修士論文

題目

マルチプロセッサ環境での
可変レベルキャッシュの
モード切換手法に関する研究

指導教員

近藤 利夫 教授

平成23年度

三重大学大学院 工学研究科 情報工学専攻
計算機アーキテクチャ研究室

城田 幸利(410M513)

内容梗概

現在，ノートパソコンやPDA，携帯電話などのモバイル端末の高性能化にともない消費エネルギーが増大し，バッテリによる駆動時間が短くなるという問題が発生している．そこで，モバイル端末の性能を落とすことなく低消費エネルギーを実現することが要求されている．現在，高性能かつ低消費エネルギーを実現する様々な手法が提案されており，その手法の一つとして可変レベルキャッシュが提案されている．

可変レベルキャッシュは，アプリケーションが多くのキャッシュ容量を必要とする場合はキャッシュ容量を通常の容量で使用する通常モードとして動作し，必要とするキャッシュ容量が少ない場合にはキャッシュ容量の半分をスリープモードにする低消費エネルギーモードとして動作する．低消費エネルギーモードにおいて，スリープモードに移行した部分は擬似的に1つ下位レベルの排他的キャッシュとして利用される．可変レベルキャッシュは通常モードと低消費エネルギーモードを動的に切換ることによって高性能と低消費エネルギーの両立を実現する手法である．

これまでに，可変レベルキャッシュはシングルプロセッサ環境において性能を低下させることなく消費エネルギーを削減できることが明らかにされている．しかしながら，近年のプロセッサで広く用いられているマルチプロセッサ環境での評価は行われていない．そこで，従来の可変レベルキャッシュのモード切換アルゴリズムをマルチプロセッサ上に単純に実装した．その結果，多くのベンチマークで性能の悪化が見られた．これは，従来の手法がキャッシュ全体の動作からモードを切換えており，キャッシュを共有しているプロセッサごとの必要容量を考慮していないためである．そこで，シングルプロセッサ時と同様にキャッシュ全体の動作からモード切換を行うのではなく，マルチプロセッサの各プロセッサで動作するプログラムそれぞれの動作からキャッシュの必要容量を判断し，モード切換を行う手法へ改良することによりマルチプロセッサでのリークエネルギー削減する手法を提案する．本論文で行ったシミュレーション結果によるとエネルギー遅延積(ED積)で比較して，従来の可変レベルキャッシュと比べて11%性能向上することが明らかとなった．

三重大学大学院　工学研究科

Abstract

Increasing power consumption has been becoming a major concern not only for mobile computing but also high-performance computing, and processors are required to achieve both low-energy and high-performance at the same time. Especially, it is important to reduce leakage energy consumed in a cache memory because power dissipation by leakage energy current is dominant factor in deep submicron technologies and the cache memory dissipates a large amount leakage energy. To achieve high-performance and low-energy simultaneously, a Variable Level Cache (VLC) is proposed. The VLC dynamically varies the cache capacity according to required cache capacity by running program. If the VLC detects that current running program does not need large capacity cache memory, half of the cache memory is put into standby mode, and is virtually treated as a lower level exclusive cache. The VLC succeeded reducing leakage energy on a single processor environment without performance degradation. However, multiprocessors have become mainstream even in mobile processors and conventional VLC does not work efficiently on a multiprocessor environment. Therefore, this paper proposes a novel control mechanism for VLC which is suitable on the multiprocessor environment. According to the simulation results, proposed control mechanism improves approximately 11% energy-delay-product compared with conventional VLC.

三重大学大学院　工学研究科

目 次

図 目 次

表 目 次

1　まえがき

現在，ノートパソコンや PDA (Personal Digital Assistant)，携帯電話などのモバイル端末の高性能化にともない消費エネルギーが増大し，バッテリによる駆動時間が短くなるという問題が発生している．そこで，モバイル端末の性能を落とすことなく低消費エネルギーを実現することが要求されている．中でも，モバイル端末の性能に大きく貢献しエネルギーを大量に消費するのがプロセッサである．

プロセッサで消費される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に分けられる．動的消費エネルギーはトランジスタのスイッチングによって消費されるエネルギーである．一方，静的消費エネルギーはトランジスタの漏れ電流 (リーク電流) によって引き起こされ，トランジスタのスイッチングに関係なく消費されるエネルギーで，リークエネルギーともいう．近年，回路の微細化にともなって，動的消費エネルギーが削減される一方，リークエネルギーが消費するエネルギーの割合が増加している．リークエネルギーはトランジスタ数に比例するため，プロセッサの高性能化に伴いプロセッサ数が増加し，そのプロセッサ数の増加に伴って容量が増大しているキャッシュシステムのリークエネルギー削減が重要である．そこでキャッシュのリークエネルギー削減手法の一つとして，可変レベルキャッシュが提案されている．

可変レベルキャッシュは，アプリケーションが必要とするキャッシュ容量が多い場合はキャッシュ容量を通常の容量で使用する通常モードとして動作し，必要とするキャッシュ容量が少ない場合はキャッシュ容量の半分をスリープモードに移行し，擬似的に 1 つ下位レベルの排他的キャッシュとして使用する低消費エネルギーモードとして動作する．可変レベルキャッシュは通常モードと低消費エネルギーモードを動的に切換ることによって高性能と低消費エネルギーの両立を実現する手法である．

従来の可変レベルキャッシュはシングルプロセッサ環境を想定しており，プロセッサが 1 つの場合においては性能を低下させることなく消費エネルギーを削減できることが示されている [1][2]．しかしながら，マルチプロセッサ環境での評価はこれまで行われていなかった．そこで，従来の可変レベルキャッシュをマルチプロセッサ上に文献 [1] の切換手法を用いて実装し，評価を行った結果，多くのベンチマークで性能の悪化が見られた．本論文では可変レベルキャッシュのマルチプロセッサ環境におけるモード切換手法の改良を行う．マルチプロセッサ環境で性能が悪化した原因を調査した結果，マルチプロセッサ環境の場合，キャッシュ必要容量の

三重大学大学院　工学研究科

違うプログラムが並列で動くことがあるが，従来のモード切換は，キャッシュ全体のキャッシュミス率を用いて切換を行っているため，キャッシュアクセスが多いプログラムに比重が寄ってしまいキャッシュアクセスが少ないプログラムのキャッシュ必要容量に対応できていないことが原因とわかった．そこで，シングルプロセッサ時と同様にキャッシュ全体の動作からモード切換を行うのではなく，マルチプロセッサの各プロセッサで動作するプログラムそれぞれの動作からキャッシュ必要容量を判断し，モード切換を行う手法へ改良することによりマルチプロセッサでのリークエネルギー削減を図る．本論文で行ったシミュレーション結果によるとエネルギー遅延積 (ED 積) で比較して，従来の可変レベルキャッシュと比べて 11%程度性能向上することがわかった．

本論文の構成については以下の通りである．第 2 章では，関連研究と従来の可変レベルキャッシュについて述べ，第 3 章でマルチプロセッサ環境での可変レベルキャッシュの問題点を述べ，第 4 章で可変レベルキャッシュのモード切換の改良手法を提案する．そして第 5 章で，性能と消費エネルギーの評価を行い， 最後に第 6 章でまとめる．

2　関連研究

これまでにキャッシュの様々なリークエネルギー削減手法が提案されてきた．これらの手法は通常状態と待機状態を切換える単位で，大きく以下の 3 つに分類する事ができる．

1. ライン単位の状態切換
   1 つ目は，キャッシュライン単位で通常状態と待機状態を切換えるものである．代表的な研究として，Drowsy Cache [3][4] が挙げられる．Drowsy Cache は定期的に全てのキャッシュラインへの電源電圧を下げ，アクセスが発生したキャッシュラインに対してのみ電源電圧を回復する事でリークエネルギーを削減する手法である．ライン単位で切換える手法は，キャッシュメモリの状態を細かく切換えることが可能である一方，センスアンプ等のライン以外にかかる消費エネルギーを削減できないといった問題がある．また，1 つのセット内で通常状態のキャッシュラインと待機状態のキャッシュラインが混在する状態となる，すなわち連想度が落ちてしまうという問題がある．更に，ライン単位での電源制御が必要なため，実装するのに，通常のメモリマクロやメモリコンパイラが利用できない．よって本論文ではこれ以上議論しない．

2. ウェイ単位の状態切換
   2 つ目は，Way-Allocatable Shared Cache [5] やウェイ予測キャッシュ [6] のようにウェイ単位で通常状態と待機状態を切換えるものである．Way-Allocatable Shared Cache は Cache Partitioning [7] と Way-adaptable Cache [8]，電力供給を部分的に停止する消費エネルギー削減手法を組み合わせた手法であり，各プロセッサがアクセスできるキャッシュ範囲を決めることでプロセッサ間の不均一な性能低下を防ぎ，同時に電力供給をウェイ単位で動的に停止することで低消費エネルギー化を行う手法である．ウェイ予測キャッシュは参照データが存在するウェイを LRU を利用して予測し，選択的に活性化する事で低消費エネルギー化を行う手法である．ウェイ単位で切換える手法は，ライン単位で切換える手法と同様，連想度が落ちるためキャッシュミスが多発し，実行時間が増大するという問題点がある．

3. セット数増減による状態切換

最後の 1 つは，DRI キャッシュ [9] のようにキャッシュシステム内を複数のバンクに分割し，セット数を増減させて通常状態と待機状態を切換えるものである．この手法は，ライン単位で切換える手法のように状態の細かな切換が出来ない反面，待機状態時にセンスアンプ等のライン以外の消費エネルギー削減が可能となる．セット数増減による切換を行う手法は，バンクを構成する場合にセット単位，ウェイ単位のどちらでも可能であるが，本論文ではセット単位でバンクを構成する事を考える．なぜなら，ウェイ単位で切換える事は前述したように実効的に連想度を低下させてしまうが，セット単位の切換の場合，各セット内の連想度が通常状態と同じに保たれる利点があるからである．本論文では，実行時間を維持しながら消費エネルギーを削減することを目的としているため，DRI キャッシュを基とした可変レベルキャッシュのモード切換手法の提案を行っている．これは，DRI キャッシュはキャッシュメモリの連想度を保つことが出来，実行時間に及ぼす影響は前述した他手法と比べ比較的少ないと考えられるからである．

