修 士 論 文

ソフトウェア制御によるキャッシュ参照ウェイ限定手法の研究

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

小林 智弘

2015 年 3 月

修 士 論 文

ソフトウェア制御によるキャッシュ参照ウェイ限定手法の研究

指導教員　田中 清史 准教授

審査委員主査　田中 清史 准教授
審査委員　井口 寧 教授
審査委員　金子 峰雄 教授

北陸先端科学技術大学院大学
情報科学研究科情報科学専攻

1310025 小林 智弘

提出年月: 2015 年 2 月

概 要

近年のプロセッサでは，データアクセス時のキャッシュヒット率の向上を図るために，ブロック配置方式としてセットアソシアティヴ方式が採用されている．従来のセットアソシアティヴ方式ではデータアクセス時間を最小化するために，データアクセス時にすべてのウェイのタグとデータ配列が並列に読み込まれ，タグ比較が行われ，ヒットするウェイが検出される．しかしながら，ヒットするウェイはただひとつであり，ヒットしないウェイのデータ配列の読み込みにおいて，エネルギーを浪費することが問題となる．この問題に対してウェイを予測限定参照することによって浪費を削減する手法が存在するが，予測のために大きなハードウェアテーブルと複雑なキャッシュ構造が必要となる．そこで，本研究では大きなハードウェアテーブルと複雑なキャッシュ構造を必要とせずにウェイの予測限定参照を行う有効な方法として，TracePC Way predection(TracePC) 法と Simple-Counter Way prediction(SC) 法を提案する．提案手法のデータアクセス時の消費電力量とウェイ予測の精度を他のウェイ予測手法と比較した評価を行い，提案手法の有効性を調査する．

目 次

第 1 章　はじめに **2**
1.1 背景 2
1.2 目的 3
1.3 論文の構成 3

第 2 章　関連研究 **4**
2.1 Predictive Sequential Associative Cache(PSA)[1] 4
2.1.1 Rehash Table 4
2.1.2 Steering Bit Table 4
2.1.3 Prediction Address 5
2.2 Reactive-Associative Caches(R-A Cache)[2] 6
2.2.1 Victim List 6
2.2.2 Access Prediction Table(APT) と Block Waynumber Table(BWT) . 6
2.2.3 inhibit list 7
2.3 Reducing Set-Associative Cache Energy via Way-Prediction and Selective Direct-Mapping[3] 7

第 3 章　本研究の提案手法 **11**
3.1 TracePC Way predeciton(TracePC) 法 11
3.1.1 メモリアクセスのトレースと参照ウェイの決定 11
3.1.2 カウントのフェーズ分け 12
3.2 Simple-Counter Way prediction(SC) 法 12
3.2.1 ハードウェアカウンタと参照ウェイの決定 12
3.2.2 カウントのフェーズ分け 13

第 4 章　評価 **14**
4.1 評価環境 14
4.2 比較対象の手法の設定 14
4.3 提案手法の設定 15
4.4 評価結果 15
4.4.1 提案手法のデータアクセス時の消費電力量 15
4.4.2 限定参照の予測精度 17

第 5 章　まとめ **19**
5.1　まとめ 19
5.2　今後の課題 19

第1章　はじめに

1.1　背景

近年のプロセッサでは，データアクセス時のキャッシュヒット率の向上を図るために，キャッシュメモリのブロック配置方式として，セットアソシアティヴ方式が採用されている．セットアソシアティヴ方式とは，各ブロックを配置する場所が少なくとも 2 つ以上存在するブロック配置方式である．従来のセットアソシアティヴ方式ではデータアクセス時間を最小化するために，データアクセス時にすべてのウェイのタグとデータ配列が並列に読み込まれ，タグ比較が行われ，ヒットするウェイが検出される (図 1.1)．しかしながら，ヒットするウェイはただひとつであり，ヒットしないウェイに対するデータ配列の読み込みにおいて，結果的にエネルギーを無駄に消費することが問題となる．例えば，4-way セットアソシアティヴ方式のデータアクセスのヒット時の場合はヒットするウェイはただ 1 つであるのにも関わらず，ヒットしない残りの 3 つのウェイのデータ配列も読み込むため，約 75%のエネルギーを浪費する．

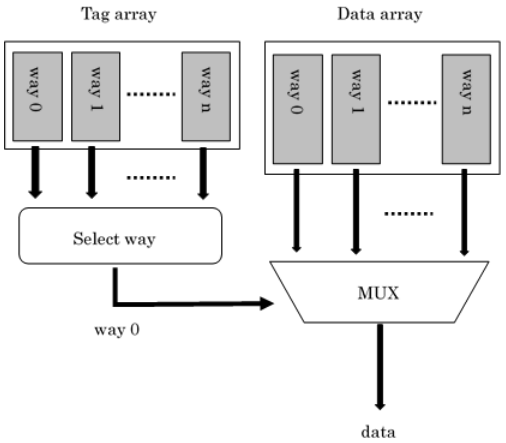

図 1.1: 従来の 4-way セットアソシアティヴ方式のキャッシュアクセス（読み込まれるウェイが影付けされている）．読み込まれるデータ配列の中でヒットするウェイは way 0 のみであるが，way 1〜n を同時に読み込んでいる．

