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

2009-ARC-181／2009-EMB-11 (15)
2009/1/14

ハイパフォーマンスコアとローパワーコアの組み合わせにおける細粒度動的スリープ制御の実装と評価

関 直臣† Lei Zhao† 池淵 大輔† 小島 悠† 天野 英晴†

† 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
E-mail: †seki@am.ics.keio.ac.jp

あらまし 演算器単位の細粒度動的パワーゲーティングとコア単位の粗粒度パワーゲーティングをヘテロジニアスマルチコアプロセッサを対象として実装し、評価を行う。ハイパフォーマンスコア (HP-Core) とローパワーコア (LP-Core) は共通の ISA で動作するが、HP-Core はスーパスカラ構成であり、動作周波数も高い。各コアは 16KB の L1 キャッシュと TLB を持っている。評価は Fujitsu 65nm のプロセスルールを用いて、性能、電力、面積とパワーゲーティングによる面積オーバーヘッドについて行った。この結果、HP-Core では 24%の性能向上し、Dijkstra、Blowfish ではリーク電力を 46%削減できたが、Qucik Sort と JPEG エンコードでは消費電力が 63%増大した。面積オーバヘッドは平均で 9.2%であった。

Fine-Grained Dynamic Sleep Control on the Combination of High-Perfomance Core and Low-Power Cores

Naomi SEKI†, Lei ZHAO†, Daisuke IKEBUCHI†, Yu KOJIMA†, and Hideharu AMANO†

† Graduate School of Science and Technology, Keio University 3-14-1, Hiyoshi, Yokohama, JAPAN 223-8522
E-mail: †seki@am.ics.keio.ac.jp

Abstract A heterogeneous multi-core processor, which is consists of the combination of fine-grained power gating for split ALU and coarse-grain power gating for cores, is designed and evaluated. Although the high-perfomance core (HP-Core) and the low-power core (LP-Core) run with the same ISA, the HP-Core is a Superscaler with higher clock frequency, compared with the LP-Core. Each core shares 16KB L1 Cache and TLB, and is designed using Fujitsu 65nm CMOS technology. As a result, the perfomance of HP-Core increased by 24%. The leakage power is reduced by 46% for executing Dijkstra and Blowfish, while the power comsuption is increased by 63% for Quick Sort and JPEG. Area overhead is 9.2%.

1. はじめに

近年、組み込み機器においても OS が搭載され複数のプロセスやスレッドが動作するものが増えてきている。プロセスルールは微細化され要求性能は高まり、一方でリーク電力の増大が顕著なりつつある。特に 90nm 世代以降のチップでは、リーク電力の増大が顕著になっている。このため、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。「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。

プロセッサにおいてキャッシュ [2] の低電力化やコアやモジュール単位 [3] のみの粗粒度なスリープ制御はこれまで行われてきた。しかしながら、急増するリーク電力に対抗するためには演算器などのデータパスにおけるリーク電力を無視することはできない。

リーク電力を削減する技術には、パワーゲーティング、Dual-Vth、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、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って、リーク電力を削減する方法であり、電源を供給する部分とその時間をアーキテクチャやソフトウェアにより制御することで、大幅なリーク電力の削減を見込むことができ、しかも他の回路レベルのリーク電力削減手法と併用することができる。

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。プロジェクトの一環として、演算器レベルのパワーゲーティングを動的に行う省電力組込みプロセッサを想定した実チップを開発し評価を行っている。[5]

本研究では、マルチプロセス、マルチスレッドにおいて有効なマルチコアをターゲットとして、コア単位と演算器単位でのスリープ制御を実装する。最大パフォーマンスを発揮する場合とバックグランドジョブのみの極低電力での動作が要求される場合など電力性能比の面で柔軟性を大きくするために、ハイパフォーマンスコア (HP-Core) とローパワーコア (LP-Core) によるヘテロジニアスなアーキテクチャを提案し実装する。各コアはL1キャッシュとTLBを持っており、L2キャッシュまたはメモリを共有することになる。

