Vol.2011-ARC-195 No.1
Vol.2011-OS-117 No.1
2011/4/13

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

小林弘明[†1] 茂木勇[†1] 木村一樹[†1]
薦田登志矢[†2] 佐藤未来子[†1] 近藤正章[†3]
中村宏[†2] 並木美太郎[†1]

本稿では，コンパイラとOSとの協調により，コア温度変化に伴う電力的損益分岐点の変動に追従する細粒度パワーゲーティング制御手法を提案する．コンパイラによる静的解析技術では，パワーゲーティング制御情報を含むオブジェクトコードを各コア温度向けに生成する．OSは，各コア温度向けのオブジェクトコードを管理し，実行時のコア温度の変化に基づいて動的に切り替えて実行する．7つのベンチマークを用いて提案機構の評価実験を行った結果，提案手法の有効性を証明した．

A Study on an Operating System for Managing Object Code Optimized for Fine-grain Power Gating Control

Hiroaki Kobayashi,[†1] Isamu Mogi,[†1]
Kazuki Kimura,[†1] Toshiya Komoda,[†2] Mikiko Sato,[†1]
Masaaki Kondo,[†3] Hiroshi Nakamura[†2]
and Mitaro Nakamura [†1]

This paper proposes a fine-grain power gating control method that deals with fluctuations in break-even point accompanying changes in core temperature by the cooperation of a compiler and an operating system. The compiler generates object code, which includes instructions with power gating directions, for various core temperatures based on static analysis. The operating system, in turn, manages multiple items of object code generated by the compiler, and selectively executes appropriate code according to changes in core temperature. The evaluation results of seven benchmark programs show that a bigger power-saving effect can be obtained with the proposed method.

1. はじめに

近年，高性能化を進めるLSIは，消費電力の増大という深刻な問題を抱えている．特に，高集積密度化に伴うリーク電力の増大が顕著である．DVFSやクロックゲーティング等の技術がダイナミック電力を低減するのに対し，ボディバイアス[8]，Dual-Vth[9]，パワーゲーティング[1]等の技術はリーク電力の低減に効果的な技術である．このうち，ボディバイアスやDual-Vthは，回路及び設計手法上の省電力技術である．これに対し，パワーゲーティングは，回路の一部に対し電源供給を遮断する技術であり，その時間的/空間的な適用範囲を動的に制御することで大幅な省電力効果が期待される．

従来のパワーゲーティング技術は，ハードウェアによって自律的に制御するものが多い[1)4)5)7)]．しかし，一般的に，パワーゲーティング技術を適用した回路では，電源供給／遮断の切り替えに伴う過渡現象や温度変化の影響で電力動向が複雑化する．そのため，ハードウェアによる自律制御だけでは期待する省電力効果を得られるとは限らない．これを踏まえ，コンパイラの静的解析に基づいて命令語レベルでパワーゲーティングを制御する手法も提案されている[2)3)6)]．これらの技術では，プログラムの静的解析時に演算ユニットのアイドルサイクルを予測し，電力的損益分岐点を指標としてコード中の適切な位置にパワーゲーティング制御用の命令を挿入することで，パワーゲーティングの時間的/空間的な適用範囲を細かく調節する．特に，文献2)では，手続き間やループ以外も解析対象とすることで，高精度の予測を実現している．これらの技術は，評価実験によってその有効性が確認されている．

従来のコンパイラ技術で静的解析の指標となっている電力的損益分岐点は，温度によって変動する．そのため，コンパイラによるパワーゲーティング制御では実行時のコア温度情報も考慮して最適化することが望ましい．しかし，コンパイラの解析技術だけでは，温度ごとに最適化したパワーゲーティング制御向けコードの生成に留まり，実際のコア温度の変動に伴う電力的損益分岐点の変動には対処できないという課題がある．

そこで，本稿では，従来のコンパイラによるパワーゲーティング制御技術を活用し，OS

†1 東京農工大学
Tokyo University of Agriculture and Technology
†2 東京大学
The University of Tokyo
†3 電気通信大学
The University of Electro-Communications

