社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－ARC－173（9）
2007／5／31

MIPS R3000における細粒度動的スリープ方式の提案

関 直臣† 長谷川揚平† 天野 英晴† 大久保直昭†† 武田 晴大††
香嶋 俊裕†† 白井 利明†† 宇佐美公良†† 近藤 正章††† 中村 宏†††

† 慶應義塾大学大学院理工学研究科〒223.8522 神奈川県横浜市港北区日吉 3.14.1
†† 芝浦工業大学工学部情報工学科〒135.8548 東京都江東区豊洲 3-7-5
††† 東京大学先端科学技術研究センター〒153.8904 東京都目黒区駒場 4.6.1
E-mail: †seki@am.ics.keio.ac.jp

あらまし 本報告はパワーゲーティング(PG)を使った細粒度動的スリープ制御による消費電力削減手法を提案する。細粒度とは演算部を乗算器、除算器、シフタとそれ以外の演算の4つに分割したそれぞれのユニットを意味する。PGを用いたこれらの制御機構をMIPS R3000プロセッサに適用し、ASPLA 90nmライブラリによる配置配線、レイアウトまで行い電力と面積の評価を行った。また、電力評価を取るためにRTLシミューレションで4つの組込み系向けベンチマークアプリケーションを動作させて各ユニットの使用頻度を解析した。この結果、アプケーションの平均での消費電力の低減効果は、リーク電力で31%、ダイナミック電力は59%であった。全体では55%の消費電力削減を達成した。スリープ制御の実装によって生じたエリアオーバーヘッドは34%であった。
キーワード 低消費電力、動的スリープ制御、パワーゲーティング, リーク電力

A Fine Grain Dynamic Sleep Control Scheme in MIPS R3000

Naomi SEKI†, Yohei HASEGAWA†, Hideharu AMANO†, Naoaki OHKUBO††, Seidai TAKEDA††,
Toshihiro KASHIMA††, Toshiaki SHIRAI††, Kimiyoshi USAMI††, Masaaki KONDO†††, and
Hiroshi NAKAMURA†††

† Graduate School of Science and Technology, Keio University 3-14-1 Hiyoshi, Kouhoku-ku, Yokohama, 223.8522 Japan
†† Department of Information Science and Engineering, Shibaura Institute of Technology 3-7-5 Toyosu, Kohtoh-ku, Tokyo 135.8548, Japan
††† Research Center for Advanced Science and Technology, The University of Tokyo 4.6.1 Komaba, Meguro-ku, Tokyo, 153.8904 Japan
E-mail: †seki@am.ics.keio.ac.jp

Abstract A novel fine grain power gating technique in a processor is proposed for saving leakage power in the future semiconductor processes. By dividing an execution unit into four small units: multiplier, divider, shifter and others and cut off the power dynamically based on the operation, both dynamic and static power can be reduced. We implemented the chip layout of MIPS R3000 with the proposed mechanism using 90nm CMOS technology, and evaluated area and consuming power. Evaluation results of some benchmark programs for embedded application show that 31% leakage power and 59% dynamic power are reduced in avarage with 34% area overhead.
Key words Dynamic Sleep Control, Power Gating, Leak Power

1. はじめに

プロセスルールの微細化に伴ってリーク電力の占める割合が増大している。90nm以降の世代のチップでは、リーク電力の増大は特に顕著であり、従来のゲーテッドクロックや動作周波数、動作電圧の動的制御など、ダイナミック電力を対象とした省電力化手法では、今後増えるリーク電力に太刀打ちすることができない。モバイル機器では、将来に渡って低電力なプロセッ

サが必要となることから、プロセッサのリーク電力を削減する技術の確立が急務である。「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」プロジェクト[1]は、回路技術、アーキテクチャ、システムソフトウェアの各階層が連携、協力し、革新的な電力制御を実現することで高性能システムLSIの消費電力を格段に低下させることを狙っている。このプロジェクトの中でも、プロセッサのリーク電力の削減技術の確立は最も大きなテーマの一つである。