Fujitsu 65nmのプロセスルールを用いて論理合成、配置配線を行い、面積、アプリーケーションを実際に動かした場合の性能、電力を評価する。

2. 動的細粒度パワーゲーティング

本節では、本研究で用いるCPUのベースとなっているGeyser-0と、動的細粒度パワーゲーティングについて簡単に紹介する。

2.1 細粒度パワーゲーティング

Geyser-0では図1に示す細粒度動的パワーゲーティングを用いている。

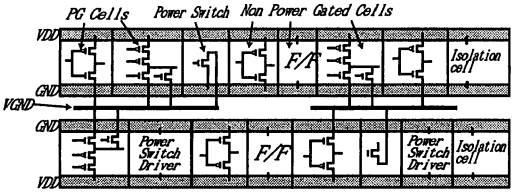

図1 細粒度パワーゲーティング

複数個のセルが仮想Groundを共有し、これに対応して一つのパワースイッチ (PS) が挿入される。粗粒度のパワーゲーティングに比べて、PS一つでドライブする回路の容量が小さいため、高速でON、OFFを切り替えることができる。

CPUの中では、命令やCPUの状態によっては、動作する必要のない部分がある。これを分離してユニットと呼ばれる単位を構成し、細粒度パワーゲーティングを適用する。このゲートを個別にOFFすることで、そのユニットのパワーをOFF、すなわちスリープさせることができる。

パワーゲーティングによるスイッチング電力はオーバーヘッドエネルギーを発生させる。また、スリープモードに入っても消費電力を最小にするまでには時間を要する。エネルギーロスを相殺するまでの時間をブレイクイーブンタイム (BET) と呼び、電力削減効果を得るためには、BETより長い時間スリープを行う必要がある。

2.2 動的なスリープ制御

Geyser-0では、MIPS R3000[6]ベースのCPUコアの演算器およびコプロセッサCP0に細粒度動的パワーゲーティングを実装した[5]。R3000は代表的なRISC型の32ビットプロセッサで、5段構成のパイプラインステージで命令を処理する。Geyser-0ではEX（実行）ステージのシフタ、乗算器、除算器をユニットとして分離し、命令に応じた動的なスリープ制御を行った。OSとプロセッサのインタフェースの役割を持つCP0も使用頻度が低いため、パワーゲーティングのターゲットとしている。

パワーゲートの制御は、スリープコントローラからの信号によって行われる。図2にスリープコントローラとパイプラインの関係を示す。

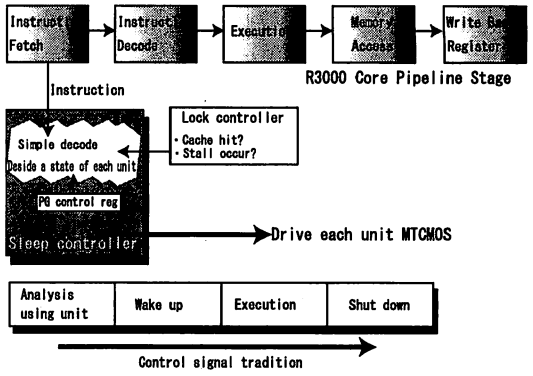

図2 実行時スリープ制御

スリープコントローラは、キャッシュミスや演算結果待ちのストールも考慮し、各ユニットのスリープを制御する。基本動作として、演算完了後に各ユニットは自動的にスリープモードに遷移する。現状では5ns以下で、各ユニットをWake-upさせることが可能だが[7]、ID（命令デコード）ステージの命令デコードの時点では直後のExステージには間に合わない。そのためIF（命令フェッチ）ステージの中で、スリープコントローラは簡易的なプリデコーダを用いてどのユニットが使われるかを判断する。

2.3 特殊なスリープ制御

2.3.1 命令によるスリープ制御

コンパイル時に静的な解析を行うことで、前述のBETに対しての最適化を行うためにPG制御情報を付加した命令を追加した。図3はPG制御情報を付加した命令の構造を示している。

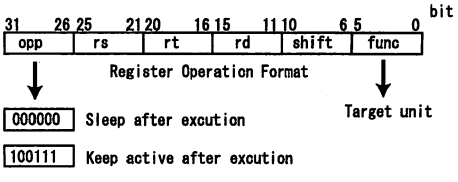

図3 PG制御命令

R3000のISAでレジスタ演算命令では仕様上上位6ビット

(ROP ビット) が 0 になる。ROP ビットに対して PG 制御情報を畳み込むことで、命令数を増加させることなく、演算後に利用した演算ユニットの ON、OFF を指示することができる。

2.3.2 パワーゲーティング制御レジスタ

リーク電流はチップの温度に影響を受け、BET は温度が高くなると短くなることが分かっている [5]。温度変化をパワーゲーティングの制御に反映させるため、CP0 内にパワーゲーティング制御用レジスタを搭載し、OS 管理のもとで BET に最適化したパワーゲーティングの運用を行うことができる。図 4 は PG 制御用レジスタの概要である。