1.2 目的

今日の高性能プロセッサに採用されているキャッシュメモリのブロック配置方式であるセットアソシアティブ方式は，複数のウェイへの並列参照による消費電力の浪費が問題となっている．この問題に対して，タグ配列のみを先に読み込み，比較し，その比較結果に基づいてマッチしているウェイのデータのみを読み込む手法が存在する．しかしながら，この手法はタグ配列とデータ配列を同時に読み込みこんでいる従来のセットアソシアティヴ方式とくらべて，毎回のキャッシュアクセス時間が大幅に増加するため，高性能な L1 キャッシュには適していない．そこで，Powell らはキャッシュアクセス時間の増加を抑えることが可能な Reacive-Associative Caches(R-A Cache)[2] を L1 キャッシュのデータアクセス時の消費電力量の削減に利用している [3]．文献 [3] における R-A Cache はデータアクセス時にすべてのタグ配列の読み込みと並列に予測されたウェイのデータ配列のみを読み込む機能を備えたキャッシュである．しかしながら，参照するウェイの予測精度を高めるために，プロセッサ内にハードウェアによってテーブルを用意する必要がある．また，ウェイ予測の精度を向上させるためにほとんどのブロックをダイレクトマップ方式のようにを配置するため，従来のセットアソシアティヴキャッシュと比べてキャッシュ全体ミス率が増加する．

本研究ではウェイを限定参照するための既存のウェイ予測手法について調査し，ソフトウェア制御による参照ウェイ限定手法である TracePC Way prediction(TracePC) 法と極めて小さなカウンターを用いて参照ウェイを限定する Simple-Counter Way prediction(SC) 法を提案する．TracePC 法は，参照ウェイを決定するためのテーブルのような特別なハードウェアを必要とせず，SC 法はウェイ予測のために極めて小さなカウンターを必要とする．どちらの方式も従来のセットアソシアティヴ方式と同じ全体ミス率である．これらの提案する方式に対して，シミュレーションによりプログラム毎のデータキャッシュに対する消費電力量の定量的評価を行うことにより有効性を示す．

1.3 論文の構成

本論文は 5 章からなる．第 2 章では，ウェイ予測を用いた関連研究について述べる．第 3 章では，提案手法の TracePC Way prediction(TracePC) 法と Simple-Counter Way prediction(SC) 法について述べる．第 4 章では評価環境について述べた後、提案手法の評価結果を述べる．第 5 章で本論文をまとめる．

第2章　関連研究

本章では, 本研究の関連研究としてプロセッサにハードウェアを追加して，参照すべきウェイを予測し，限定参照する方式である Predictive Sequential Associative Cache(PSA)[1] と Reactive-Associative Caches(R-A Cache)[2] を説明し，これらの手法を第 1 章で述べたエネルギーの浪費問題を削減することに応用した研究である文献 [3] について説明する．

2.1　Predictive Sequential Associative Cache(PSA)[1]

PSA はウェイを限定して参照するためにテーブルを用いてウェイ予測を行う初めての研究であり，セットアソシアティヴ方式のデータ配列の選択を原因としたアクセス時間の増加を改善するために提案されたキャッシュ構造である．PSA はキャッシュの限定参照を行うために Steering Bit Table と呼ばれるテーブルを使用し，予測ミス時に 2 回目の参照によるアクセス時間の増加を防ぐために Rehash Table を使用する．

図 2.1 に PSA の構造を示す．PSA のキャッシュは 2 つのバンクに分けられ，Prediction Address をインデックスとして用いて Steering Bit Table を使って 1 回目の参照 (probe0) を行う．その参照の際に，並列にもうひとつのバンクの Rehash Table 値を参照する．probe0 の結果，ヒットしていた場合にはメモリアクセスは完了し，2 回目の参照 (probe1) は行われない．probe0 がミスした場合，probe1 の有無は参照した Rehash Table の値によって決まる．

2.1.1　Rehash Table

Rehash Table の各エントリーは各ブロックのタグ情報の最下位ビットとバンク番号の排他的論理和演算値を保持している．Probe0 の際に，参照される側のバンクではないバンクの Rehash Table 値が読み込まれ，実際のデータアドレスのタグの最下位ビットと比較し，比較結果に基いて probe1 の有無が決まる．

2.1.2　Steering Bit Table

Steering Bit Table(SBT) とはウェイを限定して参照するために必要となるウェイ予測テーブルであり，テーブルの各エントリはウェイ番号によって構成される．エントリのサ

イズは $\log_2 n$ (n はセットアソシアティヴ方式の連想度) ビットである．SBT はメモリアクセスの際に Prediction Address をインデックスとして参照される．

2.1.3 Prediction Address

Prediction Address は SBT の参照のためのインデックスであり，以下に文献 [1] 内で挙げられた Prediction Address と成り得る 4 つの候補を示す．

有効アドレス
: メモリアクセス命令におけるレジスタ値とオフセットを加算した実際のデータアドレスである．

XOR 近似アドレス
: メモリアクセス命令におけるレジスタ値とオフセットを排他的論理和演算した近似データアドレスである．

レジスタ番号とオフセット
: メモリアクセス命令におけるレジスタ番号とオフセットを結合した擬似アドレスである．

プログラムカウンタ (PC)
: メモリアクセス命令の PC 値である．

実際のデータアドレスとウェイ番号を関連付けることにより，精度の高いウェイ限定参照を行うことが可能である．しかしながら，実際のデータアドレスが使用可能となるのはパイプライン処理においてメモリアクセスの直前であるため，速度的なペナルティを生じることなしに実装することが困難である．同様に，速度的なペナルティ無しにメモリアクセス命令のレジスタ値とレジスタ番号を予測の入力として用いることも困難である．速度的ペナルティを生じること無しに実装するために，パイプライン処理の早い段階で利用可能となるプラグラムカウンタ値を予測の入力とすることが現実的である．しかしながら，有効アドレスや XOR 近似アドレスを予測の入力とする方法と比較予測の精度は低くなる．

2.2 Reactive-Associative Caches(R-A Cache)[2]

R-A Cache は PSA と同様にセットアソシアティヴ方式のデータ配列の選択を原因としたダイレクトマップ方式に対するアクセス時間の増加を改善するために提案されたキャッシュ構造である．