リーク電力を削減する手法としてパワーゲーティングがある。この手法は、回路の一部に対して電源供給を断つことで、リーク電力を削減する技術である。しかし電源供給を断った部分のリーク電力は削減されるが、パワースイッチを切り替える際のレイテンシや切り替え用のトランジスタのエリアオーバーヘッドなどの問題がある。このため、従来、パワーゲーティングは、マルチコアシステムにおけるコア単位等、プロセッサ全体に相当するサイズについて、長いタイムスパンで行われてきた[5][6]。

しかし、最近、パワースイッチのウェイクアップタイムの高速化やレイアウト時におけるパワースイッチの挿入の最適化など回路レベルの技術の進展[4]により、パワーゲーティングを用いるためのオーバーヘッドは軽減されている。プロジェクトの一環として、芝浦工大では乗算器を分割して演算データに応じたパワーゲーティングを施すことが可能なチップPinnacle[3]を開発している。

そこで、本研究では、この手法をCPUの比較的小さい単位に対して適用し、必要に応じて、動的にスリープさせる手法を提案する。CPUの中の論理ブロックの中で、例えば乗算器などは面積が大きく、リーク電力も大きいにも関わらず、一般的なプログラムではさほど使用頻度は高くない。このような論理ブロックをこまめにスリープさせれば、全体として消費電力を大きく削減できる可能性がある。そこで、本報告では、プロセッサの演算器部分をユニットというより小さな単位に分割し、これらをプログラムの命令に応じて、パワーゲーティングを施す手法を提案する。ここでは、これらの演算器以下のレベルでのパワーゲーティングを、従来のプロセッサ全体などのパワーゲーティングと区別して細粒度パワーゲーティングと呼ぶ。本報告では細粒度パワーゲーティング手法とその基本となる動的スリープ制御について述べ、その効果についての予備的評価を行う。

2. パワーゲーティング

本研究で用いるパワーゲーティング[4]について紹介する。

パワーゲーティングは、電源供給の制御を行うものであり、スリープ制御信号により動作するスリープトランジスタによって実現する。実際には複数のスリープトランジスタでひとつの回路ブロックを制御し、これをパワースイッチ(PS)と呼ぶ。

スリープトランジスタとして、NMOSかPMOSが用いられ、PMOSはVDDラインと回路の間に、NMOSはGroundラインと回路の間に挿入される。スリープトランジスタは、通常のCMOSに比べて動作は遅いが、高いVthのトランジスタであるため、OFFにした状態でのリーク電力を低くすることができる。

図1は、NMOSのスリープトランジスタを用いたパワーゲーティングの様子である。

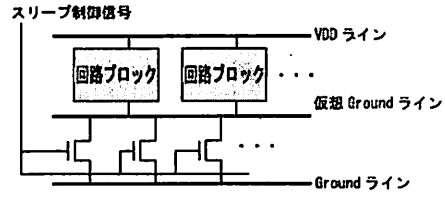

図1 パワーゲーティングの概要

一つの制御信号について、複数の回路ブロックと一つの仮想Groundライン、複数のスリープトランジスタが対応する。仮想GroundラインはPSがONの場合にはGroundラインと同じ電位になり、OFFの場合にはVDDラインと同じ電位になる。パワーゲーティングを行うときには、対象となる回路はその規模に応じて複数の回路ブロックに分割され、同じ制御信号で動作するスリープトランジスタも複数挿入される。これは回路を安定してPSでON/OFFするためである。大規模な回路を一つのスリープトランジスタで制御しようとすると、仮想Groundラインの電圧を制御するのに時間がかかってしまう。また、そのスリープトランジスタへの負荷も大きくなってしまうため、回路規模に応じて適切な個数のスリープトランジスタを挿入する。

PSのON/OFFをモード切替と言う。モード切替では、スリープ信号の伝播や仮想Groundラインの放電によって、遅延やエネルギーのオーバーヘッドが発生する。切替遅延はアクティブへの遷移(ウェイクアップ)時とスリープへの遷移(シャットダウン)時の二種類がある。動作上ウェイクアップが問に合わないと回路処理に支障が出るが、スリープトランジスタの技術的な向上で5ns以下でウェイクアップが可能である。よって、200MHz以下の動作なら1サイクル以内で対象の回路をウェイクアップさせることができるので、命令フェッチの最後やデコードの最初でスリープ制御信号を生成すればよい。

