Energy Effciency of History-Based Tag-Comparison Cashe for Media Applocations

井上, 弘士
福岡大学工学部電子情報工学科

Moshnyaga, Vasily G.
Department of Electronics and Computer Science, Fukuoka University

村上, 和彰
九州大学大学院システム情報科学研究院

出版情報：電子情報通信学会技術研究報告, CPSY2002-1-11. 102 (27), pp.55-60, 2002-04.
IEICE(CPSY)
バージョン：
権利関係：

低消費電力メディア・アプリケーション向け ヒストリ・ベース・タグ比較キャッシュの評価

井上　弘士† 　VasilyG. Moshnyaga† 　村上　和彰††

† 福岡大学 工学部 電子情報工学科　〒 814-0133 福岡県福岡市城南区七隈 8–19–1
†† 九州大学大学院 システム情報科学研究院　〒 816-8580 福岡県春日市春日公園 6–1
E-mail: †inoue@tl.fukuoka-u.ac.jp, ††vasily@fukuoka-u.ac.jp, †††murakami@i.kyushu-u.ac.jp

あらまし　これまでに我々は，ダイレクト・マップ命令キャッシュの低消費エネルギー化を目的として，ヒストリ・ベース・タグ比較 (HBTC: History Based Tag-Comparison) 方式を提案した．従来型キャッシュでは，ヒット / ミス判定のために，タグ比較が毎アクセス実行される．これに対し，HBTC キャッシュでは，プログラムの実行履歴に基づき必要に応じてタグ比較を行う．そして，無駄なタグ比較処理を動的に検出・削除し，命令キャッシュの低消費エネルギー化を実現する．本稿では，これまでに提案した HBTC キャッシュを改良し，オーバヘッドの小さい新しい実現方式を示す．また，信号処理アプリケーションを中心としたベンチマーク・プログラムを用いて，性能ならびに消費エネルギーに関するより詳細な評価を行う．

キーワード　低消費電力，キャッシュ，タグ比較，実行履歴，動的最適化，再利用

Energy Efficiency of History-Based Tag-Comparison Cache for Media Applications

Koji INOUE†, Vasily G. MOSHNYAGA†, and Kazuaki MURAKAMI††

† Department of Electronics and Computer Science, Fukuoka University
8–19–1 Nanakuma, Jonan-Ku, Fukuoka, 814–0133 Japan
†† Department of Informatics, Kushu University
6–1, Kasuga-Koen, Kasuga, Fukuoka, 816–8580 Japan
E-mail: †inoue@tl.fukuoka-u.ac.jp, ††vasily@fukuoka-u.ac.jp, †††murakami@i.kyushu-u.ac.jp

Abstract　We have proposed a *history-based tag-comparison cache (HBL cache)* architecture for low-energy consumption. In conventional direct-mapped caches, tag checks are performed to examine whether the current reference hits the cache. On the other hand, the HBTC cache attempts to reduce the frequency of tag checks by exploiting execution footprints. In this paper, we improve the proposed HBTC cache, and evaluate performance/energy efficiency by using media applications. Simulation results show that such an approach can eliminate up to 95% of tag checks, saving the cache energy by 17%, while affecting the processor performance by 0.2%.

Key words　low power, cache, tag comparison, execution history, run-time optimization, reuse

1. はじめに

拡大を続けるプロセッサ–主記憶間の性能差を隠蔽するため，現在のプロセッサ・チップには当然のようにオンチップ・キャッシュ(以下，キャッシュ) が搭載されている．プログラム実行時，全てのメモリ参照はキャッシュを介して行われるため，キャッシュがシステム性能および消費エネルギーに与える影響は重大である．

更なるキャッシュ・ヒット率の向上を目的として，キャッシュ・サイズは年々増加傾向にある．しかしながら，それに伴い，キャッシュの消費エネルギーが増大し，システム全体の消費エネルギーに大きな影響を及ぼすようになってきた．例えば，チップ全体の消費電力 (単位時間当たりの消費エネルギー) のうち，DEC 21164 CPU では約 25%，StrongARM SA-110 CPU では約 43%がキャッシュによって消費される [4]．特に，命令キャッシュへのアクセスは毎クロック・サイクル発生するため，その低消費エネルギー化が極めて重要となる．

