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

2008－ARC－178 (17)
2008／5／14

スーパースカラプロセッサにおける細粒度動的スリープ制御の実装と評価

小島 悠† 池淵 大輔† 関 直臣† 長谷川揚平† 天野 英晴†
香嶋 俊裕†† 武田 清大†† 白井 利明†† 中田 光貴†† 宇佐美公良††
砂田 徹也††† 金井 遵††† 並木美太郎††† 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
††† 東京農工大学工学部情報コミュニケーション工学科〒 184-8588 東京都小金井市中町 2-24-16
†††† 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: †yu@am.ics.keio.ac.jp

あらまし Geyser-0は、MIPS R3000の演算ステージからシフタ、乗算器、除算器を分離し、それぞれに対して動的にパワーゲーティングを行い、命令レベルでスリープ制御を行うことにより、リーク電力を節減する。Geyser-0は、単純なパイプライン構造であるため、場合によってはより大きな性能が必要である。そこで、Geyser-0を拡張したスーパースカラプロセッサGeyser-HS(Hybrid Scalar)を提案する。Geyser-HSはGeyser-0同様の演算器レベルの動的スリープ機構を持つだけでなく、パイプラインの片方全体をパワーゲーティングし、1本のパイプラインのプロセッサとしても動作可能な構成になっている。リーク電力はスリープ制御により平均34.8%削減でき、面積オーバーヘッドは20.3%となった。また、パイプラインを片方スリープさせることにより、スーパースカラモードに比べて、エネルギー効率は平均12%向上した。このことから性能を要求されない場合、モード切り替えは有効に働くことがわかる。

A Fine Grain Dynamic Sleep Control Scheme in Superscalar Processor

Yu KOJIMA†, Daisuke IKEBUCHI†, Naomi SEKI†, Yohei HASEGAWA†, Hideharu AMANO†, Toshihiro KASHIMA††, Seidai TAKEDA††, Toshiaki SHIRAI††, Mitustaka NAKATA††, Kimiyoshi USAMI†††, Tetsuya SUNATA†††, Jun KANAI†††, Mitaro NAMIKI†††, 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
††† Department of Computer and Information Science, The Tokyo University of Agriculture and Technology 2-24-16 nakamachi, koganei-shi, Tokyo, 184-8588 Japan
†††† Research Center for Advanced Science and Technology, The University of Tokyo 4-6-1 Komaba, Meguro-ku, Tokyo, 153-8904 Japan
E-mail: †yu@am.ics.keio.ac.jp

Abstract Geyser-0 is a low power MIPS R3000 processor which uses a novel fine grain power gating technique to computational units; shifter, multiplier and divider separated from the execution stage. Although Geyser-0 can reduce its leakage power, the performance is often less than that of standard embedded processors since it only has a simple pipeline structure. Here, Geyser-HS(Hybrid Scalar) is a superscalar processor as an extension of Geyser-0. The power gating technique is applied to the whole pipeline as well as computational units in Geyser-0. As the evaluation results, the leakage power was reduced by 34.8% in average, while the area overhead was 20.3%. If a pipeline is slept, the energy efficiency is improved by 12%, and this mode is suitable when the performance is not required.

1. はじめに

90nm 世代以降のチップでは、リーク電力の増大が顕著である。「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技術には、パワーゲーティング、Selective MTCMOS、基板バイアスなどが提案され、一部は実際に利用されている。この中で、我々は、ソフトウェア、アーキテクチャとの連携によりスリープ期間を制御することで大きな効果が得られることが期待されるパワーゲーティングに着目した。

パワーゲーティングは回路の一部に対して電源供給を断つことにより、リーク電力を削減する手法だが、モード遷移レイテンシによる性能ロスやパワースイッチのエリアオーバーヘッドなどの問題点がある。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]。

しかし昨今、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ハードルは低くなってきている。この技術を利用することで，パワーゲーティングを CPU 内部の比較的小さい単位に対して動的に適用することが可能である．例えば演算器の中でも乗算回路は面積が大きく、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細かくスリープさせることで，従来の方法では不可能なレベルまでリーク電力を削減することが可能である．このように分離した演算器部分をユニット呼び、実行する命令毎に利用しないユニットに対して動的にパワーゲーティングを行う。このような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 [10]。我々は、細粒度パワーゲーティングの実装についての検討を進め、その結果を集約して、MIPS R3000 と互換性のある命令セットを持つ Geyser-0 を VDEC の ASPLA 90nm プロセスを利用し、200MHz 動作を想定して試作した。

