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

2007－ARC－173（2）
2007／5／31

Drowsyキャッシュにおける活性期間の制御手法に関する検討

小林良太郎† 谷口 英樹† 島田 俊夫†

† 名古屋大学大学院工学研究科 〒464-8603 愛知県名古屋市千種区不老町
E-mail: †{kobayashi,taniguchi,shimada}@shimada.nuee.nagoya-u.ac.jp

あらまし 近年プロセス技術の進歩により、リーク電流によって消費される静的電力が問題となってきている。これに対し、チップ上のトランジスタを数多く使用しているキャッシュのリーク電流を削減する手法としてDrowsy Cacheが提案されている。本研究では、Drowsy Cacheにおいて、キャッシュ・ラインを活性状態のまま保持する期間を制御することで、目標とする性能や静的電力を達成する手法について検討を行う。

キーワード 低リーク・キャッシュ、リーク電流、ライン制御方式

A Study on Control Scheme of Awake Time in Drowsy Caches

Ryotaro KOBAYASHI†, Hideki TANIGUCHI†, and Toshio SHIMADA†

† Graduate School of Engineering, Nagoya University Furocho, Chikusa-ku, Nagoya, Aichi, 464-8603 Japan
E-mail: †{kobayashi,taniguchi,shimada}@shimada.nuee.nagoya-u.ac.jp

Abstract Recently static power due to the leakage current has been a major problem as process technology advances. Drowsy Cache is one of the techniques to reduce the leakage power consumed in a cache which contains a large amount of transistors. In this paper, we focus on Drowsy Cache and propose a scheme that controls the interval for which a cache line is kept active in order to achieve the given performance or the given static power.

Key words Low-leakage cache, Leakage current, Line control scheme

1. はじめに

近年の携帯機器では、バッテリー駆動時間の延長や発熱の抑制に加え、機能の高度化も重要である。そのため、モバイル・プロセッサの設計においては、性能の向上と消費電力の削減をともに実現することが要求されるようになってきている。

CMOS回路の消費電力は、負荷容量の充放電による動的消費電力と、トランジスタのリーク電流による静的消費電力に分けることができる。従来、動的消費電力が消費電力の大部分を占めていた。そのため、プロセス技術の進歩により、性能の向上と消費電力の削減をともに実現することができていた。しかし近年、トランジスタの閾値電圧の低下に伴ってリーク電流が増加しつつあるため、静的消費電力の削減が重要な課題となってきている。

チップ上において、キャッシュは多くのトランジスタを使用しており、多くのリーク電流が流れる。そのため、キャッシュのリーク電流を削減する研究がいくつも行われてきた[1][2][3][4][5][6][9]。いずれも、キャッシュ・ライン（以降ラインと略記する）を高リークな活性状態から、低リークな待機状態に切替えることにより、リーク電流を大幅に削減する。しかし、待機状態のラインへのアクセスはペナルティを生じさせるため、性能に悪影響を与える。そのため、どのようにしてラインの状態を切替えるかを決める、**ライン制御方式**が重要となる。

これまで様々なライン制御方式が提案されてきた。しかし、これらの方式では、電力効率にのみ着目しているので、静的消費電力や性能低下率が、アプリケーションのデータ参照の振舞いによって変化する。したがって、発熱に対する要求が厳しい携帯機器に搭載するために、ピーク消費電力をある一定の値以下に制限したいという要求に応えることができない。あるいは、携帯機器が必要とするリアルタイム性を満たすために、性能低下率をある一定の値以下に制限したいという要求に応えることができない。

そこで、本研究では、ラインの状態を切替えることでリーク電流を削減するキャッシュにおいて、静的消費電力や性能低下率をある一定の値に保つことを目的とする。そしてこの目的を達成するためのライン制御方式を提案する。提案機構では、ラインが活性状態にある期間（**活性期間**）に着目しこれを制御する。これにより、目標とする静的消費電力、あるいは、目標とする性能を達成できるようにする。