そこで我々は，ダイレクト・マップ命令キャッシュの低消費エ

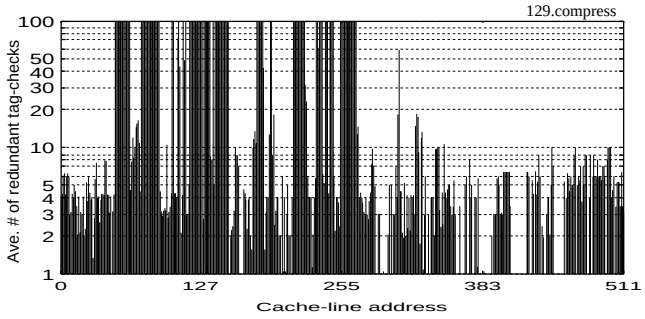

図 1　Redundant tag-check distribution in I-cache

ネルギー化を目的として，ヒストリ・ベース・タグ比較 (HBTC: History Based Tag-Comparison) 方式を提案した [3]．通常，全てのキャッシュ・アクセスにおいて，参照データがキャッシュ中に存在するか否かを調べるためにタグ比較を行う必要がある．これに対し，HBTC キャッシュでは，プログラムの実行履歴に基づき，必要に応じてタグ比較を行う．そして，プログラム実行時の総タグ比較回数を削減することで，命令キャッシュの低消費エネルギー化を実現する．文献 [3] で提案した方式では，CPU において分岐予測ミスが検出された場合，BTB に記録した実行履歴を復元する必要があった．また，文献 [3] では，実行されたタグ比較回数のみを用いて消費エネルギー削減効果を議論した．そこで本稿では，従来の HBTC キャッシュ方式を改良し，分岐予測ミス発生時でも復元作業を必要としない新アーキテクチャを提案する．また，今後のマルチメディア社会を想定し，信号処理プログラムを用いたより詳細な評価 (性能と消費エネルギー) を行う．

以下，第 2 章では，従来型キャッシュの問題点を整理し，その解決策として関連研究を紹介する．次に，第 3 章では HBTC キャッシュの基本概念とその動作について説明し，第 4 章ではベンチマーク・プログラムを用いた定量的評価を行う．最後に，第 5 章で本稿をまとめる．

2.　従来型キャッシュの問題点とその解決法

2.1　従来型キャッシュにおけるタグ比較

一般に，キャッシュはタグ・メモリとデータ・メモリで構成される．キャッシュ・アクセスが発生した際，参照アドレスで指定されたタグ情報とデータ (キャッシュ・ライン・データ) が，それぞれ，タグ・メモリおよびデータ・メモリから読み出される．そして，キャッシュ・ヒット (参照データがキャッシュに存在する) か否かを判定するため，読み出したタグ情報と参照アドレス中のタグ・フィールド値を比較する．もし，比較結果が一致であればキャッシュ・ヒットとなり，読み出したキャッシュ・ラインから参照データを選択して CPU に供給する．

通常，キャッシュの内容が更新されるのは，キャッシュ・ミスの発生により新しいデータがキャッシュにロードされた時 (または，あるデータがキャッシュから追い出された時) である．よって，連続したキャッシュ・ミスの間，キャッシュは安定した状態 (キャッシュ内のデータに変更がない状態) となる．本稿では，このような期間を**キャッシュ・ミス・インターバル**と呼ぶ．ここで，ある命令が繰返し参照される場合を考える．最初にこの命令が参照される時，キャッシュ・ミスを発生する可能性があるため，必ずタグ比較処理を行わなければならない．しかしながら，次の参照において，前回の参照から現在に至るまで一度もキャッシュ・ミスが発生していなければ (つまり，キャッシュ・ミス・インターバル内であれば)，当該命令はキャッシュ内に滞在し続けていることが保証される．よって，このような場合には，タグ比較を行うことなくキャッシュ・ヒットであると判定できる．

データ圧縮プログラム (*129.compress* SPEC ベンチマーク) の実行における，キャッシュ・ミス・インターバル当たりの平均キャッシュ・ライン参照回数を図 1 に示す (同一キャッシュ・ラインに格納された全命令は 1 つのタグ情報を共有する)．ダイレクト・マップ命令キャッシュのサイズは 16K バイト，キャッシュ・ラインサイズは 32 バイトを仮定した．なお，実験環境の詳細は第 4.2 節で示す．図 1 の横軸はキャッシュ・ライン・アドレスである．図 1 より，キャッシュ・ミス・インターバル内において，幾つかのキャッシュ・ラインは 100 回以上，また，ほとんどのラインは 3〜4 回以上参照されていることが分かる．前述したように，キャッシュ・ミス・インターバル内で本質的に実行すべきタグ比較処理は最初の 1 回だけである (当該キャッシュ・ラインに対する後続アクセスでのタグ比較結果は，最初のアクセスにおけるタグ比較結果と同じになるため)．しかしながら，従来型キャッシュでは，アクセス毎にタグ比較を行う．例えば，キャッシュ・ミス・インターバルにおける参照回数が 100 の場合，従来型キャッシュでは 100 回のタグ比較が実行される．その結果，最初を除く残り 99 回の無駄なタグ比較処理によって多くのエネルギーを消費する．

2.2　従来の解決法 (関連研究)

