Vol.2013-ARC-205 No.13
Vol.2013-OS-125 No.13
2013/4/26

FPUにおける細粒度パワーゲーティング制御手法の基礎的検討

塚本 潤[1]　和田 基[1]　嶋田 裕巳[1]　坂本 龍一[1]　佐藤 未来子[1]　近藤 正章[2]　宇佐美 公良[3]
天野 英晴[4]　中村 宏[5]　並木 美太郎[1]

概要：筆者らは，プロセッサの細粒度パワーゲーティング (PG) 制御方式を研究しており，整数演算器については GeyserCPU でその効果を検証してきた．本研究では FPU(Floating Point Unit) の省電力を実現するために PG 制御を細粒度に施した GeyserFPU の基本設計と RTL による FPU の実装を行い，FPU に細粒度 PG 制御手法を考察した．整数演算器と同様の制御手法を FPU にも適用し、電力削減効果をシミュレーションにより検証した．整数演算器と同様のハードウェアによる細粒度の動的 PG と、最適な PG 制御の効果を比較したところ，最適な PG 制御は，ユニット単体では平均 21.5%，GeyserFPU 全体では 28.8%の平均リーク電力が削減可能であることを示すことができた．

1. はじめに

スマートフォンやカメラなどの組込み機器の高性能化に伴い，FPU や GPU が搭載された組み込みプロセッサの需要が高まってきている．しかし FPU や GPU の消費電力は大きいことが問題となっている．組込み機器における消費電力はバッテリーの持続性などに影響するため，消費電力の増加は重要な問題となっている．さらにスマートフォンでは画像処理によって多くの電力を消費していることがわかっている [1]．画像処理は FPU や GPU で処理されるため，FPU や GPU の省電力化を実現する必要性は高い．そのため回路の省電力化を目指すクロックゲーティング [2]，DVFS[3]，パワーゲーティング (PG)[4] など様々な研究が行われている．また LSI のプロセスの微細化により，消費電力中に占めるリーク電力の割合が大きくなっている．リーク電力の削減には PG が有効である．

そこで PG 技術が細粒度に施されたプロセッサ Geyser が先行研究によって開発された [5]．Geyser の細粒度 PG 制御は時間的にも物理的にも細粒度であることが特徴である．時間的には一命令単位で電源の制御が可能であり，物理的には PG の適用粒度を演算器のユニットレベルまで細かく施すことが細粒度 PG 制御である．Geyser は電力的損益分岐点 (Break Even Point : BEP) を指標として PG を制御している．BEP とは PG を実行した時に，PG によって削減可能な電力と，PG によって発生するオーバヘッド電力が等しくなる点である．図 1 に示すように BEP 未満の PG を実行すると消費電力が増加するため，Geyser は BEP 未満の PG を実行しないように制御を行い，整数演算器において省電力化を達成した．

図 1　電力の推移

2. 目標

Geyser は整数演算器において省電力化を達成したが，FPU が搭載されていないという問題がある．そのため Geyser は FPU を用いない限られたプログラムで多くの評価を行なってきた．そこで Geyser に FPU(GeyserFPU) を搭載する．しかし FPU は一般的に回路規模が大きく FPU を搭載することで消費電力も増加してしまう問題もある．事前実験として RTL で記述された FPU を論理合成し回路規模を比較した結果，GeyserFPU は CPU である Geyser の約 3 割の回路規模になることが判明した．GeyserFPU の回路規模が大きいため，GeyserFPU の搭載による消費電力の増加は大きいことが予想される．

そこで本研究では FPU において省電力を実現するため

1　東京農工大学
2　電気通信大学
3　芝浦工業大学
4　慶應義塾大学
5　東京大学

ⓒ 2013 Information Processing Society of Japan

に，プロセスのコンテキストスイッチのタイミングで制御する方法と，コンパイラによって静的解析を行い BEP 未満の PG の実行を禁止する制御方法の二つを拡張し，GeyserFPU における細粒度 PG 制御手法を提案する．そして GeyserFPU における細粒度 PG 制御手法を実現するために GeyserFPU の設計および実装を行い，電力評価としてコンパイラによる細粒度 PG 制御手法を用いることによる削減されるであろう電力を見積もった．

3. Geyser プロセッサ

プロセッサ Geyser は MIPS R3000 をベースにして開発され，構成は MIPS R3000 と大部分が同じである．MIPS R3000 との主な差異としては Geyser の特徴でもある細粒度 PG 制御が施されていることである．Geyser は整数演算器に PG 技術が施されており，Geyser の PG 制御対象ユニットは ALU，SHIFT，MULT，DIV の四つのユニットである．また図 2 に示すように Geyser は CPU として，GeyserFPU は Geyser のコプロセッサとして動作する．

Geyser ではソフトウェアから細粒度 PG 制御を行うためのインタフェースとして PG 制御レジスタと PG 制御命令を備えており，このインタフェースを用いた二種類の細粒度 PG 制御手法が提案されている [6][7].