Vol.2011-ARC-195 No.1
Vol.2011-OS-117 No.1
2011/4/13

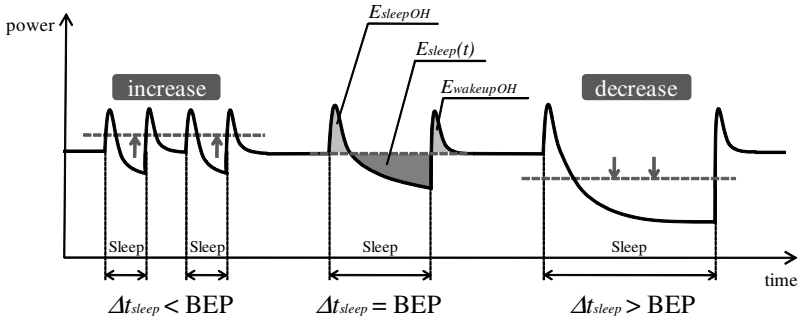

図 1 損益分岐点
Fig. 1 Break-even point

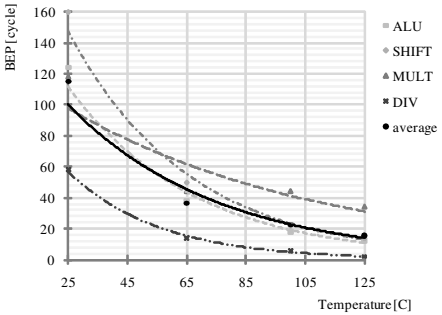

図 2 BEP と温度の関係
Fig. 2 BEP versus temperature

がコア温度の変動に応じて最適なコードを実行するというパワーゲーティング制御手法を提案する．具体的には，コンパイラは各温度向けに最適な命令シーケンスを含むオブジェクトコードを生成する．そして，OS がこれらのコードを管理し，実行時温度変化に基づいて実行するコードを動的に切り替える．すなわち，本提案手法は，コンパイラと OS とが協調し，コンパイラは温度毎の戦略を提供し，OS は実際の温度変化に応じて最適な戦略を選択する手法である．

本稿では，2 章でパワーゲーティング技術について述べ，3 章で提案手法の概要について述べる．4 章で OS の制御について述べ，5 章で実装について述べる．6 章で，パワーティング技術を搭載したプラットフォームにおける評価実験について述べ，考察を述べる．7 章でまとめを述べる．

2. パワーゲーティング関連技術

2.1 パワーゲーティング

パワーゲーティングは，アイドル状態の回路への電源供給を遮断することでリーク電力を削減する技術である．パワーゲーティングを適用した回路の電力動向を**図 1** に示す．電源の供給/遮断の遷移には，オーバヘッド電力が生じる．また，電源供給の遮断後，電力は過渡的に減少していく．そのため，ある一定期間以上電源供給を遮断し続けなければ省電力効果を得られない．この一定期間を BEP（Break-Even Point）と呼ぶ．BEP は，次式が成り立つときのスリープ期間 Δt の値である．

$$E_{sleep}(t) = E_{sleepOH} + E_{wakeupOH}$$

2.2 BEP を考慮した従来技術

効率的に電力を削減するために，BEP を考慮したパワーゲーティング制御手法がこれまでに提案されている．文献 4) では，演算ユニットへパワーゲーティング技術を適用し，二つの戦略によるパワーゲーティング制御を提案している．一つは，一定期間以上のアイドルサイクルを検出したときにパワーゲーティングを施す戦略であり，もう一つは，分岐遅延時にパワーゲーティングを施す戦略である．どちらも，ハードウェアが自律的に制御を行うものであり，そのための追加リソースが大きいことが課題として挙げられる．文献 1) では，MIPS R3000 プロセッサの演算ユニットを 4 つのモジュールに分け，それぞれにパワーゲーティング技術を適用している．こちらでは，パワーゲーティング制御用に ISA を拡張しており，拡張命令実行時に対象演算ユニットに対するパワーゲーティングの on/off を指定するパワーゲーティング制御命令を備えている．文献 2) では，コンパイラの静的解析の視点から，この命令を用いたパワーゲーティング制御手法を提案している．本技術では，静的解析により各演算ユニットのアイドルサイクルを予測する．この情報に基づき，BEP を指標として，パワーゲーティング制御情報を命令に付加する．具体的には，アイドルサイクルが BEP を上回る区間では，パワーゲーティングを on にし，下回る区間では off にする．本技術は，シミュレーションテストによりその効果を確認している．その他，文献 3) や文献 6) でも，コンパイラにより静的解析を行い，パワーゲーティング制御用の命令を挿入することで，パワーゲーティングの適用区間を細かく制御する手法を提案している．

