Vol.2011-ARC-196 No.23
2011/7/28

BTBへのBimode Cascading手法適用による分岐先アドレス予測の高効率化

石井康雄[†1,†2] 畔柳圭佑[†1]
稲葉真理[†1] 平木敬[†1]

本稿では，高精度な分岐先アドレス予測を省資源なハードウェア構成で実現するBimode Cascading BTB (BC-BTB) を提案する．BC-BTB はセットアソシアティブ型の BTB と Bimode Cascading ITTAGE 分岐予測のデータパスを融合した分岐先アドレス予測方式である．複数の分岐先アドレスを持つ間接分岐の予測には 1 つの BTB に対して 2 度の性質の異なる参照をすることで，専用のハードウェア資源を持つことなく高精度な分岐先アドレス予測を実現する．BC-BTB を CBP3 Framework で評価した結果，既存の省資源型分岐先アドレス予測方式と比較して同程度のハードウェアリソースで分岐予測ミスを 21.3%削減できることがわかった．

Cost Efficient Branch Target Prediction by Combining BTB and Bimode Cascading

Yasuo Ishii,[†1,†2] Keisuke Kuroyanagi,[†1] Mary Inaba[†1] and Kei Hiraki[†1]

In this paper, we propose Bimode Cascading BTB (BC-BTB). BC-BTB achieves accurate branch target prediction with small hardware cost because it reuses the branch target buffer to predict indirect branches that takes multiple targets. BC-BTB combines data path of a set-associative branch target buffer and Bimode Cascading ITTAGE indirect branch predictor. For indirect branch that takes multiple targets in a workload, BC-BTB makes two lookups that have different features. It reduces the dedicated hardware cost for the indirect branch target prediction. We evaluate BC-BTB by championship branch prediction framework. The simulation result shows that BC-BTB reduces the branch misprediction by 21.3% from existing cost-effective branch target predictor.

1. はじめに

近年，レジスタ間接分岐を頻繁に利用するオブジェクト指向言語の広まりとともに，レジスタ間接分岐の高速化の要求が高まっている．この要求から，Intel や IBM の最新のプロセッサは専用の分岐先アドレス予測器を導入している[11]．レジスタ間接分岐は，単一の間接分岐命令がプログラム中で単一の分岐先アドレスのみを持つ **Monomorphic 分岐**と，単一の間接分岐命令が複数の分岐先アドレスを持つ **Polymorphic 分岐**に分類できる．これらの分類のうち Polymorphic 分岐の分岐先アドレス予測は複数の分岐先アドレスから 1 つの正しい分岐先アドレスを決定する必要があるため，Taken/NotTaken を予測する分岐方向予測と比較して難しいとされる．

分岐先アドレス予測には古くから Branch Target Buffer(BTB)[7] が用いられてきた．BTB は単一の分岐命令を単一の分岐先アドレスに関連付けるため，Monomorphic 分岐は効率的に予測できるが，Polymorphic 分岐を正しく予測することは不可能である．Target Cache[2] は Polymorphic 分岐の予測を行うために，分岐履歴を用いることで単一の分岐命令を複数の分岐先アドレスに関連付ける．近年では，さらに高い予測精度を実現するために，BTB 方式に基づく分岐先アドレス予測機構と Target Cache に基づく分岐先アドレス予測機構を組み合わた Cascaded Predictor[3] が提案され，分岐先アドレス予測の精度を高めている．Cascaded Predictor は Monomorphic 分岐を効率の良い BTB 方式で予測し，残った Polymorphic 分岐だけを Target Cache 方式で予測することで高い予測精度を実現する．この方式は ITTAGE 分岐予測器[10] や Bimode Cascading ITTAGE (BCTAGE) 分岐予測器[4] といった手法で拡張され，分岐予測精度は大幅に改善されている．しかし，これらの方式は Polymorphic 分岐の分岐先アドレス予測のために専用のハードウェア資源が必要で，Polymorphic 分岐を用いないプログラムではその資源が無駄になるという問題点がある．Rehashable BTB[8] や VPC 予測[6] は BTB を Polymorphic 分岐の分岐先アドレス予測のために活用するため専用のハードウェア資源が不要である．しかし，これらの方式は BTB を Polymorphic 分岐の予測のために利用するという制約から予測アルゴリズムが単純なものとなるため予測精度は十分ではない．

†1 東京大学
The University of Tokyo
†2 日本電気株式会社
NEC Corporation

Vol.2011-ARC-196 No.23
2011/7/28

本稿では，BTB のハードウェアリソースを Polymorphic 分岐の分岐先アドレス予測に用いる高精度な分岐先アドレス予測方式 Bimode Cascading BTB (BC-BTB) を提案する．BC-BTB は BTB のハードウェア構成に対して Bimode Cascading 方式を組み合わせたものである．BC-BTB はセットアソシアティブ方式の BTB のハードウェア構成に対して最小限のデータパス構成の変更を施し，効率的に Polymorphic 分岐の分岐先アドレスを予測する．BC-BTB は，Polymorphic 分岐の分岐先アドレスを効率的に予測するために Monomorphic 分岐用と Polymorphic 分岐用の 2 回に分けた BTB 参照を行う．最初に Monomorphic 分岐用の参照を行い，その時に Polymorphic 分岐であることがわかった場合にのみ 2 度目の Polymorphic 分岐用の BTB 参照を実施する．Polymorphic 分岐用の BTB 参照では ITTAGE 分岐予測器などで用いられるマルチステージ型カスケード予測を用いるため，既存の単純なアルゴリズムを用いる省資源型分岐先アドレス予測方式と比べて高い予測精度を実現することができる．