一つ目の制御手法は PG 制御レジスタを用いた細粒度 PG 制御手法である．Geyser は PG 制御をソフトウェアから行うためにシステム制御コプロセッサ (CP0) に PG 制御レジスタ (PGStatus レジスタ) が追加されている．この PG 制御レジスタをプロセスのコンテキストスイッチのタイミングで OS が制御する．PG 制御レジスタには動的 PG，キャッシュミス時に PG を実行，常に PG を実行しない，の三種類の動作モードが設定できる．これらの三種類の動作モードを Geyser の PG 制御対象ユニット (ALU，SHIF，MULT，DIV) ごとに設定でき，$3^4 = 81$ 通りのスリープポリシと呼ばれる組み合わせがある．このスリープポリシを用いた細粒度 PG 制御である「Linux における演算ユニットの電力特性を考慮した細粒度 PG 制御手法」[6] では，専用のパフォーマンスカウンタを用いて各演算ユニットの BEP ミス率を計算し，その相対に応じて動的に上記のスリープポリシを変更することで，タスクの特性に応じた細粒度 PG 制御を行い，整数演算器において省電力化を達成した．

二つ目の制御手法は PG 制御命令による細粒度 PG 制御手法である．この PG 制御命令による制御は，Geyser は命令セットのビットを書き換えることで実現する．MIPS R3000 アーキテクチャには R 形式の命令のオペコードが 6 ビットあり，この 6 ビットを書き換え次の二つの制御モードを指定する．

- ユニット使用後に電源供給を遮断 $(000000)_2$
- ユニット使用後も電源供給を持続 $(100111)_2$

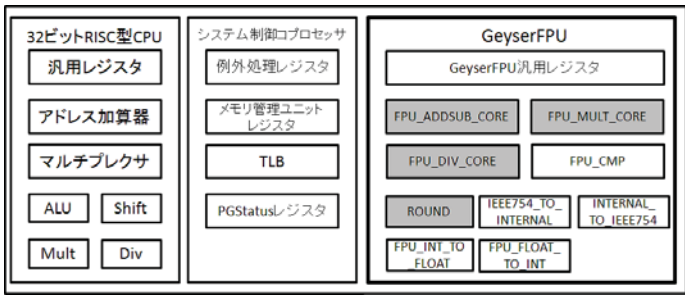

図 2 全体構成

通常の R 形式のオペコードは $(000000)_2$ であるが，オペコードを $(100111)_2$ とすることで PG 制御命令を実現する．この命令は事前に計測した BEP を用いて，コンパイル時にユニットの使用間隔であるアイドルサイクルを算出し，このアイドルサイクルが BEP 未満であれば PG を実行させない制御を行う．この PG 制御命令を用いた「OS における細粒度パワーゲーティング向けオブジェクトコードの実行時管理機構の研究」[7] の研究では，OS がユニットの温度向けのオブジェクトコードを管理し，実行時にユニットの温度に基づいて動的に最適なコードに切り替えて実行し，整数演算器において省電力化を達成した．

4. GeyserFPU の設計

この章ではまず細粒度 PG 制御を実現する GeyserFPU の構成と設計方針について述べる．次に，Geyser の整数演算器の電力制御手法を、GeyserFPU に拡張する方式について述べる．

4.1 GeyserFPU の構成

細粒度 PG 制御手法が施された GeyserFPU の設計について述べる．詳しい値は 5 章で述べるが，GeyserFPU は浮動小数点演算を行うため回路規模が大きく，演算器が GeyserFPU 全体の回路の多くを占め，その演算器の消費電力が大きいことが判明したため，演算器に PG 技術を施し，GeyserFPU の省電力化を目指した．また演算器の他に丸め処理を行うユニット (ROUND) も回路規模が大きいことがわかったため，このユニットにも PG 技術を施した．そして GeyserFPU の演算器を FPU_ADDSUB，FPU_MULT，FPU_DIV の三つの演算器に分割し，それぞれに PG 技術を施した．演算器を分割せずに PG を行った場合，実際には三つのユニットの中で一つのユニットしか使用されていない時に，他の二つのユニットは使用されていないにもかかわらず電源供給をするため無駄な電力が発生してしまう．そのため演算器を三つのユニットに分割し，使用しているユニットだけ電源供給を行い，演算器をまとめて PG を行う場合よりも多くの消費電力を削減することが狙いである．

GeyserFPU は Geyser のコプロセッサとして動作する．

ⓒ 2013 Information Processing Society of Japan

図 3　GeyserFPU を含むパイプラインの概要

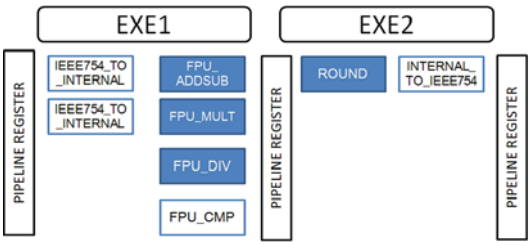

図 4　GeyserFPU のユニットとパイプラインの関係

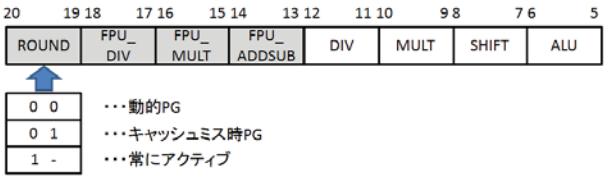

図 5　PG 制御レジスタ (CP0) のスリープポリシビット

表 1　命令と制御対象ユニットの対応

命令	ユニット
add.d add.s	FPU_ADDSUB, ROUND
sub.d sub.s	FPU_ADDSUB, ROUND
mul.d mul.s	FPU_MULT, ROUND
div.d div.s	FPU_DIV, ROUND
cvt.d.s cvt.s.d	FPU_ADDSUB, ROUND