なお、本論文では、動的消費電力が最も高くなった場合を考慮して、静的消費電力の目標値が与えられると仮定する。また、

ラインの待機状態を実現する方式として、Drowsy Cache [1] で用いられている方式を用いる。この方式は、待機状態のライン (**待機ライン**) へのアクセスが性能に与える影響を比較的小さくすることができるという特徴を持つ。

2 章ではリーク電流を削減手法について説明する。3 章ではラインの活性期間を制御する手法と、提案手法を実現するための機構を示す。4 章では提案機構の評価を行う。最後に本論文をまとめる。

2. リーク電流削減手法

ラインの状態を切替えることでリーク電流を削減するキャッシュでは、待機状態の実現方式と、ライン制御方式が重要となる。

まず、待機状態の実現方式について述べる。

待機状態を実現する代表的な方式として、Cache Decay [3] で用いられている方式と、Drowsy Cache [1] で用いられている方式を挙げることができる。

Cache Decay では、電源供給を停止することでラインを待機状態にし、リーク電流を削減する。この場合、SRAM セルの記憶を保持することはできないため、待機ラインへのアクセスはキャッシュミスとなり、性能に悪影響を与える。一方、Drowsy Cache では、動的に電源電圧を下げることでラインを待機状態にし、リーク電流を削減する。この場合、SRAM セルの記憶は保持されるが、それを参照するためには、電源電圧を上げてラインを活性状態に戻す必要があり、それに要する時間がペナルティとなる。

リーク電流の削減に関しては、待機状態を実現するために電源供給を停止する Cache Decay の方式が有利である。一方、性能低下の抑制に関しては、待機状態において SRAM セルの記憶を保持できる Drowsy Cache の方式が有利である。単純にどちらの方式が良いかを決めることはできないが、本論文では、性能低下の抑制を重視し、Drowsy Cache の方式を用いてラインを待機状態にすることとする。

つぎに、ライン制御方式について述べる。なお本論文では、ラインを待機状態から活性状態に切替えることを**活性化**と呼び、活性状態から待機状態に切替えることを**非活性化**と呼ぶこととする。

待機状態の実現方法に関わらず、待機ラインへのアクセスはペナルティを生じ、性能に悪影響を与える。そこで、性能低下の抑制とリーク電流の削減をできるだけ両立させることを目的として、様々なライン制御方式が提案された。これらのほとんどは、データ参照の局所性を利用してラインの制御を行う。例えば、Flautner らは、一定の間隔で、活性状態のライン (**活性ライン**) を全て非活性化する方式を提案している [1]。Kaxiras らは、各ラインへのアクセスを監視し、一定サイクル以上参照されなかったラインを非活性化する方式を提案している [3]。小宮らは、Kaxiras らの方式に改良を加え、アクセスが集中するラインを常に活性状態に保つ、常活性ライン方式を提案している [6]。常活性ライン方式は、リーク電流の削減効果を維持しつつ、性能低下を大きく抑制することに成功しており、非常に効率的なアプローチである。

本論文の目的は、静的消費電力や性能低下率をある一定の値に保つことにあるが、それによって、効率が大幅に低下することは許されない。そこで、本論文では、効率の高い常活性ライン方式の概念を導入することとする。

しかし、従来のライン制御方式では、電力効率にのみ着目しているので、アプリケーションのデータ参照のパターンが変化すると、静的消費電力や性能低下率が変化する。例えば、Drowsy Cache において、32K サイクル毎に全ラインを非活性化する Simple 方式 [1] を用いると、リーク電流を削減しない場合に対し、静的消費電力は最小で 27 %、最大で 55 %となる (評価の詳細は 4 章で述べる)。したがって従来方式では、静的消費電力や性能低下率をある一定の値以下に制限したいという要求に応えることができない。

3. 提案方式

静的消費電力や性能低下率をある一定の値以下に制限するため、ラインの活性期間に着目する。ラインの活性期間は、静的消費電力や性能と相関がある。具体的には、ラインの活性期間が長ければ、静的消費電力は増加し、性能低下率は減少する。一方、ラインの活性期間が短ければ、静的消費電力は減少し、性能低下率は増加する。そこで本論文では、ラインの活性期間を制御することで、目標とする静的消費電力や性能低下率を達成する方式を提案する。

