Vol.2013-ARC-206 No.21
2013/8/1

ブロックバイパス機構による キャッシュのエネルギ効率化に関する研究

高井 拓実[1]　佐藤 雅之[2,3]　江川 隆輔[2,3]　滝沢 寛之[1,3]　小林 広明[2,3]

概要：エネルギ効率の高いキャッシュメモリの実現を目的として，動的キャッシュリサイズ機構が提案されている．動的キャッシュリサイズ機構は，アプリケーションが必要とするキャッシュ容量に応じて分割されたキャッシュ領域への電力供給を制御することで，性能を維持しつつキャッシュの消費エネルギを削減できる．しかし，動的キャッシュリサイズ機構によって有効化されるキャッシュ領域の大部分が，再利用されないデータに占有される場合がある．このため，キャッシュ中の再利用されないブロックを削減することで，有効化されたキャッシュ領域を更に削減できる可能性がある．本報告では，再利用されないブロックを削減するためのキャッシュバイパス手法を提案する．提案手法では，再利用されないブロックがキャッシュを占有する割合に応じて，キャッシュをバイパスする頻度を変更する．これにより，キャッシュ中の再利用されないブロックを削減し，動的キャッシュリサイズ機構によって有効化されるキャッシュ領域をさらに削減することが可能である．評価により，提案手法はキャッシュの消費エネルギを最大48.3%，平均 5.9%削減可能であることを明らかにした．

1. はじめに

近年，トランジスタの微細化による集積密度とリーク電流の増加 [1] に伴い，マイクロプロセッサの消費エネルギの増加が問題になっている．特に，キャッシュメモリ（以下，キャッシュ）は，マイクロプロセッサの性能向上に重要な役割を担っているが，マイクロプロセッサの消費エネルギに占めるキャッシュの消費エネルギは大きい [2]．したがって，マイクロプロセッサの消費エネルギ削減のためには，キャッシュのエネルギ効率の向上が非常に重要である．

キャッシュのエネルギ削減を目的として，動的キャッシュリサイズ機構が提案されている．動的キャッシュリサイズ機構は，キャッシュを複数の領域に分割し，アプリケーションが性能維持に必要とするキャッシュ領域へ電力を供給して有効化する．また，性能維持に不要な領域への電力は遮断して無効化する．この電源管理によって，キャッシュが必要以上の電力を浪費せず，性能を維持しつつキャッシュのエネルギ削減が可能である．

しかし，キャッシュ領域の大部分が，キャッシュに保存されてから追い出されるまでに 1 度も再利用されないデータブロック（デッドオンフィルブロック）に占有される場合がある．この場合，再利用されないブロックを保持するためにエネルギが浪費され，動的キャッシュリサイズ機構のエネルギ削減効果が抑制される．

本報告では，動的キャッシュリサイズ機構によるさらなる消費エネルギの削減を目的とし，デッドオンフィルブロック削減のためのブロックバイパス機構を提案する．本手法では，キャッシュにデッドオンフィルブロックを保存せず，バイパスを行う．これにより，デッドオンフィルブロックに占有されていたキャッシュ領域の電源を遮断し，キャッシュの消費エネルギをさらに削減することが可能である．

本報告の構成は以下の通りである．本章では研究背景と目的について述べた．第 2 章では関連研究について述べる．第 3 章ではデッドオンフィルブロック削減のためのブロックバイパス機構を提案し，その動作について述べる．第 4 章で提案手法の評価を行う．第 5 章で本報告をまとめる．

2. 関連研究

2.1 キャッシュバイパス機構

キャッシュヒット率の向上を目的として，再利用されないブロックをバイパスする手法が数多く提案されている．Kharbutli ら [3] は，過去のブロックの参照履歴を基にブロックの置換とバイパスを行う機構を提案している．本手法では，ブロックアドレスとそのブロックを参照した命令

[1] 東北大学 大学院情報科学研究科
[2] 東北大学サイバーサイエンスセンター
[3] JST CREST

のプログラムカウンタを索引として，ブロックの参照履歴を記録する．参照の空間的局所性により，キャッシュで再利用されたブロックと空間的に近いブロックは，キャッシュで再利用される可能性が高い．また，ある命令によってキャッシュに保存されたブロックが再利用された場合，同一の命令によって保存された他のブロックもキャッシュで再利用される可能性が高い．これらの性質から，同一の索引を持つブロックは，キャッシュで再利用されるか否かが類似していると考えられる．これにより，デッドオンフィルブロックの予測が可能になる．しかし，参照履歴の保存のために，キャッシュに対して約 7.9%の追加のハードウェアオーバーヘッドが必要である．したがって，キャッシュのエネルギ効率の向上には適していないと考えられる．

Nam ら [4] は *Protecting Distance based Policy*（PDP）を提案している．PDP は，ブロックが保存されてから再利用されるまでの，そのブロックが属するキャッシュセットへの参照回数（参照間隔）を予測し，参照間隔が短いブロックを優先的に保存する．キャッシュセット中に保存されている全てのブロックが参照間隔に達していない場合，新しいブロックはキャッシュをバイパスする．しかし，参照間隔を予測するために，追加のバッファを用いて参照間隔毎のヒット数の集計を行う必要がある．このオーバーヘッドによって追加のエネルギ消費が発生するため，キャッシュのエネルギ効率の向上には適さないと考えられる．

さらに，有効なキャッシュ容量が動的に変化する場合の参照間隔の予測方法はこれまでに提案されていないため，動的キャッシュリサイズ機構との組み合わせは困難である．

2.2 バイパス以外のデッドオンフィルブロック削減手法

デッドオンフィルブロックによるキャッシュ占有の抑制を目的として，多くの置換ポリシが提案されている．Qureshi ら [5] は，ミス数に基づいて LRU 置換ポリシと *Bimodal Insertion Policy*（BIP）を動的に切り替える *Dynamic Insertion Policy*（DIP）を提案している．BIP は，大部分のブロックを LRU に挿入し，一部のブロックを MRU に挿入することで，多くのブロックを早期に追い出す．このため，デッドオンフィルブロックが早期に追い出され，デッドオンフィルブロックによるキャッシュ占有が抑制される．