そのため R3000 のコプロセッサである R3010 を参考にして GeyserFPU の設計を行った．これにより，GeyserFPU 用に命令セットの修正や追加を行う必要がない．また R3010 は六段パイプラインであるが，CPU である Geyser が五段パイプラインであり，異なる段数のパイプライン処理は制御が複雑になるため，GeyserFPU は Geyser の同じ段数の五段パイプラインとして設計を行った．パイプラインの全体構成を図 3 に示す．Geyser のパイプラインは IF，ID，EXE，MEM，WB である．浮動小数点演算を行う場合は IF，ID までは Geyser と共有し，ID ステージで浮動小数点演算であるか判定し，浮動小数点演算であれば EXE1，EXE2，FWB と進む．EXE1 で浮動小数点数の四則演算を行い，EXE2 で丸め処理を行う．そして FWB で GeyserFPU のレジスタに演算結果を格納する．ID ステージで GeyserFPU の処理へ分岐するようにすることで，Geyser の ID ステージのみを修正すれば良い．

また図 4 中の FPU_ADDSUB，FPU_MULT，FPU_DIV，ROUND のユニットが PG 制御対象ユニットである．EXE1 ではユニット IEEE754_TO_INTERNAL で演算を行うために IEEE754 の形式の値を内部で演算する形式に分割し，演算器で演算を行う．EXE2 では EXE1 での演算結果を ROUND ユニットで丸め処理を行い，INTERNAL_TO_IEEE754 で演算結果の値を IEEE754 の形式に戻しレジスタに値を格納する．

4.2　GeyserFPU における細粒度 PG 制御プログラミングインタフェース

ここでは GeyserFPU において PG をソフトウェアから制御を実現する二つのインタフェースについて述べる．Geyser で用いていた PG 制御レジスタと PG 制御命令の二つのインタフェースを，GeyserFPU の PG 制御対象ユニットに対しても実行できるように拡張を行った．

一つ目の制御方法は PG 制御レジスタによる細粒度 PG 制御方法である．3 章で述べたように，Geyser は細粒度 PG 制御をソフトウェアから行うために CP0 に PGStatus レジスタが追加されており，この PG 制御レジスタは特権モードで OS で制御する．PG 制御レジスタには Geyser の PG 制御対象ユニットは四つであるため，$3^4 = 81$ 通りのスリープポリシがある．GeyserFPU の PG 制御対象ユニットである四つのユニットに，Geyser と同様にそれぞれのユニットに対して三つの動作モードを割り当てると，$3^8 = 6561$ 通りのスリープポリシが存在する．また，PG 制御レジスタも図 5 のように GeyserFPU の四つのユニットの動作モードを格納するためにビット拡張する．図 5 中の 13 ビットから 20 ビットが GeyserFPU 用に拡張したビットである．このようにして PG 制御レジスタによって PG を制御する．

二つ目は PG 制御命令による制御方法である．この PG 制御命令による制御方法では，GeyserFPU の PG 制御対象ユニットを実行する R 形式の命令のオペコードを書き換え細粒度 PG 制御を行う．制御対象の FPU 命令は表 1 に示すとおり，EXE1 ステージと EXE2 ステージに含まれるユニットを用いて実行される．したがって，それぞれのユニットの電源供給をオペコードで指示できるようにするために，MIPS R3000 では使用されていない三種類のオペコードを追加し，これらを用いてユニット使用後に PG を実行させず電源供給を持続させる．浮動小数点演算を行う R 形式の命令を図 6 に示す．通常浮動小数点演算を行う R 形式の命令のオペコードは $(010001)_2$ である．オペコードが $(010101)_2$ のときは EXE1 ステージの演算ユニットのみ電源供給を持続させ，オペコードが $(010110)_2$ のときは EXE2 ステージの ROUND ユニットのみ電源供給を持続させ，オペコードが $(010111)_2$ のときは EXE1 ステージと EXE2 ステージのユニット共に電源供給を持続させる．このようにオペコードを書き換えることによって PG 制御対象ユニットの電源を制御する．

4.3　GeyserFPU の PG 制御機構

ここでは回路に対して電源のスイッチングを行うスリープコントローラについて述べる．スリープコントローラは

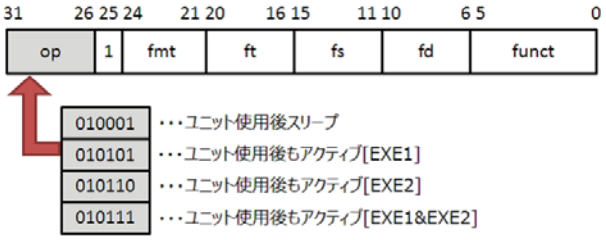

図 6 R 形式の命令セット

電源を供給している状態 (アクティブ) または電源を遮断している状態 (スリープ) のスイッチングを PG 制御対象ユニットに対して行う．GeyserFPU におけるスリープコントローラは，Geyser で用いられていたスリープコントローラを拡張させて実現する．図 7 に示すように，スリープコントローラは Geyser の PG 制御対象ユニット (ALU，SHIFT，MULT，DIV) と接続していたが，GeyserFPU の PG 制御対象ユニット (FPU_ADDSUB，FPU_MULT，FPU_DIV，ROUND) にも接続するように拡張を行う．