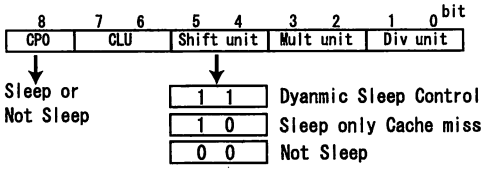

図 4 PG 制御用レジスタ

レジスタの各ビットは各ユニットがスリープをキャンセルするかどうかのフラグ (各 2 ビット) となっている。2 ビット使うのは、キャッシュミス時にのみスリープするモード、使用後は常にスリープモードするモード、スリープを行わないモードの 3 状態を識別するためである。

3. Geyser-Hetero アーキテクチャ

Geyser-0 は、細粒度動的スリープ制御により細かいレベルでリーク電力を低減できるが、単純な 5 段パイプラインを持つ MIPS R3000 互換のプロセッサであり、性能の点では不十分な場合がある。一方で、性能がほとんど要求されない場合でも、演算器以外の部分をスリープさせないために、制御できる電力-性能の幅が大きくないという問題点がある。

そこで、本研究では、細粒度動的スリープ制御だけでなく、ヘテロジニアスマルチコアにおける粗粒度と細粒度のスリープ制御を組みあわせた運用を提案する。ヘテロジニアス構成によりシングルスレッドでの最大パフォーマンスを大きくし低電力モード時のリーク電力を小さくすることができる。本アーキテクチャを Geyser-Hetero と呼ぶことにする。

3.1 アーキテクチャの全体構成

図 5 に提案するヘテロジニアスマルチコアのアーキテクチャを示す。

Geyser-Hetero は一つの HP-Core と二つの LP-Core を持っている。OS のインターフェースやページテーブル管理を行うための CP0 コアは HP-Core のみに搭載する。これら 3 つのコアは内部で細粒度 PG によるリーク削減を動的に行う他、コア単位でのスリープすることで、パフォーマンスの変化に柔軟に対応することができる。

パフォーマンスが必要無い場合は LP-Core だけを動作をさせてリーク電力を含めて電力消費を削減でき、逆に 3 つのコアを同時に使用してピークパフォーマンスを得ることもできる。

HP-Core は LP-Core と比べて高い周波数で動作し、パイプ

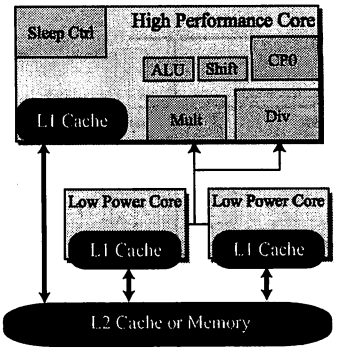

図 5 Geyser-Hetero アーキテクチャ概観

ラインの段数も多い。また、命令スケジューリングやスーパスカラ、乗除算回路、CP0(例外処理、TLB エントリ操作) を搭載している。

一方、LP-Core は OS のカーネルやパフォーマンスが不要なバックグランドジョブなどを実行させるのに適しており、性能が低いかわりに消費電力を小さく抑えられるように設計した。L1 キャッシュは全てのコアが持っており、L2 またはメインメモリの階層でコヒーレンシーを解消する。

HP-Core と LP-Core で共通の ISA で運用するため、性能と電力の以外の部分ではヘテロジニアスな構成を意識する必要はない。OS やコンパイラのレイヤーからは共通のアーキテクチャとしてバイナリを生成し、実行することが可能である。

3.2 High Performance Core

HP-Core は In-Order、2way Super-Scaler の 5 段パイプラインで、周波数は LP-Core の 2 倍の 260MHz〜300MHz を想定している。図 6 に HP-Core のパイプラインアーキテクチャを示す。[8] 命令キャッシュは 2 命令同時読み出しで、データキャッシュは 2 リード、1 ライトである。

一般演算ユニットとシフタを各パイプラインに配置し、乗算器と除算器は 2 本のパイプラインで共有する。MIPS R3000 では乗算又は除算命令を実行した場合、その結果を専用レジスタに格納した後、専用レジスタから汎用レジスタへ転送する命令を用いて計算結果を利用する仕様となっている。このため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ので、インオーダースーパ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。よって今回は乗算器、除算器を 2 本のパイプラインで共有し、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乗除算器コントローラを用意した。

