九州大学学術情報リポジトリ
Kyushu University Institutional Repository

待機状態ラインに対する参照局所性を考慮した低リーク・キャッシュの性能低下抑制方式

小宮, 礼子
福岡大学工学研究科電子情報工学専攻

井上, 弘士
九州大学大学院 システム情報科学研究院

モシニャガ, ワシリー
福岡大学工学部電子情報工学科

村上, 和彰
財団法人九州システム情報技術研究所 | 九州大学大学院 システム情報科学研究院

https://hdl.handle.net/2324/6202

出版情報：デザインガイア2004, 情報処理学会研究報告2004-ARC-160, pp.95-100, 2004-12. 情報処理学会ARC研究会
バージョン：
権利関係：ここに掲載した著作物の利用に関する注意 本著作物の著作権は（社）情報処理学会に帰属します。本著作物は著作権者である情報処理学会の許可のもとに掲載するものです。ご利用に当たっては「著作権法」ならびに「情報処理学会倫理綱領」に従うことをお願いいたします。

KYUSHU UNIVERSITY

待機状態ラインに対する参照局所性を考慮した低リーク・キャッシュの性能低下抑制方式

小宮礼子[†‡]　井上弘士[§]　モシニャガ・ワシリー[#]　村上和彰[‡§]

福岡大学　工学研究科　電子情報工学専攻[†]　財団法人九州システム情報技術研究所[‡]
九州大学大学院　システム情報科学研究院[§]　福岡大学　工学部　電子情報工学科[#]
arch-ccc-lpc@c.csce.kyushu-u.ac.jp

概要

これまでに多くの低リーク・キャッシュが提案されてきた．しかしながら，これらの手法を用いると待機状態ラインへの低速なアクセスが発生するため，必然的に性能が低下する．そこで本稿では低リーク･キャッシュにおける性能低下抑制方式として，常時活性化(always-awake)ライン方式を提案する．具体的には，性能低下の原因となる待機状態ライン・アクセスの局所性を考慮し，アクセスが集中するラインは常時活性化状態にする．これまでに提案された Drowsy 方式では，15%程度の性能低下をもたらす事で 84%のリーク削減率を達成した．これに対し，本稿で提案する always-awake ラインを用いた場合，同程度のリーク削減率を維持しつつ，性能低下を 8～11%に抑制することができた．

A Cache Management Technique via Sleep-Hit Locality to Alleviate Performance Impact of Low-Leakage Caches

Reiko Komiya[†‡]　Koji Inoue[§]　Vasily G. Moshnyaga[#]　Kazuaki Murakami[‡§]

Department of electronics engineering and computer Science, Fukuoka University[†#]
Institute of Systems & Information Technologies/KYUSHU[‡]
Department of Informatics, Kyushu University[§]
arch-ccc-lpc@c.csce.kyushu-u.ac.jp

Abstract

A number of techniques to reduce cache leakage energy have so far been proposed. However, in these techniques, low speed accesses to a standby mode line degrade processor performance. We have analyzed the detail of cache-access behavior, and have found that there is a locality of accesses to the standby-mode lines. Based on this observation, we propose a cache management technique to alleviate the negative effect of low-leakage caches. In our approach, cache lines having high degree of sleep-hit locality are forced to stay in the high-speed but high-leakage mode. In our evaluation, it has bee observed that the Drowsy cache can achieve 84% of leakage reduction with 15% of performance degradation, while the proposed scheme worsens the performance by only 8～11% with the same degree of energy reduction of the Drowsy approach.