Vol.2011-ARC-195 No.1
Vol.2011-OS-117 No.1
2011/4/13

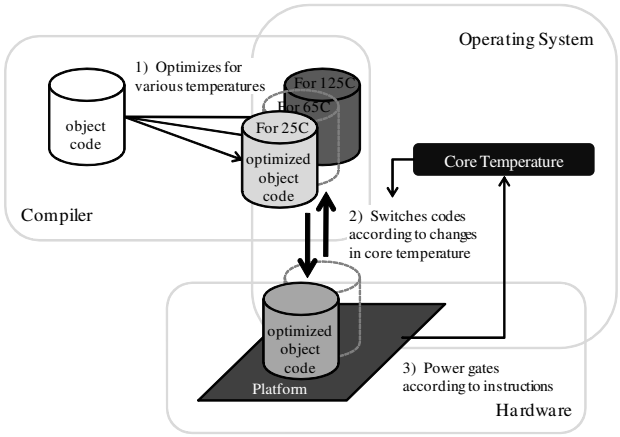

図 3 提案手法の概要
Fig. 3 Overview of the proposed method

これらの技術の課題として，回路の温度変化に伴う BEP の変動に対処できない点が挙げられる．BEP は，**図 2** に示すように，回路の温度変化の影響を受けて大きく変動することが知られている．先行研究では，コア温度変化に伴う BEP の変動に対処した例はない．提案手法では，コア温度変化に伴う BEP の変動に対処することで，効果的にリーク電力を削減することが期待できる．

3. 提案手法の概要

提案手法の概要を**図 3** に示す．提案手法では，まず 1) コンパイラにより，コア温度ごとに，最適な命令シーケンスを含むオブジェクトコードを生成する．次に 2) OS が温度ごとのオブジェクトコードを管理し，実行時のコア温度変化に基づき，最適なオブジェクトコードを選択して実行する．そして 3) オブジェクトコードに埋め込まれたパワーゲーティング制御情報に基づき，ハードウェアがパワーゲーティングを行う．

提案手法を実現するにあたり，ハードウェアアーキテクチャに求める要件として，以下の 2 点が挙げられる．これに関しては，先行研究でいくつか提案されている[1)3)]．

(1) パワーゲーティング技術を搭載していること．
(2) パワーゲーティング制御用の命令を備えていること．

また，コンパイラに求める要件として，以下の 3 点が挙げられる．(3) は，OS がオブジェクトコードを切り替えたときのプログラムの一貫性を保証するために必要である．

(1) パワーゲーティング制御用の命令を含むコードを生成すること．
(2) BEP のように温度に影響される値を指標とし，目的温度によって異なるコードを生成すること．
(3) 目的とする温度によって，命令番地やコードサイズが変化しないこと．

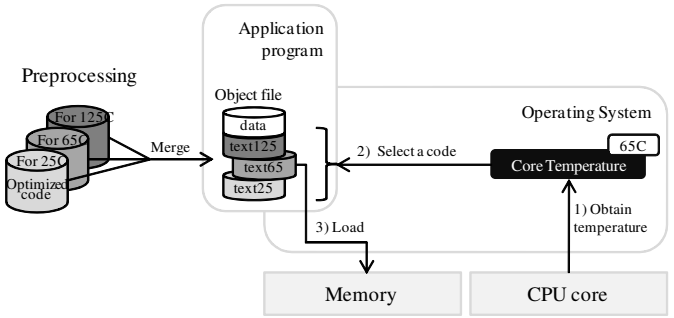

図 4 OS の処理
Fig. 4 Work of operating System

先行研究で提案されてきたコンパイラ技術は，大きく 2 種類に分類することができる．一つは，パワーゲーティング制御用の命令をコードに挿入するものである[3)6)]．この場合，(3) を満たすために，各温度向けのコードの命令番地やコードサイズを揃える処理を施す必要がある．もう一方は，命令中の特定ビットの値により，パワーゲーティングを指示するものである[2)]．こちらは，新たな命令を挿入せず，特定ビットを書き換えるだけであるため，目的とする温度が変わっても命令番地やコードサイズが変化しない．コードサイズの増大による性能への影響を考えると，提案手法では後者が望ましい．

4. OS におけるパワーゲーティング制御向けコードの管理

4.1 概　　要

提案手法における，OS の処理の概要を**図 4** に示す．OS は，コンパイラにより生成された各コア温度向けのオブジェクトコードを管理する．そして，プログラム実行時に，1) コア温度を取得し，2) 取得したコア温度に基づきコードを選択し，3) 選択したコードをメモリへ展開する．