2.1 Way-Allocatable Shared Cache

マルチプロセッサ環境でのキャッシュ消費エネルギー削減手法として，Way-Allocatable Shared Cache を簡単に説明する．Way-Allocatable Shared Cache とは，図 2.1 にようにマルチプロセッサの共有キャッシュにおいて，Cache Partitioning，Way-adaptable Cache と電力供給を部分的に停止する消費エネルギー削減手法を組み合わせることにより，性能の維持と低消費エネルギーの両立を図る手法である．

Cache Partitioning とはマルチプロセッサの各プロセッサで使用するキャッシュ領域を割り当てることでプロセッサ間のキャッシュ領域の競合を防ぐ手法である．また，Way-adaptable Cache とはプログラムの参照局所性を実行時に計測することでプログラムが要求しているキャッシュ領域を予測し，割り当てるウェイ数を増減させる手法である．Cache Partitioning により同じキャッシュ領域を取り合うことから発生するキャッシュミスの増加による性能低下を防ぎ，Way-adaptable Cache によって動的に動作しているアプリケーションのキャッシュ必要容量を判断する．つまり，Cache Partitioning と Way-adaptable Cache を組み合わせる (ウェイアロケーション) ことにより，よりキャッシュ競合による性能低下を防ぎ，キャッシュ

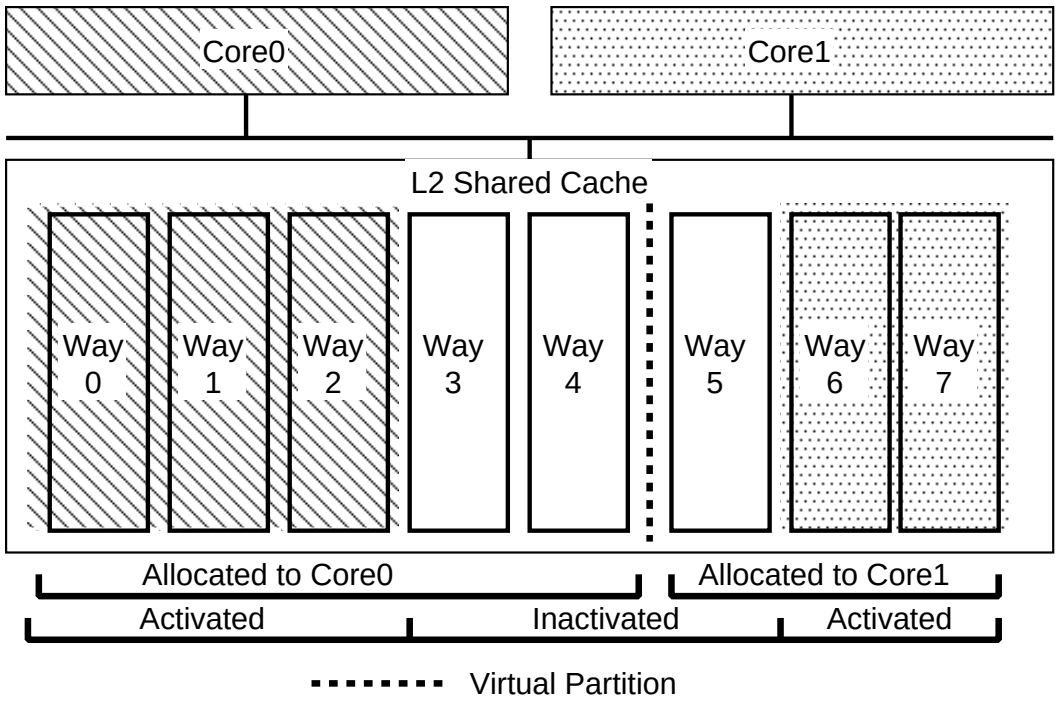

図 2.1: Way-Allocatable Cache

アクセス時にアクセスするウェイ数を減らせるため動的消費エネルギーの削減も期待できる．更に，Way-adaptable Cache でキャッシュ容量に余裕があると判断した場合，ウェイの電力供給を動的に停止することにより，性能を維持しつつ動的，静的消費エネルギーの削減を図っている．

2.2 DRI キャッシュ

2.2.1 DRI キャッシュの概要

図 2.2 に DRI キャッシュの概要図を示す．DRI キャッシュはある一定時間間隔 (interval) でキャッシュミス数をカウントする (miss counter)．そして，ミス数がある閾値 (miss-bound) より小さい場合には，キャッシュサイズを縮小しても性能には大きな影響を与えないと判断する．一方，ミス数が閾値よりも大きい場合にはキャッシュサイズを増大して性能低下を防ぐ．キャッシュサイズを減らす場合はその時の容量の半分にし，逆にキャッシュサイズを増やす場合は倍にする．これを複数段階に分けて実装

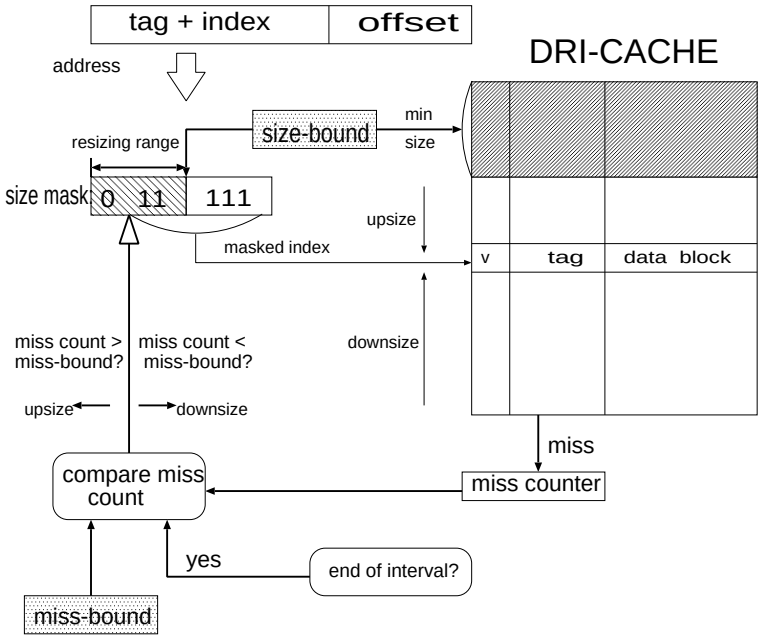

図 2.2: DRI キャッシュ

を行う事で，キャッシュサイズを必要に応じて変更する．例えば，動的に 256KB，128KB，64KB，32KB のキャッシュサイズに変更することができる．キャッシュラインへのアクセスは，アドレスにマスク (size mask) を掛ける事でキャッシュサイズの変化に対応させている．また，容量を小さくすると参照するインデックスのビット幅が減少する．すなわちタグフィールドのビット幅が増加するため，タグのビット幅を冗長に確保している．

このようにして DRI キャッシュでは，キャッシュサイズを縮小した場合の未使用領域の SRAM セルに対して電源電圧の供給を停止することによってリークエネルギーの削減をしている．また，DRI キャッシュは電源を切る部分のキャッシュを 1 つのバンクとして電源管理を行う事で，センスアンプやビット線などの，SRAM セル以外の回路の電源も切る事が出来るメリットがある．

2.2.2 DRI キャッシュの問題点

従来の DRI キャッシュでは，キャッシュサイズを縮小した場合，未使用領域の SRAM セルに対して電源電圧の供給を停止することでリークエネルギーを削減している．従って，電源を落とした部分に格納されていたデータは破棄されるためキャッシュヒット率が低下するという問題がある．

更に，DRI キャッシュは命令キャッシュ用に開発された手法であり，当該手法をデータキャッシュに適用する場合，縮小する部分への電力供給を完全に停止し，データを破壊するため，その部分にあるデータを下位の記憶層へと書き戻す必要がある．また，キャッシュサイズによってデータの配置が異なるために，キャッシュサイズを増大させる場合には，現在キャッシュに入っているデータを下位の記憶層へと書き戻さなければならない．DRI キャッシュをデータキャッシュに適用するとこれらの処理によって性能へ悪影響を与えるという問題がある．

2.3 従来の可変レベルキャッシュ

DRI キャッシュの上記の問題点を解決する手法として，メモリのスタンバイモードを利用し，かつ待機状態への切換時の書き戻しを抑制することで性能を向上させる可変レベルキャッシュが提案されている [1] [2].