1. はじめに

携帯電話やノート型 PC といったバッテリー駆動型機器の普及に伴い，マイクロプロセッサ・システムの低消費エネルギー化が重要視されるようになった．一般に，CMOS 回路の消費エネルギーは，動的消費エネルギーと静的消費エネルギーの 2 つに大別される．前者は回路負荷容量の充放電によって，また，後者はトランジスタの漏れ電流によって消費されるエネルギー(リーク消費エネルギー)である．従来の CMOS 回路では動的消費エネルギーが多くの割合を占めていた．しかしながら，微細化加工技術の進歩に伴い，リーク消費エネルギーによる影響が大きくなってきた．例

えば, Pentium4 プロセッサでは全消費電力の 20%がリークに起因しており, 更なるプロセス技術の進歩に伴いこの割合はより大きくなると予想される[2]．特に，大量のトランジスタで構成されるキャッシュ･メモリにおいては，リーク消費エネルギーの削減が極めて重要となる．例えば，0.07μm プロセスを想定した場合，キャッシュ全消費エネルギーの 70%はリークに起因するとの予測もある[2].

この問題を解決するため，これまでに様々なキャッシュ･リーク消費エネルギー削減手法が提案された[1]．これらの手法は，以降参照されないと予測されたキャッシュ･ラインを動的に待機状態へ切替えることによってリークを削減する. しかしながら，待機状態ラインへの参照はアクセス･ペナルティを引き起こし，その結果，プロセッサの性能が低下する．今後，プロセッサ動作速度の更なる向上に伴い, 待機状態ラインに対するアクセス・ペナルティは増加すると予測される．そのため, 高い性能と低消費エネルギーを両立させるためには，低リーク・キャッシュにおける性能低下の抑制が重要となる．

そこで本稿では，低リークキャッシュにおける性能低下の抑制を目的として，待機状態ライン・アクセスの局所性を活用したキャッシュ制御方式を提案する．また,ベンチマーク･プログラムを用いたシミュレーションを行い, 提案手法の有効性を評価する．本手法では，待機状態時に集中して参照されるラインを常に活性状態で動作させる．これにより，待機状態ラインへのアクセスを回避し，性能低下を抑制する．本方式を適用した結果, 従来の低リークキャッシュと同程度のリーク削減率を維持しつつ，性能低下を 4～12%抑制できた.

以下, 第 2 節では従来のリーク削減手法を簡単に説明する．次に，第 3 節で性能低下を抑制するキャッシュ制御方式を提案し, 第 4 節でベンチマーク・プログラムを用いた定量的評価を行う．最後に第 5 節で簡単にまとめる.

2. 従来キャッシュ・リーク削減手法

本節では，これまでに提案された代表的なリーク削減手法である Drowsy キャッシュ[1]について説明し，その問題点を述べる.

2.1 Drowsy キャッシュ

Drowsy キャッシュは待機状態(低速かつ低リーク)と活性状態(高速かつ高リーク)をライン単位で切替えることでリーク消費エネルギーを削減する．本稿では待機状態で動作するラインを **sleep ライン**，活性状態のラインを **awake ライン**と呼ぶ. リーク削減手法を用いない従来型キャッシュ･メモリの場合，全ラインが awake ラインとして動作する.

sleep ラインは，SRAM セルに記憶されたデータを失わない程度まで電源電圧を下げることで実現される. 電源電圧の供給を完全に停止する場合と比較して, ライン当たりのリーク消費エネルギーは大きくなるものの, 従来型キャッシュと同じヒット率を維持できる．ただし，awake ラインと比較して，sleep ラインへのアクセス時間は長くなる．これは，sleep ラインに供給されている電源電圧を回復し，活性状態(つまり awake ライン)へと移行させる必要があるためである.

sleep ライン数が多い場合，高いリーク削減効果を得られる一方, 極端に性能が低下する．つまり, ラインの状態がリーク削減効果と実行時間へ多大な影響を与える．したがって，キャッシュにおける sleep-awake 間の動作状態切替え制御が非常に重要となる．Drowsy キャッシュでは，定期的に全てのキャッシュ・ラインを sleep ラインへと変更し, アクセスが発生したラインに対してのみ awake ラインに変更する.