各コア温度向けのオブジェクトコードを一元管理するために，前処理として，各コア温度

Vol.2011-ARC-195 No.1
Vol.2011-OS-117 No.1
2011/4/13

向けの複数のテキストセグメントを格納するオブジェクトファイルを生成する．この詳細については，4.2 節で述べる．OS は，各コア温度向けのテキストセグメントの仮想アドレスを等しくすることで，任意のタイミングで実行するオブジェクトコードを切り替えても命令シーケンスの一貫性を保証する．各タスクの仮想アドレス空間の詳細については，4.3 節で述べる．コア温度変化に基づく実行時のコード切り替え処理の詳細については，4.4 節で述べる．

4.2 オブジェクトファイルの生成

プログラムのオブジェクトコードは，ヘッダ情報やデバッグ情報等の各種情報と共にオブジェクトファイルに格納される．主要なオブジェクトファイルフォーマットの一つである ELF では，ELF ヘッダ，セクションヘッダ，プログラムヘッダの 3 種類のヘッダを格納する．ELF ヘッダはファイル全体の情報を，セクションヘッダは「セクション」と呼ばれる管理単位の情報を，プログラムヘッダは「セグメント」と呼ばれる管理単位の情報を記述する．コンパイラやアセンブラ，リンカは，セクションヘッダテーブルが記述するセクションの集合として ELF ファイルを扱う．一方，ローダは，プログラムヘッダテーブルが記述するセグメントの集合として ELF ファイルを扱う．そのため，ローダによって実行時コア温度に最適なコードをロードするには，コア温度ごとに独立したテキストセグメント，及びその情報を記述するプログラムヘッダが必要になる．また，各コア温度向けのオブジェクトコード間で，プログラムの一貫性を保証するため，各コア温度向けのテキストセグメントの仮想アドレスをすべて等しくする必要がある．

以上を踏まえ，**図 5** に示す手順により，各コア温度向けのオブジェクトコードを一つのファイルに統合する．この処理によって生成されるオブジェクトファイルの中身は**図 6** のようになる．各コア温度向けにテキスト領域（セグメント／セクション）があり，それに対応するプログラム／セクションヘッダがある．各コア温度向けのテキスト領域には，目的とするコア温度において最適なパワーゲーティング制御を行うための命令シーケンスが格納される．テキスト以外の領域は温度の影響を受けないため，一つを共有する形になる．OS は，このオブジェクトファイルを用いて各コア温度向けの複数のコードを一元管理する．

4.3 タスクのアドレス空間

一般的に，OS は，オブジェクトファイルに格納されている各種ヘッダ情報が記述するセグメント情報に基づき，各タスクに一意の仮想アドレス空間を提供する．ELF ファイルでは，プログラムヘッダテーブルからセグメント情報を参照できる．

図 6 に示すオブジェクトファイルは，ファイル内の構造的には，各テキストセグメントが

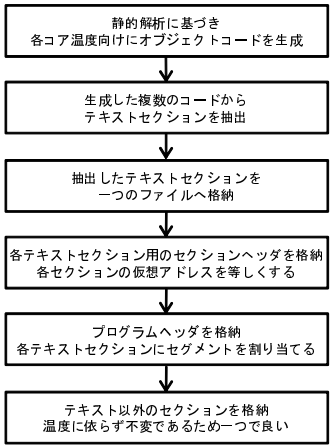

図 5 オブジェクトファイル生成フロー
Fig. 5 Object-file generation flow

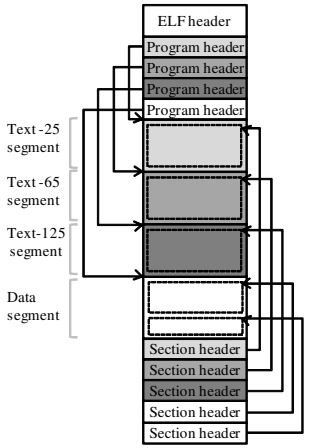

図 6 生成されるオブジェクトファイル
Fig. 6 Contents of generated object file

線形に並んでいる．しかし，プログラムヘッダテーブルに格納されている各テキストセグメントの仮想アドレスはすべて等しい．そのため，プログラムヘッダテーブルによって定義される仮想アドレス空間は，**図 7** - 左のように，仮想アドレスの等しい多重のテキストセグメントを持つものとなる．OS は，このようなアドレス空間をタスクに提供し，温度変化に応じてテキストセグメントを切り替えることで，コア温度に最適なコードを動的に選択して実行する（図 7）．

4.4 実行時のコード切換処理

OS は，4.2 節で述べたオブジェクトファイルを用い，コア温度変化に基づいて最適なコードを選択・実行する．本節では，実行時のテキストセグメントの切り替え処理について述べる．