2.3.1 可変レベルキャッシュの概要

可変レベルキャッシュは，DRI キャッシュと同様に，ある一定時間間隔でキャッシュミス数をカウントする事で，キャッシュへの要求性能をキャッシュミス率から動的に判断し，キャッシュ容量を増減させる．しかし，DRI キャッシュのように単純に容量を減少させるだけでは，キャッシュミス回数が増加してしまう．そこで，容量の半分の電源供給を遮断するのではなく，スリープモードとする．スリープモードとは，電源の共有を完全に停止するのではなく，データの内容が破壊されない程度に電源電圧を下げた状態の事を言う．スリープモードにはデータの内容が保存されるというメリットがある．また，電源供給を停止する場合よりはリークエネルギーの削減率が低くなるが，通常モードよりは大幅にリークエネルギーが削減できるというメリットもある．しかし，単純にスリープモードにしては，スリープモードとなっている領域へのアクセスは通常のアクセスより時間がかかるため，スリープモードになっている領域へアクセス

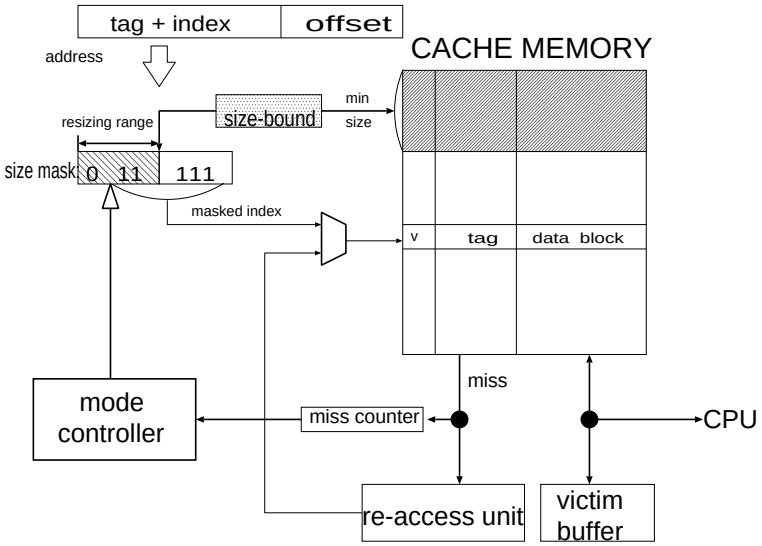

図 2.3: 可変レベルキャッシュ

が多発する場合，性能が悪化する恐れがある．そこで，スリープモードの領域を 1 つ下位レベルの排他的キャッシュ [10] [11] とすることでスリープモードでのキャッシュアクセスを減らし，消費エネルギーを削減する．

例えば，256KB の L2 キャッシュに可変レベルキャッシュを適用した場合，性能があまり必要でないと判断したときは，半分の 128KB はスリープモードへと移行し，擬似的な L3 キャッシュとして動作する．この時，擬似 L3 キャッシュは排他的キャッシュとして動作させる．

図 2.3 に可変レベルキャッシュの概要図を示す．可変レベルキャッシュは主に，DRI キャッシュの回路に再アクセスユニット (re-access unit) とバッファ (buffer) を加えた形で構成される．再アクセスユニットとバッファは共に L2 と L3 にキャッシュを分割したときに利用し，再アクセスユニットは L2 キャッシュでキャッシュミスをした場合に，L3 キャッシュ領域をスリープモードから通常モードへ切換え、L3 キャッシュにアクセスする際に用いられる．バッファは排他的キャッシュとして L2 キャッシュのリプレイスデータを L3 キャッシュへ書き戻すときに用いる．

以降では，全てのラインがアクティブの時を「通常モード」，キャッシュの半分をスリープモードにし，排他的キャッシュとして動作するときを「低消費エネルギーモード」とする．

排他的キャッシュ(Exclusive Cache)

排他的キャッシュは AMD 社が開発したキャッシュアーキテクチャで，L1 キャッシュと L2 キャッシュのデータを排他的にすることでキャッシュメモリを有効に利用する手法である．

以降の説明で，キャッシュの構成を L1 キャッシュは 128KB，L2 キャッシュは 256KB と想定する．

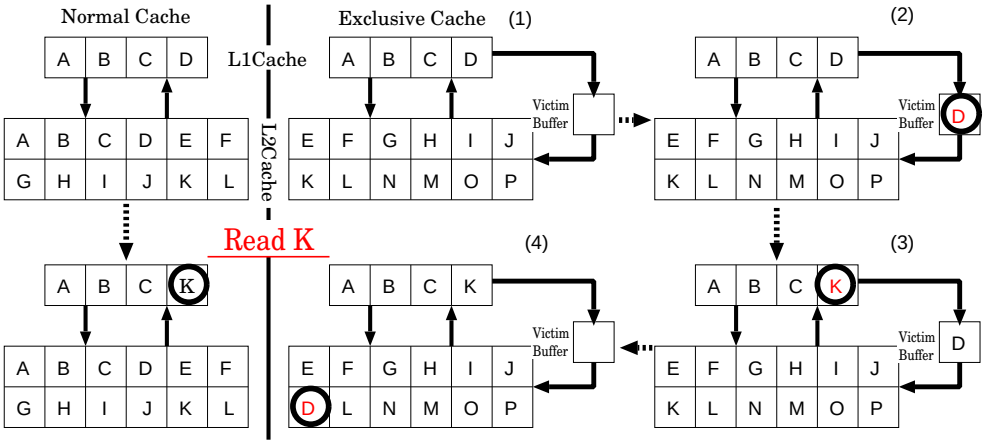

図 2.4: 通常キャッシュと排他的キャッシュ

従来のキャッシュでは，すべてのデータはまず L2 キャッシュに格納され，その後 L2 キャッシュから L1 キャッシュへとコピーされる．そのため図 2.4 の Normal Cache のように，データ K の読み出しが合った場合でも，L1 キャッシュ内の LRU を追い出し，L2 キャッシュからコピーするだけである．そのため，L1 と L2 の割り当てがあっても全体的なキャッシュ・サイズは L2 キャッシュに相当する 256K バイトであるといえる．それに対し，排他的キャッシュでは，図 2.4 のように，L1 キャッシュから L2 キャッシュへと書き戻されるデータを一度バッファに移し (図 2.4(2))，その後 L2 キャッシュから必要なデータをロードし (図 2.4(3))，最後にバッファのデータを L2 キャッシュに書き戻している (図 2.4(4))．このように，データを L1 と L2 との間で交換する事 (排他的に処理する事) で全体的なキャッシュ・サイズが 128+256=384K バイトとなり，キャッシュサイズを有効に活用できる

書き戻しペナルティ軽減

可変レベルキャッシュは動的に要求性能を判断し，通常モードと低消

費エネルギーモードを切換えている．通常モードから低消費エネルギーモードへと切換える際はデータを下位層に書き戻す必要はないが，低消費エネルギーモードから通常モードに切換える際は，キャッシュ内のデータの配置が通常モードでの本来の位置と異なるため，データを下位層に書き戻す必要がある．

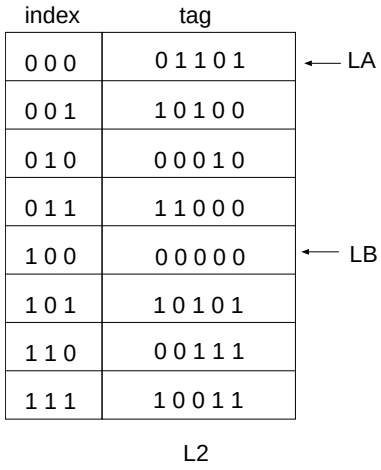

図 2.5: 通常モードへ切換わる前のキャッシュ

そこで，低消費エネルギーモードから通常モードに切換える際のペナルティを低減する手法が提案されている．書き戻しをする際に不良配置ラインを選別する事で，主記憶に書き戻しをするデータ数を減少させる手法である．データが正しい場所にあるか否かの判定はタグとインデックスを用いて行う．可変レベルキャッシュではキャッシュを均等に 2 つのバンクに分割している．そのため，bankA のインデックスの最上位ビットは 0 であり，bankB のインデックスの最上位ビットは 1 となる．また，低消費エネルギーモード時にセット数が半減する，すなわち参照するインデックスの範囲が半分になり最上位ビットは無視されるため，アドレスのタグフィールドはインデックスの最上位ビットと重複するように 1bit 分冗長にとってある．よって，タグフィールドの冗長な 1bit の値が低消費エネルギーモード時には無視されているインデックスの最上位ビットの値と一致しなければ，不良配置ラインだと断定する事が出来る．例えば図 2.5 では，LA のインデックスの最上位ビットが 0 なのに対して，タグの最下位ビットが 1 になっており，冗長な 1bit が一致しておらず，LA のラインは低消費エネルギーモード時にリプレイスされたラインだとわ

かる．このようにタグとインデックスの重なる部分を比較し，ダーティビットの立っている不良配置ラインだけを書き戻す事でオーバヘッドの軽減を図っている．また，1 ビットの比較を行うのみなので，ハードウェア量がほとんど増えない．

2.3.2 可変レベルキャッシュの動作詳細

通常モード時は，通常のキャッシュとして動作する．そして低消費エネルギーモードへの切換条件を満たした場合は，低消費エネルギーモードへと切換わる．その際，図 2.3 に示すマスクのビット幅を変更し，参照するインデックスに制限をかけ，キャッシュメモリの半分をスリープモードへと切換える．このように可変レベルキャッシュは DRI キャッシュと異なり電源供給を停止するのではなくスリープモードへと切換えることによって，切換える部分のデータを書き戻さなくてよいという利点がある．低消費エネルギーモード時はマスクによってキャッシュメモリへのアクセスに制限をかけ，図 2.6 のように容量が通常モード時の半分になっている L2 キャッシュの部分へアクセスを行う．キャッシュミスをした場合，L3 キャッシュへと書き戻すデータを図 2.3 に示すバッファへと移動し，同じく図 2.3 に示す再アクセスユニットにより L3 キャッシュの部分へとアクセスを行う．L3 キャッシュでキャッシュヒットした場合は，ヒットしたデータを L2 キャッシュへと移動し，L3 キャッシュの部分でもキャッシュミスした場合はメモリにアクセスを行う．バッファに入れられたデータはキャッシュアクセスの完了後，バックグラウンドで下位の記憶層に書き戻しされる．通常モードへの切換条件を満たした場合，低消費エネルギーモードから通常モードへと切換えられる．このとき，キャッシュメモリ内のデータの配置が変わってしまうためキャッシュメモリ内のデータをメモリへと書き戻し，マスクのビットをすべて 1 にし，インデックスへの制限を解除する．可変レベルキャッシュはこのように動作を行い，低消費エネルギーと高性能を両立させる．

このようにして可変レベルキャッシュはキャッシュ必要容量を判断し，動的に通常モードと低消費エネルギーモードを切換えることによってシングルプロセッサでは通常キャッシュと比較して約 30%の性能向上が得られた．しかし，近年マルチプロセッサがスマートフォンなどのモバイル端末においても使用されている．そこで，次章でマルチプロセッサ環境での可変レベルキャッシュについて述べる．