HP-Core は例外処理や割り込みの管理を行うコプロセッサ CP0 を搭載し、各 LP-Core の TLB エントリも HP-Core から設定する。CP0 は命令デコーダやメモリアクセスの際、アドレス計算などでコア内のハードウェアリソースを使うためメインパイプラインと太いバスで接続する必要がある。CP0 のパワーゲーティングを考える場合、CP0 からの出力ポートにはホルダーセルを入れる必要がある。このため CP0 はパワーゲーティングによる面積オーバーヘッドが比較的大きいユニッ

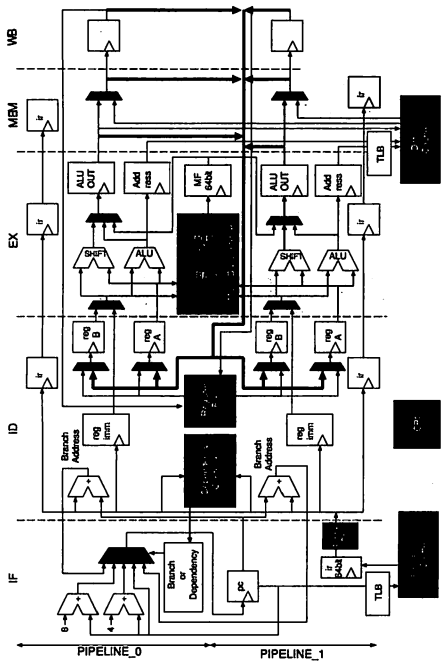

図 6　HP-Core のアーキテクチャ

トであるが、利用頻度は他の演算器に比べて非常に少ないため、HP-Core のみに搭載することとした。

3.3 Low Power Core

低電力での動作を可能にするために、LP-Core は図 7 のような単純で小さなコアを提案する。

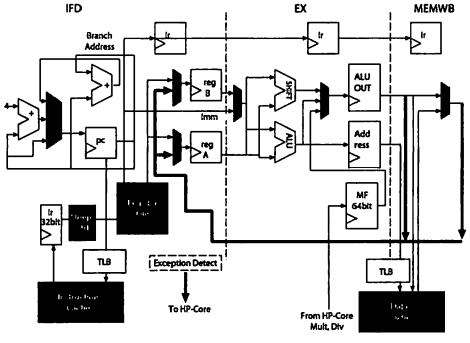

図 7　LP-Core のアーキテクチャ

In-Order、Single-Scalar で CP0 命令には対応しない。パイプラインは 3 段で、周波数は 130〜150MHz を想定している。乗除算器はなく HP-Core のものを共有して使用する。LP-Core は例外処理や割り込みを制御するための CP0 を持っていないため、OS の特権命令のうちこれらを操作する命令を単体で処理することはできない。従って、自身の TLB テーブルエントリについても操作することはできない。LP-Core で処理不能な命令をフェッチした場合には、例外を発生してその処理を HP-Core に委託する。

3.4 キャッシュと TLB

HP-Core のキャッシュは 2 リード、1 ライトとし、LP-Core は 1 リード 1 ライトで Geyser-0 と共通のものを用いた。どちらも方式は共通で命令キャッシュ、データキャッシュともに 8KB の 2way セットアソシアティブ方式でデータキャッシュはライトバック方式である。

TLB は全てコアで個別に持っているがアーキテクチャやエントリ数は同じである。ページアドレス変換用のテーブルのエントリ数は 16 とした。

図 8 はキャッシュアクセスの流れを示す。仮想アドレスの上位 20 ビットを仮想ページアドレスとし、下位 12 ビットをページ内オフセットとして一階層のページアドレス変換を行う。ページサイズと 1way のサイズを同じにしたので、キャッシュを引くと同時に TLB で変換を行うことができる。

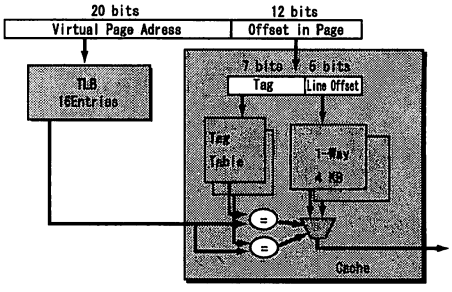

図 8　アドレス変換フロー

3.5 Geyser-Hetero の実装

3.5.1 概　　要

Geyser-Hetero の動作周波数は HP-Core を 300MHz、LP-Core を 150MHz として実装し、評価を行った。将来的には各コアごとに DVFS を行い柔軟性を持たせるのが妥当だと思われる。