一般的に，仮想記憶を提供する OS は，ページテーブルを用いて VPN（Virtual Page Number）と PFN（Page Frame Number）との対応を管理する．そして，各タスクに一意の仮想アドレス空間を提供する．さらに，デマンドページング方式を採用する場合は，アクセス要求があったときにはじめて物理メモリをページに割り当てることで，メモリ使用量を最小限に抑える．

提案手法では，図 6 に示すオブジェクトファイルを使用する．本オブジェクトファイル

Vol.2011-ARC-195 No.1
Vol.2011-OS-117 No.1
2011/4/13

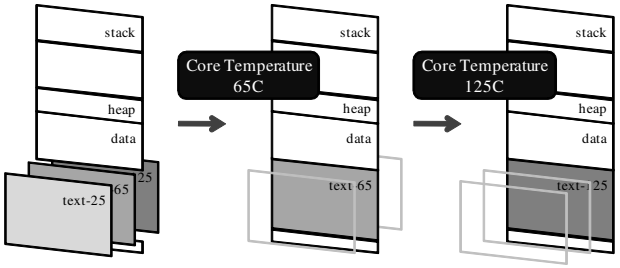

図 7 仮想アドレス空間
Fig. 7 Virtual address space

は，各温度用に複数のテキストセグメントを持つため，すべてをメモリに展開した場合，メモリオーバヘッドが大きい．そこで，デマンドページングを採用し，必要なときに必要なコードをロードすることとする．必要なときとは，すなわち，ページフォルトが発生したときである．

ページフォルト発生時の処理の流れを**図 8** に示す．OS は，はじめに，セグメント情報を参照し，ページフォルトを引き起こしたアクセス要求の正当性を判断する．アクセス要求が不当である場合，プログラムを強制終了する．アクセス要求が正当である場合，その後の処理は，アクセス要求のあった仮想アドレスがテキストセグメントに属するか，そうでないかで異なる．以下に，それぞれの処理を示す．(1) の処理により，コア温度変化に基づいて最適なコードを選択・実行することが可能となる．

(1) テキストセグメント

はじめに，コア温度を取得する．そして，取得したコア温度に基づいてテキストセグメントを選択する．ここで，温度 t_1，t_2 用の 2 つのテキストセグメントがある状況を考える．取得したコア温度が t（$t_1 < t < t_2$）のとき，どちらのテキストセグメントを選択するのが適切であるか．これを決めるために，温度 t_1，t_2 の境界となる温度を定義する．文献 1) で採取した BEP のデータを分析したところ，BEP と温度との関係は指数関数的に近似できることを確認した．つまり，温度 t における BEP の値は次式で近似的に求まる．係数 A，B は，回路の規模等により決まる．

$$BEP(t) = Ae^{Bt}$$

温度 t_1，t_2 における BEP の平均を $BEP(t_{boundary}(t_1,t_2))$ とすると，$t_{boundary}(t_1,t_2)$ は次式により求まる．

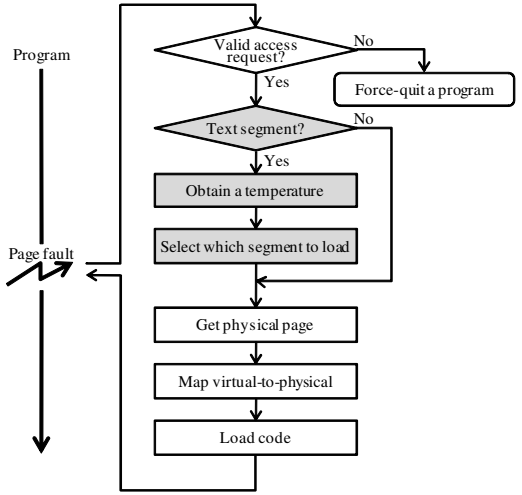

図 8 ページフォルト処理
Fig. 8 Page fault handling

$$t_{boundary}(t_1,t_2) = \frac{1}{B}\ln\frac{e^{-Bt_1}+e^{-Bt_2}}{2}$$

この $t_{boundary}(t_1,t_2)$ を，温度 t_1，t_2 の境界となる温度と定義する．OS は，この値に基づき，適切なテキストセグメントを選択する．例えば，取得したコア温度が t（$t_1 < t < t_2$）のとき，$t < t_{boundary}(t_1,t_2)$ であれば t_1 を，そうでなければ t_2 を選択する．テキストセグメントを選択したら，アクセス要求のあった仮想アドレスが属するページに実ページを割り当て，対応するコードをロードする．