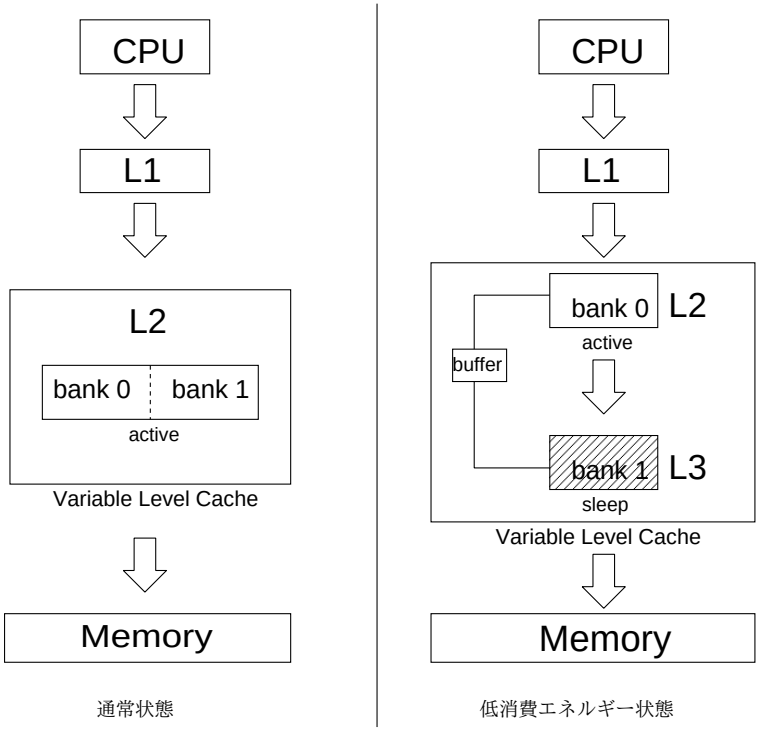

図 2.6: 可変レベルキャッシュへのアクセス

3　マルチプロセッサ

近年汎用プロセッサにおいてマルチプロセッサが注目を集めており，スマートフォンなどのモバイル端末においてもマルチプロセッサが使用されている．しかし，従来の可変レベルキャッシュはシングルプロセッサを想定しており，プロセッサが 1 つの場合においては消費エネルギーの削減に成功しているが，マルチプロセッサでの評価は行われていなかった．そこで，本論文では可変レベルキャッシュをマルチプロセッサ環境へ適用し評価を行った．

マルチプロセッサシミュレータ上に可変レベルキャッシュを組み込み，SPEC2000 ベンチマークを用いて予備評価を行った．結果を図 3.7 に示す．図 3.7 は縦軸が実行時間，横軸がベンチマークであり，全ての値は通常キャッシュを 1 として正規化してある．図 3.7 より，通常キャッシュと比較して従来手法では約 15%実行時間が悪化していることがわかる．これは共有キャッシュ構成のマルチプロセッサの場合，シングルプロセッサと

違ってキャッシュ上に複数のプロセッサが存在し，それぞれのプロセッサ上で必要とするキャッシュ容量が異なるプログラムが実行される．しかし，従来のモード切換手法を単純にマルチプロセッサに適用した場合，全プロセッサのキャッシュミス率でモード切換を判定してしまう．例えば図 3.8 のように，プロセッサ 0 のキャッシュアクセスが 7000 回あり，キャッシュミスが 1000 回，プロセッサ 1 のキャッシュアクセスが 1192 回あり，キャッシュミスが 800 回あったとすると，キャッシュミス率 21%となり低消費エネルギーモードになってしまうが，プロセッサ 1 のプログラムではキャッシュ容量を必要としている．このように，キャッシュアクセスが多いプログラムにモード判定の比重が偏ってしまい柔軟に対応ができないことが原因だとわかった．そこで本論文では，複数のプログラムが並列に処理を行っていても，各プログラムの必要キャッシュ容量に対応できるよう，新たなモード切換手法を提案する．

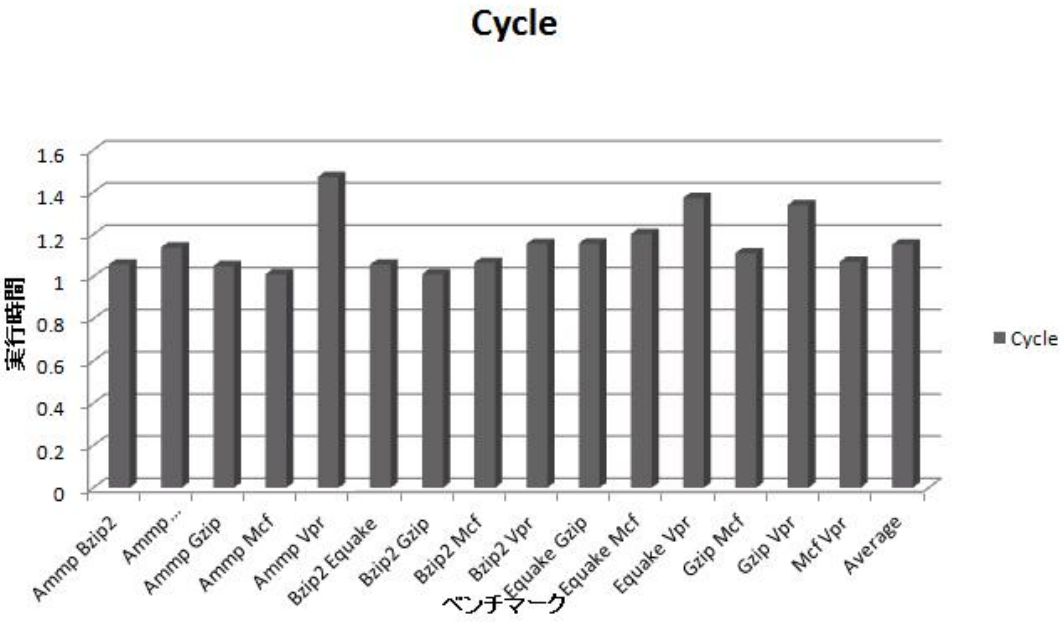

図 3.7: 実行時間

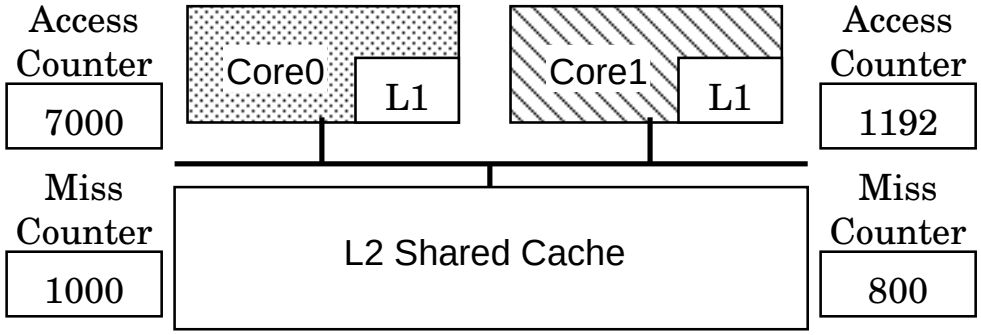

図 3.8: キャッシュアクセス例

4　モード切換手法

4.1　従来のモード切換手法

文献 [1] で提案されているモード切換制御のブロック図を図 4.9 に示す.

図 4.9 のように一定アクセス回数 (Constant value) ごとにキャッシュミス率を測定し，閾値 (Threshold) と比較している．その際,

$$\text{キャッシュミス率} \geq \text{閾値} \tag{1}$$

だった場合通常モード,

$$\text{キャッシュミス率} < \text{閾値} \tag{2}$$

だった場合は低消費エネルギーモードへ遷移する．また，一定のアクセス回数を 2^X と 2 の乗数することによってキャッシュミス率を算出する際に，図 4.9 のように除算ではなく図 4.10 のようにシフト演算で算出し，ハードウェアコストを削減する手法が提案されている [2].

4.2　モード切換手法の提案

予備評価から従来のモード切換を単純にマルチプロセッサ環境へ実装すると性能が悪化することが明らかとなった．これはシングルプロセッサとは違い，キャッシュ必要容量の異なる複数のプログラムが並列に動作することにより，キャッシュアクセス数が多いプログラムにモード切換の比重が偏ってしまうことが原因である．そこで，本論文では複数のプロ

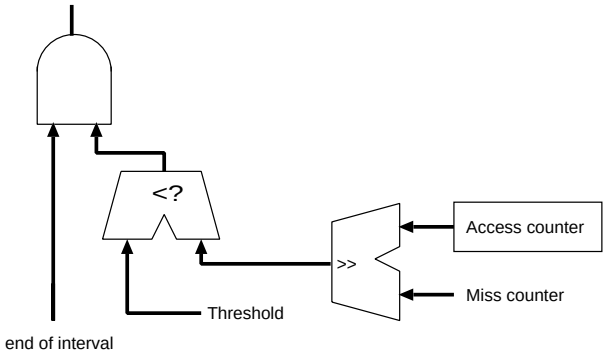

図 4.9: 切換制御のブロック図

グラムが並列で動作していても，キャッシュアクセスの多いプログラムにモード切換比重が偏ることのない 2 種類の新たなモード切換手法を提案する．

4.2.1 改良案 1

1 つ目の手法は，マルチプロセッサにおいて可変レベルキャッシュに対する全プロセッサのキャッシュミス率の平均を測定するのではなく，各プロセッサでキャッシュミス率を求める方法である．この手法では従来の可変レベルキャッシュと同じように可変レベルキャッシュに対する全キャッシュアクセスのカウントを行い，キャッシュミス率を算出する．しかし，キャッシュミス率算出する際は全体のキャッシュミス率ではなく各プロセッサでの可変レベルキャッシュに対するキャッシュミス率を測定し，モード切換判定を行う．各プロセッサで別々にキャッシュミス率を算出することにより，キャッシュアクセスが多いプログラムに比重が偏ることを防いでいる．