図 8 では EXE1 ステージに存在するユニットに対するスリープコントローラの通常の動作 (自動スリープ方式) を示している．スリープコントローラは IF ステージで命令をフェッチした時に命令を先見してどのユニットが使用されるか判定する．その後 ID ステージで命令がデコードされている間に，演算器の電源を ON にする．そして EXE1 ステージでユニットを使用する．ROUND ユニットの場合は EXE1 ステージで電源を ON にさせ，EXE2 ステージでユニットを使用する．ユニット使用後の EXE2 または FWB でそのユニットをスリープする．

PG 制御命令による制御手法ではユニットを使用した後，つまり EXE1 ステージのユニットは EXE2 で，EXE2 ステージのユニットは FWB でスリープするかアクティブを保持するのか判定する．PG 制御命令によってユニット使用後も電源供給を持続させるオペコードに書き換えられていた場合は，スリープコントローラはスリープさせずアクティブな状態を保持する．ユニット使用後に電源供給を遮断するオペコードの場合は PG のを実行する．

PG 制御レジスタによる制御では，PG Status レジスタに動的 PG のモードが割り当てられていれば，スリープコントローラは割り当てられている間は自動スリープ方式を実行する．キャッシュミス時に PG のモードであれば，スリープコントローラはキャッシュミス時にユニットに対して自動スリープ方式の PG を実行し，その他では PG は実行しない．常にアクティブであれば，PG を全く実行させない．このようにスリープコントローラが PG 制御対象ユニットの電源を管理する．

4.4 GeyserFPU の適切な PG 制御を行う細粒度 PG 制御手法

本研究では 4.2 節で述べた PG 制御命令を用いた細粒度

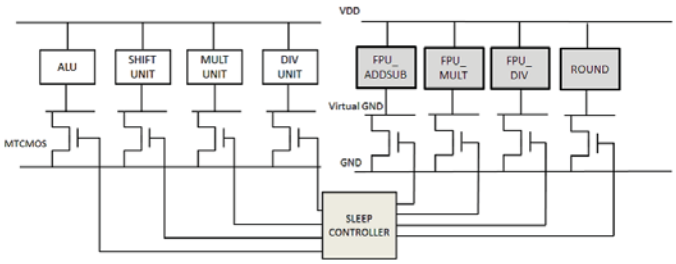

図 7 スリープコントローラの拡張

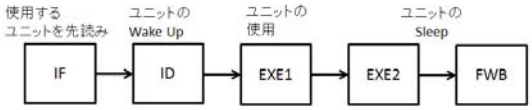

図 8 パイプラインとスリープコントローラの関係

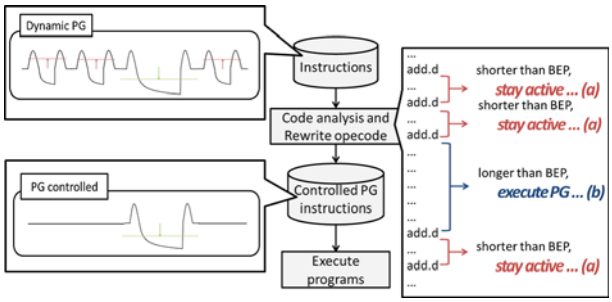

図 9 細粒度 PG 制御命令による制御の様子

PG 制御手法を提案する．提案手法ではアイドルサイクルが BEP 未満の場合は電力が増加するため PG を実行させず，BEP よりも長い場合のみ PG を実行し，GeyserFPU の省電力化を実現する．本 FPU は，PG 制御命令によってユニット使用後に PG を行うか，または PG せずにアクティブ状態を保持するかを制御できるため，アイドルサイクルが BEP 未満の場合はアクティブ状態を保持する命令列を実行する．また，OS のコンテキストとしてタスク全体のスリープポリシーを制御できる．本研究では，整数演算器と同様に，コンパイラにより PG 制御を行う命令列を生成する手法と，OS によりタスク全体の挙動特性を反映したスリープポリシー制御の両方を可能とする．

一つ目の手法では，整数演算器と同様に，コンパイラを用いて FPU を含む省電力な命令列を生成する．図 9 に示すように，コンパイラで実行命令列の静的解析を行う．コンパイラの省電力コード生成モジュールでは，GeyserFPU の PG 制御対象ユニットを使用する命令に着目し，ユニットを使用した最後の命令からその次にそのユニットを使用するまでの FPU の命令の間隔をアイドルサイクルとしてユニットごとに算出する．そのアイドルサイクルが BEP 未満の場合 PG を実行しないように命令のオペコードをアクティブ状態を保持する命令とする．整数演算器の場合でも，コンパイラの省電力効果は高く，FPU では整数演算器

ⓒ 2013 Information Processing Society of Japan

と BEP 特性が違うので，適切な命令列を生成することは重要である．

二つ目の手法は，CP0 の PG 制御を行うレジスタを，タスクのコンテキストとして保持し，温度やキャッシュミスなどの実行時情報を反映する方式である．整数演算器では，Geyser 向けの Geyser OS [8] を開発し，方式を Linux のスケジューラに導入し効果を確認した．さらに，BEP は温度に強く依存することから，コンパイラの生成するコードも特定温度向けのロードモジュールを生成する．実行時に適切な温度の命令列を実行する必要があるが，ページングを用いて適切な温度の命令列を実行する方式を筆者らは GeyserOS および Linux で成果をあげてきた．この方式を FPU にも適用できるように OS を拡張する．

このようにしてソフトウェアとハードウェアが連携することが提案する細粒度 PG 制御手法の特徴である．ただし，いずれの手法においても，重要なパラメータはユニットの BEP である．そこで，本研究ではまず第一に FPU の各ユニットの BEP をシミュレーションにより求め，各方式の効果を検証した．まず，ユニットごとの BEP を求め，その値から通常の PG 制御を行ったときの電力，さらに上記のような方式により理想的な PG 制御となったときの電力を机上計算により確認した．詳細を次章に示す．