2.2 問題点

第 2.1 節で述べたように sleep ラインへのアクセスは性能低下を引き起こす．以降，キャッシュにヒットした sleep ラインへのアクセスを **sleep ヒット**, その際に生じるアクセス時間オーバヘッド(サイクル数)を **SHP**(Sleep Hit Penalty)と呼ぶ. SHP は実際には SRAM セルの回路構成やプロセス技術に依存するが，多くの研究報告では SHP を 1 クロックサイクルと仮定している. しかしながら，第 1 節で述べたように，プロセッサ動作速度の更なる向上に伴い, SHP は増加すると予測される．その結果，リーク削減に伴う性能低下はより顕著に表れるようになる．高いキャッシュ・リーク削減率を保ち，かつ，性能低下を最小限に抑えるためには, アクセスが発生する直前に当該ラインのみ awake ラインへと変更する必要がある. Drowsy キャッシュでは，sleep ヒットが発生すると awake ラインへと移行するため，必ず SHP による性能低下が生じる. SHP が 3 クロックサイクルの場合, 従来のキャッシュ・メモリと比較して最大 22.7%，平均 8.5%性能が低下する[3]．この性能低下を改善するためには，sleep ヒット回数を削減しなければならない.

3. 性能低下を抑制するキャッシュ制御方式

第 2.2 節で述べたように, sleep ヒット回数の削

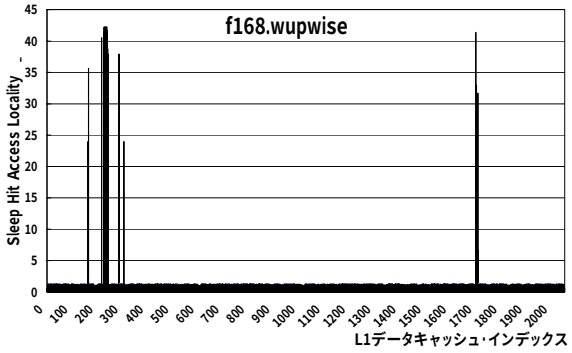

図 1 : SHL_i (高局所性)

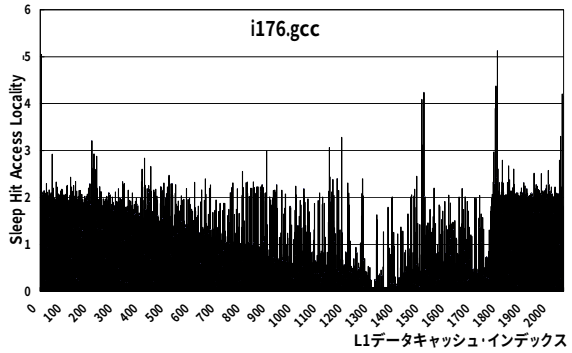

図 2 : SHL_i (低局所性)

減は低リーク･キャッシュの性能低下を抑制する. 本節では, 性能低下抑制キャッシュの制御方式について詳細を述べる.

3.1 sleep ヒット・ローカリティ

一般に, メモリ参照には局所性が存在するため, sleep ヒットに関しても何らかの局所性が存在すると予測される. そこで, プログラム実行における sleep ヒットの局所性を調査した. ここで, キャッシュ・ライン *i* における sleep ヒットの局所性 SHL_i(*Sleep Hit Locality*)を以下の式で表す.

$$SHL_i = \frac{N_{line-i}}{N_{avg}} \qquad (1)$$

N_{line-i} はライン *i* における sleep ヒット回数, N_{avg} は全ラインに関する sleep ヒット回数の平均である. 例えば, SHL_2 が 1 以上の場合, ライン 2 は平均以上の sleep ヒットが発生していることになる.

128KB の L1 データキャッシュを想定し, 各ラインにおける SHL_i を測定した. 2 つのプログラムに関する結果を図 1(f168.wupwise) と図 2(i176.gcc)に示す. f168.wupwise の場合, sleep ヒットが一部のラインに集中している. したがって, 非常に局所性が高いと言える. 逆に, i176.gcc では全ラインが平均的に参照されている. このことから, 実行するベンチマークごとに sleep ヒット