Jaleel ら [6] は，ブロックの参照間隔を *near-immediate*，*intermediate*，*long*，*distant* の 4 種類に分類する *Re-Reference Interval Prediction*（RRIP）を提案している．RRIP では，新しく保存するブロックの参照間隔は *long* と予測され，比較的早期に追い出されるため，デッドオンフィルブロックによるキャッシュ占有を抑制することが可能である．さらに，ブロック毎に付与されるビット数が削減されるため，LRU 置換ポリシよりもオーバーヘッドが小さい．

表 1 シミュレーションパラメータ

パラメータ	値
コア	2 GHz，1 コア
L1 命令キャッシュ	32 kB，2 ウェイ 64 B ブロック，1 サイクル
L1 データキャッシュ	32 kB，2 ウェイ 64 B ブロック，1 サイクル
L2 キャッシュ	256 kB，8 ウェイ 64 B ブロック，10 サイクル
L3 キャッシュ	1 MB，32 ウェイ 64 B ブロック，20 サイクル ウェイ適応型キャッシュ機構
メインメモリ	4 GB，200 サイクル

DIP と RRIP は性能向上を目的とする置換ポリシであるため，動的キャッシュリサイズ機構と組み合わせた評価は行われていない．したがって，これらの置換ポリシのエネルギ効率への影響は明らかになっていない．

2.3 動的キャッシュリサイズ機構

キャッシュのエネルギ効率の向上のために，一部のキャッシュ領域の電源を遮断する機構が提案されている．小林ら [7] は，各キャッシュウェイ毎に電源を管理し，性能維持に必要のないウェイの電源を遮断するウェイ適応型キャッシュ機構を提案している．性能維持に必要なウェイ数は *stack distance profiling* [8] による局所性評価量を基に算出する．参照の時間的局所性が高い場合，短い間隔でブロックが参照されるため，より多くのウェイの電源を遮断して無効化ウェイとする．一方，参照の時間的局所性が低い場合，長い間隔で参照が起きるため，長時間ブロックを保持する必要がある．このため，多くのウェイに電源を供給し，有効ウェイとする．これにより，性能維持に必要な最小のウェイ数のみを有効化することが可能である．

しかし，ウェイ適応型キャッシュ機構では，デッドオンフィルブロックによるキャッシュ占有が起きる場合がある．デッドオンフィルブロックによってキャッシュの大部分が占有されると，ウェイ適応型キャッシュ機構はデッドオンフィルブロックを保持するために多くのウェイを有効化するため，エネルギ効率が低下する．

予備実験として，動的キャッシュリサイズ機構を用いるキャッシュ中のデッドオンフィルブロックの割合を調査する．実験には，gem5 シミュレータ [9] とキャッシュの電力モデル CACTI version 6.5 [10] を用いる．プロセッサのシミュレーションパラメータを表 1 に示す．L3 キャッシュには，動的キャッシュリサイズ機構 [7] を採用する．また，全てのキャッシュのデータ管理には LRU 置換ポリシを用いる．ベンチマークは，SPEC CPU2006 ベンチマーク集 [11] を利用する．

アプリケーション実行中の，L3 キャッシュを占有するブ

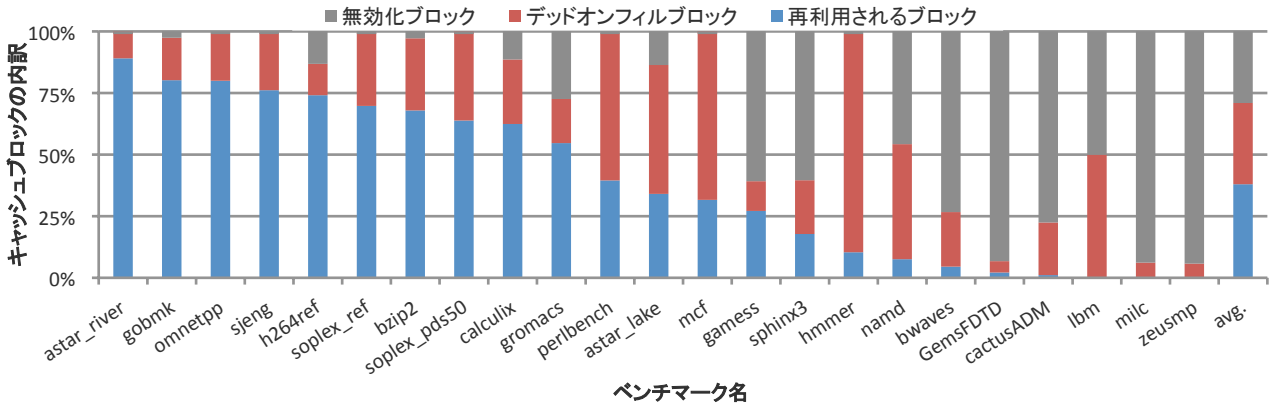

図 1 L3 キャッシュブロックの内訳

ロックの内訳を図 1 に示す．内訳は，再利用されるブロック，デッドオンフィルブロック，無効化ブロックに分類されている．無効化ブロックは動的キャッシュリサイズ機構によって電源が遮断されている領域に相当するブロックである．図 1 から，キャッシュ領域全体の内，平均で 33.0%，最大 88.6%がデッドオンフィルブロックによって占有されていることがわかる．無効化ブロックは平均 29.0%であるため，動的キャッシュリサイズ機構を用いる場合，有効化されるキャッシュ領域の約半分がデッドオンフィルブロックに占有され，エネルギを浪費している．このため，デッドオンフィルブロックを削減することで，性能維持に必要なキャッシュ領域が減少し，キャッシュの消費エネルギを削減することが可能である．