5. GeyserFPU の実装

ここでは細粒度 PG 制御が施された GeyserFPU の実装について述べる．まずは GeyserFPU を構成するユニットの実装について述べ，その後 PG 制御対象ユニットに施した PG 技術の実装方法について述べる．そしてその PG 技術が施されたユニットに対して BEP を計測した結果を述べる．

5.1 GeyserFPU を構成するユニットの実装

GeyserFPU はハードウェア記述言語である Verilog-HDL で記述した．本研究では GeyserFPU は試作段階であるため FPGA 向けに実装を目指しているが，ASIC への展開を考慮しているため IP コア使用せずに，フルスクラッチで実装を行った．

GeyserFPU の内部は IEEE754 の形式の値を内部へ分割を行う FPU_IEEE754_INTERNAL，浮動小数点数の四則演算を行う FPU_ADDSUB，FPU_MULT，FPU_DIV，浮動小数点の比較演算を行う FPU_CMP，丸め処理を行う ROUND，そして演算結果を IEEE754 の形式に戻す INTERNAL_TO_IEEE754 のユニットが存在する．開発は Xilinx 社の Xilinx ISE Design Suite で行い，ISim 上で動作テスト行った．

GeyserFPU における細粒度 PG 制御手法の電力評価を行うためには，実際に PG 技術が施された回路を実装する必要がある．本研究で行った PG の実装方法は Row-Base 型 [9] である．65nm プロセスを使用して実装を行った．PG 制御対象ユニットを記述した RTL により記述されたファイルを Synopsys 社の Design Compiler で論理合成を行う．次に論理合成の出力結果を用いて Synopsys 社の IC Compiler で配置配線を行う．この配置配線では PG を実行するために必要なスイッチトランジスタなどを回路に配置する．表 2 を見ると PG 技術を施した PG 制御対象ユニットの面積が GeyserFPU 全体の多くを占めているのがわかる．次の節ではこの PG 技術を施した回路においてシミュレーションを行い BEP を計測する方法について述べる．

表 2 回路面積

ユニット	面積 [μm^2]
FPU_ADDSUB	184194
FPU_MULT	204912
FPU_DIV	223938
ROUND	115911
PG 制御対象外ユニット合計	39645

5.2 PG 制御対象ユニットの BEP

BEP は細粒度 PG 制御において重要な指標である．Geyser では各ユニットに対して PG 技術を施し，その回路でシミュレーションを行ない，PG によって削減可能な電力と PG によるオーバヘッド電力を計測し BEP を求めていた．そこで GeyserFPU の PG 制御対象ユニットでも Geyser と同様の手法で BEP を計測する．5.1 節で示した配置配線後に，配置配線の出力結果のストリームファイルから SPICE シミュレーションで必要なネットリストを取得する．そして Synopsys 社の HSIM で SPICE シミュレーションを行い，実際に回路を対して PG を実行した際の消費電力を計測する．この HSIM ではユニットがスリープしている時間を変化させて BEP を求める．PG を行わないアクティブな状態の消費電力と T[ns] だけ PG を実行したときの消費電力を比較する．この二つの消費電力が等しくなる点が BEP となる．

表 3 にユニットのアクティブ時のリーク電力を示す．アクティブ時のリーク電力も回路面積の結果と同様，PG 制御対象ユニットが GeyserFPU 全体の大部分を占めることが判明した．そして計測した BEP を表 4 にそれぞれの温度における BEP を示す．シミュレーション環境として，常温の 25 ℃と，回路の温度が上昇した場合の 65 ℃の二種類の環境で計測を行った．表 4 中のサイクル数は Geyser のユニットの BEP と同様に 200Mhz で換算したものであり，GeyserFPU のユニットは ISim において 200MHz での動作を確認している．この計測した BEP と細粒度 PG 制御手法を用いて次の 6 章で電力評価を行う．

6. 電力評価

本章では 4.4 節で述べた細粒度 PG 制御手法の電力削減

ⓒ 2013 Information Processing Society of Japan

表 3　アクティブ時のリーク電力

ユニット	25 ℃ [μW]	65 ℃ [μW]
FPU_ADDSUB	21.7	79.5
FPU_MULT	12.9	42.1
FPU_DIV	13.9	54.4
ROUND	8.4	31.5
PG 制御対象外ユニット合計	14.3	45.8
合計	71.2	253.3

表 4　GeyserFPU の BEP

	25 ℃	65 ℃
FPU_ADDSUB	260ns(52 サイクル)	75ns(15 サイクル)
FPU_MULT	535ns(107 サイクル)	215ns(43 サイクル)
FPU_DIV	735ns(147 サイクル)	185ns(37 サイクル)
ROUND	715ns(143 サイクル)	205ns(41 サイクル)

効果を検証する．ソフトウェアによる制御である細粒度 PG 制御手法の効果を GeyserFPU の実装前に見積もるために，本研究ではシミュレーションによる電力評価を行った．比較対象として，PG を実行しない場合 (常にアクティブ)，ユニット使用後必ず電源供給を遮断する場合 (動的 PG)，および細粒度 PG 制御の一つである細粒度 PG 制御手法の場合 (細粒度 PG) の三つの場合で比較を行う．電力評価はプログラムを実行した時の 1 サイクルあたりの平均リーク電力を算出して行う．