Batson らは競合を引き起こすキャッシュブロックは僅かであり，ほとんどのキャッシュブロックはダイレクトマップポジションでヒットする傾向があると主張し，ほとんどのキャッシュブロックをダイレクトマップポジションに配置し、競合しているブロックのみをセットアソシアティヴポジションに置き直す振る舞いをするキャッシュ格納方式である R-A cache を提案した．この振る舞いにより R-A Cache は高い精度のウェイ予測限定参照を実現している．しかしながら，ウェイ限定参照による精度は向上するが，LRU 管理の従来のセットアソシアティヴ方式に対してキャッシュ全体ヒット率は低下する．R-A Cache は，データアクセス時に，すべてのウェイのタグ配列と予測によって選択されたウェイのデータのみを並列に読み込む．そして，タグ比較の結果，読み込んだウェイのデータがヒットしている場合はデータアクセスを完了する．異なるウェイにヒットするデータが存在する場合は，ヒットしているデータを読み込むための 2 回目の参照を行う．そして，どのウェイにもデータが存在していない場合は，全体キャッシュミスとなり，2 回目の参照は行われない．

2.2.1 Victim List

R-A Cache は競合しているブロックのみをダイレクトマップ配置からセットアソシアティヴ配置へと置き換えるために，Victim List と呼ばれるテーブルを必要とする．Victim List の各エントリーはブロックアドレスと飽和カウンタによって構成される．ブロックがキャッシュから追い出される時に，Victim List が参照され，そのブロックのエントリーが存在していた（すなわち，過去に追い出されたことがある）場合は，カウンタの値をインクリメントする．エントリが存在しなければブロックアドレスが挿入される．カウンタの値が定められた値に達した場合にはそのブロックは競合していると判断され，Block Waynumber Table(BWT) と呼ばれるテーブルにそのブロックアドレスのタグが登録され，そのブロックがキャッシュ内に再度フィルされる際にセットアソシアティヴ配置へと置き直される．

2.2.2 Access Prediction Table(APT) と Block Waynumber Table(BWT)

R-A Cache はあるメモリアクセス命令がダイレクトマップ配置のブロックへのアクセスなのか，セットアソシアティブ配置のブロックへのアクセスであるのかを判断するために，メモリアクセス命令のプログラムカウンタ（PC) 値と Access Prediction Table(APT)

と Block Waynumber Table(BWT) を関連付けている．APT の各エントリーはメモリアクセス命令の PC 値のタグとその命令の過去にアクセスしたブロックのブロックアドレスにより構成され、BWT の各エントリーは Victim List によって競合していると判断されたブロックアドレスのタグとウェイ番号とウェイ予測ミス飽和カウンタにより構成される．APT はメモリアクセス命令の際に必ずアクセスされ，メモリアクセス命令の PC 値をインデックスとして使ってアクセスされる。その PC 値に対応する APT のエントリーが存在しない場合は，ダイレクトマップポジションのウェイへのアクセスと判断され，エントリーが存在する場合はそのエントリの保持するブロックアドレスをインデックスとして BWT にアクセスする (図 2.2)．また，BWT 内に対応するエントリが存在していない場合はダイレクトマップ配置へのウェイ限定アクセスと判断され，存在する場合には BWT エントリー内のウェイ番号を用いてセットアソシアティヴ配置へのウェイ限定アクセスとなる．そして，セットアソシアティヴ配置へのウェイ限定アクセスが予測ミスした (同じセット内の他のウェイがヒットしている）場合は BWT エントリー内のウェイ予測ミスカウンタがインクリメントされ，予測が当たっていた場合にはミス予測カウンタがデクリメントされる．ウェイ予測ミスカウンタの値が定められた値に達した場合には inhibit list と呼ばれるテーブルに情報が登録され，今後はダイレクトマップポジションでヒットすることを期待し，キャッシュ内から対応するブロックを追い出し，次回のアクセスからはそのメモリアクセス命令はダイレクトマップポジションへのウェイ限定アクセスとなる．

2.2.3 inhibit list

inhibit list はセットアソシアティヴポジションへの再度の置き直しを禁止するために使用される．inhibit list のエントリーはセットアソシアティヴポジションへの置き直しが禁止されているかを表す 1 ビットのみで構成され，命令キャッシュ内にメモリアクセス命令と関連付けられて置かれる．

2.3 Reducing Set-Associative Cache Energy via Way-Prediction and Selective Direct-Mapping[3]

Powell らは R-A Cache[2] をセットアソシアティヴ方式のデータ配列のウェイ読み込みにおけるエネルギー浪費問題の改善へと応用した．

R-A Cache[2] は図 2.3 のようにデータ配列のウェイを限定して参照するため，エネルギー浪費を抑える事が可能である．ウェイ限定参照の予測がはずれていた (同じセット内の他のウェイにヒットするデータが存在する) 場合は 2 回目の参照を行う．2 回目の参照の際は 1 回目の参照のタグ比較の結果に基づいて，ヒットしているウェイのデータ配列のみを参照する．4-way セットアソシアティヴ方式のウェイ限定参照の予測として PC を Prediction Address として用いている PSA 方式と R-A Cache を使用した場合の，L1 D-cache アクセ

ス時におけるエネルギー削減量の評価を行っており，それぞれ従来の 4-way セットアソシアティヴ方式に対して平均 63%，平均 69%のエネルギー遅延を削減している．