一方シャットダウンの場合、対象の回路の電位が下がりきるまでかなり時間がかかることが分かっている。つまり、切替頻度が多い場合には電位の下がりきった状態の時間が短くなってしまい十分な効果が得られない。ただし、今後のスリープトランジスタの性能向上や実装方法の改善によって、シャットダウン遅延を縮めていくことは可能である。そこで、本報告ではPSのモード切替に関するオーバーヘッドは考慮しない。

また、パワーゲーティングするにあたってPS挿入による面積の増加というオーバーヘッドも発生する。スリープトランジスタの数は制御する回路規模に比例して大きくなる。大きな回路をパワーゲーティングするには、多くのスリープトランジスタでドライブする必要があるためだ。面積オーバーヘッドについては、本報告の評価の対象としている。

3. 動的スリープ制御の設計

本研究では、動的スリープ制御を実装する対象プロセッサに、ベーシックで組込み用途でもよく使われる MIPSR3000 プロセッサを用いた。この実行ステージの演算部を複数のユニットに分割してスリープ制御行う。

3.1 対象とする R3000 プロセッサ

R3000 は、代表的な RISC 型の 32 ビットマイクロプロセッサである。32 個の 32 ビット汎用レジスタと 32 ビットのプログラムカウンタ、および整数の乗除算結果を格納する 64 ビットレジスタがある。命令パイプラインは 5 段構成になっており、多くの場合、1 プロセッササイクルで 1 命令ずつ終了することができる。用意されている命令はロードストア命令と基本的な演算命令だけである。

仮想記憶機構、例外処理機構は、プロセッサ内に内蔵された CP0 と呼ばれるコプロセッサでサポートし、浮動小数点演算はコプロセッサ R3010 でサポートされるが、本報告の評価モデルには含めていない。

3.2 演算部のユニット分割

今回は、研究の第一歩として、スリープ制御を実装する対象を演算部のみとした。

R3000 の演算部を細かく見ていくと、加算器や AND、OR 回路などは最も高い頻度での使用が予想されるが、シフト回路や乗算器、除算器の使用頻度はそれほど多くないと考えられる。また、アプリケーションによってはシフト回路を多めに使うなどの偏りも予想される。

そこで演算部から回路をいくつか独立させて、スリープできるユニットとして構成する。まず演算部の中で最も大きな面積を占める乗算回路と除算回路をそれぞれ独立させる。また、次いで大きな回路であるシフト回路もユニットとして独立させることにする。最後に、その他の一般的な演算を行う部分もユニットとし、演算を行わない場合にスリープできるようにする。すなわち、以下の 4 つのユニットから構成する。

- 一般演算ユニット: 加減算、比較、AND、OR、ビット反転など
- シフトユニット: 左右論理シフトと算術右シフト
- 乗算ユニット: 符号付、符号なし乗算
- 除算ユニット: 符号付、符号なし除算

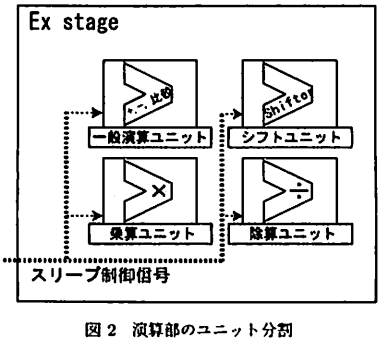

図 2 演算部のユニット分割

以上に基づいて設計した R3000 の実行ステージのユニットの概略を図 2 に示す。

3.3 ユニットに対する動的スリープ制御

各ユニットには、実際に演算が行われる場合にのみ電力が供給されるように電力供給の是非を信号として送る。この信号をスリープ制御信号と呼ぶ。

さらに、いくつかの場合では全てのユニットを使わない状況がある。それは次のケースである。

- NOP 命令の場合

演算を行わないため、全てのユニットをスリープできる。

- メモリアクセス命令でオフセット値の計算が不要の場合

ロード命令やストア命令では、オペコードに指定したレジスタ値などにオフセットを加算して、実際にアクアセスするメモリ番地を計算する必要ある。しかし、オフセット値が 0 の場合やゼロレジスタ (常に 0 が格納されているレジスタ) を指定している場合には演算が不要であるため、一般演算ユニットを使わず全てのユニットをスリープできる。