この Geyser-0 は動的に細粒度パワーゲーティングを行う最初のプロセッサとして、実装したため、単一パイプラインで構成されている。しかし、最近は組込み用途のプロセッサにおいても演算性能が要求されるようになり、比較的単純な構成を持つインオーダーの 2way スーパースカラプロセッサを利用する場合が増えてきている。そこで、本稿では Geyser-0 を拡張したインオーダーの 2way スーパースカラプロセッサ Geyser-HS(Hybrid Scalar) を提案する。Geyser-HS は Geyser-0 同様の演算器レベルの動的スリープ機構を持つだけでなく、パイプラインの片方全体をパワーゲーティングし、1 本のパイプラインのプロセッサとしても動作可能な構成になっている。ここでは、Geyser-HS の設計を示し、そのパワーゲーティングの効果と必要面積について評価する。

2. Geyser-0

本節では細粒度パワーゲーティング機構を備えた MIPS R3000 互換の CPU である Geyser-0 について説明する。

一般に CPU の中には、命令や状態によっては動作させる必要が無い部分がある。Geyser-0 ではこれらの動作する必要が無い部分をユニットと呼ばれる単位に分離し、それぞれにフッタ型のパワーゲーティングを適用した。これらのパワーゲーティングされたユニットは、独立にパワーゲーティングできるようになっている。

Geyser-0 がベースとする MIPS R3000 は RISC 型の 32 ビットのプロセッサで、32 個の汎用レジスタを持ち、5 つのパイプラインステージで構成されている。Geyser-0 では Ex(実行) ステージで用いる演算ユニットおよび CP0 にパワーゲーティングを行った。CP0 は OS とのインターフェースであり、各種の例外を処理する際に用いられ、使用頻度が低いため、パワーゲーティングが有効であると考えられる。また、演算ユニットを一般演算ユニット、シフタ、乗算器、除算器の 4 つに分割することで、使われていないユニットに対してパワーゲーティングを行えるようになった。各演算ユニットに関して以下に示す。

- ALU : 一般演算ユニット

加算、減算、論理演算を担うユニットである。1 サイクルで演算を行う。

- SHIFT : シフタ

1 サイクルで演算を行う。

- MULT : 乗算器

マルチサイクルのユニットで演算に 4 サイクルかかる。

- DIV : 除算器

マルチサイクルのユニットで演算に 10 サイクルかかる。

レジスタ、TLB、キャッシュなどは、CPU を占める面積が大きく、パワーゲーティングが有効と考えられるが、データ保持の問題が有るため、Geyser-0 ではパワーゲーティングの実装は行わなかった。

また、モードの遷移にはスリープトランジスタの切り替えや、回路中の電荷の放電により電力オーバーヘッドが発生する。この電力オーバーヘッドを相殺するためには、一定のスリープ時間が必要となる。そのため、頻繁なモード遷移はエネルギーロスになり、結果として消費エネルギーの削減にならなくなる可能性が有る。遷移によるエネルギーロスを埋め合わせるスリープ期間をブレイクイーブンポイントと呼ぶ。実質的に消費エネルギーを削減するには、このブレイクイーブンポイント以上のスリープ期間が必要となる。

さらにモードの遷移にはレイテンシがあり、特にスリープから起動する際のレイテンシが問題となる。しかし、今回利用したフッタ方式のパワーゲーティングは、200MHz 動作で動作させた場合、起動のためのレイテンシは 1 サイクル以内に収まることが確かめられた。

そこで、Gyser-0 では、各ユニットを制御するスリープコントローラーを IF ステージと並行して実装することで起動のためのレイテンシを隠蔽した。このスリープコントローラーでは、IF ステージで命令をフェッチするのと同時にどのユニットが使われるのかを判定し、使われるユニットを起動し、そのユニットでの演

算が終了すると自動的にスリープさせるようにしている。

しかし、この方法を用いてスリープ制御を行った場合、利用頻度の高いユニットでは、ブレイクイーブンポイントに達していないスリープが多発した場合には、エネルギーロスが大きくなる。そのため、演算終了後もスリープに移行しないような専用命令を用意している。あらかじめコンパイル時に頻繁に使われるユニットを予測し、そこに専用命令を入れることで、過剰なモード遷移を削減することができる。