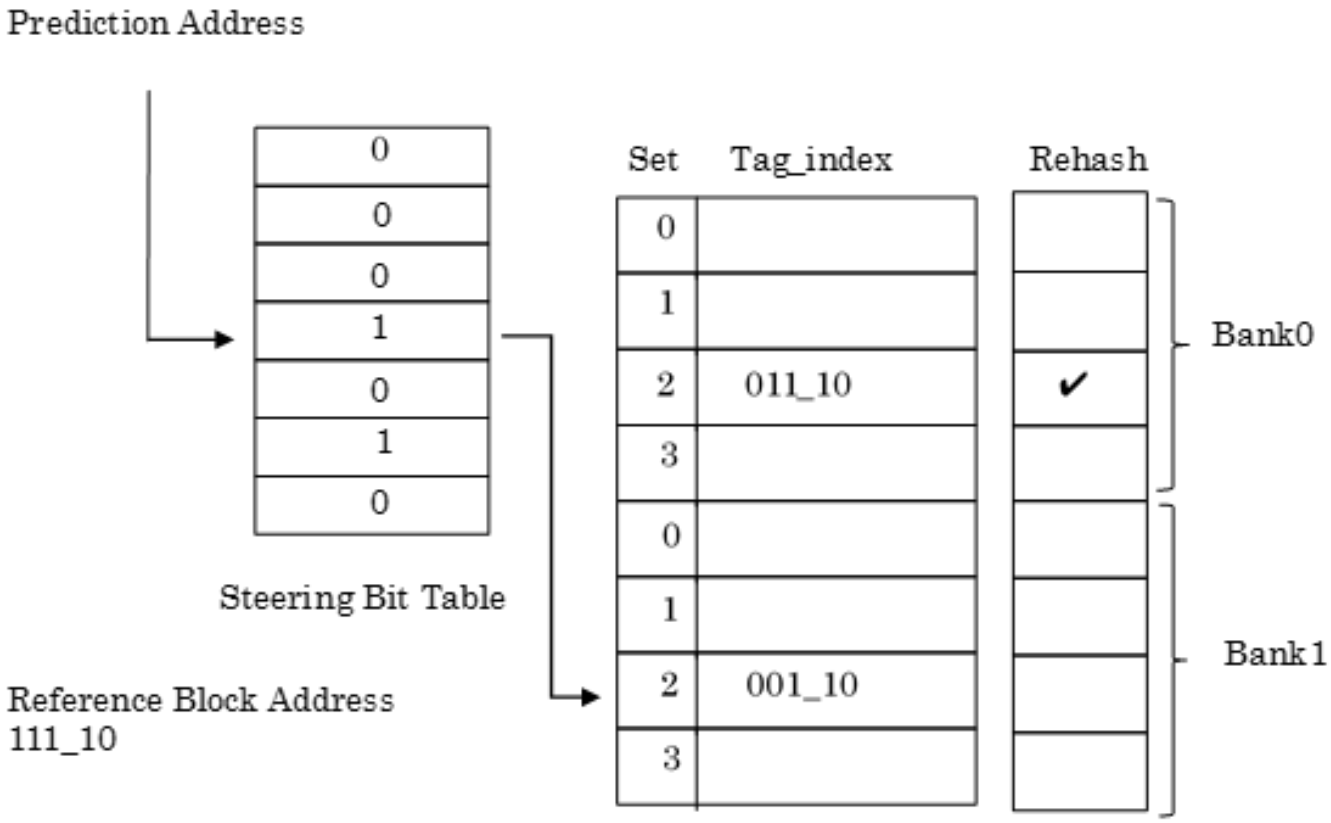

図 2.1: ブロックアドレス 110_10 を参照する際の PSA の振る舞い．まず，Prediction Address をインデックスとして Steering Bit Table のエントリーを参照し，そのエントリーを用いて Bank1 のセットを限定参照 (probe0) している．Bank1 の参照の際に Bank0 の対応するセットの Rehash Table を参照し，Rehash 値が有効となっているため Bank0 への probe1 を行う．しかしながら，Bank0 への probe1 はキャッシュミスとなり，LRU 管理によってブロックを置き換える．