- 分岐命令の場合

条件分岐・ジャンプ命令などでは、実行ステージは演算を行わないため、全てのユニットをスリープできる。

- データハザードが起こった場合

前節で述べたデータハザードが起こった場合、実行ステージは演算を行わないためスリープできる。制御ハザードの場合は、乗算器か除算器が演算中のため、それ以外のユニットのみスリープできる。

スリープ制御信号は命令内容が判明する命令デコードステージで生成し、使用するユニットをアクティブにし、それ以外をスリープさせる。このようにすることで、ユニット単位での動的スリープ制御を行う。

4. 実装と評価手法

4.1 評価環境の実現

評価時の R3000 の動作環境は、組み込み用途を想定して動作周波数は 100MHz とし、動作電圧は、ASPLA 90nm ライブラリで規定されている 1V とした。

Verilog HDL で RTL 記述した R3000 を Synopsys 社の Design Compiler を用いて論理合成を行った。この論理合成は R3000 のコア部分のみで、キャッシュメモリなどは含まれていない。合成には ASPLA が提供する 90nm プロセスルールのライブラリを用いた。

この合成によって得られたネットリストを元にゲートレベル検証を行った。この検証でダイナミック電力を評価するために用いる各ゲートのスイッチング情報を取得した。最後に、このネットリストを元に Synopsys 社の Astro を用いてレイアウトを行った。

図 3 は、R3000 のレイアウトのスクリーンショットである。R3000 の旧演算部にかわる 4 つのユニット（一般演算ユニット、シフトユニット、乗算ユニット、除算ユニット）と、それ以外の R3000 の本体の部分 (R3000 メイン部) で、合計 5 つのエリアから構成される。フロアプランにおいて、R3000 メイン

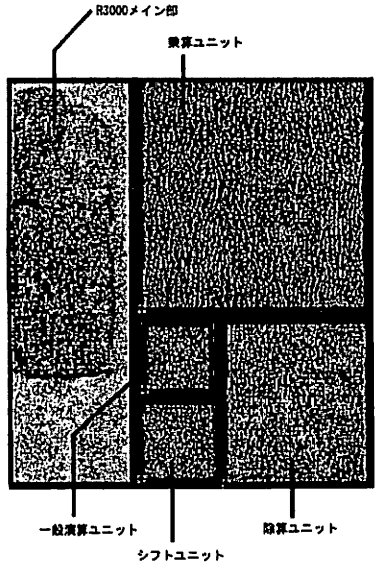

図 3 R3000 のレイアウト

部以外は、ユーティライゼーション(注1)を 40%とした。これは、PS の挿入分の面積を事前に確保するためである。また、このフロアプランの段階で PS の挿入を行った。自動配置を終えた後、Astro から DEF 形式でレイアウト情報ファイルを出力し、シーケンスデザイン社の Cool Power で PS の最適化を行った。

Cool Power は、PS の抵抗値 (トランジスタ幅) と個数を回路規模に応じて最適化する。このときの制約条件として、PS が (常時) オンしている状態で PS を共有している複数の論理ゲートが動作して放電したときに、仮想 Ground ラインの最大電圧が VDD の 10%未満になるように行った。本研究のモデルでは、回路がアクティブな場合に瞬発的に発生する電圧が 100mV 未満になるような制約で PS の種類と個数を最適化している。

この結果から、PS の ON/OFF 時のリーク電力を算出した。また、配線長から計算したキャパシタンスと前の段階で得た各ゲートのスイッチング情報を用いてダイナミック電力を算出した。さらに、PS の挿入に伴うエリアオーバーヘッドも算出した。

4.2 アプリケーションによる予備評価

アプリケーションには主に MiBench を用いた。MiBench は組み込み系プロセッサでも動くように設計された小規模なベンチマークアプリケーション群で、いくつかのパッケージから構成されている。パッケージには、数学演算、Office アプリケーション、ネットワーク処理など、分野ごとにいくつかの代表的なアルゴリズムや演算処理が集められている。

本報告ではこれらのアプリケーションのうち、数学演算パッケージから Quick Sort、ネットワークパッケージから Dijkstra、セキュリティパッケージから Blowfish、また、MiBench 以外からメディア系の処理として JPEG エンコードを用いた。