以下では、まず、提案方式の概要について述べる。その後、提案機構の各構成要素について詳しく説明する。

3.1 概　　要

提案方式では、ラインの状態を 3 種類用意する：待機状態、活性状態、常活性状態。効率的にラインを制御するため、アクセスが集中するラインだけを常活性状態に遷移させる。また、目標とする消費電力や性能を達成するため、活性ラインの数と常活性ラインの数を制御し、それによって間接的に活性期間を制御する。

ここで、アクセスが集中するラインを検出する方策について述べる。従来機構では、アクセスが集中しているかどうかを調べるために、ライン毎に用意されたカウンタを用いて、アクセス回数をカウントしている。しかし、この機構を導入すると、提案方式のハードウェア・コストがさらに増加してしまう。そこで、本論文では、待機状態を維持していた最後の期間 (**最終待機期間**) に着目し、最終待機期間が短ければ、アクセスが集中しているとみなすこととする。そして、最終待機期間が短いかどうかを調べるために、非活性化したラインのインデクスの履歴を用いて、現在活性状態にあるラインが最後に非活性化された時期を調べる。最近の履歴のみ保持すれば良いので、ハードウェア・コストは低いと期待できる。

図 1 に、提案方式を備えたキャッシュの例を簡略化して示す。図において、点線で囲まれた部分はキャッシュを示し、アドレス・デコーダ、リーク電流削減回路、タグ・アレイ、データ・アレイを持つ。リーク電流削減回路は、ラインを非活性化することにより、リーク電流を削減する。なお、前章で説明した通り、

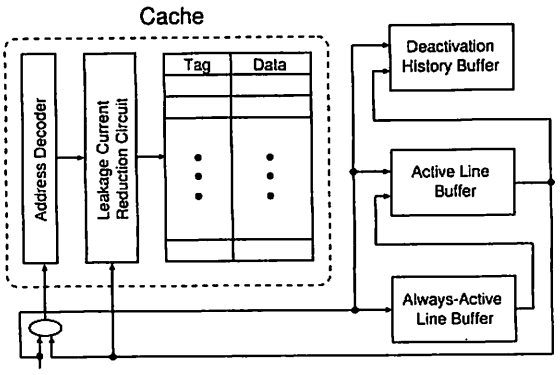

図 1 提案方式

本論文では、Drowsy Cache と同じ方式を用いて、ラインの待機状態を実現することとする。

図において、キャッシュの右側は、提案方式を実現する機構である。提案機構は、以下 3 つの要素で構成される。

- 非活性化履歴バッファ
  最近、非活性化したラインのインデクスの履歴を保持する FIFO 形式のバッファである。活性ラインの最終待機期間を調べるために用いる。
- 活性ライン・バッファ
  活性ラインのインデクスを保持する FIFO 形式のバッファである。活性ラインの数がある閾値 (Th_{normal}) を超えないようにするために用いる。
- 常活性ライン・バッファ
  常活性ラインのインデクスを記録する FIFO 形式のバッファである。常活性ラインの数がある閾値 (Th_{always}) を超えないようにするために用いる。

ライン制御動作の概要を以下に示す。

待機状態にあるライン (Ls と表記) が活性化されると、非活性化履歴バッファを参照して、各エントリのインデクスと Ls のインデクスを比較する。Ls のインデクスが、非活性化履歴バッファに存在する場合、Ls が最後に非活性化されたのは最近なので、最終待機期間は短かいと判定する。そうでない場合、最終待機期間は長いと判定する。

Ls の最終待機期間が短い場合、Ls のインデクスを常活性ライン・バッファに書き込む。ただし、常活性ライン数が Th_{always} を超える場合は、書き込みと同時に、最も古いライン (La と表記) のインデクスをクリアする。

Ls の最終待機期間が長い場合、あるいは、常活性ライン・バッファから La のインデクスがクリアされた場合、Ls、あるいは、La のインデクスを活性ライン・バッファに書き込む。ただし、活性ライン数が Th_{normal} を超える場合は、書き込みと同時に、最も古いライン (Ln と表記) のインデクスを読み出してクリアする。また、Ln のインデクスを用いてキャッシュを参照し、対応するラインを非活性化する。さらに、Ln のインデクスを非活性化履歴バッファに書き込む。ただし、非活性化履歴バッファに空きが無い場合は、書き込みと同時に、最も古いラインのインデクスをクリアする。