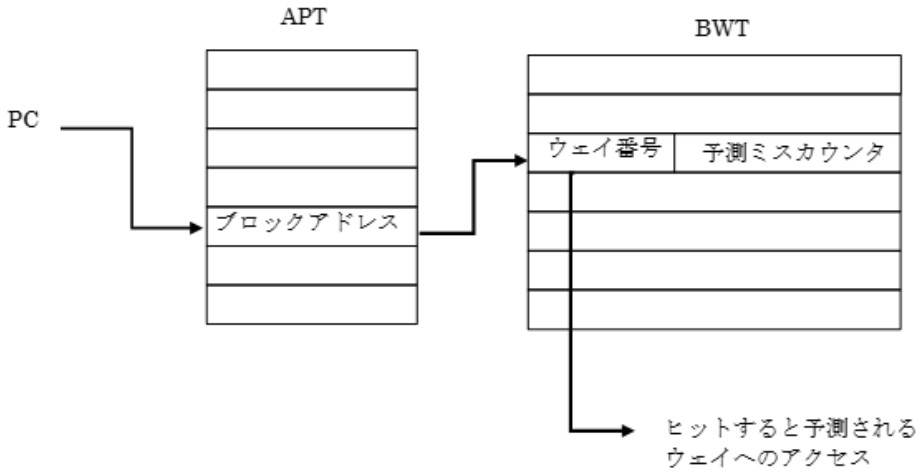

図 2.2: APT と BWT へのアクセスの流れ

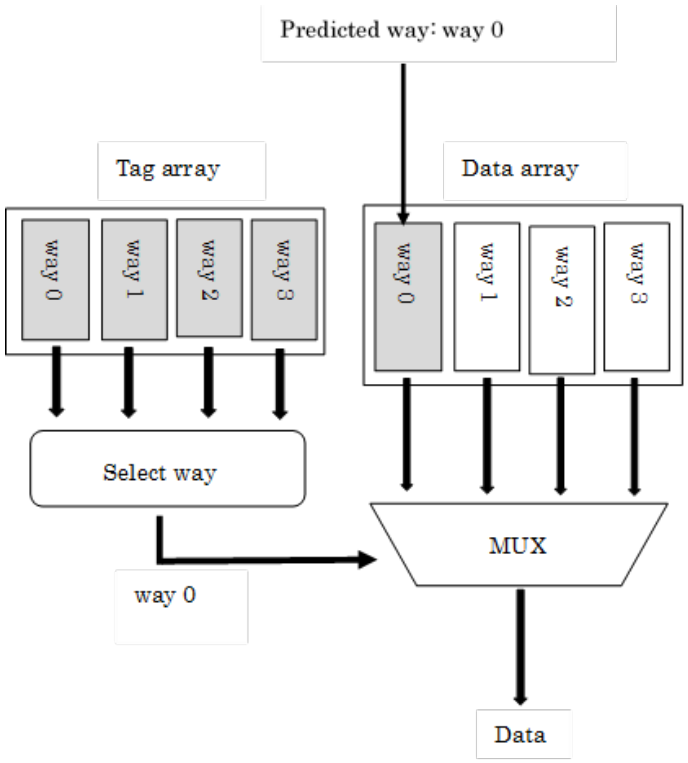

図 2.3: 予測によるウェイ限定参照を行う 4-way セットアソシアティヴ方式のキャッシュアクセス (読み込まれるウェイが影付けされている).

第3章　本研究の提案手法

本章では，事前実行によるメモリアクセスのトレース情報を用いて静的に参照ウェイを限定することによって低ハードウェアコストで電力を削減する手法である TracePC Way prediction(TracePC) 法と，小さなカウンターを用いて参照ウェイを限定する手法である Simple-Counter Way prediction 法の提案を行う．

3.1　TracePC Way predeciton(TracePC) 法

TracePC 法では事前実行によって得るメモリアクセスのトレース情報を用いて各メモリアクセス命令に対して静的に参照すべきウェイを決定し，ウェイ限定参照を行う．

TracePC 法を使ったキャッシュアクセスでは各メモリアクセス命令の PC に対して静的に定められたウェイ番号が与えられ，ウェイを限定参照を行う．また，R-A Cache のキャッシュアクセスのように 1 回目の参照 (probe0) 時にすべてのウェイのタグ配列を読み込み，同時に限定されたウェイのデータのみを読み込む．タグ比較の結果，予測したウェイがヒットしていた場合にはアクセスが完了する．もしウェイの予測が外れていた場合には 2 回目の参照 (probe1) を行う．Probe1 では probe0 のタグ比較の結果に基いて参照を行うため，ヒットしているウェイのデータのみを読み込む．

また，他のウェイ予測手法が用いてるようなハードウェアテーブルや複雑な構造を必要としない．事前実行時のウェイ使用状況と実際の実行時のウェイ使用状況を近づけるために，実装する際にはプログラムの実行時に LRU 情報をリセットするハードウェアを備える．

3.1.1　メモリアクセスのトレースと参照ウェイの決定

まず，TracePC 法ではプログラムの事前実行によりメモリアクセスのトレース情報を得る必要がある．トレース情報は，メモリアクセス命令の PC 値と参照したウェイ番号を使用する．事前実行はキャッシュブロック置き換えアルゴリズムとして LRU を使用しているセットアソシアティヴ方式によって行う．本研究では，このトレース情報を CPU シミュレータである SimpleScalar3.0 を用いてキャッシュメモリシミュレーションを行い取得した．SPEC2000 の各プログラムの 1 億の命令に対してキャッシュシミュレーションを行い，その中のメモリアクセス命令のトレース情報を解析する．

TracePC 法はトレース内の各メモリアクセス命令の PC 値に対するヒットウェイをカウントし，最も多くアクセスされたウェイを限定参照するウェイとして決定する．ウェイ限定参照の予測精度を高めるためにトレース情報をフェーズに分けてカウントし，それぞれのフェーズの各メモリアクセス命令に対して限定参照すべきウェイを与える．そのため，同じメモリアクセス命令であったとしても，フェーズが異なる場合は異なるウェイへの限定参照となる場合が存在する．本研究では全体の命令数が 1 億であるトレース情報に対して 100 フェーズに分けて (1 フェーズの命令数は 100 万)，限定参照ウェイを決定した．フェーズ分けを行うことにより，フェーズ分けを行わない場合と比較してウェイ予測の精度が向上する．

3.1.2 カウントのフェーズ分け

予測精度の向上を図るために，TracePC 法ではメモリアクセスのトレースに対してフェーズに分けてカウントしている．そして，10，100，1000，10000 フェーズに分けての予測精度の調査を行ったが，フェーズの分け方による予測精度には大きな差は生じなかった．そのため，TracePC 法は僅かではあるが予測精度が最も高かった 1 億命令を 100 フェーズに分けたトレース解析法を採用している．

3.2 Simple-Counter Way prediction(SC) 法

Simple-Counter Way prediction(SC) 法ではプロセッサ内に少数のカウンターを用意し，動的に参照ウェイ予測することによってウェイ限定参照を行う．キャッシュアクセスは，TracePC 法と同様に R-A Cache のキャッシュアクセスの様に行われる．

3.2.1 ハードウェアカウンタと参照ウェイの決定

SC 法では動的なウェイ限定参照を行うためにハードウェアカウンタを必要とする．そのサイズは 1 つのウェイに対して 9 ビットであり，PSA や R-A Cache が用いるテーブルのような大きなサイズではない．プログラムの実行中に，メモリアクセス命令においてヒットしたウェイをカウントする．フェーズに分けてカウントし，カウントした結果，一番参照数の多いウェイが次のフェーズで限定参照するウェイとなる．次のフェーズに移った場合にはカウンター値をリセットした上でカウントを行う．一番最初のフェーズでは限定参照するウェイを決定するためのカウント情報が存在しないため，0 番目のウェイを限定参照する．

3.2.2　カウントのフェーズ分け

予測精度の向上を図るために，SC 法では実行中のプログラムをフェーズに分けてカウントしている．

本研究では，10，100，1000，10000，100000，200000 フェーズに分けて調査した結果，フェーズ数を増やす毎にウェイ予測の精度が向上することが観測できた．そのため，本研究で SC 法とは 1 億命令を 20 万フェーズに分けた SC 法を指す．このことから，500 命令の中のメモリアクセス命令でヒットするウェイをカウントする．