各プロセッサのキャッシュミス率を出した後の状態遷移を表 4.1 示す．表 4.1 のように全てのプロセッサでキャッシュミス率が閾値を下回った場合のみ低消費エネルギーモードに切換え，逆に全プロセッサで上回れば通常モードに切換えを行うことで性能悪化を低減させる．その他の場合は状態を維持する．

以下の例では，図 3.8 が通常モードで動作している可変レベルキャッシュであり，モード切換間隔が 2^{13}，キャッシュミスの閾値が 30%であると仮定する．

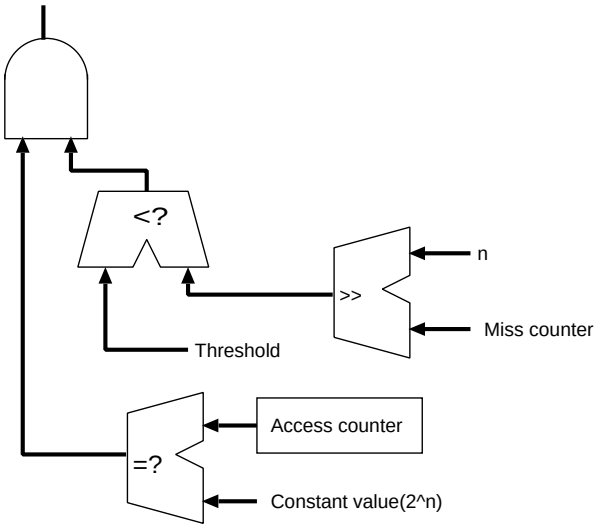

図 4.10: シフト演算を用いた切換制御のブロック図

表 4.1: 状態遷移

プロセッサ 1 / プロセッサ 0	閾値以下	閾値より上
閾値以下	低消費エネルギーモード	状態維持
閾値より上	状態維持	通常モード

プロセッサ 0 ではキャッシュアクセスが 7000 回あり，そのうちの 1000 回がキャッシュミス，プロセッサ 1 ではキャッシュアクセスが 1192 回，800 回のキャッシュミスが起こったとする．この場合従来手法では全体のキャッシュミス率を求めるためキャッシュミス率が約 21%となり，低消費エネルギモードへモード切換を行う．改良案 1 では各プロセッサでキャッシュミス率を求めるため，プロセッサ 0 のキャッシュミス率が約 8%，プロセッサ 1 のキャッシュミス率が約 45%となりモード切換を行わず通常モードで動作する．

4.2.2 改良案 2

2つ目の手法は，改良案1を更に改良したものである．可変レベルキャッシュは動的に要求性能を判断し，通常モードと低消費エネルギーモードを切換えている．通常モードから低消費エネルギーモードへと切換える際はデータを下位層に書き戻す必要はないが，低消費エネルギーモードから通常モードに切換える際は，キャッシュ内のデータの配置が通常モードでの本来の位置と異なるため，データを下位層に書き戻す必要がある．しかし，シミュレーションの結果，通常モードから低消費エネルギーモードへ切換わり，次のモード切換時に低消費エネルギーモードから通常モードへ切換わる場合があることがわかった．モード切換持にペナルティが発生する可変レベルキャッシュでは，このような頻繁なモード切換は性能の悪化を招く恐れがある．そこで，一度だけモード切換判定が行われただけでモード切換を行うのではなく，モード切換を行う前にビットカウンタを追加し，ビットカウンタの値からモード切換を行うようにすることにより，頻繁なモード切換を抑制することができる．更に，改良案1と同様に各プロセッサでそれぞれキャッシュミス率を求めると同時に，全体のキャッシュミス率も考慮したモード切換を行う．これにより，キャッシュ全体の動作と共に各プロセッサでのプログラムの動作を追従することが可能になる．

つまり，手法2では図 4.11 の制御フローのようにマルチプロセッサでプロセッサが2つの場合，可変レベルキャッシュに対する一定間隔のプロセッサ0のキャッシュミス率，プロセッサ1のキャッシュミス率，全体のキャッシュミス率を算出して閾値と比較し，通常モード，または低消費エネルギーモードの過半数 ((プロセッサ数 + 1)/2 以上) を占めたモードへ切換を行う．例えば，プロセッサ0が通常モード，プロセッサ1が低消費エネルギーモード，キャッシュ全体が通常モードと判定した場合，可変レベルキャッシュは通常モードで動作する．また，モード切換信号の前に，ビットカウンタを組み込むことによって，一時的に通常モードから低消費エネルギーモードになり，通常モードへ戻る時に起こる無駄な WriteBack を減らすことにより，性能の悪化抑制を図っている．

図 3.8 を使用して説明すると，改良案2ではプロセッサ0のキャッシュミス率 8%，プロセッサ1のキャッシュミス率 45%，そして全体のキャッシュミス率 21%の3つのキャッシュミス率を使用し，ビットカウンタの増減を決定する．今回は閾値 30% 以下が過半数を占めるためビットカウンタは加算され，もしビットカウンタが2の場合は低消費エネルギーモー

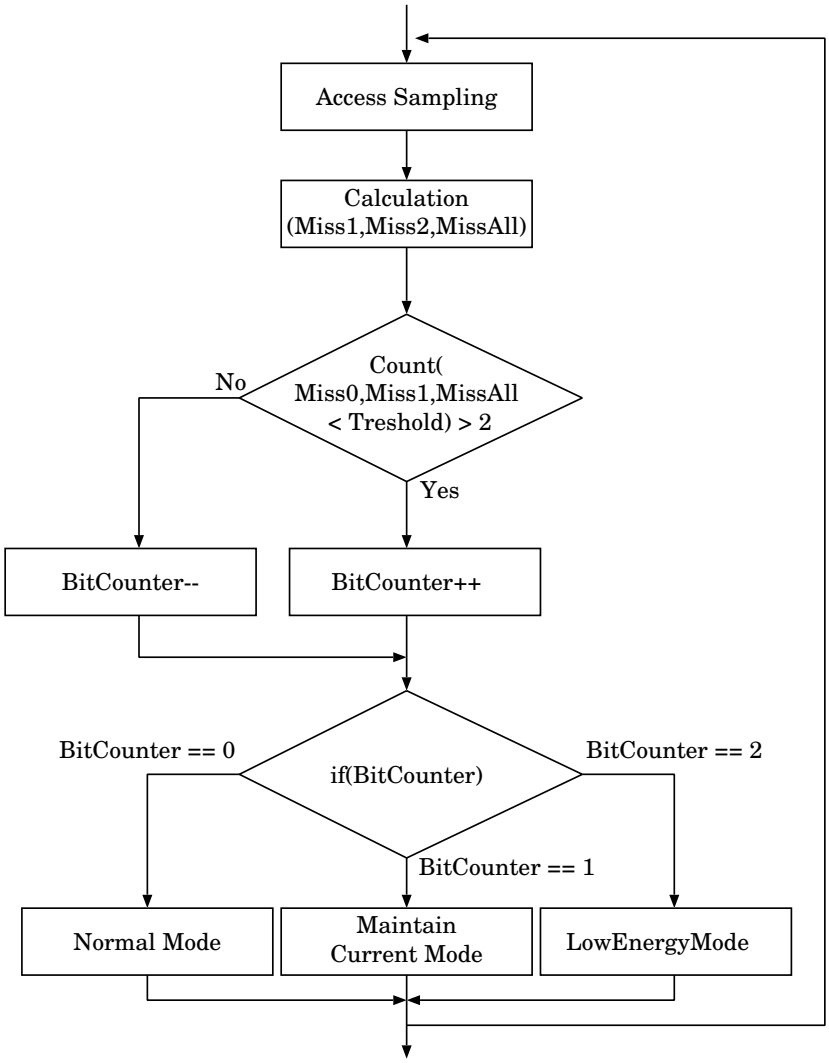

図 4.11: 改良案 2 の制御フロー

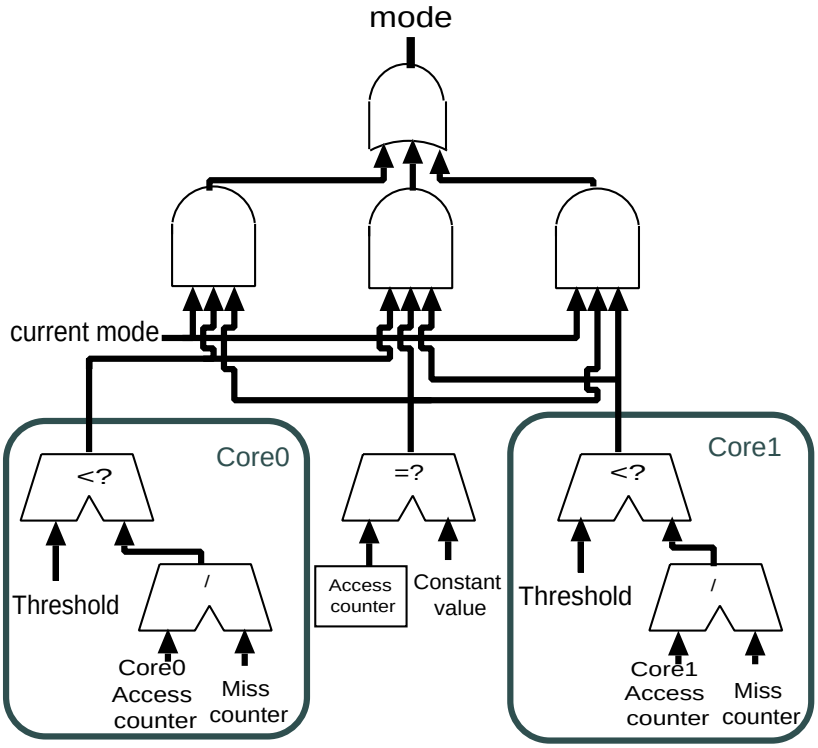

図 4.12: 改良案 1 の可変レベルキャッシュコントローラ

ドへモード切換を行い，1 の場合はモード切換を行わずに現在のモードを維持する．ここで，もし閾値以上が過半数を占めた場合，ビットカウンタは減算され，0 の場合は通常モードへモード切換を行う．つまり，ビットカウンタが 0 を示している場合は通常モードで動作し，ビットカウンタが 2 を示している場合は低消費エネルギーモードで動作する．

