C-027

マルチ・スレッド実行を前提としたキャッシュ・リーク削減アルゴリズムの評価
A Quantitative Evaluation of Cache Leakage Reduction Algorithms on Multi-Thread Processors

小宮 礼子†1†2 Reiko Komiya
井上 弘士†3 Koji Inoue
モシニャガ・ワシリー†3 Vasily G. Moshnyaga
村上 和彰†1†4 Kazuaki Murakami

1. はじめに

近年，プロセッサの低消費電力化を阻害する 1 つの要因として，キャッシュ・メモリにおけるリーク消費電力の増大が挙げられる．リーク消費電力は実装されたトランジスタ数に比例するため，大容量化されたオンチップ・キャッシュでは，そのリーク消費電力削減が極めて重要となる．これまでに我々は，提案された様々なキャッシュ・リーク削減アルゴリズムに関する分類・比較・評価を行い，文献[1]で提案された方式が有効である事を示した[3].

しかしながら，これまでの評価では単一スレッド実行を想定しており，今後主流になるであろうマルチ・スレッド実行での有効性は明らかにしていない．異なる複数のスレッドを同時に実行した場合，単一スレッド実行時と比較して，キャッシュ・メモリへのアクセス・パタンは大きく異なる．そのため，メモリ参照の局所性を活用するリーク削減アルゴリズムでは，単一スレッド実行時ほど高いリーク削減率を達成できない可能性がある．そこで本稿では，文献[1]で提案された方式を対象とし，マルチ・スレッド実行におけるキャッシュ・リーク削減率ならびに性能オーバヘッドを評価する．また，同時実行されるスレッドの種類（スレッドの組合せ）が与える影響を調査する．

2. 対象とするリーク削減アルゴリズム

リーク削減手法は動的に低速・低リークな **sleep モード**と高速・高リークな **awake モード**を切替えることで実現される．効果的にリークを削減するには“いつ，どのようにして，どの範囲のモードを切替えるか”が重要となる．本稿では，リーク削減手法をマルチ・スレッド環境下で適用した場合に最大どれだけのリーク消費エネルギーを削減できるか明らかにするため，以下のような条件でモードの切替えを行う．

まず，いつモードを切替えるか？であるが，これは前節で述べたように，単一スレッド実行時に最も有効であったアルゴリズム，つまり，一定サイクル経過後に sleep モードにし，アクセスが発生した時点で awake モードに切替える方法を用いる．一般にメモリ参照には時間的・空間的局所性が存在する．この手法ではアクセスが発生した時点で awake モードへ切替わるため，その後局所的に発生するアクセスは高速に行われ，性能の低下は抑えられる．また，定期的に sleep モードへ切替えるので制御機構も簡単である．

本稿ではこのアルゴリズムをライン単位で制御する．つまり，各ラインにカウンタを設け，閾値を 4K サイクルとし，その値を超えたら sleep モードに切替える．また，sleep ラインにアクセスが発生したら直ちに awake モードに切替える．

最後にどのようにしてモードを切替えるか？であるが，本稿では sleep モードに切替えた場合に内部状態を破壊しない方式(低電源電圧化[1]や閾電圧の変更)を想定する．つまり，sleep モード時でもキャッシュに格納されたデータは保持される．そのため，リーク消費電力削減手法を適用していない従来型キャッシュでのミス数を維持できる．

†1（財）九州システム情報技術研究所，ISIT
†2 福岡大学大学院工学研究科電子情報工学専攻
†3 福岡大学工学部電子情報工学科
†4 九州大学大学院システム情報科学研究院

3.定量的評価

本節では，第 2 節で説明した手法に関して，マルチ・スレッド実行環境下でのリーク削減率ならびに性能オーバヘッドを評価する．

3.1 リーク消費エネルギー・モデル

キャッシュ・メモリのリーク消費エネルギー(*LEtotal*)は，プログラム実行サイクル数(*CC*)，サイクル当りの 1 ビット SRAM セル平均リーク消費エネルギー(*LEbit*)，ならびに，キャッシュ・サイズ(*CSize*)によって近似できる(式(1)).

$$LEtotal = CC * LEbit * CSize \quad (1)$$
$$CC = CCconv + CCextra \quad (2)$$
$$LEbit = SR * LEsbit + (1 - SR) * LEabit \quad (3)$$

ここで，*CCconv* はリーク削減手法等を用いない場合のプログラム実行サイクル数であり，*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* はキャッシュ・サイズに対する sleep モード SRAM セル数の割合である．さらに，*LEsbit* ならびに *LEabit* は，それぞれ，sleep モード/awake モード時における 1 ビット SRAM セルの平均リーク消費エネルギーである．例えば，リーク削減手法を用いない従来キャッシュの場合は *CCextra* ならびに *SR* が 0 となる．