タグ比較回数を削減する一般的な手段の 1 つとして，レベル 1 キャッシュと CPU の間にレベル 0 キャッシュを搭載する方法がある [2] [4] [6] [7]．レベル 0 キャッシュにヒットした場合，レベル 1 キャッシュへのアクセスは発生しない (つまり，レベル 1 キャッシュでのタグ比較処理は実行されない)．しかしながら，レベル 0 キャッシュ・ミス時には従来方式と同様，タグ比較に基づくレベル 1 キャッシュ・アクセスが発生する．

その他の技術として，連続実行される命令間のアドレス関係を利用する方法がある．通常，あるメモリ参照においてキャッシュ・ミスが発生した時，参照データと同一キャッシュ・ライン内に存在する全てのデータは同時にキャッシュにロードされる．よって，連続する 2 つの命令が同一キャッシュ・ラインに存在する場合，後者の命令参照に関するタグ比較は不要となる．文献 [7] では，このようなタグ比較省略条件の動的検出方法が提案された．これを本稿では ITC(Interline Tag-Comparison) 手法と呼ぶ．

3.　ヒストリ・ベース・タグ比較キャッシュ

3.1　基 本 概 念

第 2.1 節で述べたように，ある命令を参照する時，**当該命令の前参照から現在に至るまで一度もキャッシュ・ミスが発生していなければ** (つまり，前参照と現参照がキャッシュ・ミス・インターバル内であれば) タグ比較を行うこと無しにキャッシュ・

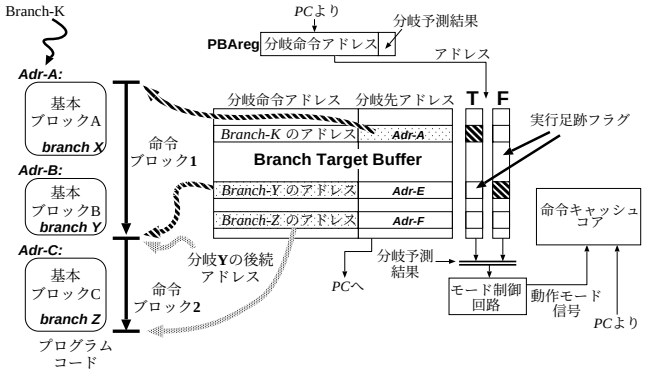

図 2　HBTC キャッシュの内部構成

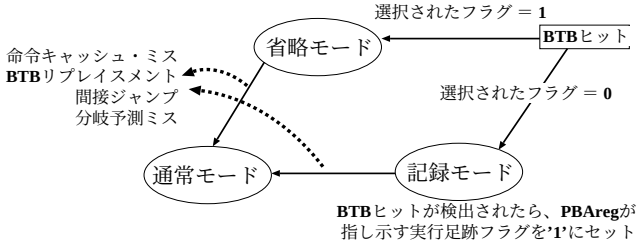

図 3　HBTC キャッシュの状態遷移図

ヒットであると判定できる．HBTC キャッシュでは，多くの高性能 CPU に搭載されている BTB(Branch Target Buffer) を利用して，このようなタグ比較省略条件を動的に検出する．

通常，BTB の各エントリには，分岐命令アドレスとその分岐先アドレスが格納される．プログラム実行中，条件が成立した分岐命令は BTB に登録される．よって，BTB には，最近実行された命令のトレース情報が (分岐命令の粒度で) 存在する．そこで HBTC キャッシュでは，ある分岐の後続命令列が実行された時，それに対応する BTB エントリに実行足跡 (実行履歴) を残す．そして，キャッシュ・ミスが発生した際，全ての実行足跡を消去する．したがって，当該命令列の次参照時に実行足跡が検出された場合には，当該命令列が同一キャッシュ・ミス・インターバル内で過去に参照された実績がある (つまり，タグ比較を省略するための条件を満たしている) ことを意味する．このようにして，HBTC キャッシュは無駄なタグ比較処理を検出ならびに削除し，低消費エネルギー化を実現する．

HBTC キャッシュの新規性は，分岐予測のために実装された BTB を活用し，無駄なタグ比較処理によって生じる消費エネルギーを削除する点にある．第 2.2 節で示した ITC 方式とは異なり，実行履歴に基づきタグ比較省略条件を判定するため，連続実行される 2 つの命令が異なるキャッシュ・ラインに存在する場合にも対応できる．また，メモリ階層構造には影響しないため，レベル 0 キャッシュなどの低消費エネルギー化技術と組み合わせて実装する事も可能である．

3.2 内 部 構 成

HBTC キャッシュは，図 2 に示すように，**命令キャッシュ・コア**，**BTB**，**前分岐アドレス・レジスタ (PBAreg)**，および，**モード制御回路**といった 4 つのハードウェア・コンポーネントを用いて実現できる．