4.2.3 モード切換コントローラのハードウェア詳細設計

これまで可変レベルキャッシュの切換コントローラの消費エネルギーについては，シフト演算などの少量のハードウェア量で済むと考えられていたため詳しく調査していなかった．しかし，従来手法ではシフト演算でキャッシュミス率を求めているのに対し，改良案 1 では除算器を使用している．そこで，本論文では可変レベルキャッシュのモード切換コントローラのハードウェア設計を行った．

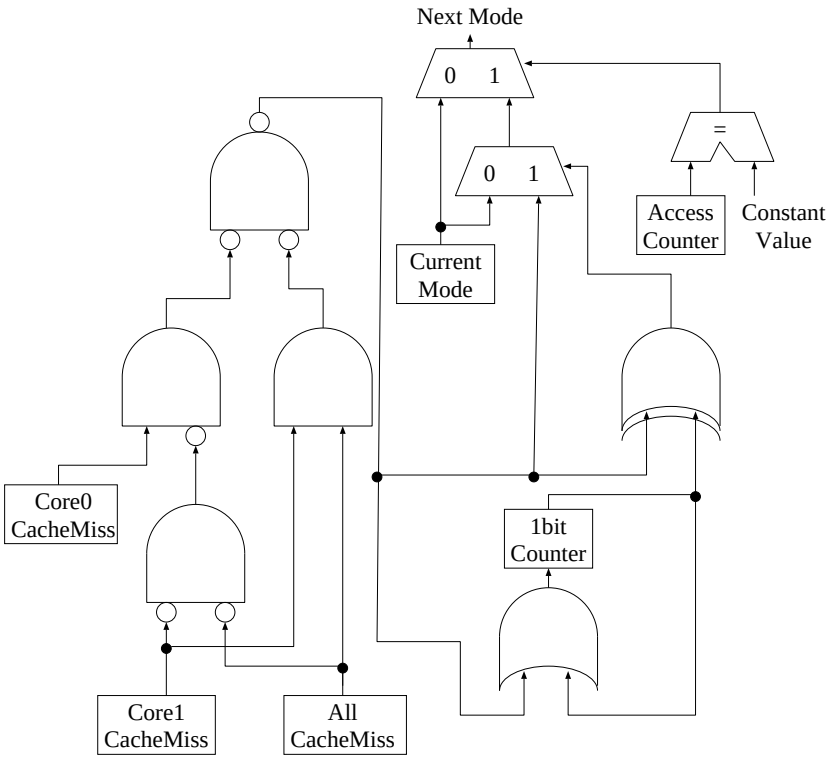

図 4.13: 改良案 2 の可変レベルキャッシュコントローラ

改良案 1 のコントローラ

図 4.12 に設計した改良案 1 のコントローラのブロック図を示す．今回設計したコントローラの除算器はキャッシュミス率と比較するために，十分な精度を確保するために 16bit の除算器を用いることにした．

コントローラは入力として各プロセッサのキャッシュアクセス回数，各プロセッサのキャッシュミス回数，そして今現在のモードを受け取り，次のサイクルからのモードを決定する．

改良案 2 のコントローラ

図 4.13 に設計した改良案 2 のコントローラのブロック図を示す．今回ビットカウンタは 1 ビットカウンタを使用したため，加算器は使用していない．そのため，改良案 1 のコントローラにマルチプレクサ 2 個と XOR 回路 1 個，OR 回路 1 個，そしてビットカウンタを追加しただけである．こちらのコントローラでも入力として各プロセッサのキャッシュアクセス回数，各プロセッサのキャッシュミス回数，そして今現在のモードを受け

取り，入力されたデータとビットカウンタから次のサイクルからのモードを決定する．

5　性能評価

5.1　評価方法

5.1.1　エネルギー遅延積

本論文では，可変レベルキャッシュについて，実行時間と消費エネルギーの評価を行う．

DRI キャッシュとの比較は既に文献 [1] でシングルプロセッサにおいて有用性が示されている．また，DRI キャッシュは命令キャッシュに適用させるために提案されているため，共有キャッシュに適用させる可変レベルキャッシュとは性質が違う．そのため，今回比較は行わなかった．

評価指標として通常，高性能なプロセッサを評価する場合には実行時間を見比べればよく，低消費電力プロセッサにおける評価では主に消費エネルギーを見比べればよい．しかし，今回は高性能かつ低消費エネルギーのプロセッサを作成することを目標としているため，実行時間と消費エネルギー両方について見比べる必要がある．そのため，高性能かつ低消費エネルギーのプロセッサ評価を行う場合によく用いられるエネルギー遅延積を用いる．エネルギー遅延積は消費エネルギー E と実行時間 T を用いて以下の式で求めることができる．

$$\text{エネルギー遅延積} = E \times T \tag{3}$$

このエネルギー遅延積が小さいほど低消費エネルギーと高性能の両立が達成できている．エネルギー遅延積では性能に依存しない消費エネルギーに性能に依存する実行時間をかけているため，消費エネルギーの指標とは異なり，性能に依存する値となる．また，エネルギー遅延積は消費エネルギーに実行時間の 2 乗をかけても求めることができる．そのため，消費エネルギーと性能について評価する場合は性能の方が重要視された指標となるが，今回は消費エネルギーと性能についての評価であるのでエネルギー遅延積を用いた．

5.1.2　消費エネルギー

消費エネルギーについては，文献 [12] [13] を参考に以下のように近似した．

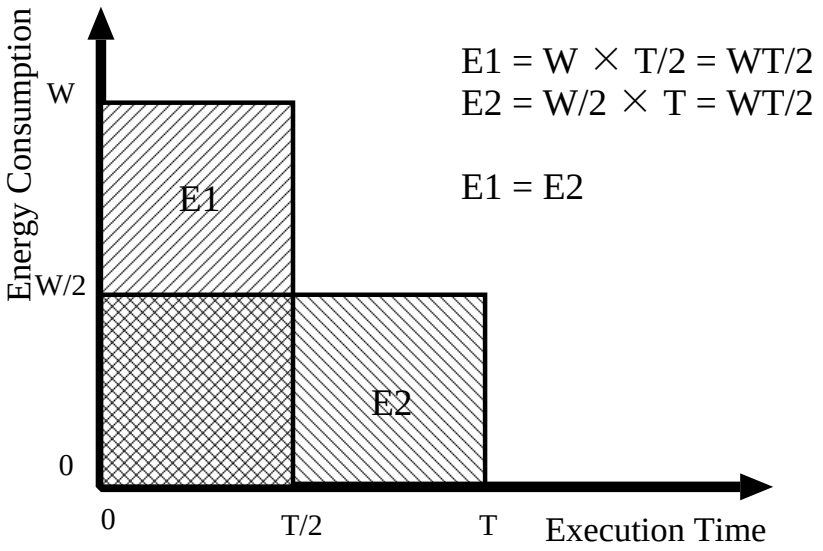

図 5.14: 実行時間と消費エネルギー

まず，キャッシュアクセスによる動的消費エネルギーの総和 DE_{total} はラインアクセス当りの平均動的消費エネルギー DE_{line} とアクセス回数 $Access$ の和で求められる．よって近似式は

$$DE_{total} = DE_{line} \times Access \tag{4}$$

となる．

次にリークエネルギー LE_{total} は 1 クロックサイクルで消費するライン当りの平均リークエネルギー LE_{line} に総ライン数 $CSize$ とプログラム実行クロックサイクル数 CC の積で求まる．よって近似式は

$$LE_{total} = CC \times LE_{line} \times CSize \tag{5}$$

となる．

式 (2) の LE_{line} は，キャッシュメモリ全体のうちスリープモードとなっているラインの割合 SR とスリープモードのライン当たりのリークエネルギー LE_{sline} の積に，通常モードのラインの割合 $(1 - SR)$ と通常モードのライン当たりのリークエネルギー LE_{aline} の積を加えたものであるため，以下のように

$$LE_{line} = SR \times LE_{sline} + (1 - SR) \times LE_{aline} \tag{6}$$

となる．

スリープモード時に L3 キャッシュにアクセスする場合, 通常モードに切換える必要がある, L3 キャッシュアクセスの際，キャッシュラインをスリープモードから通常モードに切換えるエネルギーは

$$CE_{total} = CE_{line} \times BSize \times Access_{sline} \tag{7}$$

と表される．CE_{line} はキャッシュライン当たりのモード切替エネルギーであり，$BSize$ はモードを切換えるライン数 (バンクの大きさ)，$Access_{sline}$ は通常モードへの切換回数，つまり可変レベルキャッシュにおけるスリープモード時の L3 キャッシュに当たる部分へのアクセスの回数である．

よって，キャッシュ全体の消費エネルギー E_{total} は，式 (1)，式 (2)，式 (4) の和，すなわち，

$$E_{total} = DE_{total} + LE_{total} + CE_{total} \tag{8}$$

となる．

尚，通常キャッシュや DRI キャッシュでは，スリープモードのラインは存在しないため，$CE_{total} = 0$ となる．

キャッシュのエネルギーを評価する値は Cacti [15] を用い, 32nm プロセスを想定し，ラインサイズを 64B として求めた, 求めた値を以下に示す.