そこで Geyser がベースとした MIPS アーキテクチャの CPU と GeyserFPU がベースとした R3010 と同様のコプロセッサで構成されている仮想マシン [10] 上でプログラムを実行させ，そのプログラムを実行するのに使用された命令をトレースする．そしてユニットの使用間隔であるアイドルサイクルを計測し，このアイドルサイクルを PG が実行された間隔として置き換えることで，実機上でプログラムを動作させた場合と同様の評価を行うことが可能だと考えている．またプログラム実行に使用した命令列に対して命令列解析を行うため，このシミュレーションで得られる結果は細粒度 PG 制御手法の電力削減効果の理想値となる．

シミュレーションによる電力評価の概要を図 10 に示す．命令列を見て常にアクティブであればすべての命令において PG を実行せずに電源供給を続けたときの電力を算出し，動的 PG は BEP 未満のアイドルサイクルであっても PG を実行したときの電力を，細粒度 PG の場合は BEP 未満であれば PG を実行せず，BEP より長いアイドルサイクルのときのみ PG を実行する．このようにして各ユニットにおいて前記の三種類の PG 制御時の平均リーク電力を算出して評価を行った．

評価プログラムは，組込み分野のベンチマークである Mibench から高速フーリエ変換を実行する FFT と，浮動小数点演算性能を計測する Whetstone を用いた．

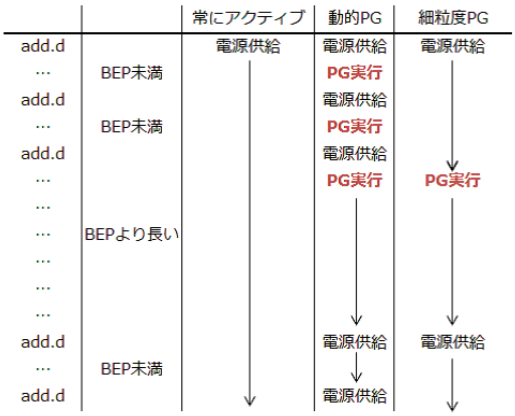

図 10　FPU_ADDSUB における電力評価の例

6.1　命令列解析によるアイドルサイクルの取得

アイドルサイクルの算出には Geyser プロジェクト内で開発されたアイドルサイクル算出プログラム [11] を使用する．このプログラムは Geyser 向けに PG 制御命令を埋め込む際に，一つの命令が 1 サイクルで実行するとした時，あるユニットを実行した命令から，次にまたそのユニットを実行する命令まで何サイクルあるかを算出する．そのサイクル数と BEP の情報を用いて PG を実行するかどうか決定する．しかしこのアイドルサイクル算出プログラムは Geyser 向けに開発されたため浮動小数点演算命令には対応していない．そのため本研究ではこのアイドルサイクル算出プログラムを GeyserFPU に対応するように修正を行った．4.2 節の表 1 で示した命令と使用するユニットの対応情報を追加し，GeyserFPU の PG 制御対象ユニットのアイドルサイクルも計測可能となった．これにより，仮想マシンで取得した命令からユニットの使用間隔をこのアイドルサイクル算出プログラムで計測し，その間隔を PG を実行された間隔とすることで本研究のシミュレーションによる電力評価が可能になる．

6.2　平均リーク電力算出方法

細粒度 PG 制御手法の電力削減効果を検証するために電力評価を行う．電力評価では比較対象である常にアクティブの場合，動的 PG の場合，4.2 節で示した細粒度 PG の場合の三つの場合それぞれの 1 サイクルあたりの平均リーク電力を算出して評価を行う．

本研究においてアクティブ時のリーク電力を $\overline{P_{active}}$，スリープ時のリーク電力を $\overline{P_{sleep}}$ と定義し，あるユニットにおける平均リーク電力 $\overline{P}$ は次のように定義する．

$$\overline{P} = \frac{\overline{P_{active}} \times T_{active} + \overline{P_{sleep}} \times T_{sleep}}{T_{total}} \quad (1)$$

T_{active} はアクティブ時のサイクル数，T_{sleep} はスリープ時のサイクル数であり，この二つのサイクル数の合計が T_{total} である．$\overline{P_{active}}$ と $\overline{P_{sleep}}$ は 5.2 節で述べた HSIM で求めている．よってユニットが使用されているまたは PG 制御命令によってアクティブ状態となっているサイクル数と，