3. ブロックバイパス機構

本研究は，デッドオンフィルブロックによるキャッシュ占有を抑制し，動的キャッシュリサイズ機構を用いるキャッシュのエネルギ効率を高めることを目的としている．

これまでに提案されているバイパス機構は，ブロックの参照間隔の予測により，デッドオンフィルブロックを削減することが可能である．しかし，参照間隔の予測のために必要なハードウェアオーバーヘッドが大きいため，キャッシュのエネルギ削減には適していないと考えられる．また，DIP や DRRIP などの置換ポリシは，デッドオンフィルブロックの早期追い出しが可能である．しかし，これらの置換ポリシではバイパスを行わないため，全てのデッドオンフィルブロックがキャッシュに保存される．したがって，最適なバイパスを行った場合と比較すると，デッドオンフィルブロックの削減量が小さいと考えられる．

本報告では，ハードウェアオーバーヘッドの小さいバイパス機構の提案を目的とし，ブロックをバイパスする頻度を動的に調整する機構を提案する．本手法により，デッドオンフィルブロックをバイパスすることで，動的キャッシュリサイズ機構のエネルギ効率を向上させることが可能である．

3.1 提案手法の概要

本章では，デッドオンフィルブロックによるキャッシュ占有を抑制するブロックバイパス機構を提案する．本手法は，ハードウェアオーバーヘッドを抑制するために，個々のブロックがデッドオンフィルブロックか否かを予測せず，デッドオンフィルブロックのキャッシュ中の占有率に基づき，バイパスする頻度を調整する．頻度が高い場合，再利用されるブロックを誤ってバイパスし，キャッシュミスを増加させる可能性が高くなる．そこで，再利用されるブロックを誤ってバイパスしても問題ない程度にバイパス頻度を抑制する．これにより，デッドオンフィルブロックを削減しつつ，再利用されるブロックのバイパスを抑制する．

ブロックのバイパス頻度は，デッドオンフィルブロックのキャッシュ中の占有率に基づき判断することが可能である．デッドオンフィルブロックの占有率が高い場合，最近保存されたブロックの大部分はデッドオンフィルブロックであると考えられる．したがって，今後キャッシュに保存されるブロックのほとんどがデッドオンフィルブロックであると予測することができる．

図 2 に提案手法の概要を示す．図 2 において，ラストレベルキャッシュには，ウェイ適応型キャッシュ機構が適用されている．提案するバイパス機構は *Bypass Frequency Controller*（BFC）と *Insertion Counter Array*（ICA）から構成される． BFC は，再利用されたブロックによるキャッシュ占有率を算出し，バイパスの頻度を決定する．ICA は，BFC によって決定された頻度に基づいてバイパスを行う．

3.2 バイパス頻度調整機構

3.2.1 キャッシュ占有率のモニタリング

バイパスの頻度を調整するための指標として，再利用されたブロックによるキャッシュ占有率を用いる．式 (1) にその算出式を示す．

$$FRAC_{reused} = \frac{N_{reused}}{N_{act}}. \quad (1)$$

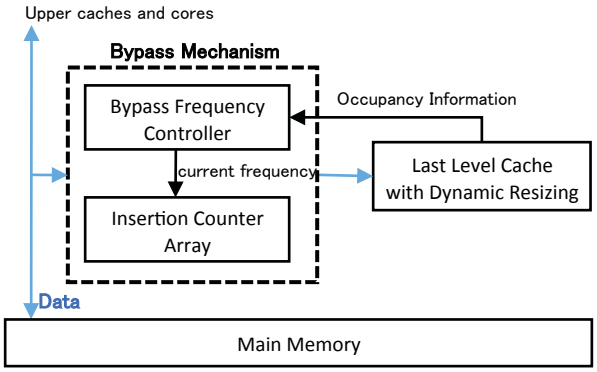

図 2 提案手法の概要

式 (1) において，$FRAC_{reused}$ は再利用されたブロックによるキャッシュ占有率である．N_{reused} はキャッシュ中の再利用されたブロックの数，N_{act} はウェイ適応型キャッシュ機構で有効化されたブロックの数を表す．$FRAC_{reused}$ により，キャッシュの有効化されたブロックに占める再利用されたブロックの割合がわかる．$FRAC_{reused}$ の値が大きい場合，最近保存されたブロックの大部分が再利用されているため，今後保存されるブロックの大部分が再利用されるブロックであると考えられる．この場合，バイパスは行うべきではないため，バイパスの頻度を低くする．一方，$FRAC_{reused}$ の値が小さい場合，最近保存されたブロックの大部分がデッドオンフィルブロックであるため，今後保存されるブロックも，その大部分がデッドオンフィルブロックであると考えられる．この場合，より多くのブロックをバイパスすることでデッドオンフィルブロックを削減できるため，バイパスの頻度を高くする．

提案手法では，$FRAC_{reused}$ の算出のために再利用されたブロックの数を観測する必要がある．そのため，キャッシュ中の各ブロックに 1 ビットの *referenced* ビットを付加する．*referenced* ビットは，キャッシュにブロックが保存される時に 0 にセットされ，そのブロックがキャッシュで再利用されると 1 にセットされる [12]．BFC には，再利用されたブロックの数を集計するためのカウンタを用意する．キャッシュへの参照とキャッシュからのブロックの追い出しを監視し，参照されたブロックの *referenced* ビットが 0 から 1 になった場合は，再利用されたブロックが 1 つ増えるので，カウンタをインクリメントする．一方，キャッシュから追い出されるブロックの *referenced* ビットが 1 の場合，再利用されたブロックが 1 つ減るので，カウンタをデクリメントする．これにより，カウンタの値が現在キャッシュ中に存在する再利用されたブロックの数と一致する．