$$DE_{line} = 1.75E - 10(J) \tag{9}$$

$$LE_{aline} = 8.38E - 14(J) \tag{10}$$

$$LE_{sline} = 1.10E - 17(J) \tag{11}$$

$$CE_{line} = 6.36E - 20(J) \tag{12}$$

以上の式で求まった，実行サイクル数と消費エネルギーの積を求め比較を行う．

5.2 シミュレータ

これまでマルチプロセッサ環境での可変レベルキャッシュの評価は SimMc [16] ベースの 1 命令/$cycle$ の単純なシミュレータで評価を行っていた．しかし，SimMc による評価は 1 命令/$cycle$ であるため，モバイル端末などで実際に使われているプロセッサとキャッシュアクセスの動作が大幅に違う．そ

表 5.2: シミュレーションのキャッシュに関するパラメータ

キャッシュ容量	
L1 命令-cache	32KB(64B/entry, 1way, 512entry)
L1 データ-cache	32KB(64B/entry, 2way, 256entry)
L2 cache	512KB(64B/entry, 16way, 512entry)
ヒット・レイテンシ	
L1 cache	1 cycle
L2 cache	16 cycle
主記憶	250 cycle
モード切換のオーバーヘッド	
レイテンシ	10cycle

こで，より現在の汎用プロセッサに近いシミュレータを使用して評価を行う．シミュレータは東京大学で研究されているサイクルアキュレートマルチプロセッサシミュレータである鬼斬弐 [17] を使用する．鬼斬弐はスーパースカラのシミュレータであり，分岐予測やパイプラインストールなどの正確な動作のシミュレーションが行えるシミュレータである．鬼斬弐を使用することで，より実機に近いアクセスタイミングでのシミュレーションが行える．

5.3 実験環境

可変レベルキャッシュに関して，性能と消費エネルギーの評価を行うため，鬼斬弐を改造し，可変レベルキャッシュを実装した．ここで可変レベルキャッシュは鬼斬弐の L2 の統合キャッシュに実装する事を想定た．プロセッサ構成を表 5.2 に示す．

今回の評価において，可変レベルキャッシュは，1 つの 512KB の L2 キャッシュとして扱う通常モードと，L2 キャッシュが 256KB，L3 キャッシュが 256KB である排他的キャッシュとして動作する低消費エネルギーモードの 2 種類を動的に切換える．可変レベルキャッシュは 16384(2^{14}) 回のキャッシュアクセス毎に各プロセッサでキャッシュヒット率を測定し，全てのプロセッサでキャッシュミス率が閾値以下ならば通常モードに，全てのプロセッサで閾値を超えていれば低消費エネルギーモードに切換えるものとする．また，キャッシュミス率の閾値は 15%とする．なお，これらの閾値

表 5.3: ベンチマーク

ベンチマーク	処理内容
ammp	計算科学
bzip2	圧縮
equake	地震波電波シミュレーション
gcc	C 言語コンパイラ
gzip	圧縮
mcf	組み合わせ最適化
vpr	FPGA の配置配線

は各手法についてシミュレーション評価により実験的に求めた最適値である．可変レベルキャッシュと改良したモード切換を組み込んだ鬼斬弐上でベンチマークプログラムを実行し，それぞれの手法の性能を測定した．ベンチマークプログラムは表 5.3 に示すように，SPEC2000 [18] の 7 種類を使用し，各プロセッサで別々のプログラムを実行して評価を行った．その際，プログラムの実行安定時の評価を行う為に，プログラム実行開始より 10 億命令実行後の 10 億命令を評価対象とした．

5.4 評価結果

5.4.1 シミュレーション評価

実験によって得られた可変レベルキャッシュの実行時間を図 5.15，モデル式から算出した消費エネルギーを図 5.16，エネルギー遅延積による評価結果を図 5.17 に示す．縦軸は図 5.15 では実行時間 (cycle)，図 5.16 では消費エネルギー，図 5.17 ではエネルギー遅延積であり，横軸は図 5.15，図 5.16，図 5.17 全てにおいて，使用したベンチマークを表している．縦軸はそれぞれ通常のキャッシュの結果で正規化したものである．図 5.15 では各ベンチマークで通常キャッシュで動作した実行時間と，低消費エネルギーモードで動作した実行時間を色分けしてある．「$Normal$」が通常キャッシュ,「$Conv$」が従来の可変レベルキャッシュ,「$Inte$」が改良案 1 を組み込んだ可変レベルキャッシュ,「$BitCounter$」が改良案 2 を組み込んだ可変レベルキャッシュである． 実行時間を見ると，$Conv$ と比較して $Inte$，$BitCounter$ ともに実行時間の悪化を抑えることができた．特に

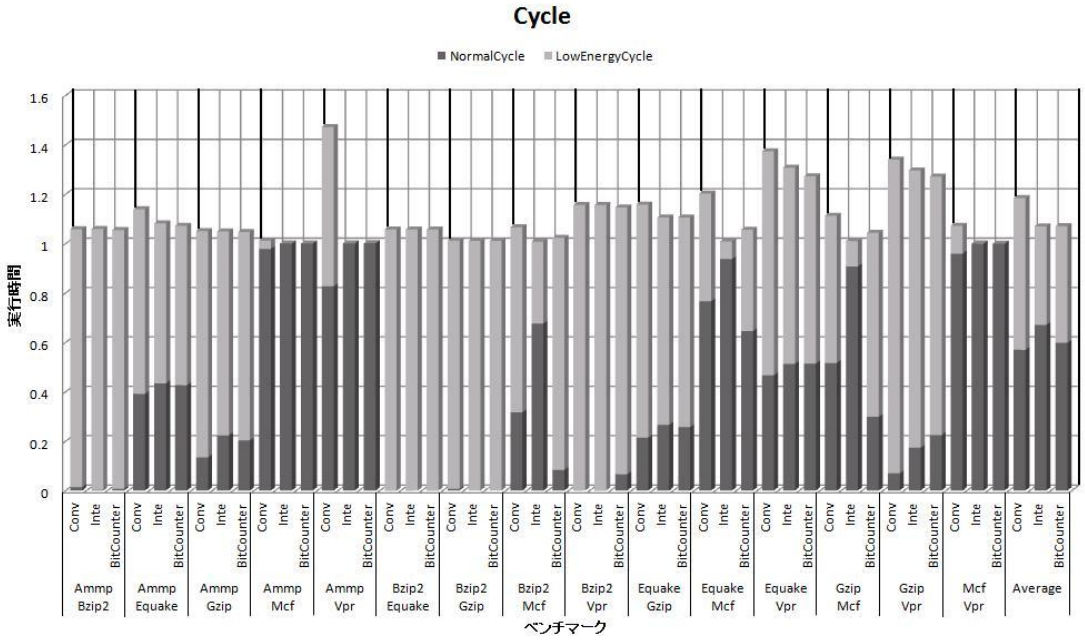

図 5.15: 実行時間

$AmmpVpr$ では通常キャッシュの 1.45 倍ほどの実行時間の悪化があったのに対し，提案手法である $Inte$，$BitCounter$ は通常キャッシュとほぼ同じ実行時間で実行を終了している．これは $Ammp$, Vpr 共にキャッシュ必要容量が高いが，従来手法では低消費エネルギーモードでも動作しているのに対し，提案手法では常に通常モードで動作しているからである．しかし，平均を見ると低消費エネルギーモードから通常モードへモード切換を行う際に行う書き戻しの影響で通常キャッシュよりは実行時間が延びている．

消費エネルギーをみると，通常キャッシュと比べて平均で約 15%消費エネルギーが低いという結果がでている．一部のベンチマークでは通常キャッシュとほぼ同じ消費エネルギーを示しているものがあるが，これはキャッシュミスが多く，ほとんど，または全て通常モードで動作しているからだ．また，実行時間でほとんどが低消費エネルギーモードで動作していた $Bzip2$ を含んだベンチマークを見ると最大 30%の消費エネルギー削減に成功している．しかし，$Conv$ と $Inte$, $BitCounter$ を比較すると，$Inte$ よりも $Conv$ の消費エネルギーの方が小さいベンチマークがある．こ

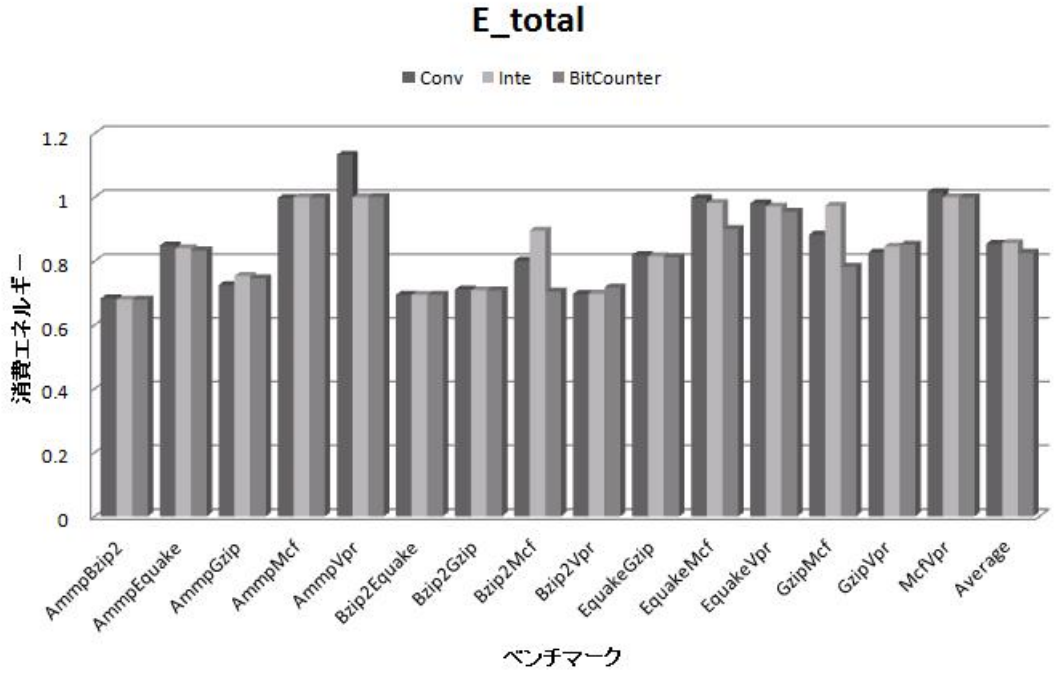

図 5.16: 消費エネルギー

れは $Inte$ がモード切換を行う際に，各プロセッサのキャッシュミス率が閾値以下と少し厳しいモード切換条件をつけていること，そして，キャッシュミス率の閾値を従来手法では 30%だったのに対して提案した 2 手法は共に 15%で行っていることが原因だと考えられる．