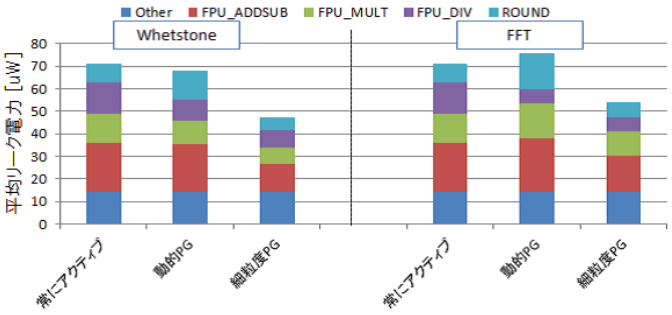

図 11 25 ℃環境における平均リーク電力

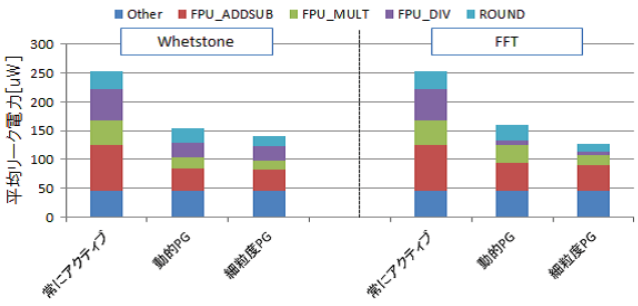

図 12 65 ℃環境における平均リーク電力

PG が実行されユニットがスリープしているサイクル数がわかれば，この式を用いて 1 サイクルあたりの平均リーク電力を算出して評価を行うことができる．

6.3 25 ℃環境における評価結果

25 ℃環境での Whetstone における平均リーク電力の評価の結果を図 11 に示す．PG を行わず常にアクティブにした場合の平均リーク電力は FPU_ADDSUB が 21.7[μW], FPU_MULT が 12.9[μW], FPU_DIV が 13.9[μW], ROUND が 8.4[μW]，そしてすべてのユニットを合計した GeyserFPU 全体では 71.2[μW] であった．

Whetstone において，細粒度 PG の場合は，動的 PG の場合と比較すると，FPU_ADDSUB が 42.1%, FPU_MULT が 27.9%, FPU_DIV が 18.1%, ROUND が 56.2%, GeyserFPU 全体で 28.8%の平均リーク電力が削減可能であることがわかった．FFT において，細粒度 PG の場合は，動的 PG の場合と比較すると，FPU_ADDSUB が 32.8%, FPU_MULT が 32.3%，FPU_DIV が 0%，ROUND が 55.2%，GeyserFPU 全体で 28.4%の平均リーク電力が削減可能であることが判明した．ROUND は四則演算の全てで使用されるためアイドルサイクルが短く，BEP 未満のスリープが頻出していた．そのため従来のユニット使用後に必ず PG を行う動的 PG の場合は，PG を行わない常にアクティブの場合よりも平均リーク電力が高いことが判明した．一方 FFT において FPU_DIV を使用する浮動小数点数の除算を行う演算が全くなかったため，FPU_DIV の平均リーク電力は動的 PG と細粒度 PG 制御手法の差が無かった．

25 ℃環境では動的 PG と細粒度 PG 制御手法の電力削減効果の差が無いユニットも存在したが，多くのユニットで細粒度 PG 制御手法の電力削減効果が動的 PG よりも優れていた．よって 25 ℃の環境では提案する細粒度 PG 制御手法は FPU の省電力化を達成する手法として有効であることが確認できた．

6.4 65 ℃環境における評価実験結果

65 ℃環境での Whetstone における平均リーク電力の評価の結果を図 12 に示す．PG を行わず，常にアクティブにした場合の平均リーク電力は FPU_ADDSUB が 79.5[μW], FPU_MULT が 42.1[μW], FPU_DIV が 54.4[μW], ROUND が 31.5[μW]，そしてすべてのユニットを合計した GeyserFPU 全体では 253.3[μW] であった．

Whetstone において，細粒度 PG 制御手法の場合は，動的 PG の場合と比較すると，FPU_ADDSUB が 7.9%, FPU_MULT が 14.5%, FPU_DIV が 0%, ROUND が 28.2%，GeyserFPU 全体で 8.3%の平均リーク電力が削減可能であることがわかった．FFT において，細粒度 PG 制御手法の場合は，動的 PG の場合と比較すると，FPU_ADDSUB が 8.6%, FPU_MULT が 48.5%, FPU_DIV が 0%, ROUND が 49.2%，GeyserFPU 全体で 20.4%の平均リーク電力が削減可能であることが判明した．25 ℃環境の場合と異なり，動的 PG のみでも平均リーク電力は大きく削減できており，特に Whetstone における FPU_ADDSUB では動的 PG の場合は，常にアクティブの場合より約 50%の平均リーク電力を削減可能である．ROUND は 25 ℃環境の場合と同様で，BEP 未満のアイドルサイクルが頻出しているため細粒度 PG 制御手法による電力削減効果は大きい．FFT においては動的 PG 場合の約 50%の平均リーク電力を削減可能である．

65 ℃環境では動的 PG であっても常にアクティブの状態と比較すると大幅な電力削減が可能であったが，さらにソフトウェアによる制御を行う細粒度 PG 制御手法を施すことで動的 PG よりもさらに電力が削減可能できることが判明した．25 ℃環境でも FPU における細粒度 PG 制御手法よる制御手法の高い電力削減効果を確認できたため，チップのコア温度が変化しても細粒度 PG 制御手法の電力削減効果が高いことを本研究の電力評価によって確認できた．

6.5 温度変化に応じた制御手法

電力評価の結果，細粒度 PG 制御手法は改良すべき点として温度変化に応じた制御が挙げられる．ROUND は動的 PG の場合と比較すると，大幅に平均リーク電力を削減可能であることが判明したが，25 ℃環境では，常にアクティブと比較すると，削減可能な平均リーク電力は少ない．そこで 25 ℃環境で，ROUND のみ PG を実行させずに常にアクティブにさせ，他のユニットは細粒度 PG 制御手法を実行させた場合 (細粒度 PG-2) について検証を行った．そ

ⓒ 2013 Information Processing Society of Japan

表 5　平均リーク電力の比較

	細粒度 PG[μW]	細粒度 PG-2[μW]
Whetstone	47.3	50.0
FFT	54.3	55.6

の結果を表 5 に示す.