Fujitsu 65nm のプロセスルールを用いて、キャッシュメモリ及びそのタグメモリは、VDEC が供給する SRAM を用いた。TLB およびキャッシュは、パワーゲーティングの対象としていない。

3.5.2 実装フロー

Verilog-HDL で RTL 記述した Geyser-Hetero を Design Compiler(注1) で論理合成し、合成結果のネットリストを Astro(注1)を用いて配置配線を行った。

Fujitsu 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。パワーゲーティング用のセルは使用頻度の高いものを優先的に作ったが時間の関係上全てをパワーゲーティング用に直すことはできなかった。このため一部の複合セルなどが使えず、面積が増加してしまうケースが生じる。

配置配線後のレイアウトデータにおけるパワースイッチを

(注1)：Synopsys, Inc

Cool Power(注2) を用いて最適化した。このときの制約条件として、スリープからのウェイクアップ時に Ground ラインの最大電圧が VDD の 20%未満になるように行った。

最適化済みのレイアウトデータを再び Astro で配線しマスク用の GDS を構築した。

4. 評価方法

4.1 ダイナミック電力とアクティブ時のリーク電力

細粒度パワーゲーティングを行うため全てのセルの高さを変更しているので、SPICE モデルからタイミング、電力の情報を含むライブラリを生成し使用した。ダイナミック電力とアクティブ時のリーク電力は、レイアウト後のネットリストから PowerCompiler(注1)を用いて算出した。スイッチング確率情報 (SAIF) は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。

4.2 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、シ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少し複雑である。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 HSIM(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。

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

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 から 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。

5. 評価結果

5.1 演算器の使用頻度

図 9 に HP-Core における各演算器の使用頻度を示す。

(注2)：Sequence Design, Inc

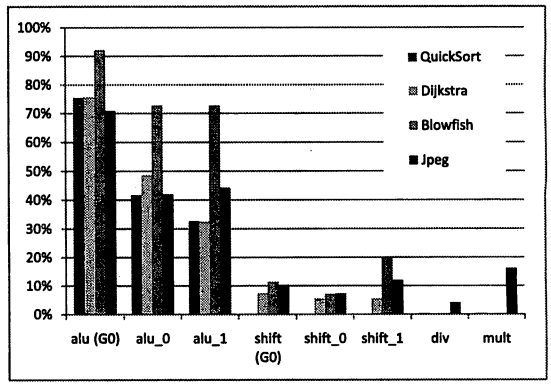

図 9 使用頻度の比較

HP-Core は ALU と Shift を二つずつ持っているので、それぞれ alu_0、alu_1、shift_0, shift_1 とした。alu(G0) と shift(G0) は Geyser-0 の演算器の使用頻度である。これらと比較して、使用頻度の高い ALU は二つのパイプラインを有効に活用し、使用頻度を減少させている。

ALU_0 と ALU_1 の頻度の合計が 100%を超えてしまう場合があるが、これは演算器を 1 サイクル早く起動させた後、依存関係の問題が発生して片側のパイプラインには命令が流れなかったというケースが存在するためである。この場合には演算器は 1 サイクルだけ起動してすぐにスリープする。この 1 サイクルは HP-Core のアーキテクチャでは避けられず、SuperScalar による電力オーバーヘッドと考えられる。

5.2 各コアの消費電力

各ユニットのブレイクイーブンポイントを表 1 に示す。表中の数値は、1 サイクルは 5ns としたサイクル数である。

表 1 Break Even Time for Each Unit (cycles)

Temperature (℃)	25	65	100	125
ALU	136	35	15	9
SHIFT	168	40	17	2
MULT	111	31	13	8
DIV	80	24	10	6

温度が上がるとアクティブ時のリークが大きくなるため、BET は短くなる。90nm で実装された Geyser-0 と比較して、BET が長くなっている。

図 10 は HP-Core と Single-Scalar の Geyser-0 の性能を比較した結果である。縦軸はサイクル数 (単位: 100 万) でアプリケーションごとに掲載した。HP-Core は Geyser-0 と比較して平均 24%の性能向上になった。HP-Core と Geyser-0 を同じ周波数で動かしたと仮定した場合の値であるため、ダイナミック電力の増大を許容すればさら HP-Core をさらに高い周波数で動作させてシングルスレッドのパフォーマンスを向上させることができる。

図 10 にアプリケーションごとの HP-Core の演算器のリーク電力を示す。縦軸は電力 (mW) で横軸に温度とアプリケーションを取ってある。Active の項はスリープを行わない場合