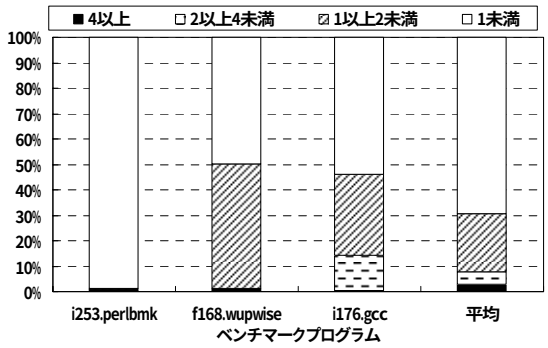

図 3: SHL_i に関するライン数の内訳

の局所性は異なることが分かる. 図 3 はキャッシュ中で SHL_i の値が 1 未満, 1 以上 2 未満, 2 以上 4 未満, または, 4 以上であったラインの占める割合を示している. SHL_i の値が 1 以上であるラインが全ラインに占める割合は平均 30%にも達する. したがって, 一般的には sleep ヒットには局所性があると言える.

3.2 性能低下抑制方式

第 2.2 節で述べたように, 性能低下を抑制するにはアクセス対象ラインを事前に awake ラインへと変更しておく必要がある. その手段として, 集中的に sleep ヒットが発生するラインを常に awake ラインとして動作させる制御方式を提案する.

sleep ラインへの参照局所性の判定には SHL_i の値を用いる. SHL_i がある閾値以上ならばそのラインは性能低下を引き起こしていると判断し, 常に awake ラインとして動作させる. 今後, このラインを **always-awake ライン**と呼ぶ. SHL_i が閾値以下の場合, 当該ラインは性能低下に影響を与えていないと推測されるため通常の低リーク･キャッシュと同様に動作する.

always-awake ライン数が過剰になると, 性能低下は改善されるがリーク消費エネルギーの削減効果が得られなくなる. 一方, always-awake ライン数が極端に少ない場合は従来の低リーク･キャッシュと比較して性能が改善されない.

4. 評価

本節では, 性能低下抑制方式を適用した低リーク・キャッシュの評価を行う. 具体的には性能オーバヘッドおよびリーク消費エネルギー削減率を測定し, 従来の低リーク・キャッシュである Drowsy キャッシュと比較することで本方式の有効性を議論する.

4.1 リーク消費エネルギー・モデル

キャッシュのリーク消費エネルギー(*LEtotal*)は, 以下に示すように, プログラム実行時間(*CC*), クロックサイクル当りの 1 ライン平均リーク消費エネルギー(*LEline*), ならびに, キャッシュ全体のライン数(*Nline*)によって近似できる.

$$LEtotal = CC * LEline * Nline \quad (2)$$

$$CC = CCconv + CCextra \quad (3)$$

$$LEline = SR * LEsline + (1 - SR) * LEaline \quad (4)$$

ここで, *CCconv* はリーク削減手法を用いない場合のプログラム実行時間(クロックサイクル数)であり, *CCextra* はリーク削減手法の採用に伴う実行時間オーバヘッドを表す. また, *SR* は全キャッシュ容量における sleep ライン数の割合である. リーク削減手法を用いないキャッシュの場合は *CCextra* ならびに *SR* が 0 となる. *LEsline* ならびに *LEaline* は, それぞれ, 各 sleep ラインならびに awake ラインにおいて 1 クロックサイクルで消費される平均リーク・エネルギーである. 本稿では 0.07μm プロセスを想定した文献[1]の測定結果を参考にし, *LEaline* と *LEsline* の比が 100 対 8 と仮定した. なお, 本評価では, キャッシュ・アクセスやモード変更に伴う動的消費エネルギーは考慮しない.

4.2 実験環境

トレースドリブン・シミュレーションを実施し, 前節で定義した *CCconv*, *CCextra*, *SR* を測定した. 具体的には, SPARC64 プロセッサ用シミュレータである ZonC[4]を利用して各ベンチマークの実行安定期における約 10,000,000 命令の実行を抽出し, 作成したキャッシュ・シミュレータへ入力した. なお ZonC では高精度なプロセッサならびにメモリモデルを実現しており, 実論理設計レベル・シミュレーションとの誤差は 5%以下である. ベンチマーク・プログラムとしては, SPEC2000 ベンチマークより 12 個の整数プログラムと 14 個の浮動小数点プログラムを用いた. 本評価では, 128KB の L1 データキャッシュ(連想度 2)を対象とする. always-awake ラインに関しては, 事前にプロファイル情報を採取して決定した.