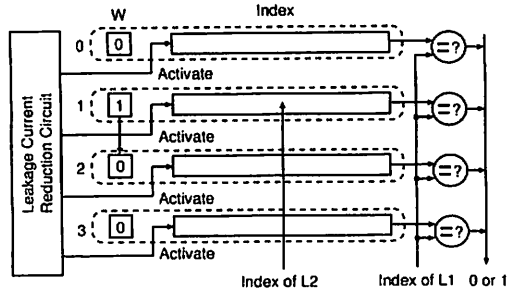

図 2 非活性化履歴バッファ

最後に、目標とする静的消費電力や性能低下率を達成する方法を示す。

リーク電流はラインの状態によって一意に決まる。そこで、Th_{normal} と Th_{always} の合計をある一定の値に保つことで、静的消費電力が目標値を超えないようにする。一方、性能低下率は、プログラムの実行時にしか分からない。そこで、プロセッサ性能を監視しながら、Th_{normal} や Th_{always} を動的に変化させ、性能低下率が目標値を超えないようにする。なお、後者を実現する具体的な機構の検討は今後の課題とするが、間所らの提案したスループット制御機構 [8] を応用すれば容易に実現できると考えられる。

3.2 非活性化履歴バッファ

非活性化履歴バッファは、最近、非活性化したラインのインデクスの履歴を保持する FIFO 形式のバッファである。更新の際は、ラウンドロビン方式で、非活性化したラインのインデクスを書き込んで行く。これにより、最も古いインデクスが最も新しいインデクスで上書きされる。つまり、FIFO 形式で更新が行われる。一方、参照の際は、活性化されたばかりのラインのインデクスを全エントリにブロードキャストし、各エントリ毎に読み出したインデクスと比較する。

非活性化履歴バッファの構成を図 2 に示す。リーク電流削減回路は、エントリを非活性化することにより、リーク電流を削減する。初期状態では、全エントリを待機状態にする。そして、キャッシュ上の待機状態にあるラインが活性化される度に全エントリを活性化し、参照／更新が終了すると、全ラインを再び非活性化する。ただし、参照／更新が連続する場合は、それらが終了した後で、全ラインの非活性化を行う。なお、全エントリを活性化するのは、1 回の参照につき、全てのエントリがアクセスされるからである。

図において、点線で囲まれた部分はエントリを示し、左側に付加した数字がエントリ番号を示す。各エントリは、W flag、非活性化したラインのインデックスを保持する。W flag が 1 であるエントリに、最も新しいインデクスを書き込む。図では、エントリ数を 4、1 番エントリの W flag を 1 とする。

図において、エントリの右側は最終待機期間を判定する機構であり、比較器で構成される。この機構は、最終待機期間が短

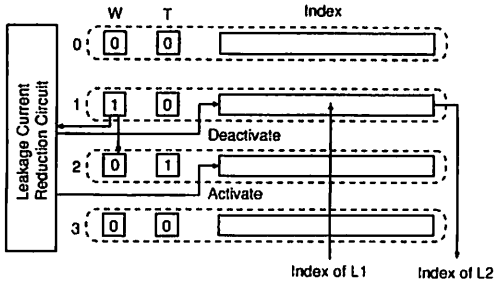

図 3　活性ライン・バッファ

表 1　測定条件

命令発行方式	イン・オーダ
フェッチ／発行幅	2 命令
機能ユニット数	iALU 2, iMULT/DIV 1, fpALU 2, fpMULT/DIV/SQRT 1
命令キャッシュ	32KB, 4 ウェイ, 32B ライン, ミスペナルティ 20 サイクル
データキャッシュ	32KB, 4 ウェイ, 32B ライン, 1 ポート, ミスペナルティ 20 サイクル
分岐予測機構	gshare 6 ビット履歴, 8K エントリ PHT, 予測ミスペナルティ 5 サイクル