3.2.2 モニタリングに基づくバイパス頻度の調整

$FRAC_{reused}$ に基づくバイパス頻度の調整は，一定時間間隔で行う．一定時間毎に BFC の再利用されたブロックの数を集計しているカウンタを参照して式 (1) の N_{reused} とする．N_{act} は，キャッシュセット数とウェイ適応型キャッシュ機構が有効化しているウェイ数の積で求められる．閾値 $T_{dec}, T_{inc}(T_{dec} > T_{inc})$ を導入し，算出された $FRAC_{reused}$ の値が $FRAC_{reused} > T_{dec}$ の場合，バイパス頻度を低くする．一方，$FRAC_{reused} < T_{inc}$ の場合，バイパス頻度を高くする．$T_{inc} \leq FRAC_{reused} \leq T_{dec}$ の場合は現在の頻度を維持する．

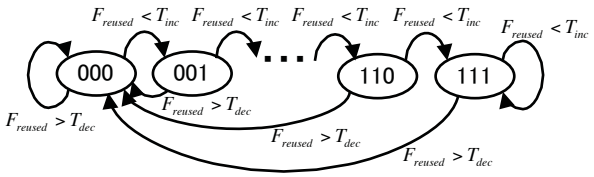

図 3 3 ビット非対称ステートマシン

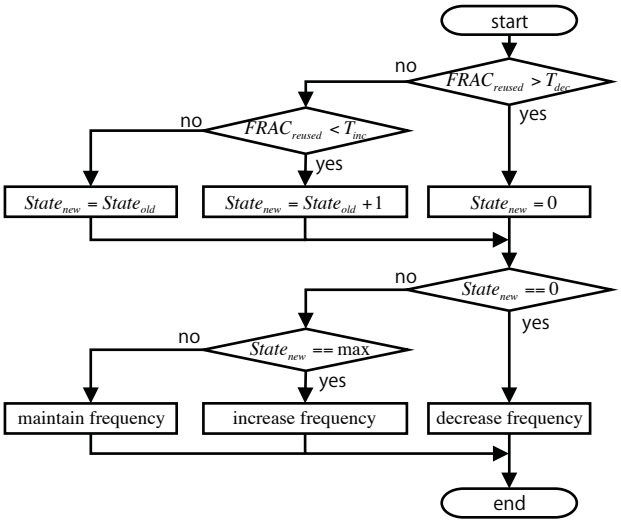

図 4 ステートとバイパス頻度の調整方法

しかし，提案手法では個々のブロックがデッドオンフィルブロックであるか否かの予測は行わないため，再利用されるブロックのバイパスを引き起こす恐れがある．したがって，バイパス頻度を高める判断は消極的に行う必要がある．そこで，最終的な頻度の変更判断に非対称ステートマシンを利用する．3 ビット非対称ステートマシンの状態遷移図を図 3 に示す．3 ビットカウンタで表される 8 種類の状態のうち，カウンタが 000 の状態になるとバイパス頻度を低くする．また，111 の状態になるとバイパス頻度を高くする．この非対称ステートマシンを用いることで，バイパス頻度を高くする判断を消極的に行い，再利用されるブロックのバイパスによる性能への影響を防ぐことが可能である．

$FRAC_{reused}$ の値と非対称ステートマシンに基づく，BFC によるバイパス頻度の調整アルゴリズムを図 4 に示す．BFC では，一定時間毎に図 4 のアルゴリズムによってバイパス頻度を更新する．

3.3 キャッシュ挿入のカウンタ配列

バイパスによってキャッシュのエネルギ効率を向上させるためには，単純な機構を用いて複数の頻度のバイパスを実現する必要がある．提案手法では，各キャッシュセットにカウンタを設けることで，ブロックのキャッシュへの保存を周期的にバイパスする．また，周期を変更することによって頻度の調整を実現する．このカウンタの配列が ICA である．

以下，ICA の詳細を示す．ICA は各キャッシュセットでのミス発生時にカウンタをインクリメントし，カウンタの値に応じてキャッシュへの保存とバイパスの切り替えを行う．X 回のミスによってキャッシュに保存されるブロックのうち 1 回をバイパスする場合を考える．このバイパス頻度を $1/X$ と表す事とする．キャッシュセットでミスが発生し，ブロックをキャッシュに保存する時に，そのセットに対応するカウンタの値をインクリメントする．キャッシュへの保存を繰り返し，カウンタの値が $X-1$ になった後，次にミスが発生した時にバイパスを行い，カウンタの値を 0 に戻す．これにより，X 回の保存のうち 1 回がバイパスされる．各カウンタが n ビットの場合，この手法によって $1/2$，$1/2^2$，$1/2^3$，$\cdots$，$1/2^n$ の頻度でバイパスを行うことが可能である．BFC で決定されたバイパス頻度が $1/2$ より大きい場合，カウンタの値が $X-1$ より小さい時にバイパスを行い，$X-1$ の時に保存を行う．これにより，$(X-1)/X$ の頻度でブロックをバイパスできる．これにより，より多くのバイパスの頻度を設定することが可能である．各カウンタが n ビットの場合，上記のバイパス頻度に加えて，$(2^2-1)/2^2$，$(2^3-1)/2^3$，$\cdots$，$(2^n-1)/2^n$ の頻度でバイパスを行うことが可能である．