以降, リーク削減手法を用いない通常の高リークキャッシュを基準として, リーク消費エネルギー削減率と実行時間増加率を用いて評価する. 評価対象モデルは以下のように表す.

- Drowsy：Drowsy キャッシュ
- AA_x：SHL_i の値が x 以上であるラインは always-awake ラインとして動作

なお, SHP は 5 クロックサイクルとして実験を行った. また, 各 always-awake のラインは, プログラムの実行開始時から常に活性状態であるとする.

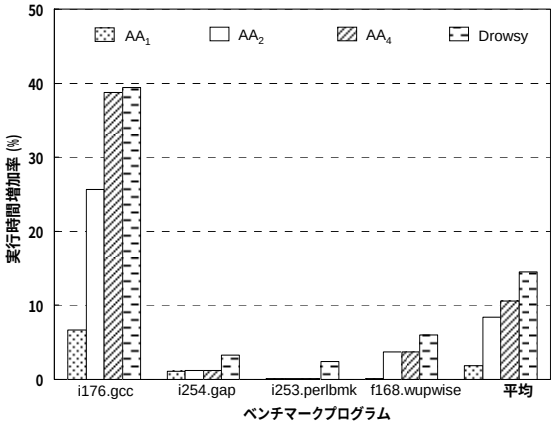

図 4: 実行時間増加率

4.3 実験結果

本節では, 従来の低リークキャッシュである Drowsy キャッシュと, 性能低下抑制方式を適用したキャッシュの比較を行う. なお, 紙面の都合上, 代表的であった 4 つのベンチマーク(i176.gcc, i254.gap, i253.perlbmk, f168.wupwise)ならびに全ベンチマークの平均結果を用いて議論する.

4.3.1 実行時間増加率

本提案方式が性能低下抑制に与える影響について評価する. 評価モデルは AA_1, AA_2, AA_4 と Drowsy であり, それぞれの実行時間増加率を図 4 に示す. この図から, 全評価モデルにおいて, Drowsy キャッシュよりも性能低下が改善されていることが分かる. i176.gcc の場合, 閾値が小さくなるにつれて性能低下が改善される. これは第 3.1 節の図 3 で示したように, SHL_i の値が 1 または 2 以上であるラインの数が多いためである. i253.perlbmk では, 提案方式の適用により, リーク削減に伴う性能オーバヘッドををほぼ完全に隠蔽している. SHL_i の値が閾値以上であるライン数は非常に少ないにもかかわらず高い性能改善が得られていることから, Drowsy キャッシュにおいて性能低下を引き起こしていたのは当該ラインであったといえる. また, f168.wupwise では図 3 から分かるように SHL_i の値が 2 以上のラインが極端に少ない. したがって, AA_1 では非常に高い性能低下抑制効果が得られるが, それ以外の閾値を用いると極端な性能改善は見込めない.