GeyserFPU 全体における二つの制御の平均リーク電力の違いはとても小さいことがわかる．これは 25 ℃環境ではアイドルサイクルが短いものが多く，BEP よりも長い電力を削減可能なスリープが少ないため，常にアクティブの場合の平均リーク電力との差がわずかとなってしまった．また今回の細粒度 PG 制御手法の評価結果は BEP 未満のスリープが全くない最適な場合の評価であるが，実際には静的解析では予測の難しい分岐によって，禁止した BEP より短い時間のスリープが実行されてしまう可能性がある．よって，25 ℃では ROUND ユニットの PG は実行せず，65 ℃では PG 制御を行うといった細粒度 PG 制御手法を提案するとより効果的である．

7. おわりに

本研究では FPU の省電力化を実現する細粒度 PG 制御手法とそれを実現する GeyserFPU を提案した．また GeyserFPU の実行モデルであるパイプラインの設計，構成するユニットの設計と実装を行い，GeyserFPU の四つのユニットに PG 技術を施した．そして細粒度 PG 制御手法の電力評価を行った．シミュレーションによる電力評価の結果，細粒度 PG 制御手法は，ユニット使用後必ず PG を実行する動的 PG の場合と比較して，ユニット単体では最大で 56.2%，平均で 28.2%，GeyserFPU 全体では最大で 28.8%，平均で 21.5%の平均リーク電力を削減可能であることが判明した．本研究が提案する細粒度 PG 制御手法は，各ユニットごとに BEP を計測し BEP 未満のスリープを禁止させる制御を実行することで，従来の動的 PG では解決できなかった電力的に不利なスリープの問題を解決し，その BEP 未満のスリープ禁止する制御手法である細粒度 PG 制御手法の電力削減効果が高いことが確認できた．これにより細粒度 PG 制御手法の有効性を示すことが達成できた.

今後の課題として GeyserFPU の実装の完了が挙げられる．今回の電力評価は GeyserFPU の Geyser への組込みが完了していないため，シミュレーションによる評価であった．そのため今後は GeyserFPU を Geyser へ移植し，実機上で提案する制御手法の電力評価を行う必要があると考えている．さらに 6.5 節で述べた温度変化に対応するための細粒度 PG 制御手法の改良をする必要があると考えている.

謝辞　本研究は東京大学大規模集積システム設計教育研究センターを通し，シノプシス株式会社の協力で行われたものです.

参考文献

[1] Aaron Carroll, Gernot Heiser ：”An Analysis of Power Consumption in a Smartphone”, Proceeding of USENIX-ATC’10 Proceedings of the 2010 USENIX conference on USENIX annual technical conference.

[2] Nam Sung Kim, Jun Seomun, Abhishek Sinkar, Jungseob Lee, Tae Hee Han, Ken Choi, Youngsoo Shin ：”Design of Lowpower RISC Processor by Appying Clock Gating Technique”, International Journal of Engineering Research and Applications(IJERA), Vol.2, Issue 3, May-Jun 2012, pp.94-99.

[3] Hiroshi Sasaki, Yoshimichi Ikeda, Masaaki Kondo, Hiroshi Nakamura : ”An Intra-task Dvfs Technique Based on Statistical Analysis of Hardware Events”, Proceeding CF ’07 Proceedings of the 4th international conference on Computing frontiers, pp.123-130 (2007-5).

[4] Zhigang Hu, Alper Buyuktosunoglu, Viji Srinivasan, Victor Zyuban, Hans Jacobson, Pradip Bose, IBM T.J. Watson Reserch Center :”Microarchitectural Techniques for Power Gating of Execution Units”, Proceedings of the 2004 International Symposium on Low Power Electronics and Design (ISLPED 04). pp32-37 (2004-4).

[5] 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「計算機アーキテクチャ研究会報告」，計算機アーキテクチャ研究会報告 2007(55), 79-84, 2007-5.

[6] 高橋昭宏，小林弘明，坂本龍一，佐藤未来子，並木美太郎：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（OS）」，Vol.2012-OS-120，No.4，pp.1-8 (2012-2).

[7] 小林弘明，茂木勇，木村一樹，薦田登志矢，佐藤未来子，近藤正章，中村宏，並木美太郎：OS における細粒度パワーゲーティング向けオブジェクトコードの実行時管理機構の研究，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（OS）」，Vol.2011-OS-117，No.1，pp.1-8 (2011-4).

[8] 砂田徹也，関直臣，中田光貴，香嶋俊裕，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 プロセッサにおける OS の試作とシミュレーションによる電力評価，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（OS）」，Vol.2008-OS-108，No.35，pp.163-170 (2008-4).

[9] 王蔚涵，太田雄也，LEI Zhao，石井義史，宇佐美公良，天野英晴：細粒度パワーゲーティングを適用した演算モジュールの構成方式に関する研究電子情報通信学会技術研究報告「CPSY, コンピュータシステム」, Vol.111, No.163, CPSY2011-9，pp.1-6，(2011-7).

[10] 松尾和弥，佐藤未来子，並木美太郎：QEMU を用いた命令, アドレストレーサの実現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（OS）」，Vol.2007-OS-105，No.36，pp.39-46 (2012-4).

[11] 薦田登志矢，佐々木広，近藤正章，中村宏：リーク電力削減のためのコンパイラによる細粒度スリープ制御情報処理学会研究報告 SACSIS2009，pp.11-18 (2009-5).

ⓒ 2013 Information Processing Society of Japan