(2) その他のセグメント

テキストセグメントは異なり，温度を考慮する必要がない．つまり，一般的なページフォルト処理と同様に，アクセス要求のあった仮想アドレスが属するページに実ページを割り当て，コードをロードする．BSS 領域である場合は，割り当てたページを初期化する．

5. 実　　装

文献 12) で提案されている MIPS R3000 アーキテクチャ向けの軽量 OS をベースに，4 章に述べた機能を OS に実装する．実行基盤としては，文献 1) で提案されている MIPS R3000

Vol.2011-ARC-195 No.1
Vol.2011-OS-117 No.1
2011/4/13

表 1 追加／修正コード数
Table 1 number of changed or added codes

モジュール	追加／修正行数
ローダ	350
実ページ管理	90
メモリアロケータ	90
ページテーブル	140
TCB 管理	100
ページフォルト処理	100
TLB 例外ハンドラ	50
UTLB 例外ハンドラ	80

互換のプロセッサを Xilinx 社の ML501[11] 上に構築したものを用いる．本プロセッサでは，演算ユニットを ALU，シフタ，乗算器，除算器の 4 つに分割し，それぞれにパワーゲーティング技術を適用している．コンパイラ技術としては，文献 2) で提案されているものを用いる．本コンパイラは，命令中特定ビットの値によりパワーゲーティングを指示するものであり，生成するオブジェクトコードのサイズ，命令番地は目的とする温度によって変化しない．

OS に追加／修正したコード数を**表 1** に示す．組込み向けの軽量 OS ということもあり，もともとメモリ管理に関する機能があまり提供されていなかったため，基本的な機能も追加した．Linux 等の汎用 OS では，メモリ管理に関する基本的な機能をあらかじめ備えているため，より少ない追加／修正で提案手法を実現できると考えられる．

6. 評　　価

実装した OS 機構による消費電力削減効果を評価する．

6.1 評 価 方 法

評価に使用する FPGA ボード上では，実際の消費電力を計測することができない．そこで，以下に述べる評価式により消費リーク電力を計算する．まず，総スリープサイクルを T_s，i サイクルスリープの回数を N_i，i サイクルスリープにおける 1 サイクルあたりの平均消費リーク電力を L_{ws_i} として，次式によりスリープ時平均リーク電力を求める．

$$L_{ws} = \sum_i \left(L_{ws_i} \frac{iN_i}{T_s} \right)$$

次に，総サイクルを T，アクティブ時の平均消費リーク電力を L_{wa} として，次式により全体の平均リーク電力を求める．なお，アクティブ時の平均リーク電力は，文献 1) で求めた値を用いる．

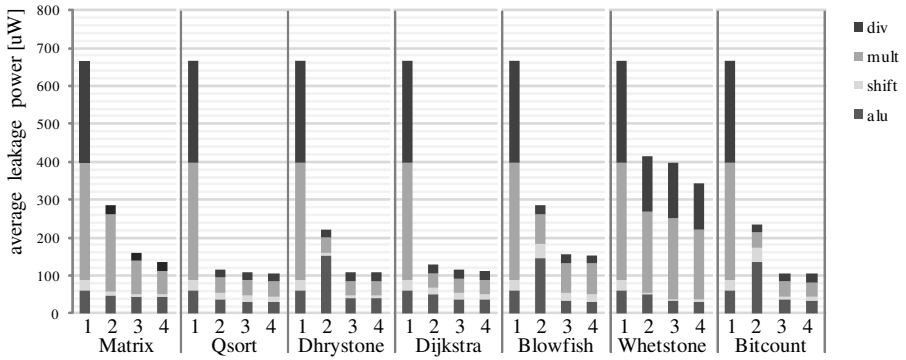

図 9 平均リーク電力
Fig. 9 Average lekage power

$$L_w = L_{ws} \frac{T_s}{T} + L_{wa} \frac{1 - T_s}{T}$$

各種サイクル情報はパフォーマンスカウンタを用いて計測する．また，本環境では，コア温度を計測することができないため，0〜130 の範囲でソフトウェアからランダムに与える．ベンチマークは，Matrix,，Qsort，Dhrystone，Dijkstra，Blowfish，Whetstone，Bitcount の計 7 個を使用する[10]．なお，すべて最適化オプション-O2 でコンパイルする．