第4章　評価

本章では提案手法のデータアクセス時の消費電力量の削減効果を評価する．また，提案手法と PSA と R-A Cache のキャッシュアクセスのウェイ予測限定参照の予測精度について評価する．

4.1　評価環境

評価は SimpleScalar3.0[4] を用いてキャッシュシミュレーションを行って得られるメモリアクセス情報を提案手法のシミュレータ (C 言語で記述) 上で実行することによって，データアクセス時の消費電力量を算出することで行った．また，評価に用いるベンチマークプログラムは SPEC2000 ベンチマーク [5] の中から 18 種類を使用した．そして，評価の際のプログラムの入力として ref を使用する．

評価した L1 データキャッシュは probe0 ですべてのウェイのタグ配列と予測されたウェイのデータのみを参照する 4 ウェイセットアソシアティヴ方式である．L2 キャッシュのブロック格納方式は従来の 4 ウェイセットアソシアティヴ方式である．L1 データキャッシュの容量は 16K バイト，L2 キャッシュの容量は 128K バイト，キャッシュブロックサイズは 32 バイト，命令長は 32 ビットとする．1 つのウェイのデータ配列 (4K バイト) へのアクセスに掛かる消費電力量を 1 としてデータアクセス時の消費電力量の評価を行う．

4.2　比較対象の手法の設定

提案手法の比較対象として，PSA と R-A Cache をあげる．PSA は動的予測のために 1024 エントリーからなるテーブルを使用し，メモリアクセス命令の際に参照するため，テーブル参照による消費エネルギーも考慮する．1 エントリーは 2 ビットからなるため，サイズは 256 バイト (2048 ビット) である．また，R-A Cache もメモリアクセス命令の際にそれぞれ 128 エントリーから成る APT と BWT を参照する．APT は 1 エントリー 51 ビットからなり，BWT は 1 エントリー 26 ビットからなるため，テーブルのサイズは合計 1392 バイトである．

4.3　提案手法の設定

提案手法である TracePC 法ではプログラムの事前実行が必要となるが，評価に使用するプログラムへの入力と異なる入力を与えたプログラムを事前実行することにより静的に参照ウェイを限定しておく．評価の SPEC2000 ベンチマークの入力には ref を使用し，事前実行を行うプログラムには入力として train を与える．

同じく提案手法である SC 法ではテーブルを用いて動的な予測をするが，テーブル全体のサイズは 4 バイトという極めて小さいサイズであるため，テーブル参照の際の消費電力量を評価において考慮しない．

4.4　評価結果

4.4.1　提案手法のデータアクセス時の消費電力量

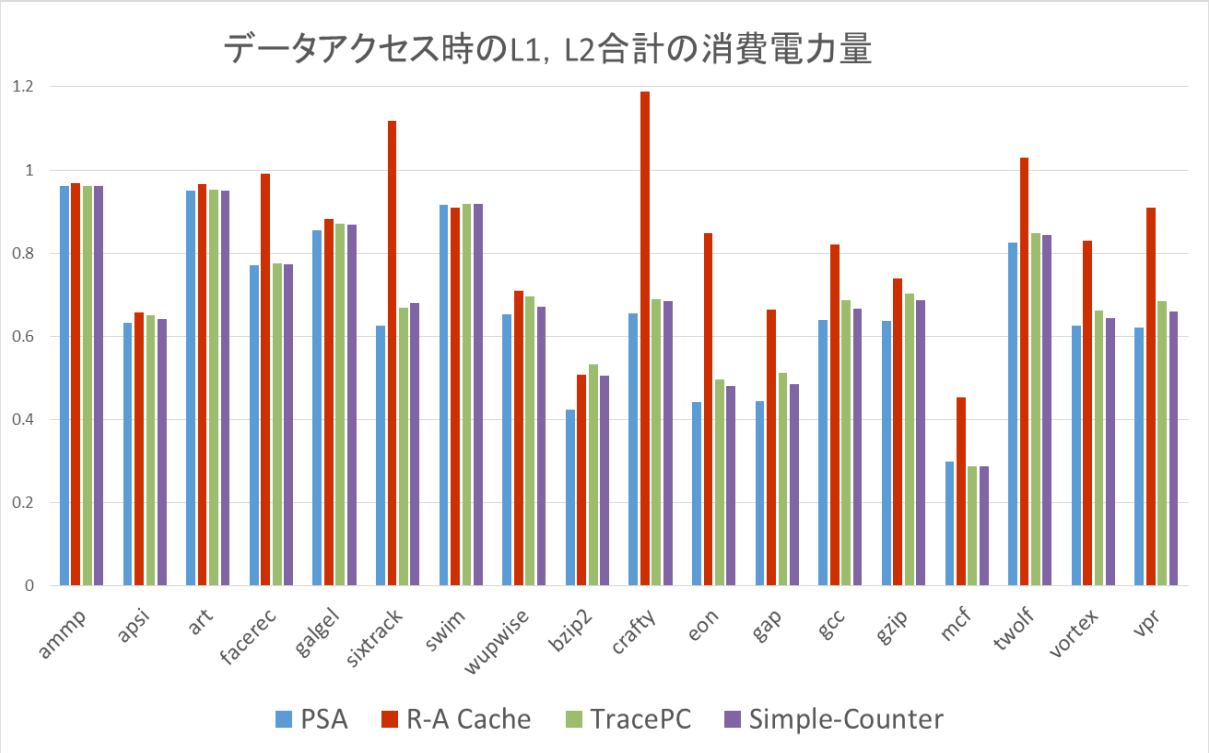

図 4.1: 各手法のデータアクセス時のプログラム毎の消費電力量比．

図 4.1 に L1 データキャッシュに従来の 4 ウェイセットアソシアティヴ方式を採用した場合の消費電力量を基準として，各手法のデータアクセス時のプログラム毎の消費電力量比を示す．各プログラムに対して 4 つの棒が存在する．4 つの棒は左から PC をインデック