このように，キャッシュセット毎にカウンタを設けることで，BFC で決定されたバイパス頻度に基づく周期的なバイパスを行う．

4. 性能評価

4.1 評価環境

シミュレーションにより，提案手法の有効性を評価する．実験条件は，第 2.3 節の予備実験における条件と同様である．L3 キャッシュにおいて，提案手法によりキャッシュバイパスを行う．提案手法の ICA は各キャッシュセットに対して 4 ビットとし，バイパス頻度は 0，1/8，1/4，1/2，3/4，7/8，15/16 の 7 通りで調整を行う．BFC には 3 ビット非対称ステートマシンを付加する．$FRAC_{reused}$ の閾値 T_{dec}，T_{inc} はそれぞれ 0.4，0.3 とする．これらは実験により経験的に求められた値である．

再利用されるブロックによるキャッシュ占有率と，再利用されるブロックへの平均参照回数を用いて，ベンチマークのカテゴリ分けを行う．各カテゴリのベンチマークを表 2 に示す．再利用されるブロックによるキャッシュ占有率が低いほど，提案手法によってバイパスの頻度が高く設定されるため，多くのブロックのキャッシュへの保存がバイパスされると考えられる．したがって，occ-H カテゴリよりも occ-L カテゴリの方が提案手法による消費エネルギ削減効果が大きいと予想される．再利用されるブロックへの平均参照回数は，デッドオンフィルを除く各ブロックが，キャッシュに保存されてから追い出されるまでの間に参照される平均回数を表す．この回数が多いほど，再利用されるブロックをバイパスした時の性能への影響が小さいと考えられる．したがって，ref-L カテゴリよりも ref-S カテゴリの方が提案手法によるミス数増加量が大きいと予想される．以下，occ-H/ref-L，occ-H/ref-S，occ-L/ref-L，occ-L/ref-S の 4 種類のカテゴリのベンチマークで評価を行う．

表 2 ベンチマークの分類

再利用されるブロックへの平均参照回数	再利用されるブロックによるキャッシュ占有率 *High*（occ-H）	*Low*（occ-L）
Large（ref-L）	astar_river，bzip2 calculix，h264ref perlbench soplex_pds50	hmmer lbm namd sphinx3
Small（ref-S）	gobmk gromacs omnetpp spolex_ref	GemsFDTD，astar_lake bwaves，cactusADM gamess，mcf，milc sjeng，zeusmp

4.2 評価結果

4.2.1 非対称ステートマシンの評価

まず，非対称ステートマシンの効果について評価を行う．3 ビット非対称ステートマシンを用いる場合と用いない場合の提案手法によるミス数の変化を図 5 に示す．図 5 において，ミス数は提案手法を用いない場合のミス数によってカテゴリ毎に正規化されている．ステートマシンを用いない場合，occ-H/ref-L カテゴリ，occ-H/ref-S カテゴリでそれぞれ 69.4%，25.2%のミス数増加が見られる．一方，ステートマシンを用いる場合は，occ-H/ref-L カテゴリ，occ-H/ref-S カテゴリでそれぞれ 2.5%，2.2%のミス数が増加する．したがって，ステートマシンを用いることでバイパス頻度を高める判断が消極的に行われ，再利用されるブロックを誤ってバイパスすることによるミス数の増加が抑制される．

特に，occ-H/ref-L カテゴリの *h264ref* では，ステートマシンを用いない場合にミス数が大きく増加するが，ステートマシンを用いる場合はミス数は増加しない．各場合における，*h264ref* 実行時間中の再利用されるブロックによるキャッシュ占有率の変化をそれぞれ図 6，図 7 に示す．実行時間が $0ms$ から $300ms$ の間は，キャッシュ占有率が比