命令キャッシュ・コアは，タグ比較処理の実行/省略を選択できることを除いては，従来型ダイレクト・マップ命令キャッシュと全く同じである．また，HBTC キャッシュで使用される BTB では，従来の BTB と比較して，各エントリに 2 つの実行足跡 1 ビット・フラグ (**T** と **F**) が追加される．実行足跡フラグ **T** は，対応する BTB エントリの分岐先命令ブロック (target instructions) が過去に参照された実績があるか否かを示す．一方，実行足跡フラグ **F** は，分岐後続命令ブロック (Fall-through instructions) の実行履歴を表している．ここで，分岐先命令ブロックとは分岐先アドレスを先頭する連続した命令列を，また，後続命令ブロックとは分岐命令の次アドレスを先頭とする連続した命令列を表す．なお，これら命令ブロックの最終アドレスは，次に実行される (BTB に登録された) 分岐命令のアドレスによって指定される．例えば，図 2 では，分岐命令 K の分岐先命令ブロックは基本ブロック A および B で構成される (分岐命令 X は BTB に未登録のため)．一方，分岐命令 Y の後続命令ブロックは基本ブロック C となる．プログラムの実行と共に BTB エントリは更新されるため，分岐先/後続命令ブロックのサイズは動的に変化する．例えば，分岐命令 X が条件成立として実行された場合，それが BTB に登録される．その結果，分岐命令 K の分岐先命令ブロックは基本ブロック A のみで構成されることになる．

PBAreg は，BTB ヒットした分岐命令のアドレス，ならびに，その際の分岐予測結果 (成立か不成立) を保存する．そして，上述した実行足跡フラグをセットする際に BTB アクセス用アドレス・レジスタとして使用される．

HBTC キャッシュは，**省略モード**，**通常モード**，および，**記録モード** といった 3 つの動作モードを有する．省略モードでは，命令キャッシュ・アクセスにおいてタグ比較は省略される (キャッシュ・ヒットが保証される)．また，通常モードでは，従来型キャッシュと同様，タグ比較に基づくキャッシュ・アクセスを行う．一方，記録モードでは，通常モードと同様にタグ比較が行われる．それに加え，本モードにて BTB ヒットが検出された時，対応する実行足跡フラグがセットされる．なお，動作モード制御の詳細については第 3.3 節で説明する．

3.3 動　　作

HBTC キャッシュにおける動作モードの状態遷移を図 3 に示す．BTB ヒットが発生した際，分岐先アドレスの読み出しと同時に，実行足跡フラグ **T** と **F** を読み出す．そして，分岐予測結果に従い，何れか一方のフラグを選択する．選択された実行足跡フラグが' 1' にセットされていれば，HBTC キャッシュは省略モードへと遷移する．一方，選択されたフラグが' 0' の場合，第 3.1 節で説明したタグ比較を省略するための条件を満足しない．よって，動作モードは記録モードとなり，それと同時に分岐命令アドレス (現在の PC) および分岐予測結果を PBAreg に保存する．

HBTC キャッシュではキャッシュ・ミスが発生した時，全ての実行足跡フラグを無効化 ('0' にリセット) すると同時に，通常モードへと状態を遷移する．したがって，記録モード動作中に次の BTB ヒットが検出された場合，これは，前 BTB ヒットから現 BTB ヒットまでの間に 1 度もキャッシュ・ミスが発

生しなかった (全ての命令参照はキャッシュにヒットした) ことを意味する．よって，記録モード動作中に BTB ヒットが発生した場合には，PBAreg(前 BTB ヒット時の分岐命令アドレス) が指し示す実行足跡フラグを'1' にセットする．

また，第 3.2 節で説明したように，各実行足跡フラグに対応する命令ブロックの最終アドレスは，BTB に登録されたある分岐命令アドレスによって指し示される．そのため，BTB エントリの追い出しが発生した場合，ある命令ブロックの最終アドレス情報を失う可能性がある．よって，HBTC キャッシュでは，キャッシュ・ミス発生時だけでなく，BTB エントリの置換えが発生した際にも全ての実行足跡を無効化し，通常モードに遷移する．また，間接ジャンプ実行時，ならびに，分岐予測ミス検出時においても，現在の動作モードに関係なく通常モードへと移行する (ただし，実行足跡フラグの無効化は発生しない)．これにより，HBTC キャッシュの動作モード制御を大幅に単純化できる．

文献 [3] で示した方式では，分岐予測実行時，それに続く命令ブロック内の全命令参照はキャッシュにヒットすると仮定し，投機的に実行足跡フラグをセットする．これに対し，本稿での提案方式では，分岐先/後続命令ブロック内の全命令参照がキャッシュにヒットすることを確認した後，実行足跡フラグをセットする．したがって，文献 [3] の方式とは異なり，分岐予測ミスが検出された場合においても実行足跡を復元する必要はない．