3. スーパースカラプロセッサの設計

本研究では、Geyser-0 を 2way のインオーダーのスーパースカラに拡張した Geyser-HS を提案し、スーパースカラにおける細粒度パワーゲーティングの効果を調べた。本研究では CPU のコア部分のみを対象とし、CP0、TLB、キャッシュの実装はしていない。

3.1 スーパースカラプロセッサの構成

Geyser-HS の構成を図 1 に示す。Geyser-HS は、Geyser-0 同

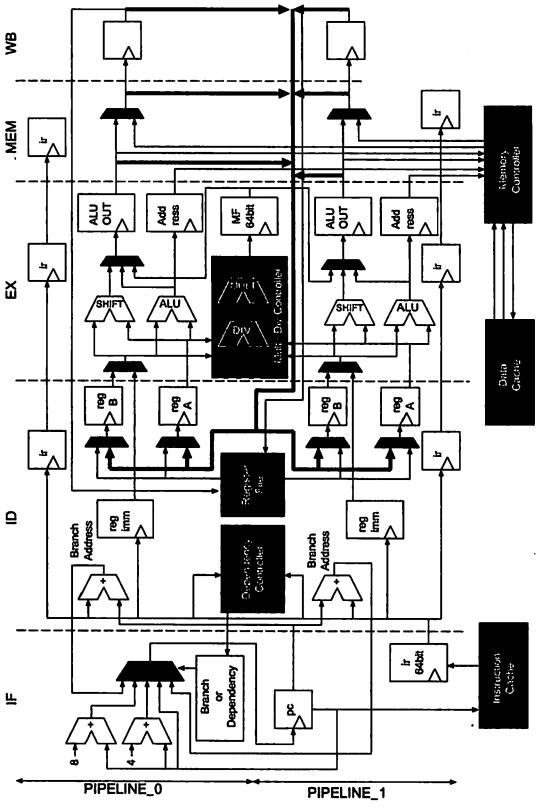

図 1 Geyser-HS の構成

様、MIPS R3000 互換の命令セットを持つため、32bit の汎用レジスタを 32 個保持し、標準的な 5 段パイプライン構成で実装されている。命令の依存関係の除去は ID ステージの Dependency コントローラーで行い、インオーダーに命令を発行し、汎用レジスタにインオーダーに結果が格納されるようにした。

一般演算ユニットとシフタを各パイプラインに配置し、乗算器、除算器は 2 つのパイプラインで共有する構成とした。MIPS R3000 では乗算、除算命令を実行した場合、それぞれの結果を専用レジスタに格納した後、専用レジスタからロードする専用命令を用いて計算結果を汎用レジスタにロードする仕様となっている。すなわち、乗除算命令と専用レジスタからロードする専用命令が交互に実行されるため、インオーダースーパースカラプロセッサにおいて乗算器、除算器を同時に 2 つ使うことはない。そのため、今回は乗算器、除算器を 2 つのパイプラインで共有するようにし、それぞれのパイプラインからの命令を制御する機構として MULT/DIV コントローラを用意した。

3.2 細粒度パワーゲーティングの手法

細粒度パワーゲーティングの制御を設計するため、まず各ユニットのブレイクイーブンポイントを測定した。この結果を表 1 に示す。表中の数値は、1 サイクルを 5ns としたサイクル数である。

表 1 各ユニットのブレイクイーブンポイント (サイクル数)

温度	ALU	Shift	Mult	Div	CP0
25 度	74	114	74	44	92
65 度	26	38	22	14	28
100 度	12	16	10	6	12
125 度	8	10	8	2	8

本研究では組み込み用途での利用を考えているため、65 度以下での動作を想定している。

予備評価としてスーパースカラプロセッサにおける各ユニットのスリープの頻度を解析した。この頻度解析では、各ユニットの全実行サイクルに対するスリープ率を各アプリケーションごとに求めた。さらに、スリープ率をブレイクイーブンポイント以下のスリープの総サイクル数とブレイクイーブンポイント以上のスリープの総サイクル数に分けて求めた。結果を図 2 に示す。パワーゲーティングが効果を発揮するためには、図 2 中で示す、ブレイクイーブンポイントより長いスリープが多数を占める必要がある。