評価用のアプリケーションプログラムは、Linux 環境で GCC(2.95.29) を使って MIPS 用にクロスコンパイルした。これらのアプリケーションを本研究のシミュレーション環境で動作されるためにコンパイル後のオブジェクトデータを GCC のサブセットである Binary utility の objdump(2.16.1) を用いて逆アセンブルし、機械語をリスト化して整形した。この整形後のデータを Verilog のテストベンチから読み込ませた。

(注1) : 実際にセルなどに占有される面積の割合

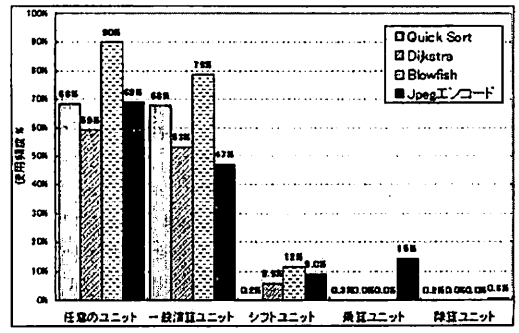

図 4 各ユニットの利用率

5. 評　　価

5.1 ユニットの利用率

分割した 4 つの演算ユニットについて、それぞれの使用率を 4 つのアプリケーションそれぞれで調べた。この結果を図 4 に示す。

任意のユニットとは、4 つのユニットのうちどれかの PS が ON の場合をカウントした結果である。この項目からは、全ユニットが OFF の割合を読み取ることができる。

シミュレーションに用いるデータのサイズは、3 段階 (JPEG エンコードは 2 段階) に変化させてそれぞれの評価を取った。これは、アプリケーションの中核となる処理が全体の占める割合に対して少ないとアプリケーションの特性を見ることができないため、十分な処理量かを確認する必要があるからである。Dijkstra は、経路を 3 種類用意して処理量を変化させている。

どのアプケーションでも、処理量による PS の ON になる頻度はそれほど大きく変わらないので評価において十分な演算量を行っていると考えられる。

JPEG エンコードを除いて、乗算器と除算器の使用率は非常に低い。シフトユニットも次いで低い。JPEG エンコードは、内部で DCT を行っているため、この演算で乗算ユニットを多用したものと考えられる。また、Blowfish においてのみシフトユニットの使用率は 10%を超えている。これは、Blowfish のブロック暗号化ルーチンの中で、ブロック内をシャッフルするためにシフト演算を多く用いるためであると考えられる。Dijkstra と Blowfish においては、乗算器と除算器を全く用いていない

ことが分かる。

一般演算ユニットの使用率は、6割から8割とアプリケーションごとにばらつきが大きい。それに伴って、任意のユニットの使用率もアプリケーションごとにばらついているが、Blowfishが約90%と最も高く、Dijkstraが約59%と最も低い。

5.2 電力削減効果

各ユニットの使用頻度とON時及びOFF時のリーク電力から、全体のリーク電力の評価を行った。ON時及びOFF時の各ユニットのリーク電力は配置配線後のデータを用いて算出した。リーク電力については、PSそのもののリーク電力も考慮してある。つまり、PSセルによる面積増加分のリーク電力を含んだ結果を以下の評価結果として示す。

PS駆動のためのダイナミック電力は、考慮していない。これは、モード切替による電力損失がかなり小さい値になることを予想しているからである。

表1に各アプリケーションごとの、PS動作時におけるリーク電力をまとめる。

表1 頻度解析を加味したアプリケーションごとのリーク電力

アプリケーション	リーク電力 (低減率)
Quick Sort	1.714 mW (32.0%)
Dijkstra	1.708 mW (32.3%)
Blowfish	1.718 mW (31.8%)
JPEG エンコード	1.781 mW (29.4%)

低減率とは、PSを常にONにし続けた場合と比較したとき消費電力削減の割合を表している。どのアプリケーションでも1.71mW前後のリーク電力で、低減率でいうと約32%の削減を実現している。

また、スリープすることにより、そのユニットで消費されるダイナミック電力も削減される。この低減率を表2に示す。

表2 頻度解析を加味したアプリケーションごとのダイナミック電力

アプリケーション	ダイナミック電力 (低減率)
Quick Sort	6.20 mW (59.8%)
Dijkstra	5.29 mW (61.9%)
Blowfish	5.61 mW (59.6%)
JPEG エンコード	6.03 mW (56.6%)