かいと判定すれば 1 を、そうでなければ 0 を出力する。具体的には、比較器を用いて、各エントリのインデクスと、ブロードキャストされたインデクスを比較し、両者が一致するエントリが 1 つでも存在すれば 1 を、そうでなければ 0 を出力する。

図を用いて、参照／更新動作を説明する。

まず、参照動作を示す。ここで、待機状態にあるライン (L1) が活性化され、L1 のインデクスが全エントリにブロードキャストされたとする。このとき、エントリ毎に、自身の保持するインデクスと、L1 のインデクスを比較し、両者が一致すれば 1、そうでなければ 0 を出力する。そして、全比較結果の論理和をとった信号を出力する。

つぎに、更新動作を示す。ここで、活性ライン・バッファから、最も古いライン (L2) のインデクスがクリアされたとする。このとき、まず、L2 のインデクスを W flag が 1 である 1 番エントリに書き込む。そして、1 番エントリの W flag を 0 にリセットする。また、書き込み先のエントリがラウンドロビンに決まるように、別の W flag をセットする。この場合、書き込みの終了した 1 番エントリは一番最後ではないので、それに隣接する 2 番エントリの W flag を 1 にセットする。

3.3　常活性／活性ライン・バッファ

常活性ライン・バッファと活性ライン・バッファは、動作がほぼ同じであるため、どちらか一方だけ説明すれば十分である。そこで本節では、活性ライン・バッファのみ説明する。

活性ライン・バッファは、活性ラインのインデクスを保持する FIFO 形式のバッファである。活性ライン数が Th_{normal} を超えないようにするため、0 番目から $(Th_{normal}-1)$ 番目のエントリを用いる。更新の際は、ラウンドロビン方式で、活性化したラインのインデクスを書き込んで行く。これにより、最も古いインデクスが最も新しいインデクスで上書きされる。つまり、FIFO 形式で更新が行われる。ただし、最も古いエントリのデータは、ラインの非活性化のために必要なので、上書きされる前に読み出す。

活性ライン・バッファの構成を図 3 に示す。リーク電流削減回路は、エントリを非活性化することにより、リーク電流を削減する。初期状態では、全エントリを待機状態にする。そして、次回更新されるエントリをあらかじめ活性化し、更新の終了したエントリを非活性化する。更新するエントリはラウンドロビン方式に基づいて決まるため、活性化／非活性化するエントリは容易に特定できる。

図において、点線で囲まれた部分はエントリを示し、左側に付加した数字がエントリ番号を示す。各ラインは、W flag、T flag、活性化ラインのインデクスを保持する。W flag が 1 であるエントリから、最も古いインデクスを読みだし、そのエントリに、最も新しいインデクスを上書きする。T flag が 1 であるエントリは、エントリ番号が $(Th_{normal}-1)$ であることを示す。したがって、0 番エントリから、T flag が 1 であるエントリまでが、更新の対象となる。図では、エントリ数を 4、1 番エントリの W flag を 1、2 番エントリの T flag を 1 とする。

図を用いて、参照／更新動作を説明する。ここで、待機状態にあるライン (L1) が活性化され、その後、L1 の最終待機期間が長いと判定されたとする。このとき、まず、W flag が 1 である 1 番エントリから最も古いライン (L2) のインデクスを読み出し、そのエントリに L1 のインデクスを書き込む。そして、書き込みの終了した 1 番エントリにおいて W flag を 0 リセットし、エントリの非活性化を行う。また、書き込み先のエントリがラウンドロビンに決まるように、別の W flag をセットする。この場合、書き込みの終了した 1 番エントリの T flag は 1 ではないので、それに隣接する 2 番エントリにおいて W flag を 1 にセットし、エントリの非活性化を行う(注1)。