3.4 性能オーバヘッド

通常，フェッチされた命令の種類に関係なく，分岐予測を行うために BTB アクセスは毎クロック発生する．したがって，記録モードにおいて実行足跡フラグをセットする際，ならびに，キャッシュ・ミスまたは BTB リプレイスによって実行足跡フラグを無効化する際，BTB アクセス競合が発生する．このような場合には，HBTC キャッシュの BTB アクセスが優先され，プロセッサでは 1 クロック・サイクルのストールが生じる．一方，実行足跡フラグの読み出しに関しては，分岐先アドレス読み出しと並列に行えるため，ストール・サイクルは発生しない．

4. 評　　価

4.1 消費エネルギー・モデル

HBTC キャッシュにおける消費エネルギー (E_{TOTAL}) は，以下の式で表すことができる．

$$E_{TOTAL} = E_{CACHE} + E_{BTBadd} \quad (1)$$

ここで，E_{CACHE} および E_{BTBadd} は，それぞれ，命令キャッシュ・アクセスおよび BTB の実行足跡フラグ・アクセスに伴う消費エネルギーである．なお，E_{BTBadd} において，従来の BTB 構造によって消費されるエネルギーは含まない．また，E_{CACHE} は以下の式で近似できる．

$$E_{CACHE} = E_{tag} + E_{data} + E_{output} + E_{dec} \quad (2)$$

ここで，E_{tag} および E_{data} は，それぞれ，タグ・アクセスおよびデータ・アクセスによって消費されるエネルギーである．また，E_{output} は外部バスや I/O ピン駆動による消費エネルギーを，E_{dec} はアドレス・デコードで消費されるエネルギーを示す．なお，E_{dec} は他の要素と比較して 3 桁程度小さいことが報告されており，本稿では考慮しない [1] [6]．一方，E_{BTBadd} は以下の式で近似できる．

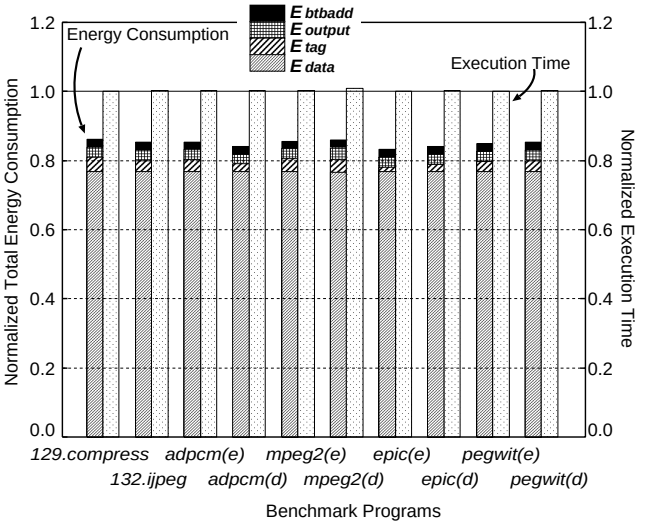

図 4　プログラム実行時間と消費エネルギー

$$E_{BTBadd} = E_{BTBef} + E_{BTBlogic} \quad (3)$$

ここで，E_{BTBef} は実行足跡フラグへのアクセスに伴う消費エネルギーであり，$E_{BTBlogic}$ は HBTC 方式のためのロジック部分 (PBAreg およびモード制御回路) における消費エネルギーである．ただし，PBAreg やモード制御回路は極めて単純な論理回路で実現可能なため，本評価では $E_{BTBlogic}$ を 0 と仮定する．

4.2 実験環境

HBTC キャッシュの有効性を明らかにするため，SimpleScalar シミュレータ [11] を用いた定量的評価を行った．本評価では，SPEC95 ベンチマーク・サイト [12] より 2 つのプログラム (*129.compress*，*132.ijpeg*)，ならびに，Mediabench [10] より 4 つのプログラム (*adpcm*，*mpeg2*，*epic*，*pegwit*) のデコーダおよびエンコーダを用いた．なお，命令キャッシュ・サイズは 16K バイト，ラインサイズは 32 バイト，BTB エントリ数は 2048(セット数 512 の 4 ウェイ) と仮定した．また，他のパラメータに関しては，SimpleScalar シミュレータのデフォルト値を用いた．

一方，全消費エネルギー (E_{TOTAL}) に関しては，文献 [4] で提案されたモデルを参考にして求めた．なお，各回路ノードの負荷容量については，0.8um の CMOS テクノロジを仮定し，文献 [5] [8] で示された値を参照した．