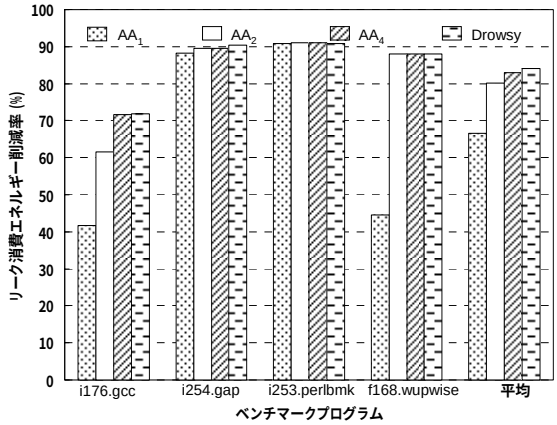

図 5: リーク消費エネルギー削減率

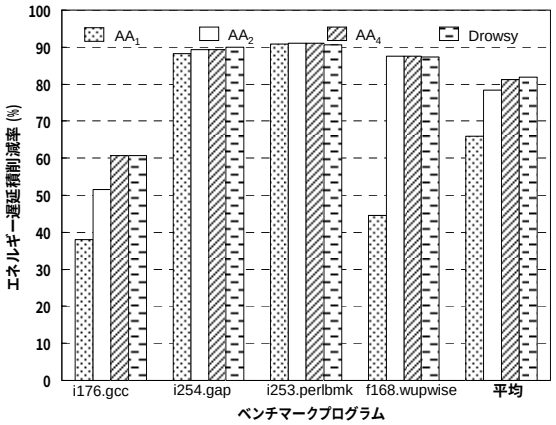

図 6: エネルギー遅延積削減率

つまり, 本提案手法は閾値を適切に設定することにより，高い性能改善が期待できると言える.

4.3.2 リーク消費エネルギー削減率

次にリーク消費エネルギーへの影響について議論する.各キャッシュのリーク消費エネルギー削減率を図 5 に示す．i176.gcc では閾値に比例してリーク削減効果が低下する. これは前節で述べたように閾値が小さくなるにつれて always-awake ライン数が増加するためである. 次に i253.perlbmk の場合, 全モデルにおいて Drowsy キャッシュと同等のリーク消費エネルギー削減率を得られることが分かる．また,第 4.3.1 節で示したように, 本ベンチマークの実行時間増加率は 0.1%と極めて小さい．つまり，これは従来型キャッシュの高い性能と，Drowsy 方式による高いリーク削減率の両方を同時に満足していることを意味する．f168.wupwise では，AA_1 のリーク削減効果のみ極端に低下している．したがって，性能低下が改善されるとリーク削減効果は低下するといえる. 全ベンチマークの平均においては, 閾値に比例してリーク消費エネルギー削減率は低下する．しかしながら, 前節で述べたように性能は改善する. したがって性能へ高い要求がある場合, 閾値を 1 として性能低下抑制方式を用いれば良い. 逆に, 高いリーク削減率が求められる場合には閾値を大きくする，もしくは，Drowsy キャッシュを採用することが適切である.

4.3.3 ED 積

エネルギー遅延積に基づいた評価結果を図 6 に示す. エネルギー遅延積が閾値の上昇に伴い向上していることが分かる．i253.perlbmk ベンチマークにおいて AA_4 では Drowsy キャッシュよりも僅かに削減率が高くなっている．しかしながら, 閾値が小さい場合, 性能低下抑制方式を適用するとエネルギー遅延積は悪化する. これは小さい閾値で性能低下抑制方式を用いると性能低下抑制効果よりもリーク消費エネルギーの増加が著しくなるためであると考えられる. 本評価においてエネルギーとはリーク消費エネルギーのみを意味している．したがって, 実行時間の低下に対してエネルギーの増加が極端に大きい場合, エネルギー遅延積削減率も大幅に低下する.

4.3.4 SHP の影響

ここまで, SHP は 5 クロックサイクルと仮定してきた．本節では SHP を 1 から 5 クロックサイクルまで変化させた場合の実行時間とリーク消費エネルギーを評価する．SHL_i の閾値は 1 に固定する．図 7 と図 8 は，それぞれ，実行時間増加率およびリーク消費エネルギー削減率を表す. 横軸の各ベンチマーク上に記されている 1〜5 の数字が SHP を表している.

SHP クロックサイクル数が増加するにつれ, 提案方式の性能低下抑制効果が高くなることが図 7 より明らかである．一方, 図 8 より SHP のクロックサイクル数に関わらずリーク消費エネルギー削減率は一定であることが分かる．SHP が 5 クロックサイクルの場合, 性能低下抑制方式を適用すると Drowsy キャッシュに比べて性能が 12.6%改善されている(全ベンチマークの平均). したがって，本提案方式は SHP のクロックサイクル数が長くなるにつれ,リーク消費エネルギーに影響を与えることなく性能低下を改善できることが分かった.