最後にエネルギー遅延積を見ると，$Conv$ と比較して $BitCounter$ は平均 11%，$Inte$ は平均 7%性能改善に成功している．また，消費エネルギーでは $Conv$ に負けていたベンチマークにおいても実行時間の悪化を抑制したことによってほぼ $Conv$ と同等のエネルギー遅延積になっている．しかし，一部ベンチマークにおいて通常キャッシュよりもエネルギー遅延積の悪化が見られる．悪化した原因はこれらのベンチマークは消費エネルギーでは通常キャッシュよりも低い消費エネルギーを示しているが，実行時間が延びており，低消費エネルギーモードで動作することによる消費エネルギー削減部分よりも，書き戻しペナルティなどによる実行時間悪化が大きく，エネルギー遅延積の悪化に繋がったと考えられる．

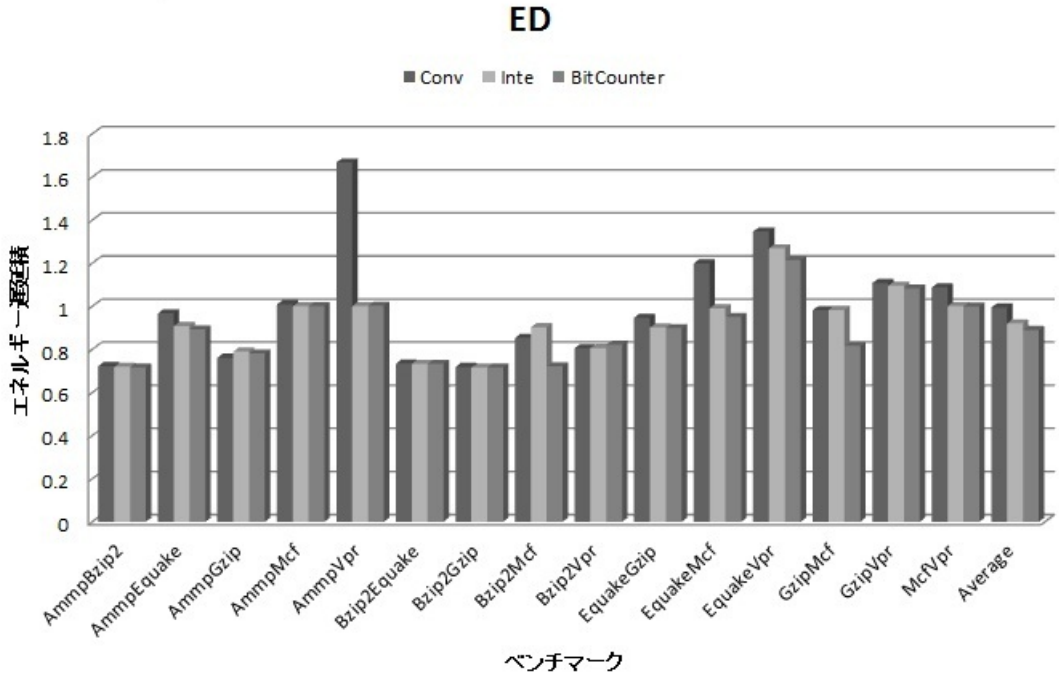

図 5.17: エネルギー遅延積

5.4.2 コントローラ評価

改良案 1 のモード切換コントローラのハードウェア規模および消費エネルギーの評価を，$Rohm$0.18 μ m テクノロジを用いて行った．コントローラの設計には Verilog HDL，論理合成には Synopsys Design Compiler，消費エネルギー評価には Synopsys PrimeTimePX を使用した．評価の結果，

表 5.4: 切換コントローラとキャッシュ(512KB) の面積・消費エネルギー

	キャッシュ(512KB)	切換コントローラ
面積	2580 μ m^2	2 μ m^2
消費エネルギー	307175 μ $Watt$	30.9 μ $Watt$

改良案 1 の切換コントローラのハードウェア量，消費エネルギーともに 512KB のキャッシュと比較して共に 3 桁以上の差があり，非常に小さく，無視できるレベルであることがわかった．なお，改良案 2 の切換コント

ローラのハードウェア評価は，改良案 1 のコントローラと差異が少なく，改良案 1 の結果から改良案 2 のハードウェア量も非常に小さくなることがわかるため，今回評価を行っていない．

6　結論

本論文では，キャッシュリークエネルギー削減手法である可変レベルキャッシュをマルチプロセッサ環境に適用した．その結果，可変レベルキャッシュの従来のモード切換手法ではベンチマークの半数において実行時間での性能悪化が見られた．そこで，新たなモード切換手法として2手法提案を行った．1つめの手法として，マルチプロセッサの各プロセッサのキャッシュミス率を求め，全プロセッサがキャッシュ容量を必要としないときのみ低消費エネルギーモード，必要としたときのみ通常モードへモードを遷移する手法を提案し，2つめの手法として，改良案1に対して，通常モード→低消費エネルギーモード→通常モードというモード切換直後のモード切換による書き戻しペナルティを低減するためにビットカウンタを挿入し，不用意なモード切換を防ぐ手法を提案した．その結果，従来の可変レベルキャッシュのモード切換よりも改良案2においてエネルギー遅延積で平均11%程度改善が見られた．また，同時に今まで詳細評価を行っていなかったモード切換コントローラの詳細設計を行い，切換コントローラの面積，消費エネルギーの評価を行った結果，両手法の面積，消費エネルギーともに非常に小さく無視できるレベルであることがわかった．今後の展望としては実際のハードウェアを設計し，より詳細な評価を行う，又はCache Partitioningと可変レベルキャッシュを組み合わせた評価などが考えられる．

謝辞

本研究を行うにあたり，多くの助言をいただきました近藤利夫教授，大野和彦講師，並びにご指導，ご助言いただきました下さいました佐々木敬泰助教に深く感謝いたします．また，様々な局面にてお世話になりました計算機アーキテクチャ研究室の皆様にも心より感謝いたします．

参考文献

[1] 恩賀琢也, 佐々木 敬泰, 大野 和彦, 近藤 利夫, “キャッシュ階層動的切り替えによる低消費電力化” , 情処学研報,2007-ARC-174,pp.115-120. August 2007

[2] 松原 伸幸，佐々木 敬泰，大野 和彦，近藤 利夫，“高性能かつ低消費電力を実現する可変レベルキャッシュのモード切換アルゴリズムの改良と評価 “ ，信学会技報，CPSY2009-44, pp.7-12，December 2009

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

[4] 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,

[5] 小寺　功，江川隆輔，滝沢寛之，小林広明，“ウェイアロケーション型共有キャッシュ機構の性能評価，” ，情処研報，2010-ARC-190 No.12, pp.1-8，2010.

[6] 田中秀和，井上弘士，モシニャガ・ワシリー，“低消費電力を目的とした適応型ウェイ予測キャッシュとその評価，” 信学会技報, VLD2004-139, ICD2004-235, pp.13-18, March 2005.

[7] Suh, G.E., Rudolph, L. and Devadas, “Dynamic Partitioning of Shared Cache Memory“, Journal of Supercomputing, Vol.28,No.1,pp.7-26, 2004

[8] Kobayashi, H., Kotera, I. and Takizawa, H.: Locality analysis to control dynamically way-adaptable caches, SIGARCH Comput. Archit. News, Vol.33, No3, pp.25-32(2005)

[9] S.H. Yang, M.D. Powell, B. Falsafi, K. Roy, and T.N. Vijaykumar, “An Integrated Circuit / Architecture Approach to Reducing Leakage in Deep-Submicron High-Performance I-Caches,” Proc. of the 7th Int. Symp. on High-Performance Computer Architecture, pp.147-157, February 2001.

[10] Ying Zheng，Brian T. Davis，Matthew Jordan“ Performance Evaluationof Exclusive Cache Hierarchies ”lEEE International Symposium of Performance Analysis of Systems and Software，ISPASS，pp.89-96, September 2004.

[11] Advanced Micro Deices，AMD, http://www.amd.com/us-en/. (CurrentJune 2003).

[12] 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論文集, pp.235-240, April 2004.

[13] 図子純平，冨山宏之，高田広章，井上弘士，“Drowsy キャッシュにおけるモード切替アルゴリズムの評価，” 情処学研報, 2006-ARC-170, pp.37-41, December 2006.

[14] 城田 幸利，佐々木 敬泰，大野 和彦，近藤 利夫，“可変レベルキャッシュ用モード切換手法のマルチコア環境への適用と評価 “ ，情処研報，2010-ARC-190 No.12, pp.1-8，2010.

[15] CACTI 5.3 Shyamkumar Thoziyoor, Naveen Muralimanohar, Jung Ho Ahn, and Norman P. Jouppi

[16] 藤枝直輝, 渡邉伸平, 吉瀬謙二, ”SimMips：教育・研究に有用な Linux が動く５０００行の MIPS システムシミュレータ” , 情報処理学会シンポジウム論文集, Y0978B, pp.143-150, December 2009

[17] 塩谷 亮太, 五島 正裕, 坂井 修一, 先進的計算基盤システムシンポジウム, SACSIS2009, pp.120-121，2009.

[18] “SPEC -Standard Performance Evaluation Corporation-,” URL: http://www.spec.org/.