4.　評　　価

本章では、提案機構の有効性を検討するために、プロセッサ性能と、キャッシュの静的消費電力を測定する。

4.1　評価環境

シミュレータには、SimpleScalar Tool Set のスーパスカラ・プロセッサ用シミュレータを用い、提案手法を組み込んで評価した。測定条件として表 1 を用いた。命令セットには MIPS R10000 を拡張した SimpleScalar/PISA を用いた。ベンチマーク・プログラムは、SPECint2000 の bzip2、gcc、gzip、mcf、paser、perl、votex、vpr の 8 本を使用した。gcc では 10 億命令、その他では 20 億命令をスキップした後、1000 万命令を実行した。

以下のモデルに対して評価を行った。

(注1)：次回の更新では、2 番エントリの T flag は 1 なので、それに隣接する 3 番エントリではなく、0 番エントリにおいて W flag を 1 にセットし、エントリの非活性化を行うことになる。

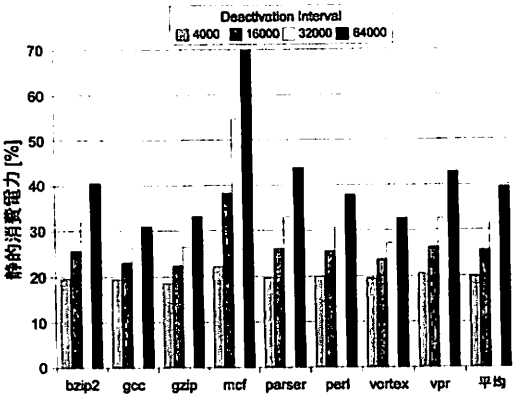

図 4 Simple モデルの静的消費電力

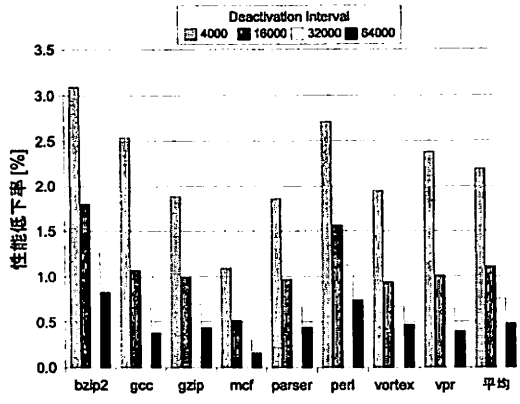

図 5 Simple モデルの性能低下率

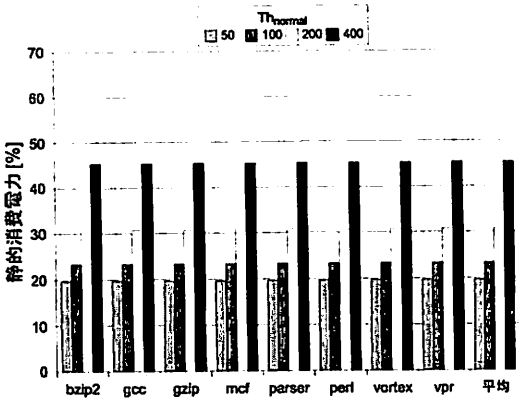

図 6 Normal モデルの静的消費電力

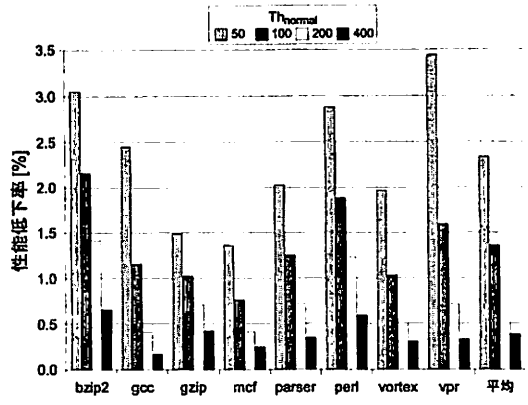

図 7 Normal モデルの性能低下率

- **Base モデル**: リーク電流の削減を行わないモデルである。
- **Simple モデル**: Simple 方式 [1] を用いて、一定サイクル毎に全ラインを非活性化するモデルである。
- **Normal モデル**: ラインを常活性状態に遷移させないようにした提案方式を用いるモデルである。Th_{always} は常に 0 となる。
- **Always モデル**: 提案方式をそのまま用いるモデルである。