4.3 実験結果

4.3.1 性能と消費エネルギー

キャッシュの全消費エネルギー，ならびに，プログラム実行時間 (所要クロック・サイクル数) を図 4 に示す．全ての値は，従来型キャッシュにおける実験結果で正規化している．図から分かるように，HBTC 方式を採用することで 8%〜17%の低消費エネルギー化を達成できた．これは，実行足跡フラグ読み出しによるオーバヘッド (E_{BTBadd} の増加) に比べ，タグ比較省略による消費エネルギー削減効果 (E_{tag} の削減) が大きいた

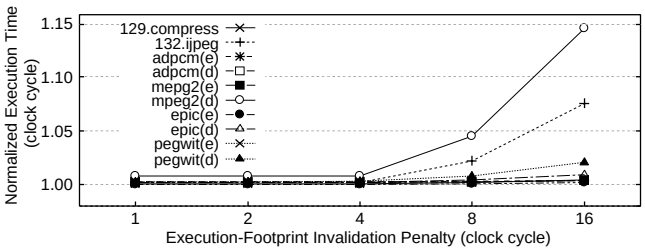

図 5　実行足跡フラグの無効化遅延時間が与える影響

めである．なお，HBTC 方式はキャッシュ・ヒット率に影響しないため，図中の E_{data} ならびに E_{output} に関しては，従来型キャッシュのそれらと同じ値である．

一方，プログラム実行時間に関しては，全てのプログラムにおいて 1%未満の性能低下であった．この性能オーバヘッドは，分岐予測のための分岐先アドレス読み出しと，HBTC 方式における実行足跡フラグ更新処理との間で生じる，BTB アクセス競合が原因である．したがって，BTB 実装時，実行足跡フラグ更新用書込みポートを新たに設け，BTB アクセス競合を解消することで性能低下を回避できる．

4.3.2　実行足跡フラグ無効化ペナルティ

第 4.3.1 節では，キャッシュ・ミスならびに BTB リプレイスメントに伴う実行足跡フラグの無効化は 1 クロック・サイクルで終了すると仮定した．しかしながら，BTB の全エントリにおいて実行足跡フラグをリセットする必要があるため，無効化処理ペナルティは BTB の実装に大きく依存する．そこで，無効化処理の所要クロック・サイクル数を 1〜16 に変化させ，HBTC キャッシュが CPU 性能へ与える影響を調査した．その結果を図 5 に示す．全ての結果は，従来型キャッシュを用いた場合のプログラム実行時間で正規化している．

図より，無効化ペナルティが 4 サイクル以下の場合には大幅な性能低下は発生しないことが分かる．これは，実行足跡無効化ペナルティのほとんどが，キャッシュ・ミス・ペナルティにより隠蔽されたためである (本シミュレーションでは，命令キャッシュ・ミス・ペナルティは 6 クロック・サイクルと仮定した)．命令キャッシュ・ミスが発生した場合，実行足跡フラグの無効化処理は，命令キャッシュ・リプレイスメント処理と並列に行われる．実際，無効化発生原因の内訳を解析した結果，その 95%以上が命令キャッシュ・ミス (残り 5%が BTB エントリの追い出し) に起因することが判明した．以上より，無効化ペナルティがキャッシュ・ミス・ペナルティより小さい場合，HBTC 方式の採用に伴う性能低下は無視できる程度に小さいと考える．

4.3.3　キャッシュ・サイズ

集積可能なトランジスタ数の増加に伴い，キャッシュ・サイズは年々増加傾向にある．そこで，キャッシュ・サイズが HBTC 方式の消費エネルギー削減効果に与える影響を調査した．キャッシュ・サイズを 4K バイト〜64K バイトに変化させた際の消費エネルギーを図 6 に示す．各キャッシュ・サイズそれぞれにおいて，HBTC キャッシュの結果は，従来型キャッシュの消費エネルギーで正規化している．図より，キャッシュ・サイズの増加と共に HBTC 方式による消費エネルギー削減効果も高くなっ

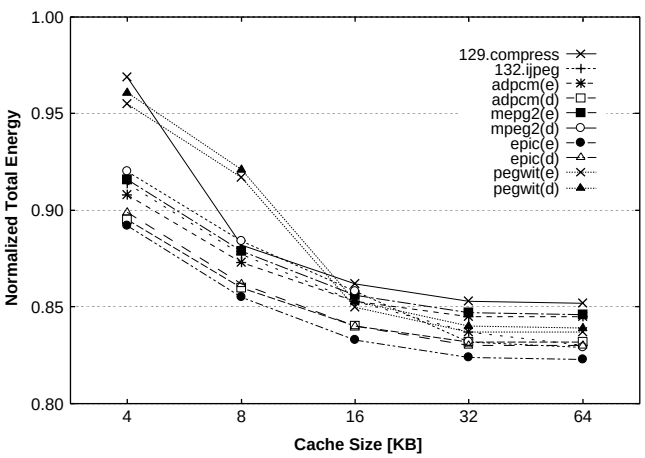

図 6　キャッシュ・サイズと消費エネルギー

ていることが分かる(注 1)．これは，キャッシュ・サイズの増加に伴いキャッシュ・ヒット率が高くなり，実行足跡フラグの無効化回数が削減されたためである．以上より，HBTC 方式は今後の大規模 LSI においても有効であると考える．