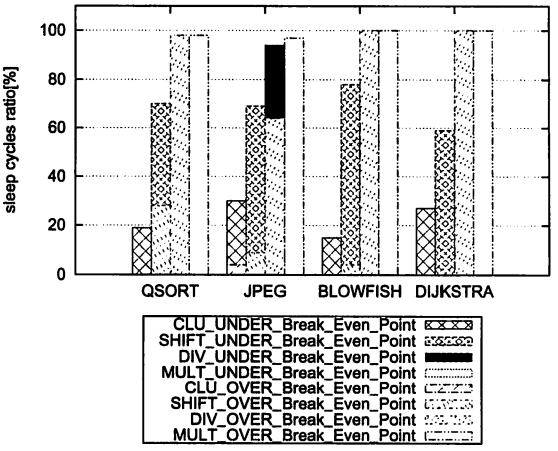

図 2 各ユニットのスリープ率とブレイクイーブンポイントによる分割

図 2 によると、一般演算ユニットはスリープしているサイクル数が実行サイクル数に対して 3 割以下と少ない。シフタはスリープサイクル数が 6 割から 8 割程度を占める。しかし、ブレイクイーブンポイント以上のスリープサイクルが多い QSORT の場合

においても、実行サイクルの28%であり、全体の平均では10%ということがわかった。一方、乗算器、除算器ではブレイクイーブンポイント以上のスリープサイクル数が乗算器で64%以上、除算器では97%以上確保できた。ただし、DIJKSTRAやBLOWFISHでは乗算器、除算器を一切使用しないので、スリープ率は100%となっている。また、JPEGでは乗算器を他のプログラムと比較して頻繁に使うため、ブレイクイーブンポイント以下のスリープが多めになった。

上記の予備評価より、今回パワーゲーティングの対象として、乗算器、除算器のみを選んだ。

3.3 スリープ制御

Geyser-0と同様にスリープコントローラを設計した場合、スーパースカラの場合、IDステージの命令の依存関係や分岐をチェックしてから、IFステージのレジスタに次のクロックで実行される命令をセットする。そのため、IDステージにある命令の依存関係や分岐をチェックし、次のクロックで実行される命令が決定した後に、使用されるユニットを起動しなくてはならない。さらに、IDステージで2本のパイプラインからのフォワーディングや命令依存の排除を行うため、一本のパイプラインのプロセッサに比べて、IDステージにおいて長いパスが形成されてしまう。以上の点から、Geyser-0と同様に実装した場合、200MHzでの動作は見込めなくなる。

このため、Geyser-HSではGeyser-0とは異なる手法をとることとした。スリープ制御をより単純にするため、IDステージで命令を判別し起動する。このIDステージで起動する場合、起動にかかるレイテンシが問題となる。しかし、今回スリープさせることとしたユニットはそれぞれマルチサイクルのユニットであるので、それぞれのユニットにおける演算サイクル数を1増やすことで解決できる。

また、これらのユニットはスリープした場合、100サイクル以上のスリープとなる確率が高い。そのため、演算が終了し次第スリープさせるようにしてもブレイクイーブンポイント以下のスリープになる確率が低いため、演算終了後、即時にスリープさせるようにした。

3.4 パイプラインに対するパワーゲーティング

CPUで演算を実行する際、常に高速な処理が要求されるわけでは無い。その様な場合、片方のパイプラインのみを動作させ、もう一方のパイプラインをスリープさせることで、電力あたりの処理効率を向上させることができる。

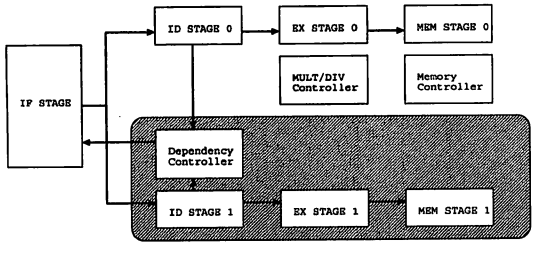

図3 パイプラインに対するパワーゲーティング

図3の灰色になっている部分のように、使わない片方のパイプライン及びそのパイプラインに属する一般演算ユニットとシフタ、Dependencyコントローラに対して、パワーゲーティングを行い消費エネルギーを削減することにした。このパイプラインのパワーゲーティングの制御には、その時に要求される処理速度の判断をOSに任せる必要が有るが、今回CP0の実装を行っておらず、OSを走らせることができないため、ハードウェアにより動的にパイプラインのパワーゲーティングを制御する機構を設けることはしていない。しかし、機構自体を設けることは容易である。