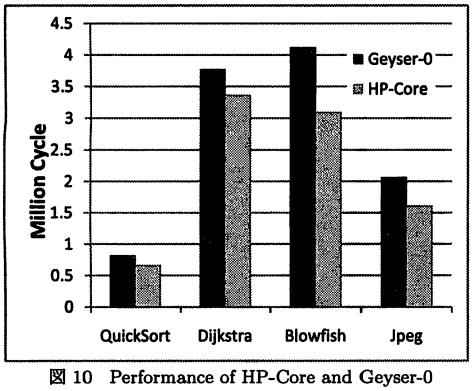

図 10 Performance of HP-Core and Geyser-0

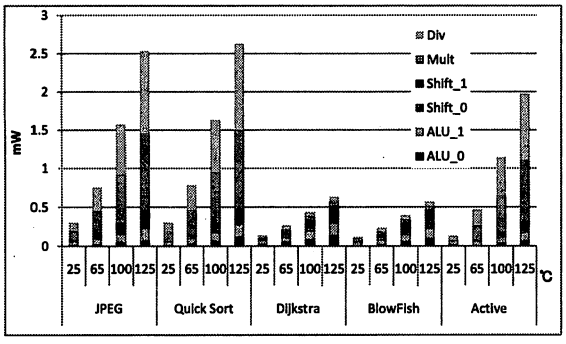

図 11 HP-Core Leakage Power

のリーク電力を表している。JPEG エンコードと Quick Sort ではスリープを行わない場合よりも電力を消費してしまっている。先に説明した表 1 を併せて考えると、スイッチングオーバヘッドが原因で電力が増大してしまったと考えられる。ALU や SHIFT はスイッチング頻度の高いため、特にオーバーヘッドの影響を大きく受けるためスリープはキャッシュミス時に限定するなどスイッチングイベントを制限する必要がある。65 ℃のときの演算器の平均リーク電力は 0.51mW であり、スリープをしない場合の 0.47mW を越えてしまっている。Dijkstra と Blowfish では削減効果の大きい乗算器と除算器を全く使わないため、電力削減に成功している。

5.3 エリアとパワーゲーティングによるオーバヘッド

表 2 に各演算器の面積と PS によるオーバーヘッドを示す。

表 2 Area Overhead (μm^2)

	Area	PS	Holder	Overhead
ALU	3330.8	275.2	79.2	10.6%
SHIFT	3030.8	298.8	76.8	12.4%
MULT	22982.8	1762	153.6	8.3%
DIV	27639.6	1301.2	153.6	5.3%

PS と Holer Cell による面積の増加は平均で 9.2%であった。Holer Cell はスリープの対象となったユニットから中間電圧が出力されるのを防ぐための Cell である。PS は回路容量や配線の混雑度に応じて最適化されている。今回の実装では比較的小さなオーバーヘッドで細粒度パワーゲーティングを行うことができた。

6. おわりに

本研究では細粒度パワーゲーティングを用いたヘテロジニアスなマルチコアアーキテクチャを 65nm のプロセステクノロジーで実装し評価した。

HP-Core では 24%の性能向上をするとともに Dijkstra、Blowfish においてはリーク電力を 46%削減できた。一方で、Qucik Sort と JPEG エンコードでは消費電力が 63%増大した。HP-Core は Geyser-0 と比較して平均 24%の性能向上になった。

パワーゲーティングによる面積の増加は平均で 9.2%であった。

今後はマルチコアを生かしたマルチスレッドアプリケーションとコア単位の周波数制御を行った場合の省電力効果について評価を行いたい。

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 株式会社半導体理工学研究センター、(株) イー・シャトルおよび富士通株式会社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] M.Ishikawa, et al.: “A 4500 mips/w, 86μa resume-standby, 11μa ultra-standby application processor for 3g cellular phones”, IEICE Trans. on Electronics, **E88-C No.4**, pp. 528–535 (2005).

[3] Y.Kanno: “Hierarchical power distribution with 20 power domains in 90-nm low-power multi-cpu processor”, ISSCC2006, pp. 540–541 (2006).

[4] K.Usami, N.Ohkubo: “An approach for fine-grained run-time power gating using locally extracted sleep signals”, ICCD (2006).

[5] N.Seki, et al.: “A fine grain dynamic sleep control scheme in mips r3000”, ICCD (2008).

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

[7] 白井 利明, et al.：“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”, 電子情報通信学会技術研究報告, VLD2007-112, pp. 43–48 (2008).

[8] 小島 悠, et al.：“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”, 情処研報 ARC, pp. 87–92 (2008).