3.2 実験環境

SPEC2000 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，第 3.1 節で示した式(2)ならびに(3)における *CCconv*, *CCextra*, *SR* を測定した．本評価では SPARC64 プロセッサを前提としており，L1 データ・キャッシュの構成は 128KB サイズの連想度 2 とした．また，最大 4 個の完全に独立したスレッドを実行可能であり，L2 キャッシュ・ミスが発生した際に実行スレッドの切替えを行う場合を想定している（つまり，時間的にスレッドの切替えを行う方式であり，SMT のように同一パイプライン・ステージで複数スレッドが混在する事はない）．実行するスレッドの組合せに関しては，単一スレッド実行時，リーク消費エネルギー削減率ならびに実行時間増加率がそれぞれ最大，最小，中間であったベンチマークを選択した．表 1 に実行スレッ

表 1：入力ベンチマーク・プログラム

	Max	Mid	Min
LE	i253.perlbmp i254.gap f200.sixtrack i256.bzip2	f188.ammp i186.crafty i255.vortex i197.parser	f179.art i176.gcc i181.mcf f178.galgel
ET	i176.gcc i164.gzip f179.art f173.applu	f191.fma3d i255.vortex i252.eon i175.vpr	i253.perlbmp i254.gap f168.wupwise f200.sixtrack

ドの組合せを示す. 表中の LE はリーク消費エネルギー削減率，ET は実行時間増加率を示しており，それぞれ Max(最大)，Min(最小)，Mid(中間)の組合せがある．実際には，SPARC64 プロセッサ用シミュレータである ZonC[4]を使用し，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て各スレッドを構成した．

一方，*LEabit* と *LEsbit*(awale モード/sleep モード時における SRAM セル当たりの平均リーク消費エネルギー)の比は，0.07μm プロセスを想定した文献[1]の測定結果を参考にして 100 対 8 と仮定する．なお，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ていない．

3.3 実験結果

データ・キャッシュのリーク消費エネルギー削減率(棒グラフ)ならびに実行時間増加率(折れ線グラフ)を図 1 に示す. 横軸は入力ベンチマーク・パターンを表す．図中の SHP(Sleep Hit Penalty)は，sleep モードのキャッシュ・ラインに対するアクセス時間オーバヘッド(サイクル数)である．

まず，SHP がリーク削減率に与える影響を評価する．全ての入力パターンにおいて，SHP の増加に伴いリーク削減率が低下している．これは，図 1 の折れ線グラフで示すように，プログラム実行時間の増加が主な原因である．つまり，第 3.1 節で示した式(2)の *CCextra* が増大し，その結果，全リーク消費エネルギー(*LEtotal*)が増加したものである．

次に，入力ベンチマークの違いがリーク削減効果に与える影響を考察する. 図 1 より入力パターン ET-Min が最もリーク消費エネルギーを削減し，性能低下もわずかであることがわかる．SHP が 1 サイクルの場合，リーク削減率は 88.2%，実行時間増加率は 1.84%である．2 番目に高いリーク削減効果を達成している LE-Max の実行時間増加率は 2.08%である．これらの入力ベンチマークの違いは 1 組（256.bzip2 と 168.wupwise）だけである．したがって，マルチ・スレッド環境下でリーク消費エネルギー削減に向いているプログラムは，シングル・スレッド実行時に性能への影響が低いプログラムと言える．また，全入力パターンを比較してみると実行時間増加率が低いものほどリーク消費エネルギー削減率が高くなっている．これも先ほど述べように，式(2)の *CCextra* の低減により *LEtotal* が減少したためである．

最後に，シングル・スレッド実行時とマルチ・スレッド実行時の比較を行う． SHP=1 サイクルの場合，シングル・スレッド実行時のリーク消費エネルギー削減率(SPEC2000 全ベンチマーク・プログラムの平均)は 85.2%，実行時間増加率は 2.80%であった[3]．一方，マルチ・スレッド実行時の 6 入力パターンの平均を取ると，リーク消費エネルギー

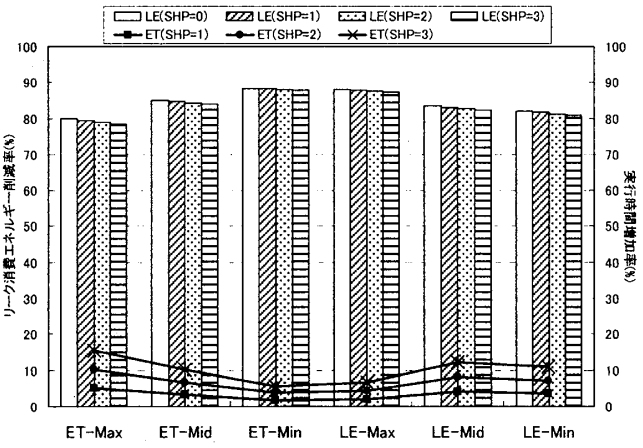

図 1：実験結果

削減率は 84.1%，実行時間増加率は 3.30%である．このことから，マルチ・スレッド環境下でも大幅なリーク削減効果が期待できることが分かる．

4 おわりに

本稿ではデータ・キャッシュのリーク消費エネルギー削減手法をマルチ・スレッド実行環境下で評価した．その結果，実行時間増加率がリーク消費電力削減効果に影響を与えることが分かった．また，シングル・スレッド実行時に実行時間増加率が低いプログラムはマルチ・スレッド環境下で高いリーク削減効果を期待できることが分かった．リーク削減に影響を及ぼす実行時間増加をいかに抑えるかが今後の課題である．

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, 14580399）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