5. おわりに

本稿では, 参照局所性が高いラインに対して常に awake モードで動作させることによって性能

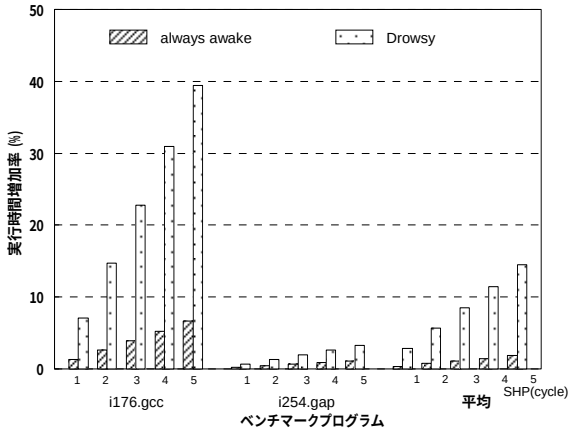

図 7: 実行時間増加率

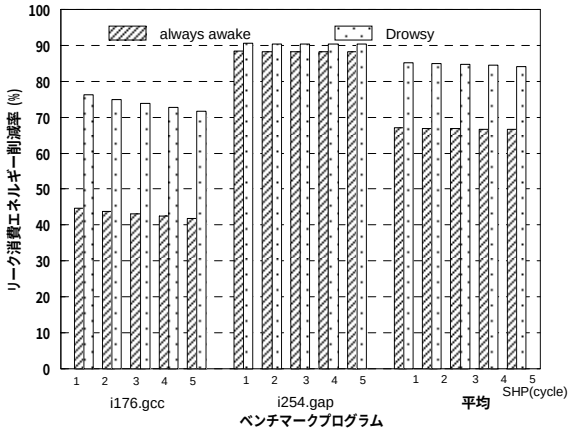

図 8: リーク消費エネルギー削減率

低下を抑制する方式を提案し, その評価を行った. その結果, 常に awake モードで動作させるライン (always-awake ライン)数に比例して性能低下は改善されるがリーク消費エネルギーが増加することがわかった. 本手法が最も有効に動作した場合, Drowsy キャッシュと同等のリーク削減率 90.8% を実現し, 性能低下は 0.1%に抑えることができた. また, SHP の増加に伴い本提案方式の有効性が増すことが分かった.

本稿では, エネルギーはリーク消費エネルギーのみに着目している. キャッシュ・メモリで消費されるエネルギーには動的消費エネルギーもある. それらも含めたキャッシュ全体のエネルギー評価が今後の課題である. また, 本評価ではライン状態の切替え等によって発生する動的消費エネルギー・オーバヘッドも含まれていない. 今後, これらを含んだより詳細な評価を行う予定である.

謝辞

本研究を進めるにあたり, 多くのご指導を頂いた富士通株式会社の池田正幸氏, 丸山拓巳氏, 富士通研究所の勝野昭氏, 坂本真理子氏に深く感謝致します. なお, 本研究は一部, 文部省科学研究費補助金 (課題番号：14GS0218, 14702064, 14102027）による.

参考文献

[1] K.Flautner, N.S.Kim, S.Martin, D.Blaauw, and T.Mudge, “Drowsy Caches: Simple Techniques for Reducing Leakage Power,” *Proc. of the 29th Int. Symp. on Computer Architecture*, pp.148-157, May 2002.

[2] N.S.Kim, K.Flautner, D.Blaauw, and T.Mudge, “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.

[3] 小宮礼子, 井上弘士, モシニャガ ワシリー, 村上和彰 “キャッシュ・リーク電力削減アルゴリズムに関する定量的評価”, 第 17 回 回路とシステム軽井沢ワークショップ,pp.235-240, 2004 年 4 月.

[4] M.Sakamoto, A.Katsuno, A.Inoue, T.Asakawa, H.Ueno, K.Morita, Y.Kimura, “Microarchitecture and Performance Analysis of a SPARC-V9 Microprocessor for Enterprise Server Systems,” *Proc. of the 9th Int. Symp. on High-Performance Computer Architecture*, pp.141-152, Feb.2003.