4. 評価手法

4.1 Geyser-HSの実装

Geyser-HSをVerilog-HDLで記述し、Geyser-0同様にASPLA 90nm CMOSプロセスを想定してSynopsys社DesignCompiler Ver.2008.06.SP2により論理合成を行った。また、乗算、除算器は起動のためのレイテンシを考慮し、乗算器を5サイクル、除算器を11サイクルのマルチサイクルパスとした。最大動作周波数を200MHzと想定し、構成している。

乗算器、除算器はGeyser-0で使ったものを利用した。これらのパワーゲーティングをしたユニットはSTARCが提供するスタンダードセルの一部をパワーゲーティング用のセルとして独自に開発し、これらを用いてスリープを行うモジュールを設計した。また、全てのセルをパワーゲーティング用に直せなかったため、一部の複合セルなどが使えず、全てのライブラリを利用可能な場合と比べると面積が17%ほど増加した。

4.2 ダイナミック電力とアクティブ時のリーク電力

まず、ダイナミック電力とアクティブ時のリーク電力を、ネットリストからPowerCompiler(注1)を用いて125度におけるライブラリを用いて算出した。スイッチング確率情報(saif)は、DesignCompilerのネットリストでゲートレベルシミュレーションを行い生成した。

4.3 スリープ時のリーク電力

パワーゲーティング対象の回路がスリープモードへ遷移する場合の電力は、ジャットダウン後の電圧の過渡状態と、その後の定常状態があり、やや複雑である。また、実行するアプリケーションがどのユニットをどれくらいの頻度で使うかによってスリープできる期間が変わってくるため頻度情報も考慮する必要がある。このため、まず、あるアプリケーションを実行したとき各ユニットにおいて“Nサイクルのスリープが M回あった”という情報を取得する。具体的な取得方法に関しては次節で述べる。

次に、それぞれのユニットがスリープモードに遷移した場合の電圧の過渡情報をHSPICE(注1)を用いて取得し、Nサイクルを連続スリープさせた場合の電力評価モデルを作る。この評価モデルを用いて、あるユニットがNサイクルスリープした後にウェイクアップする場合のの平均消費電力 Lws_N を算出した。Lws_N 値とアクティブ時のリーク電力 Lwa から各ユニットの平均リーク電力を算出できる。

Nサイクルのスリープが M_N 回あり、アプリケーションの総サイクル数がTサイクルだとすると、次の式でこのユニットの平均リーク電力Lwを算出することできる。

$$Lw = \frac{1}{T} * \{\sum_i (Lws_i * M_i * i) + Lwa * (1 - \sum_i (M_i * i))\}$$

(注1) : Synopsys, Inc

モデル作成の際には、N を 1 サイクル刻みで変えてデータを取得するのが理想的だが、計算量の制約から 2〜40 までは 2 サイクル刻みで、後は 50,60,70,80,90,100,200,300,500,750,1000 サイクルで取得し、間は線形補間した。ここでは 200MHz の動作を想定しているため、1000 サイクルは 5ms となる。これ以上は定常状態として扱った。

4.4 利用アプリケーション

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。最適化には O3 オプションを用いた。これらのアプリケーションを論理シミュレーション環境で動作されるため、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。

また、GCC で MIPS R3000 用にコンパイルした場合スーパースカラプロセッサに適したコードにはならない。そのため、一部の命令の並べ換えを行った。MIPS R3000 では、乗算、除算命令の後に専用命令で計算結果をロードするようになっている。そのため、乗算、除算をしている最中に専用命令が来た場合、その演算が終了するまで、ストールして待たなくてはならない。このストールをできるだけ減らすように、専用命令を乗算、除算命令から離すように並べ換えを行った。

5. 評価結果

5.1 速度比率

Geyser-0 をスーパースカラプロセッサにしたことにより、処理速度が向上した。各アプリケーションにおいて、Geyser-HS の Geyser-0 に対する演算処理速度の比率を求めた。図 4 に結果を示す。