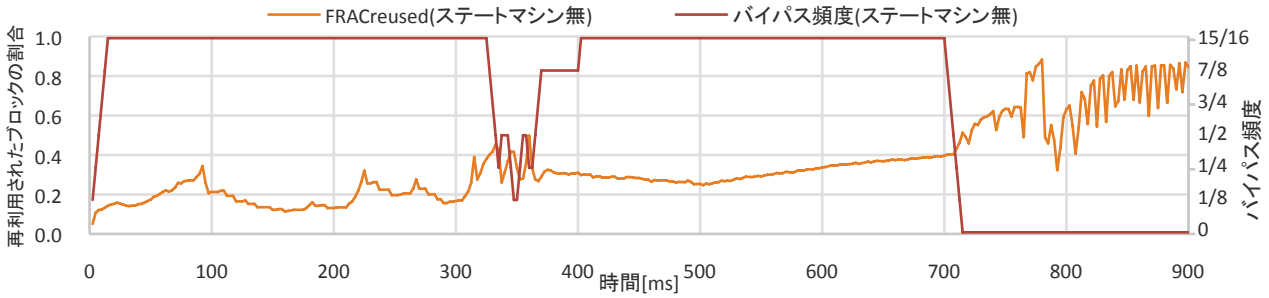

図 6 ステートマシンを用いない場合の $FRAC_{reused}$ とバイパス頻度の変化（*h264ref*）

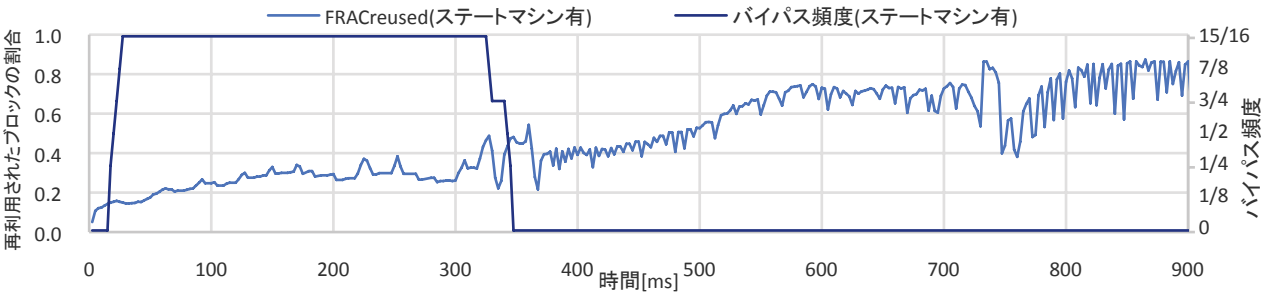

図 7 ステートマシンを用いる場合の $FRAC_{reused}$ とバイパス頻度の変化（*h264ref*）

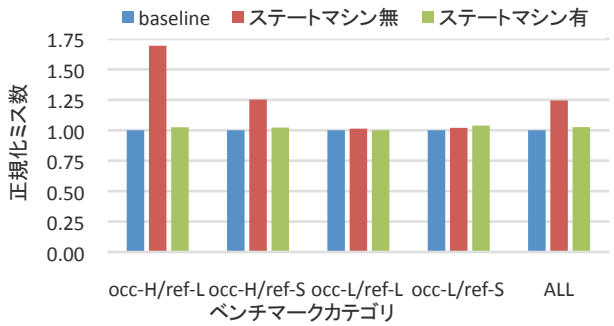

図 5 ステートマシンの有無によるミス数の変化

較的低いため，多くのブロックがバイパスされる．その後，実行時間が 320ms 付近でキャッシュ占有率が大きく変動する．この時，ステートマシンが無い場合はバイパス頻度の変化が繰り返し発生する．その結果，320ms 以降では再利用されるブロックが増加するにも関わらずバイパス頻度が高い状態となるため，ミス数の増加を招く．一方，ステートマシンを用いる場合，20ms から 300ms ではバイパス頻度が 15/16 である．しかし，320ms でのキャッシュ占有率の変動が起きると，ステートが 0 になる．その結果，バイパス頻度が低くなり，320ms 以降ではバイパスがほとんど行われない．これにより，ミス数の増加が抑制される．

一方で，occ-L/ref-S カテゴリの *gamess* では，ステートマシンを用いるとミス数が増加する．この原因として，*gamess* の L3 キャッシュへの参照パターンが挙げられる．この参照パターンでは，本来再利用されるブロックがキャッシュ容量の不足により，再利用される前にキャッシュから追い出される．このとき，高い頻度でバイパスを行うことで，バイパスされなかった少数のブロックはキャッシュに長時間留まる．このようなブロックがキャッシュで参照されることで，バイパスを行わない場合と比較してミス数が削減される．このため，ステートマシンを用いてバイパス頻度を高める判断を消極的に行うと，積極的に行う場合と比較してミス数が増加する．しかし，このような事例はごく少数であり，非対称ステートマシンを用いる方がより多くのアプリケーションにおいてミス数の増加を抑制することができる．

以上より，非対称ステートマシンを用いることで，バイパスによるミス数の増加を抑制可能であることが明らかになった．

4.2.2 消費エネルギ評価

次に，キャッシュの消費エネルギの評価を行う．提案手法を用いる場合の L3 キャッシュの消費エネルギを図 8 に示す．消費エネルギは提案手法を用いない場合の値でカテゴリ毎に正規化されている．また，ステートマシンが 2 ビット，3 ビット，4 ビットの 3 通りの提案手法を評価する．図 8 から，全ベンチマーク平均では 2 ビット，3 ビット，4 ビットステートマシンを用いる場合でそれぞれ 5.1%，5.9%，3.8%消費エネルギが削減されている．特に結果が良好であった 3 ビットステートマシンを用いる場合，occ-H/ref-L，occ-H/ref-S，occ-L/ref-L，occ-L/ref-S カテゴリでそれぞれ 0.8%，0.9%，28.6%，2.5%消費エネルギが削減されている．occ-L/ref-L カテゴリの *lbm* では消費エネルギが 48.3%削減され，最も消費エネルギ削減効果が大きい．