こちらは5割から6割の電力を削減を実現した。この大きな削減率は、元となるR3000がオベランドアイソレーションなどのダイナミック電力削減の手法を用いていないことが原因であるが、今回の手法が、従来の電力削減手法を用いなくてもダイナミック電力を削減する効果があることを示している。

リーク電力とダイナミック電力評価をまとめて、電力の低減効果がどの程度かを調べる。表3は、リーク電力とダイナミック電力を合計した総電力を各アプリケーションごとに示す。

ユニットごとにPSを挿入することによって、5割以上の電力を削減することができた。ただし、これはPSのタイミングオーバーヘッドとPSそのもののダイナミック電力を考慮していない点に注意が必要である。

表3 アプリケーションごとの総電力

アプリケーション	総電力 (低減率)
Quick Sort	7.91 mW (55.9%)
Dijkstra	7.00 mW (57.3%)
Blowfish	7.32 mW (55.3%)
JPEG エンコード	7.81 mW (52.4%)

今後、プロセスルールの微細化に伴って総電力中でダイナミック電力よりもリーク電力が支配的になっていくことが予想される。本報告で行ったユニットごとの使用頻度の解析は、電力の絶対量に関わらずMIPS命令セットとユニットの分割の仕方にのみ依存しているため、今後プロセスルールが微細化してもリーク電力削減の目安にすることができる。

5.3 エリアとそのオーバヘッド

MIPSメイン部と各ユニットごとの面積情報を表4に示す。

表4 MIPS各部の面積情報

	面積 μm^2	セル	ピン	I/O	配線
メイン部	90931.38	29616	131288	176	29536
一般演算	4969.54	1239	6951	104	1311
シフト	7890.72	2031	11467	136	2104
乗算	69908.03	18406	102300	135	18399
除算	50586.58	12234	64483	135	12176
全体	224286.25	63526	316489	686	63526

この面積には、PS挿入によるオーバーヘッドが含まれている。各部の面積、セル数、ピン数、他のユニットとの接続するI/Oピン数、配線の本数を示してる。

図5に各ユニットとMIPSメイン部が占める面積の割合をグラフで示す。

表4と図5から、乗算ユニットと除算ユニットの合計の面積がR3000全体の半分以上を占めていることがわかる。一般的に面積とリーク電力は概して比例関係に近い。前述のユニットの使用頻度解析から乗算、除算ユニットの使用頻度が極めて低いことを総合すると、大幅な電力削減を達成できた要因はリーク電力の大きいユニットを長期間スリープできたことである。

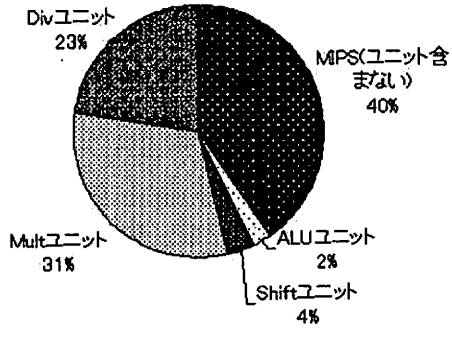

図5 各ユニットとメイン部の割合

表5にMIPS各部のオーバーヘッドと全体の面積のうちオー

表 5 エリアオーバーヘッド

	PS の面積	バッファ	合計	オーバーヘッド
メイン部	0	686	686	1%
一般演算	2250	162	2412	75%
シフト	3508	216	3724	31%
乗算	32740	2184	34924	50%
除算	30695	3328	34023	67%
全体	69193	6576	75769	34%

バーヘッドがどの程度の割合を占めているかを示した。

PS の面積は、PS セル (スリープトランジスタ) そのものの面積の合計を示す。バッファは、PS を駆動するためのバッファ用トランジスタの面積の合計を示す。合計は、これら二つオーバーヘッドの合計である。オーバーヘッドは、各部の面積のうちこのオーバーヘッドの占める割合を示す。MIPS メイン部は、スリープさせないため、PS は挿入されていないが、スリープ制御用の信号を生成しているため、そのためのバッファ用のトランジスタによる面積が増加している。バッファはクロックツリーと同様の原理で複数のスリープトランジスタを同時に制御するために必要である。

