ユニットを確認する．

図 10 に，チップ温度に対して方式(a)，方式(b)それぞれを適用した場合のエネルギー遅延積削減率の演算ユニットの内訳を示す．いずれのチップ温度においても，エネルギー遅延積削減率の内訳の大部分を ALU と Shift が占めていた．この結果を裏付けるために，本評価実験で扱った三つのタスクの演算ユニットごとの BEP 未満のスリープの発生回数の総和を調べた．チップ温度 25℃の場合，BEP 未満のスリープの総発生回数のうち Shift が最多の 60%を占め，次いで ALU が 27%，Mult が 12%，Div が 1%となり， 65℃以上の場合では BEP 未満のスリープの発生回数は Shift と ALU が全体の 99%以上を占め，特に 100℃と 125℃のケースにおいては Div が 0%となり，BEP 未満のスリープが発生しない．このように Mult および Div における引き延ばす対象となるスリープの発生回数が ALU および Shift に比べ少なかったことから，Mult と Div では提案手法によるエネルギー遅延積の削減効果を得ることができなかったと考える．そこで，Mult と Div に関して引き延ばす対象とするスリープの範囲を広げて提案方式を適用することで，エネルギー遅延積削減効果を向上できると考える．

7.3.3 残余裕時間を用いたエネルギー遅延積削減効果

残余裕時Aff'間を Mult と Div のスリープへ割り当てることでエネルギー遅延積削減効果の向上を見込めるかを概算する．本研究のプロセッサモデル Geyser では，スリープサイクル長が一定の値に達すると最小リーク電力に漸近し

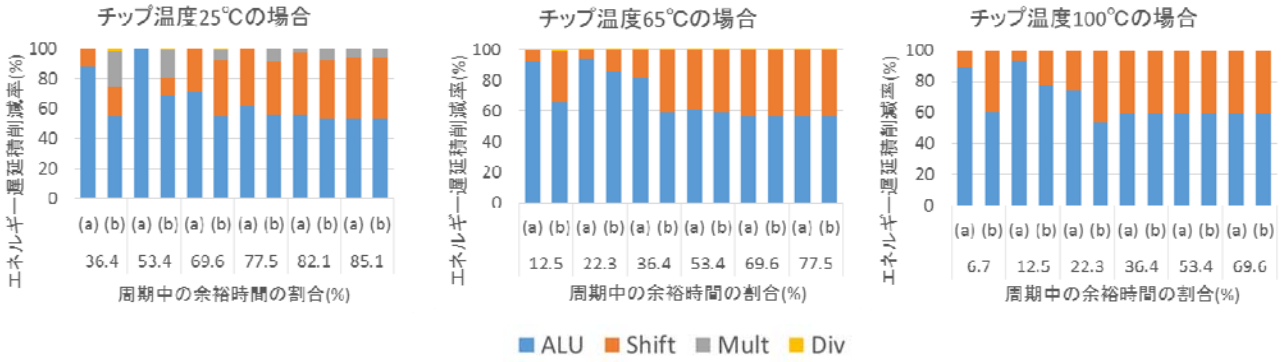

図 10 ユニット温度 25℃,65℃,100℃における方式(a)と方式(b)のエネルギー遅延積削減率内訳

てそれ以上削減できないことが先行研究[14]によって明らかとなっている．そこで，本研究ではそのスリープサイクル長を LLP(Least Leak Point)と呼び，Mult と Div については，飽和時の残余裕時間Aff'を利用して BEP 以上 LLP 未満のスリープを，残余裕時間の許す限り LLP まで引き延ばす PG 制御を施すと仮定し，7.1 節に示した評価方法と同様の方法でエネルギー遅延積削減率 Y'を求めた．その結果，BEP まで引き延ばすことで飽和していたエネルギー遅延積削減率 Y は最大 4.0%に留まったのに対して，残余裕時間Aff'を割り当てた後の遅延積削減率 Y'は最大 7.1%まで向上できるという結果を得た．

この結果から，オーバヘッドの存在により電力的に損である BEP 未満のスリープをすべて BEP まで引き延ばした状態で，もし残余裕時間Aff'が存在する場合には，BEP 以上のスリープに着目して更にスリープを引き延ばすことが省電力化に有効であると考える．本評価で扱った三つのタスクの場合には，Mult と Div のスリープを引き延ばすことが有効であった．別タスクの組み合わせを扱う場合にも，本研究で行ったようにタスクごとにスリープサイクル頻度情報 Fre_i を十分に解析することで，どのスリープを引き延ばすことが得であるかをあらかじめ算出して，強制 PG により PG 制御することは省電力化に有効であると考える．

8. おわりに

本研究では，リアルタイムシステム特有の余裕時間を活用する PG 制御手法において，複数タスクを扱う際の最適な省電力スケジューリング情報を適用する方式を提案した．BEP に基づく電源の遮断時に電力的に損となる短いスリープ期間を，余裕時間を用いて電力的に得になるよう BEP まで引き延ばす．本制御に必要となる複数タスク実行時の電力削減効果を最大とする PG 制御情報を最適省電力スケジューリング情報として事前に求め，OS が PG 制御に活用する．シミュレーション評価により，ハードウェアの自律的な PG（WIPS）の場合に比べてエネルギー遅延積を最大で 4.0%削減できることを確認した．本評価を考察して，残余裕時間の許す限りスリープを引き延ばすことが省電力化に有効であるという知見を得た．今後は本提案方式を，チップ温度変化やデッドラインミス発生などにも対応可能な PG 制御と共に OS へ実装する必要がある．

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. Computer Design, 2008, ICCD 2008. IEEE International Conference on, pp.612–617.

[2] C. L. Liu and J. Layland. Scheduling algorithms for multiprogramming in a hard real-time environment, Journal of the ACM, 10(1), 1973.

[3] Kriehi Ramamritham, John A. Stankovic. Scheduling Algorithms and Operating Systems Support for Real-Time Systems. Proceedings of the IEEE, 1994, Vol.82 No.1, pp.55-67.

[4] 林和宏，並木美太郎. EDF スケジューリングにおける DVFS を用いた CPU 省電力化手法. 情報処理学会「システムソフトウェアとオペレーティング・システム」第 113 回研究会, 2010, Vol.2010-OS-113, No.15, pp.1-8.

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

[6] 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, 2008, vol.107, no.414,VLD2007-111, pp.43-48.

[7] 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, 2008, vol.107, no.414, VLD2007-111, pp.37-42.

[8] 嶋田裕巳，小林弘明，高橋昭宏，坂本龍一，佐藤未来子，近藤正章，天野英晴，中村宏，並木美太郎. 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. 情報処理学会「システムソフトウェアとオペレーティング・システム」第 121 回研究発表会・「計算機アーキテクチャ」第 200 回研究発表会合同研究会, 2012, Vol.2012-OS-121&2012-ARC-200, No.16, pp.1-8.

[9] 嶋田 裕巳, 坂本 龍一, 塚本 潤, 和田 基, 佐藤 未来子, 並木 美太郎. リアルタイム OS による細粒度パワーゲーティング制御手法の検討. SWoPP 北九州 2013, 情報処理学会「システムソフトウェアとオペレーティング・システム」第 126 回研究発表会, 2013, Vol.2013-OS-126, No.6, pp.1-9.

[10] 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, 2007, pp.79-84.