再利用されたブロックによるキャッシュ占有率が高い場

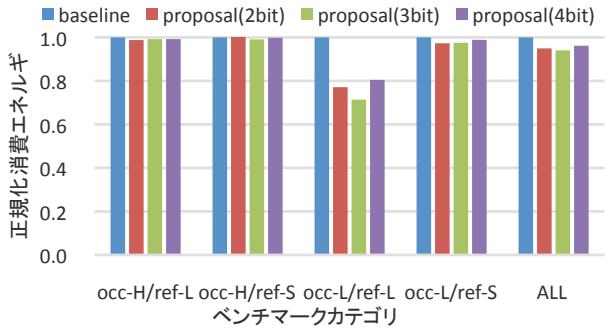

図 8 提案手法による消費エネルギの変化

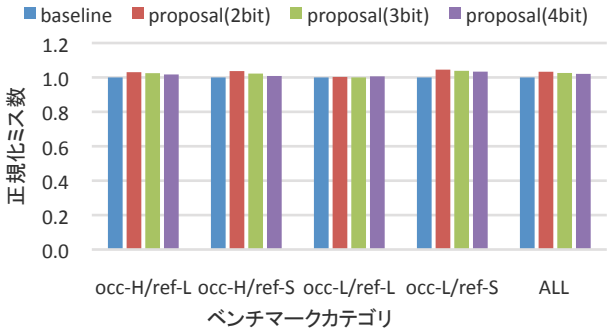

図 9 提案手法によるミス数の変化

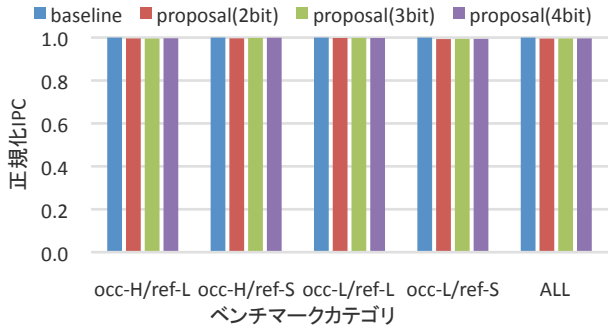

図 10 提案手法による IPC の変化

合，バイパス頻度が低くなるため，提案手法によってバイパスされるブロックが非常に少ない．このため，occ-H/ref-L カテゴリと occ-H/ref-S カテゴリでは，提案手法による消費エネルギ削減効果が小さい．一方，occ-L/ref-L カテゴリ，occ-L/ref-S カテゴリでは多くのブロックがバイパスされるため，ウェイ適応型キャッシュ機構によって有効化されるウェイ数が減少し，消費エネルギが削減される．occ-L/ref-L カテゴリと occ-L/ref-S カテゴリを比較すると，occ-L/ref-L カテゴリの方が消費エネルギが大きく削減される．occ-L/ref-S カテゴリに含まれるベンチマークの多くは，L3 キャッシュをほとんど参照しない．このようなベンチマークでは，ウェイ適応型キャッシュ機構によって有効ウェイ数が限界まで削減される．したがって，提案手法によってバイパスを行っても有効ウェイ数が減らないため，キャッシュの消費エネルギを削減することができない．この結果から，エネルギ削減の余地が大きい occ-L/ref-L カテゴリでは提案手法が適切に消費エネルギを削減できることが示された．

4.2.3 IPC 評価

2 ビット，3 ビット，4 ビットの非対称ステートマシンを用いる提案手法のミス数を図 9 に示す．ミス数は提案手法を用いない場合の値でカテゴリ毎に正規化されている．図 9 から，occ-L/ref-L カテゴリ以外では提案手法によってミス数が増加することがわかる．最も大きくミス数が増加するカテゴリは occ-L/ref-S であり，2 ビット，3 ビット，4 ビットステートマシンを用いる提案手法でそれぞれ 4.5%，3.8%，3.4%ミス数が増加する．第 4.2.2 節で述べたように，occ-L/ref-S カテゴリでは，再利用されたブロックによるキャッシュ占有率が低いため，提案手法によるバイパス頻度が高くなり，多くのブロックがバイパスされる．しかし，再利用されるブロックへの平均参照回数が少ないため，再利用されるブロックのバイパスによるミスの増加の影響が大きい．全ベンチマーク平均では，2 ビット，3 ビット，4 ビットステートマシンを用いる場合，それぞれ 3.3%，2.6%，2.1%ミス数が増加する．

ミス数の増加による性能への影響を評価する．2 ビット，3 ビット，4 ビットの非対称ステートマシンを用いる場合の IPC を図 10 に示す．IPC は提案手法を用いない場合の IPC の値でカテゴリ毎に正規化されている．図 10 から，全てのカテゴリで IPC の変化は非常に小さい．3 ビットステートマシンを用いる場合，全ベンチマーク平均で 0.4%IPC が低下する．したがって，提案手法はミス数を若干増加させるが，全てのカテゴリにおいて性能への影響は微少である．

4.2.4 キャッシュ占有率の評価

再利用されるブロック，デッドオンフィルブロック，無効化されたブロックが，それぞれキャッシュに占める割合の時間平均を図 11 に示す．occ-H/ref-L カテゴリと occ-H/ref-S カテゴリでは，提案手法を用いてもバイパスされないため，デッドオンフィルブロックの割合はほとんど変化しない．occ-L/ref-L カテゴリでは，ステートマシンが 2 ビット，3 ビット，4 ビットの場合に，それぞれ 68.8%，73.0%，71.8%のデッドオンフィルブロックが削減される．その結果，無効化ブロックがそれぞれ 1.85 倍，1.94 倍，1.75 倍に増加する．無効ブロックが大幅に増加することで，occ-L/ref-L カテゴリではキャッシュの消費エネルギが大きく削減される．occ-L/ref-S カテゴリでは，デッドオンフィルブロックがそれぞれ 18.9%，18.9%，13.6%削減される．その結果，無効化ブロックはそれぞれ 6.7%，7.5%，4.9%増加する．これにより，キャッシュの消費エネルギが削減される．全ベンチマーク平均では，デッドオンフィルブロックがそれぞれ 27.1%，28.5%，25.1%削減され，無効化ブロックがそれぞれ 26.2%，28.8%，21.4%増加する．

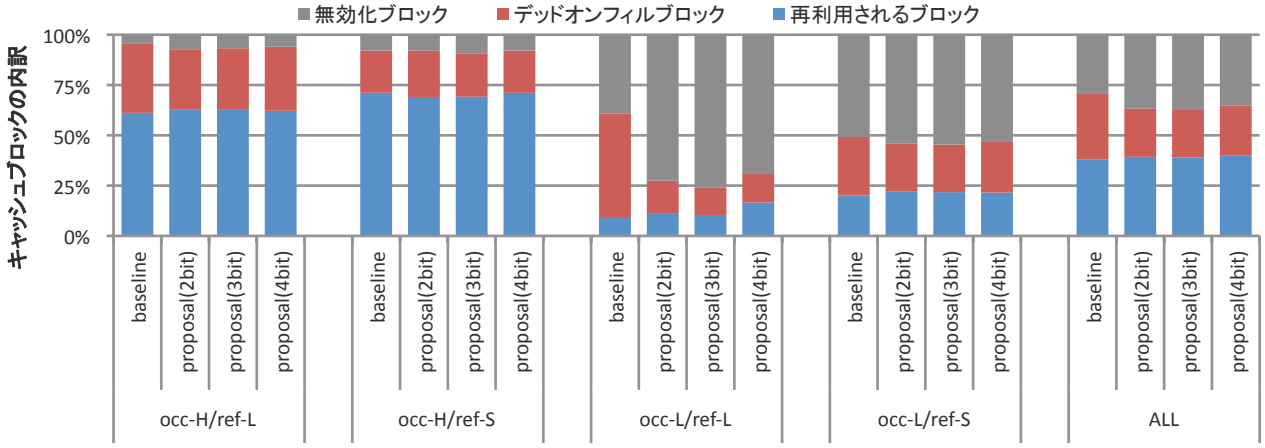

図 11 キャッシュブロックの内訳

以上より，提案手法を用いることで，デッドオンフィルブロックによるキャッシュの占有が抑制され，性能を維持しながらキャッシュの消費エネルギを削減可能であることが明らかになった．