オーバーヘッドをユニットごとに見ると、3 割から 7 割とばらつきが大きい。全体では 34%のオーバーヘッドが発生している。オーバーヘッドは、概してもとのユニットの面積やセル数に比例した大きさになるが、面積が小さいユニットでは回路中で使われるセルの種類などの影響が大きく出ている。

6. おわりに

本研究では細粒度の動的スリープ制御の実装として、MIPS R3000 プロセッサの ALU を 4 つに分けて動的にスリープ制御を行い、消費電力とエリアオーバーヘッドの評価を行った。

各ユニットの平均の使用頻度は一般演算ユニットが 62%、シフトユニットが 7%、乗算ユニットが 4%、除算ユニットが 0.3%であることがわかった。

動的スリープ制御でリーク電力は平均 31%の電力を削減でき、ダイナミック電力は平均 59%の電力を削減できた。全体で、55%の電力を削減できた。

面積の評価から乗算ユニットと除算ユニットをあわせた面積が、全体の半分以上を占めることがわかった。これらのユニットの使用頻度は低く、長時間スリープできるため大幅な電力削減の要因となった。

今回は、予備的な評価として、電源スリープにより即時に消費電力が 0 になると仮定したが、スリープ状態に入る際に付加容量に流れ込む電流による電力消費が存在するため、一定の時間スリープしないと、消費電力の低減効果が得られない。今後は回路技術の改良によりこのスリープ時の電力消費を減らすと共に、コンパイラ技術とアーキテクチャの工夫により、一定時間スリープ可能かどうかを判別する手法を導入する予定である。これらの手法を導入したチップを今年度中に ASPLA 90nm プロセスを用いて実装し、実チップにより、本稿で提案した手法を検証していく予定である。さらに、レジスタファイルなど実行ステージ以外の部分のスリープ、キャッシュ、TLB を含めたプロセッサ全体に対するスリープについて発展させる予定である。また、今回は簡単なプロセッサとして R3000 を採用したが、より複雑なプロセッサをモデルに、さらに効率的な電力運用を目指した実装、評価を行っていく予定である。

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 (VDEC) を通し，ローム (株)・凸版印刷 (株)・シノプシス株式会社・日本ケイデンス株式会社・メンター株式会社の協力で行なわれたものである。

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「情報システムの超低消費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。

文　献

[1] 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電力高性能システム LSI の構想”. 情報処理学会研究報告 ARC/電子情報通信学会技術研究報告 ICD2007、2007 年 5 月

[2] Gerry Kane 著, 前川 守 監訳. “mips RISC アーキテクチャ -R2000/R3000”. 共立出版株式会社 1992 年 10 月 1 日発行

[3] 香嶋俊裕, 武田清大, 大久保直昭, 白井利明, 宇佐美公良. “走行時パワーゲーティングを適用した低消費電力乗算器のアーキテクチャ設計”. 電子情報通信学会研究会デザインガイア

[4] 大久保 直昭, 宇佐美 公良. “細粒度動的スリープ制御による動作時リーク電力低減手法”. 情報処理学会 DA シンポジウム 2006、2006 年 7 月号

[5] M.Ishikawa, et.al, “A 4500 MIPS/W, 86μA Resume-Standby, 11μA Ultra-Standby Application Processor for 3G Cellular Phones,” IEICE Trans. on Electronics Vol.E88-C, No.4, pp.528-535, Apr. 2005.

[6] Y.Kanno, “Hierarchical Power Distribution with 20 Power Domains in 90-nm Low-Power Multi-CPU Processor, “ ISSCC2006, Feb. 2006.

[7] 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”. Low Power Electronics and Design 2004. ISLPED '04. Proceedings of the 2004 International Symposium on. Aug 2004 9-11 page32-37

[8] 近藤 正章, 中村 宏 “リーク電力削減のための細粒度命令スケジューリング手法の検討” デザインガイア 2006 No.127 研究報告 page49-54

[9] Erin Farquhar, Philip Bunce “THE MIPS PROGRAMMER'S HANDBOOK” Morgan Kaufmann Publishers