アプリケーション毎に傾向が異なるが、1.1 倍から 1.4 倍程度の速度向上が見られた。QSORT や BLOWFISH では 1.4 倍程度の向上が確認でき、スーパースカラ化が効果的である事が分かる。JPEG も命令並列性自体は高いが、乗算、除算命令の実行が完了する前に後続の専用レジスタから計算結果をロードする命令が実行されてしまいストールが発生してしまうことが多く、1.3 倍程度の性能向上に留まった。DIJKSTRA は 1.1 倍程度と他のプログラムと比較して、低い結果となった。これは、DIJKSTRA のプログラム中における分岐命令の割合が高いことに依存している。Geyser-0 では分岐命令によるストールは発生しなかったが、Geyser-HS ではストールが発生し、1 命令もしくは 2 命令の空きが発生してしまうため、分岐命令の頻度が高いと性能が低下する。

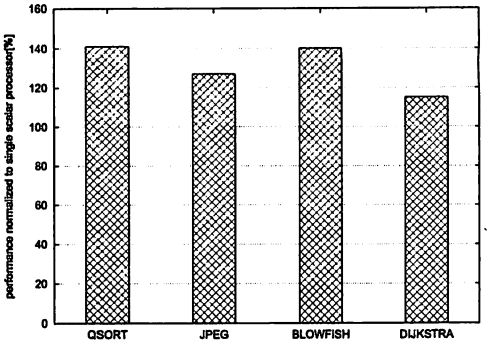

図 4 Geyser-0 に対するスーパースカラプロセッサの演算処理速度

5.2 電力削減効果

まず、乗算器、除算器におけるリーク電力の削減効果の評価を行った。各ユニットのスリープ頻度を求め、前節に記した計算モデルを用いて計算した。各アプリケーションを実行した際の平均リーク電力を、図 5 に記す。

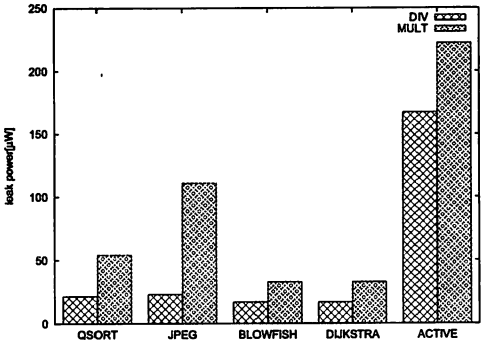

図 5 乗算器、除算器のリーク電力

アクティブ時に乗算器は 222.1μW、除算器は 167.2μW のリーク電力を消費している。これに対し、DIJKSTRA、BLOWFISH では演算にこれらのユニットを使わないため、乗算器で 16.98μW、除算器で 32.87μW と大幅にリーク電力を削減できた。一方、JPEG では乗算器でのブレイクイーブンポイント以下のスリープサイクル数が多いので、モード遷移によるリーク電力が増加し、電力の削減効果が小さくなった。

次に、コア全体のリーク電力に対する削減効果を求めた。コア全体のアクティブ時のリーク電力は 893μW であり、先程のデータと組みあわせてコア全体のリーク電力に対しては、平均で 34.8%、最大で 38.0%の削減効果があることがわかった。

一方、ダイナミック電力は全てのアプリケーションの平均で 7.31mW の電力となった。90nm プロセスで 200MHz で動作させた場合は、まだリーク電力に対してよりダイナミック電力が大きくなっている。

5.3 エリアオーバーヘッド

パワーゲーティングの際に、パワースイッチやアイソレーションセルが挿入されているため、エリアオーバーヘッドが存在する。Geyser-0 で用いられた、乗算器、除算器におけるエリアオーバー

ヘッドを用いて全体に対するエリアオーバーヘッドの値を求めた。Total の値にはトップモジュールの面積も含まれるため、表に記した項目の合計でではないことに注意が必要である。パイプラインのパワーゲーティングでは実際にパワースイッチを挿入したわけではないため、この評価では乗算器、除算器のみにパワーゲーティングを用いた値を用いて求めている。表 2 の結果より、面積

表 2　エリアの比較

	パワーゲーティングをした場合 [μm^2]	パワーゲーティングをしてない場合 [μm^2]	オーバーヘッド
Main	51052	51052	0%
Reg	53268	53268	0%
乗算器	58516	40026	44%
除算器	54863	32498	69%
ALUx2	12064	12064	0%
SHIFTx2	8604	8604	0%
Total	241848	200993	20.3%