スとしたPSA，PCをインデックスとしたRA-Cache，TracePC，SCの消費電力量の比となっている．プログラムは一番左のammpからwupwiseまでが浮動小数点型のプログラムSPECfpであり，bzip2からvprまで整数型のプログラムSPECintとなっている．

R-A Cacheは3つのプログラム(sixtrack,crafty,twolf)で消費電力量を削減することができていなかった．これは，R-A Cacheは従来のセットアソシアティヴ方式に比べてデータキャッシュミス率が増加するからである．そのためL2キャッシュの参照回数が他の方式よりも多くなり，結果としてデータアクセス時の消費電力量が増加する．図4.2に従来のセットアソシアティヴ方式とR-A Cacheのプログラムごとのデータキャッシュヒット率を示す．従来のセットアソシアティヴ方式よりも消費電力量が多くなっているsixtrack，crafty，twolfのR-A Cacheのデータキャッシュヒット率は，従来のセットアソシアティヴ方式に対して約3%低下している．一方，PSAと提案手法であるTracePC法とSC法は評価の全体を通してデータアクセス時の消費電力量を削減することが可能であることが観測できる．

平均で，PSAは33.5%，R-A Cacheは15.6%，提案手法であるTracePC法とSC法はそれぞれ平均30.0%，31.1%の消費電力量の削減効果となった．

	conventional setassociative	R-A Cache
program		
ammp	45.47%	45.13%
apsi	97.98%	97.63%
art	60.92%	60.23%
facerec	95.11%	93.06%
galgel	89.06%	88.62%
sixtrack	97.37%	94.59%
swim	79.83%	79.83%
wupwise	96.71%	96.48%
bzip2	99.37%	99.16%
crafty	97.36%	94.18%
eon	99.60%	98.02%
gap	99.44%	98.57%
gcc	97.37%	96.34%
gzip	97.12%	96.64%
mcf	99.87%	99.53%
twolf	91.14%	88.69%
vortex	97.92%	96.74%
vpr	97.47%	95.94%

図 4.2: 従来のセットアソシアティヴ方式とR-A Cacheのプログラムごとのデータキャッシュヒット率．

4.4.2　限定参照の予測精度

図 4.3 と図 4.4 に PSA と TracePC 法と SC 法の probe0 でのヒット率を示す．図 4.3 ではプログラムに対して棒が 3 つ存在し，左から PSA 法，TracePC 法，SC 法の probe0 ヒット率となる．probe0 ヒット率は予測限定参照においてヒットする確率を表し，予測限定参照によるヒット回数を全体キャッシュヒット回数で除算することで求める．

まず，PSA の平均 probe0 ヒット率は約 69%である．それに対して本研究の提案手法である TracePC 法と SC 法はどのプログラムに対しても予測の精度は劣り，それぞれ平均 36%，平均 43%のヒット率である．しかしながら，ammp，sixtrack，wupwise，mcf の 4 つのプログラムに対しては，約 42%から約 97%以上の予測精度を持っている．これは，これらのプログラムはプログラムに対する入力が変わったとしてもメモリアクセス命令の PC 値とヒットするウェイに関連があると推測される．R-A Cache はデータキャッシュヒット率が低下するが，約 96%の高い平均 probe0 ヒット率を持つ．