4.2.5 ハードウェアオーバーヘッドの評価

提案手法に必要な追加のハードウェアの評価を行う．ラストレベルキャッシュの各ブロックには，1 ビットの *referenced* ビットが必要である．これにより，ラストレベルキャッシュの容量に対して約 0.2%のオーバーヘッドが生じる．BFC には，*referenced* ビットを集計するための 14 ビットカウンタと，2 ビットから 4 ビットの非対称ステートマシン，そして現在のバイパス頻度を保持する 5 ビットレジスタが必要である．また，$FRAC_{reused}$ の算出のための論理回路も必要である．ICA の追加ハードウェアは合計 256B である．これはラストレベルキャッシュに対して 0.02%と非常に小さい．第 2.1 節において示した 2 つのバイパス手法 [3] [4] のハードウェアオーバーヘッドは，ラストレベルキャッシュの容量に対してそれぞれ 7.9%，0.8%である．

以上より，提案手法のハードウェアオーバーヘッドは非常に小さく，その消費エネルギへの影響は無視できると考えられる．

5. おわりに

本報告では，デッドオンフィルブロックを削減することでキャッシュメモリのエネルギ効率を向上させることを目的とし，ブロックバイパス機構を提案した．本機構は，再利用されたブロックによるキャッシュ占有の割合に基づいてバイパスの頻度を動的に調整する．さらに，非対称ステートマシンを用いてバイパス頻度を抑制することで，性能を維持しつつキャッシュの消費エネルギを削減することが可能である．評価では，3 ビットの非対称ステートマシンを用いる場合，平均で 5.9%，最大 48.3%の消費エネルギ削減を達成し，提案手法がキャッシュのエネルギ効率の向上に有効であることを示した．

今後の課題は，提案手法のコストの詳細な評価を行うことが挙げられる．本報告では，提案手法に必要な追加ハードウェアを示した．しかし，*referenced* ビットのモニタリングと，$FRAC_{reused}$ 算出によって増加する消費エネルギは明らかになっていないため，今後評価する必要がある．また，過去の参照履歴に基づくデッドオンフィルブロックの予測機構との比較検討を行う．

謝辞 本報告の一部は，科学技術振興機構戦略的創造研究推進事業と，文部科学省科学研究費助成事業（課題番号 22300013，25280041，25280012，24650018）の助成による．

参考文献

[1] Kim, N., Austin, T., Baauw, D., Mudge, T., Flautner, K., Hu, J., Irwin, M., Kandemir, M. and Narayanan, V.: Leakage current: Moore's law meets static power, *Computer*, Vol. 36, No. 12, pp. 68–75 (2003).
[2] Segers, S.: Low Power Design Techniques for Mircoprocessors, *ISSCC 2001*, pp. 268–273 (2001).
[3] Kharbutli, M. and Solihin, Y.: Counter-based cache replacement and bypassing algorithms, *Computers, IEEE Transactions on*, Vol. 57, No. 4, pp. 433–447 (2008).
[4] Nam, D., Dali, Z., Taesu, K., Rosario, C., Mateo, V. and Alexander, V. V.: Improving cache management policies using dynamic reuse distances, *Proceedings of the 45th Annual IEEE/ACM International Symposium on Microarchitecture*, ACM, pp. 389–400 (2012).
[5] Qureshi, M. K., Jaleel, A., Patt, Y. N., Steely, S. C. and Emer, J.: Adaptive insertion policies for high performance caching, *SIGARCH Comput. Archit. News*, Vol. 35, No. 2, pp. 381–391 (2007).
[6] Jaleel, A., Theobald, K., Steely Jr, S. and Emer, J.: High performance cache replacement using re-reference interval prediction (RRIP), *ACM SIGARCH Computer Architecture News*, Vol. 38, No. 3, pp. 60–71 (2010).
[7] Kobayashi, H., Kotera, I. and Takizawa, H.: Locality Analysis to Control Dynamically Way-Adaptable Caches, *ACM SIGARCH Computer Architecture News*, Vol. 33, No. 3, pp. 25–32 (2005).
[8] Chandra, D., Guo, F., Kim, S. and Solihin, Y.: Pre-

dicting inter-thread cache contention on a chip multi-processor architecture, *High-Performance Computer Architecture, 2005. HPCA-11. 11th International Symposium on*, pp. 340–351 (2005).

[9] Binkert, N., Beckmann, B., Black, G., Reinhardt, S. K., Saidi, A., Basu, A., Hestness, J., Hower, D. R., Krishna, T., Sardashti, S., Sen, R., Sewell, K., Shoaib, M., Vaish, N., Hill, M. D. and Wood, D. A.: The gem5 simulator, *SIGARCH Comput. Archit. News*, Vol. 39, No. 2, pp. 1–7 (2011).

[10] Muralimanohar, N., Balasubramonian, R. and Jouppi, N.: CACTI 6.0: A tool to model large caches.

[11] Henning, J. L.: SPEC CPU2006 Benchmark Descriptions, *ACM SIGARCH Computer Architecture News*, Vol. 34, No. 4, pp. 1–17 (2006).

[12] Hallnor, E. G. and Reinhardt, S. K.: A fully associative software-managed cache design, *SIGARCH Comput. Archit. News*, Vol. 28, No. 2, pp. 107–116 (2000).