前章で述べた通り、現時点において提案機構は、性能低下率を一定に保つための機構をまだ備えていない。そのため、本論文では、リーク電流の削減率を一定に保つ場合のみ評価し、性能低下率を一定に保つ場合については今後の課題とする。したがって、本論文の評価では、常活性ライン・バッファと活性ライン・バッファのエントリ数は、それぞれ、Th_{always}、Th_{normal} となる。

キャッシュの静的消費電力は、全 SRAM セルの静的消費エネルギーをプロセッサの実行時間で割ることにより得られる。ある SRAM セルの静的消費エネルギーは、活性状態で消費した静的エネルギーと待機状態で消費した静的エネルギーの和である。なお、常活性状態と活性状態は、活性期間が異なるだけなので、ここでは、常活性状態を活性状態として扱う。活性状態 (待機状態) で消費した静的エネルギーは、活性状態 (待機状態) にあったサイクル数と、活性状態 (待機状態) におけるサイクルあたりの静的エネルギーの積である。プロセッサ・シミュレータにより、各 SRAM セルが活性状態、あるいは、待機状態にあったサイクル数を求めた。文献 [1] より、SRAM セルのサイクルあたりの静的エネルギーを、活性状態において 1.63E-15J、待機状態において 2.59E-16J とした。

なお、より正確に消費電力を評価するためには、キャッシュの動的消費電力と、提案機構の静的／動的電力を知る必要があるが、それは今後の課題である。

4.2 評価結果

Simple モデルの評価結果を、図 4 と図 5 に示す。図 4 の縦軸は、Base モデルの場合で正規化した静的消費電力を示し、図 5 の縦軸は、Base モデルに対する性能低下率を示す。4 本で組になった棒グラフは、左から順に、全ラインを非活性化するインターバル (**非活性化インターバル**) が、4K, 16K, 32K, 64K の場合である。

図 4 と図 5 より、Simple モデルでは、静的消費電力や性能

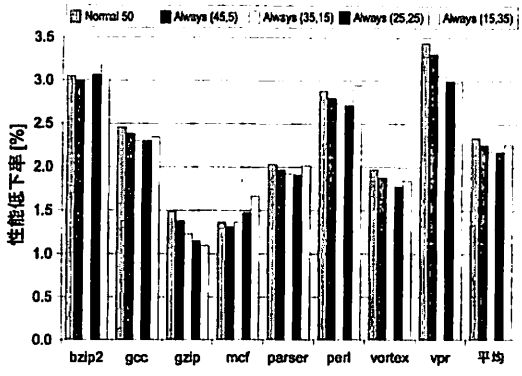

図 8 Always モデルの性能低下率 (履歴長 25)

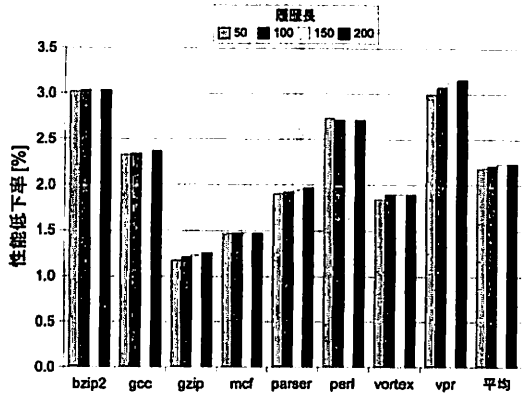

図 9 Always モデルの性能低下率 ($Th_{normal} = 25, Th_{always} = 25$)

低下率が、アプリケーションによって変化することが分かる。また、静的消費電力は、非活性化インターバルが長い程、大きく変化し、性能低下率は、非活性化インターバルが短い程、大きく変化することが分かる。

Normal モデルの評価結果を、図 6 と図 7 に示す。図 6 の縦軸は、Base モデルの場合で正規化した静的消費電力を示し、図 7 の縦軸は、Base モデルに対する性能低下率を示す。4 本で組になった棒グラフは、左から順に、Th_{normal} が 50, 100, 200, 400 の場合である。