以下に示す 4 つの手法において，ベンチマークを実行し，消費リーク電力の比較評価を行う．評価対象の回路は，ALU，シフタ，乗算器，除算器の計 4 演算ユニットとする．

- 手法 1：パワーゲーティングを適用しない．
- 手法 2：ハードウェアによる自律制御．演算実行後，毎回対象演算ユニットへの電源供給を遮断する．
- 手法 3：コンパイラによるパワーゲーティング制御．コンパイラによって 125°C 向けに生成したオブジェクトコードを実行する．
- 手法 4：コンパイラと OS との協調制御（提案手法）．コンパイラによって 25°C，65°C，125°C 向けに生成したオブジェクトコードを OS の支援により動的に切り替えて実行する．

6.2 評 価 結 果

すべてのコア温度における平均リーク電力を**図 9** に示す．グラフの下の数字は，前述の

Vol.2011-ARC-195 No.1
Vol.2011-OS-117 No.1
2011/4/13

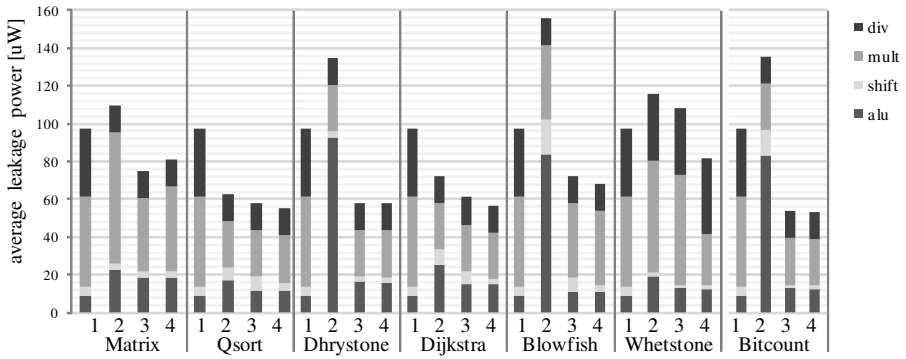

図 10 25°C におけるリーク電力
Fig. 10 Average leakage power at a core temperature of 25°C

手法の番号である．全体的に見て，パワーゲーティングを行うことで，消費リーク電力を大幅に削減できていることが確認できる．特に，本研究の提案手法（手法 4）が最もリーク電力削減率が高く，平均 77.3%のリーク電力の削減に成功している．手法 2 と手法 4 とを比較すると，手法 4 では最高 55.7%，平均 35.0%リーク電力を削減している．手法 3 と手法 4 とを比較すると，手法 4 では最高 15.6%，平均 5.5%リーク電力を削減している．

コア温度が低いほど提案手法の効果が高い傾向がみられた．コア温度 25°C における平均リーク電力を**図 10** に示す．Pattern 2 では，ベンチマークによってはパワーゲーティングを行うことでむしろ消費リーク電力が増大している．この原因は，低温化においては BEP が長く，BEP を下回るスリープが頻発したためであると考えられる．これに対し，手法 3 と手法 4 では，効率的にパワーゲーティング制御を行っていることが確認できる．特に， 手法 4 では，Matrix 以外のすべてのベンチマークにおいて，最もリーク電力削減効果が高いことが確認できる．手法 2 と手法 4 とを比較すると，手法 4 では最高 60.7%，平均 37.6%リーク電力を削減している．手法 3 と手法 4 とを比較すると，手法 4 では最高 24.5%，平均 4.9%リーク電力を削減している．以上の結果から，OS の支援によりコア温度変化に伴う BEP の変動に追従したパワーゲーティング制御を行うことで，消費電力削減効果を向上することができたといえる．

ベンチマーク実行時のサイクル数を計測し，提案 OS 機構での処理によるオーバヘッドを求めた．すべてのベンチマークにおけるオーバヘッドの平均は 3.23%であった．オーバヘッドの大半は，二次記憶からのロード時間が占めると考えられる．そのため，今後，DMA やディスクキャッシュを採用することで，さらにオーバヘッドを抑えることができると考えられる．

7. おわりに

本稿では，コンパイラと OS との協調により，コア温度変化の変動に伴う電力的損益分岐点の変動に対処するパワーゲーティング制御手法を提案した．提案手法において，コンパイラは，各温度向けに最適な命令シーケンスを含むオブジェクトコードを生成した．そして，OS がこれらのコードを管理し，実行時コア温度変化に基づいて実行するコードを動的に切り替えることで，コア温度の変動に伴う電力的損益分岐点の変動に対処した．