オーバーヘッドが全体では 20.3%になることがわかる。

5.4　片方のパイプラインに対するパワーゲーティング

この方法は、要求される処理速度に応じて、片方のパイプラインをパワーゲーティングし、電力効率を向上させる方法である。この片方のパイプラインをパワーゲーティングした状態で動作している状態をシングルスカラモードと呼び、両方のパイプラインで動作している状態をスーパースカラモードと呼ぶこととする。評価する指標として、アプリケーションの実行にかかるエネルギーを比較することにした。スーパースカラモードとして動作時のアプリケーションの実行にかかるエネルギーを 1 としたときのシングルスカラモードにかかるエネルギーを、それぞれのアプリケーションにおいて計測した。グラフを図 6 に示す。図 6 におい

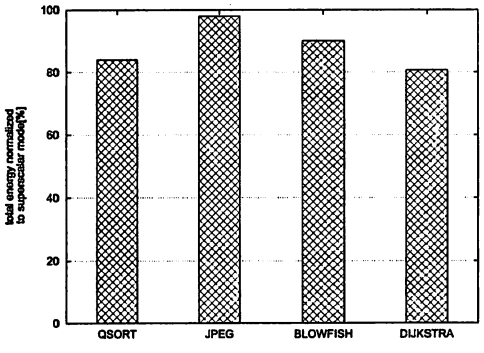

図 6　スーパースカラモードに対するシングルスカラモードのエネルギー比

て、100%より大きい場合は、スーパースカラモードで動作させた方がエネルギー効率が良くなるということになる。図 6 の結果より、プログラムにより差はあるものの JPEG 以外では、全てのアプリケーションでシングルスカラモードの方がエネルギーが少なく、平均して約 12%エネルギーが少ない。ゆえにピーク性能を要求しない状況では、シングルスカラモードの方が良いことがわかる。ただし、JPEG の場合は 98%となり、スーパースカラモードと比べてアプリケーションの実行にかかるエネルギーがほとんど変わらない。このような場合は、性能はスーパースカラモードの方が良いことから、スーパースカラモードで動かした方が良いといえる。今回 JPEG のみ電力あたりの処理速度が等しくなったのは乗算及び除算が他のアプリケーションと比較し多いためであると考えられる。

6.　おわりに

本研究では MIPS R3000 ベースの演算器レベルで動的スリープ制御を行う試作チップ Geyser-0 を改造し、Geyser-HS というスーパースカラープロセッサを作成した。Geyser-HS では演算ユニットの動的スリープ制御に加え、片方のパイプラインに対するパワーゲーティングも行った。

演算ユニットに対する動的スリープ制御でリーク電力を最大 38%削減することができた。また、それによる面積オーバーヘッドは 20%となった。片方のパイプラインに対するパワーゲーティングでは、1 アプリケーションの動作に必要なエネルギーがスーパースカラモードの方がシングルスカラモードより 2%から 24%増加する事がわかった。これにより、ピーク性能の要求に応じてモードを変化させることで消費電力を削減できることが分かった。

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，株式会社半導体理工学研究センター、富士通株式会社、松下電器産業株式会社、NEC エレクトロニクス株式会社、株式会社ルネサステクノロジ、株式会社東芝の協力で行われたものである。

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”. 情処研報 ARC/信学報 ICD, pp.79-84 (2007).

[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).

[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”. VLD No.73, pp.7-12 (2006).

[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”. DA シンポジウム 2006, pp.199-204 (2006 Nov)

[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 (2005).

[6] Y.Kanno, “Hierarchical Power Distribution with 20 Power Domains in 90-nm Low-Power Multi-CPU Processor, “ ISSCC2006, pp.540-541 (2006 Feb).

[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”.Proceedings of the 2004 International Symposium on Low Power Electronics and Design (ISLPED 04). pp32-37 (2004 Aug).

[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 No.127 研究報告, page49-54 (2006).

[9] Erin Farquhar, Philip Bunce “THE MIPS PROGRAMMER'S HANDBOOK” Morgan Kaufmann Publishers (1994).

[10] 関 直臣、他 “MIPS R3000 における細粒度動的スリープ方式の提案” 情処研報 ARC/信学報 ICD, pp.49-54 (2007).