図 6 から分かるように、Normal モデルでは、アプリケーションに関係なく、静的消費電力を一定に保つことができる。しかし、図 7 から分かるように、性能低下率は一定に保つことができない。これを解決するためには、前章で述べた通り、Th_{normal} を動的に変化させる機構を導入する必要があるが、それは今後の課題である。

Always モデルの評価結果を、図 8 と図 9 に示す。図の縦軸は、Base モデルに対する性能低下率を示す。

図 8 は、履歴長を 25 とし、Th_{normal} と Th_{always} の合計を 50 とした場合を示す。5 本で組になった棒グラフは、左から順に、Normal モデルにおいて Th_{normal} が 50 の場合、Always モデルにおいて、Th_{normal} と Th_{always} の組が、(45, 5), (35, 15), (25, 25), (15, 35) の場合である。

図より、性能低下率は、$Th_{normal} = 25$、$Th_{always} = 25$ の場合、最も抑制できることが分かる。しかし、抑制できる割合はそれほど大きくなく、最大でも 0.4%ポイントである。なお、いずれの場合も、Th_{normal} と Th_{always} の合計が等しいので、静的消費電力も等しくなる。したがって図 6 より、図 8 の構成での静的消費電力は、Base モデルの約 20%となることが分かる。

一方、図 9 は、$Th_{normal} = 25$、$Th_{always} = 25$ の場合を示す。4 本で組になった棒グラフは、左から順に、履歴長が 50, 100, 150, 200 の場合である。この図より、Always モデルの性能低下率は、履歴長にあまり依存しないことが分かる。これは、Th_{normal} や Th_{always} を変化させた場合も同様である。

これらの結果より、提案方式では、常活性化すべきラインの選択があまり適切に行われておらず、アルゴリズムの改善が必要であることが分かる。

5. ま と め

今回我々は、Drowsy Cache において、ラインの活性期間を制御する手法を提案した。提案機構では、活性状態にあるラインの数を一定に保つことができるため、目標とする静的電力を満たしつつ、より高い性能を達成することができる。しかし、電力効率を改善するアルゴリズムの検討など、まだ多くの課題が残されている。

文 献

[1] K. Flautner, et al., "Drowsy Caches: Simple Techniques for Reducing Leakage Power," *Proc. of the 29th Int. Symp. on Computer Architecture*, pp.148-157, May 2002.

[2] R. Fujioka, et al., "A Preactivating Mechanism for a VT-CMOS Cache using Address Prediction," *Proc. of the Int. Symp. on Low Power Electronics and Design*, pp.247-250, Aug. 2002.

[3] S. Kaxiras, et al., "Cache Decay: Exploiting Generational Behavior to Reduce Cache Leakage Power," *Proc. of the 28th Int. Symp. on Computer Architecture*, pp.240-251, June 2001.

[4] H. Kawaguchi, et al., "Dynamic Leakage Cut-off Scheme for Low-Voltage SRAM's," *Proc. of the Int. Symp. on VLSI Circuits Digest of Technical Papers*, pp.140-141, June 1998.

[5] N. S. Kim, et al., "Drowsy Instruction Caches / Leakage Power Reduction using Dynamic Voltage Scaling and Cache Sub-bank Prediction," *Proc. of the Int. Symp. on Microarchitecture*, pp.219-230, Nov. 2002.

[6] 小宮礼子ほか、待機ラインへの参照密度に基づく低リーク・キャッシュの動的制御、情報処理学会研究報告 2005-ARC-164, pp.73-78, 2005 年 8 月。

[7] L. Li, et al., "Soft Error and Energy Consumption Interactions: A Data Cache Perspective," *Proc. of the Int. Symp. on Low Power Electronics and Design*, pp.132-137, Aug. 2004.

[8] 間所峻洋ほか、パイプラインステージ統合のオンチップ制御機構、情報処理学会研究報告 2007-ARC-172, pp.37-42, 2007 年 3 月。

[9] 岡子純平ほか、Drowsy キャッシュにおけるモード切替アルゴリズムの評価、情報処理学会研究報告 2006-ARC-170, pp.37-41, 2006 年 11 月。