プログラム実行の前処理として，各コア温度向けのオブジェクトコードを 1 つのオブジェクトファイルへ統合することで，各コア温度向けのコードを一元管理した．OS は，オブジェクトファイルに格納される各種ヘッダ情報に基づき，各タスクに一意の仮想アドレス空間を提供した．提供した仮想アドレス空間では，同一仮想アドレスに各テキストセグメントを配置することで，各コア温度向けコード間の命令シーケンスの一貫性を保証した．デマンドページングを採用し，ページフォルトのタイミングでコア温度に最適なコードをロードすることで，動的に実行するコードを切り替えた．

評価として，パワーゲーティング技術を搭載したプラットフォームにおいて，Matrix, Qsort, Dhrystone, Dijkstra, Blowfish, Whetstone, Bitcount の計 7 つのベンチマークを用いて提案機構の実験を行った．その結果，提案機構により，ハードウェア単体と比較して平均 35%，コンパイラ単体と比較して平均 6%リーク電力が削減されることを確認した．

今後の課題としては，プログラムの挙動特性と消費電力低減効果との関係を分析し，本稿で提案した手法を改善することが挙げられる．そのために，より多くのベンチマークを利用して評価実験を行う必要がある．

謝辞 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「革新的電源制御による次世代超低消費電力高性能システム LSI の研究」によるものである．

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,

Vol.2011-ARC-195 No.1
Vol.2011-OS-117 No.1
2011/4/13

M. Namiki, M. Kondo, and H. Nakamura, “A Fine-grain Dynamic Sleep Control Scheme in MIPS R3000”, Proceeding of the 26th IEEE International Conference on Computer Design, pp. 612-617, 2008.

2) Toshiya Komoda, Hiroshi Sasaki, Masaaki Kondo, and Hiroshi Nakamura, “Compiler-Directed Fine Grain Power Gating for Leakage Power Reduction in Microprocessor Functional Units”, 7th Workshop on Optimizations for DSP and Embedded Systems, Mar. 2009.

3) Yi-Ping You, Chingren Lee, and Jenz Kuen Lee, “Compilers for Leakage Power Reduction”, ACM Transactions on Design Automation of Electronic Systems, Vol. 11, pp.147-164, Jan. 2006.

4) Zhigang Hu, Alper Buyuktosunoglu, Viji Srinivasan, Victor Zyuban, Hans Jacobson, and Pradip Bose, “Microarchitectural techniques for Power Gating of Execution Units”, Proceedings of the 2004 International Symposium on Low Power Electronics and Design, pp. 32-37, Aug. 2004.

5) Youngsoo Shin, Jun Seomun, Kyu-Myung Choi, and Takayasu Sakurai, “Power gating: Circuits, design methodologies, and best practice for standard-cell VLSI designs”, ACM Transactions on Design Automation of Electronic System (TODAES), vol. 15, pp. 281-317, Sep. 2010.

6) Soumyaroop Roy, Nagarajan Ranganathan, and Srinivas Katkoori, ”A Framework for Power-Gating Functional Units in Embedded Microprocessors”, IEEE transactions on VLSI Systems, vol. 17, pp. 1640-1649, Nov. 2009.

7) Weiping Liao, Basile J.M., and Lei He, “Microarchitecture-level leakage reduction with data retention”, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 13, pp. 1324-1328, Nov. 2005.

8) David Duarte, Yuh-Fang Tsai, Narayanan Vijaykrishnan, and Mary Jane Irwin, “Evaluation Run-Time Techniques for Leakage Power Reduction”, ASPDAC 2002, pp. 31-38, Aug. 2002.

9) James T. Kao and Anantha P. Chandrakasan “Dual-Threshold Voltage Techniques for Low-Power Digital Circuits”, IEEE Journal of Solid-State Circuits, vol. 35, pp. 1009-1018, Jul. 2000.

10) Guthaus M.R., Ringenberg J.S., Ernt D., Austin T.M., Mudge T., and Brown R.B., “MiBench: A free, commercially representative embedded benchmark suite”, 2001 IEEE international Workshop on Workload Characterization (WWWC 2001), pp. 3-14, Dec. 2001.

11) Xilinx, “ML501 Reference Design”, UG227, vol. 1.0, Jun, 2007.

12) 砂田徹也, 木村一樹, 近藤正章, 天野英晴, 宇佐美公良, 中村宏, 並木美太郎, “細粒度パワーゲーティングを制御する OS の資源管理方式”, IPSJ SIG Technical Report. 2010-ARC-189 pp. 1-8, Apr. 2010.