4.3.4　BTB 連想度

実行足跡フラグの追加により拡張された BTB は，命令実行のたびにアクセスされる．BTB の各ウェイにおいて 2 つの実行足跡フラグ (**T** と **F**) が読み出されるため，HBTC キャッシュにおける消費エネルギー・オーバヘッド E_{BTBadd} は BTB の連想度に比例する．これまでの評価では，BTB 連想度は 4 と仮定した．そこで，BTB の連想度を 1〜32 に変化させ，BTB 実装方法が HBTC キャッシュの消費エネルギー削減効果へ与える影響を調査した．実験結果を図 7 に示す．各 BTB 連想度それぞれにおいて，全ての結果は従来型キャッシュの消費エネルギーで正規化している．

実験結果より，若干ではあるが，BTB 連想度の増加と共に消費エネルギー削減効果が低下していることが分かる．連想度の増加は BTB 競合ミスを削減し，引いては，実行足跡フラグ無効化処理の発生頻度を低くする．しかしながら，第 4.3.2 節で述べたように，殆んどの無効化処理はキャッシュ・ミスに起因している．つまり，無効化発生頻度の低下による消費エネルギー削減効果 (E_{tag} の削減) に比べ，消費エネルギー・オーバヘッドが大きくなり (E_{BTBadd} の増加)，その結果，HBTC キャッシュの有効性が低下したものである．しかしながら，BTB 連想度が 32 の場合でも，従来方式と比較して 7%〜16%の低消費エネルギー化を達成している．よって，BTB 連想度が高い場合においても，HBTC キャッシュは有効であると考える．

4.3.5　ITC キャッシュとの比較

第 2.2 節で示した ITC キャッシュとの比較を行い，提案手法の有効性を評価した．ITC キャッシュと HBTC キャッシュに関して，各プログラムで実行された総タグ比較回数を図 8 に示す．全ての結果は，従来型キャッシュでのタグ比較回数で正規化している．

ITC キャッシュでは，連続して実行される 2 つの命令が同一キャッシュ・ラインに存在する時，後者の命令に関するタグ比較

(注 1)：消費エネルギーの絶対値はキャッシュ・サイズと共に増加する．

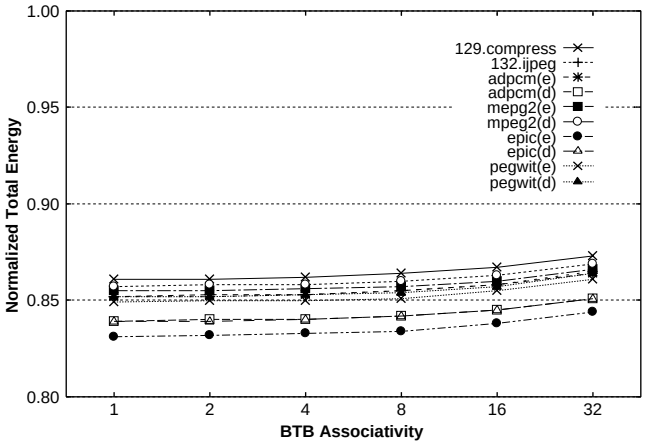

図 7　BTB 連想度が与える影響

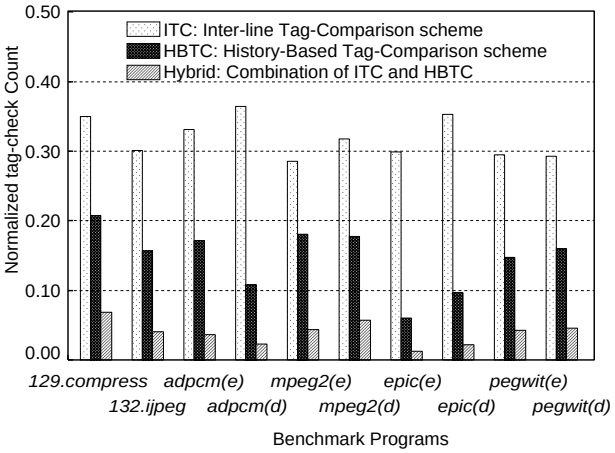

図 8　TIC キャッシュとの比較 (タグ比較回数)