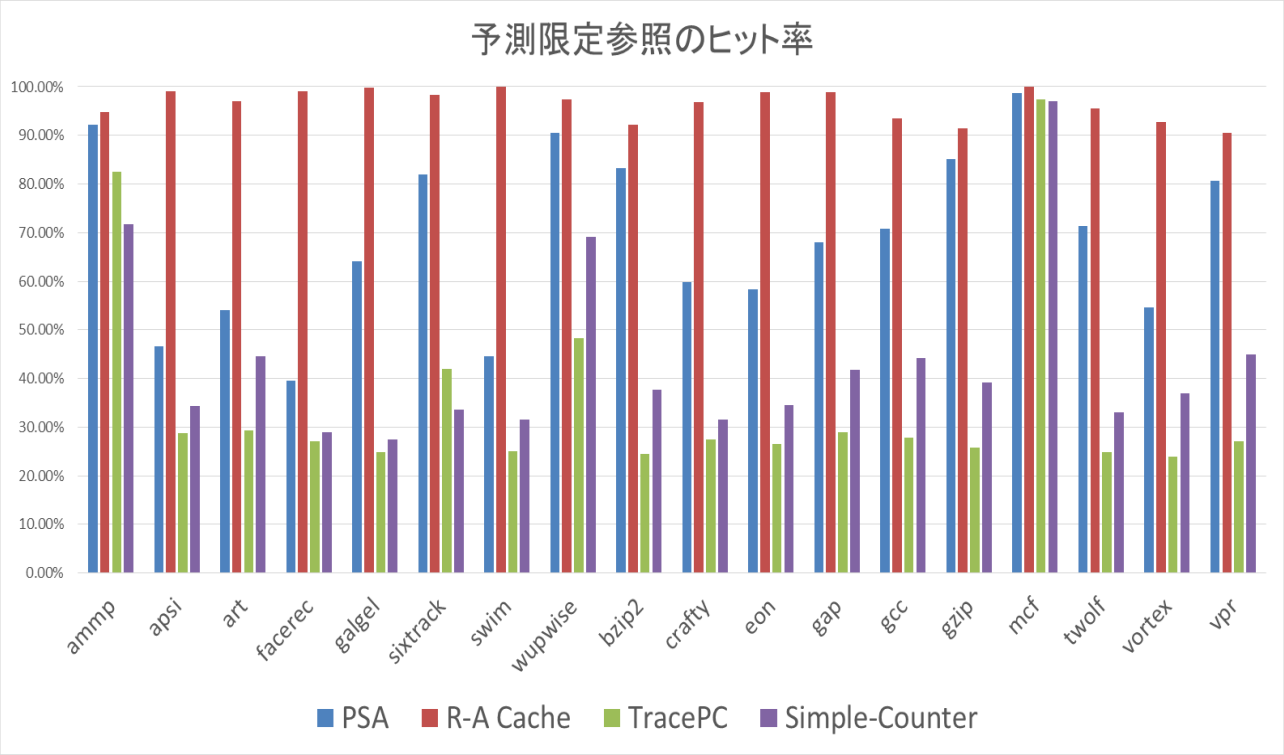

図 4.3: PSA，R-A Cache，TracePC と SC の probe0 ヒット率．

program	PSA	R-A Cache	TracePC	Simple-Counter
ammp	92.16%	94.82%	82.59%	71.67%
apsi	46.58%	99.03%	28.71%	34.35%
art	54.02%	97.07%	29.29%	44.49%
facerec	39.47%	99.04%	27.11%	28.93%
galgel	64.09%	99.85%	24.92%	27.36%
sixtrack	81.96%	98.30%	42.02%	33.50%
swim	44.64%	99.99%	25.11%	31.56%
wupwise	90.45%	97.39%	48.35%	69.05%
bzip2	83.29%	92.22%	24.43%	37.64%
crafty	59.73%	96.78%	27.50%	31.46%
eon	58.32%	98.96%	26.49%	34.59%
gap	68.03%	98.80%	28.94%	41.82%
gcc	70.87%	93.45%	27.79%	44.23%
gzip	85.19%	91.40%	25.83%	39.15%
mcf	98.73%	99.99%	97.31%	97.11%
twolf	71.39%	95.49%	24.83%	33.05%
vortex	54.57%	92.67%	23.92%	36.92%
vpr	80.72%	90.59%	27.04%	44.95%

図 4.4: PSA，R-A Cache，TracePC と SC の probe0 ヒット率.

第5章　まとめ

最後に本研究のまとめを行い，今後の課題について述べる．

5.1　まとめ

本研究では大きな追加ハードウェアを用いることなくウェイ予測限定参照を行う手法として，TracePC Way prediction(TracePC) と Simple-Counter Way prediction(SC) の提案を行った．TracePC 法は事前実行により得たメモリアクセスのトレース情報により，限定参照を行うウェイを静的に決定する．SC 法は極めて小さなサイズのハードウェアカウンタを使って，動的に限定参照を行うウェイを決定する．SPEC2000 ベンチマークプログラムを用いて，提案手法のデータアクセス時の消費電力量の評価とウェイ限定参照の予測の精度の評価を行った．評価の結果，TracePC 法はプログラムへの入力が変わったとしてもメモリアクセス命令の PC 値とヒットするウェイに関連があると推測されるプログラムにおいて，高いウェイ予測精度があり，データアクセス時の消費電力量の削減効果を持つことが示された．SC 法は極めて小さなハードウェアカウンタのみで平均 43%のウェイ予測精度を持ち，ハードウェアコストを抑えたウェイ予測限定参照として有効であり，平均 31.1 %のデータアクセス時の消費電力量の削減に成功した．

5.2　今後の課題

ウェイ限定参照の予測ミスによる性能低下を考慮した評価

本研究では従来の 4 ウェイセットアソシアティヴ方式に対して，ウェイ予測手法である PSA，R-A Cache と提案手法の比較を行い評価したが，限定参照のミスによって生じるオーバーヘッドを含めていない．よって，時間的オーバーヘッドを含めたより現実に近い評価を行う必要がある．

評価に使用したベンチマークプログラムの振る舞いの調査

提案手法の TracePC 法の評価において，プログラムへの入力が変わったとしても PC 値とヒットするウェイに関連があると推測されるプログラム (ammp，sixtrack，wupwise，mcf) が存在した．これらのプログラムの振る舞いを調査し，PC 値とヒットするウェイの関連性を明確にする必要がある．

提案手法の実装方法

提案手法の TracePC 法と SC 法の具体的な実装方法について考案する必要がある．

TracePC 法のウェイ予測精度の向上

ベンチマークプログラムへの入力のデータサイズが大幅に異なることを原因としてヒットするウェイがずれているプログラムが存在している可能性がある．ヒットするウェイがずれている場合においては，全体キャッシュミス時に LRU 管理ではなく限定参照を行ったウェイのブロックを置き換えることによりウェイ予測の精度の改善が見込める．このような改良型の置き換えアルゴリズムを採用した TracePC 法を評価する必要がある．

謝辞

本研究を完成させるために，終始熱心にご指導を頂いた田中清史准教授に深く感謝致しますと共に，ここに御礼申し上げます．

また，研究に対してご助言と意見をくださった，井口寧教授，金子峰雄教授，請園智玲助教授に深く感謝致します．

その他，日頃より支えて頂いた田中研究室をはじめとする先輩，友人，家族に対して御礼申し上げます．

参考文献

[1] B. Calder, D. Grunwald, and J. Emer. Predictive sequen- tial associative cache. In Proceedings of the Second IEEE Symposium on High-Performance Computer Architec- ture, Feb. 1996.

[2] B.Batson and T. N. Vijaykumar. Reactive associative caches. In proceedings of International Conference on parallel Architecutures and Compiliation, 2001.

[3] Michael D. Powell, Amit Agarwal, T. N. Vijaykumar, Babak Falsafi and Kaushik Roy, “Reducing Set-Associative Cache Energy via Way-Prediction and Selective Direct-Mapping” 2001.

[4] SimpleScalar <http://www.simplescalar.com/> (accessed 2015/02/09)

[5] SPEC2000 <http://www.spec2000.com/> (accessed 2015/02/09)