処理を省略する．通常，プログラムは逐次実行を基本としており，連続する命令が同一キャッシュ・ラインに存在する場合が多い．その結果，図から分かるように，ITC キャッシュは全てのプログラムにおいて約 67%のタグ比較削減を達成している．これに対し，HBTC キャッシュの実験結果にはある程度のばらつきがあるものの，全てのプログラムにおいて ITC キャッシュより高い削減率となった．HBTC キャッシュは命令の繰返し実行性を活用し，連続する命令が異なるキャッシュ・ラインに存在する場合においてもタグ比較を省略できる．基本的にメディア・アプリケーションは多くのループ構造を有するため，HBTC 方式の利点が顕著に現れたものと考える．

また，ITC 方式と HBTC 方式を組み合わせることで，より多くのタグ比較処理を省略できた (85%〜95%の削除率)．これは，特に逐次命令に対して有効な ITC 方式の利点と，履歴に基づく (つまり，分岐にも対応できる) HBTC 方式の利点とが，それぞれ相乗効果をもたらした結果である．

5. おわりに

本稿では，ダイレクト・マップ命令キャッシュ用低消費エネルギー化手法として，ヒストリ・ベース・タグ比較 (HBTC) 方式を提案した．本方式では，BTB に記録した実行履歴を利用して，無駄なタグ比較処理を動的に検出・削除する．

複数ベンチマークを用いて実験を行った結果，従来タグ比較方式と比較して，1%以下の性能低下を伴うだけで，8%〜17%のキャッシュ消費エネルギーを削減できた．また，提案キャッシュの設計空間を考慮し，BTB や命令キャッシュの構成決定パラメータが提案手法の有効性に与える影響を調査した．さらに，これまでに提案されたタグ比較削減手法との比較を行い，全てのプログラムにおいて提案手法が有効であることを明らかにした．

本評価では，HBTC キャッシュを実現するための論理回路部分で消費されるエネルギーは考慮しなかった．今後，HBTC キャッシュの実設計を行い，これらを含めた総合的な評価を行う予定である．

謝　辞

日頃から御討論頂く，九州大学 大学院システム情報科学研究院 安浦寛人 教授に感謝します．なお，本研究は一部，文部省科学研究費補助金 (課題番号：12358002, 13308015) による．

文　献

[1] R. I. Bahar, G. Albera, and S. Manne, “Power and Performance Tradeoffs using Various Caching Strategies,” *Proc. of the 1998 International Symposium on Low Power Electronics and Design*, pp.64–69, Aug. 1998.

[2] N. Bellas, I. Hajj, C. Polychronopoulos, and G. Stamoulis,“Energy and Performance Improvements in Microprocessor Design using a Loop Cache,” *Proc. of the 1999 International Conference on Computer Design: VLSI in Computers & Processors*, pp.378–383, Oct. 1999.

[3] K. Inoue, V. Moshnyaga, and K. Murakami, “Omitting Cache Look-Up for High-Performance, Low-Power Microprocessors,” *IEICE Transactions on Electronics*, vol. E85-C, no.2, pp.279-287, Feb. 2002.

[4] M. B. Kamble, and K. Ghose,“Analytical Energy Dissipation Models For Low Power Caches,” *Proc. of the 1997 International Symposium on Low Power Electronics and Design*, pp.143–148, Aug. 1997.

[5] M. B. Kamble, and K. Ghose,“Energy-Efficiency of VLSI Caches: A Comparative Study,” *Proc. of 10th International Conference on VLSI Design*, pp.261–267, Jan. 1997.

[6] J. Kin, M. Gupta, and W. H. Mangione-Smith,“The Filter Cache: An Energy Efficient Memory Stucture,” *Proc. of the 30th Annual International Symposium on Microarchitecture*, pp.184–193, Dec. 1997.

[7] R. Panwar, and D. Rennels, “Reducing The Frequency of Tag Compares for Low Power I-cache Design,” *Proc. of the 1995 International Symposium on Low Power Electronics and Design*, pp.57–62, Apr. 1995.

[8] S. J. E. Wilton and N. P. Jouppi, “An Enhanced Access and Cycle Time Model for On-Chip Caches,” *WRL Research Report 93/5*, July 1994.

[9] C. L. Su, and A. M. Despain, “Cache Design Trade-offs for Power and Performance Optimization:A Case Study,” *Proc. of the 1995 International Symposium on Low Power Design*, pp.69–74, Apr. 1995.

[10] MediaBench, *URL: http://www.cs.ucla.edu/˜leec/mediabench/.*

[11] “SimpleScalar Simulation Tools for Microprocessor and System Evaluation,” *URL:http://www.simplescalar.org/.*

[12] SPEC (Standard Performance Evaluation Corporation), *URL:http://www.specbench.org/osg/cpu95.*