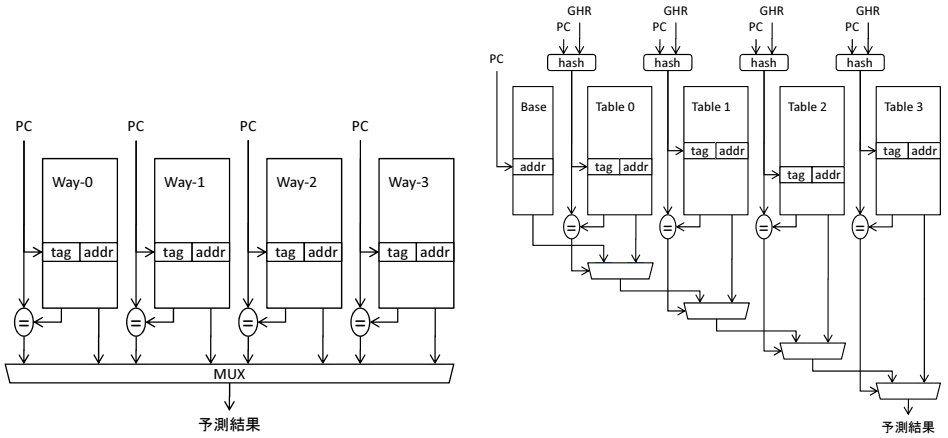
本稿の構成は以下のとおりである．2 章で既存の分岐先アドレス予測方式に関して述べる．3 章で提案手法である BC-BTB を提案する．4 章で評価を行い，5 章でまとめる．

2. 分岐先アドレス予測方式

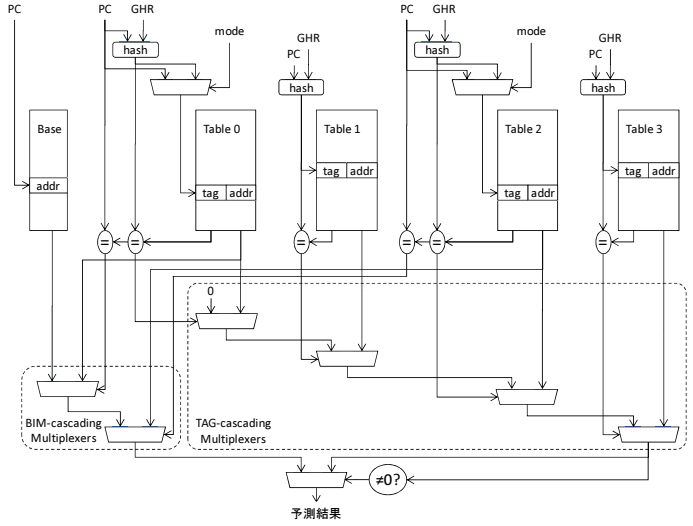
Branch Target Buffer (BTB)[7] はプログラムカウンタ (PC) の値からフェッチラインに含まれる分岐命令の検出を行い，次にフェッチする命令アドレスを決定する分岐先アドレス予測方式である．一般に，セットアソシアティブ型のハードウェア構成をとることが多い．**図 1**(a) に 4 ウェイの BTB 構成を示す．BTB の参照は命令フェッチに用いる PC の下位ビットによってエントリを読み出し，読みだされたエントリのタグ部分が PC の上位ビットと一致するかを調べる．タグ部分の比較が一致した場合には，そのエントリの予測結果が有効になる．BTB は PC と分岐先アドレスを必ず一対一で対応付けるため，Monomorphic 分岐の予測においては効率の良い方式であるが，Polymorphic 分岐を正確に予測することができない．

BTB の入力情報を PC から分岐履歴と PC のハッシュ値に変更したものは Tagged Target Cache (TTC)[2] と呼ばれる．分岐履歴としては過去の分岐の Taken/NotTaken 履歴であるグローバル履歴や過去の分岐の分岐先アドレスのハッシュ値を用いるターゲット履歴などが用いられる．TTC は分岐履歴を利用することで単一の分岐命令と複数の分岐先アドレスの対応付けが可能で，Polymorphic 分岐の予測を行うことができる．

ITTAGE 分岐予測器[10] は BTB と TTC をハイブリッドした方式である．図 1(b) にそ



(a) Branch Target Buffer　　(b) ITTAGE 分岐予測器



(c) BCTAGE 分岐予測器

図 1　Branch Target Buffer のハードウェア構成.
Fig. 1　Block Diagram of Branch Target Buffer.

の構成を示す．ITTAGE 分岐予測器は Monomorphic 分岐を BTB と似た特性を持つ Base 予測器を用いて予測し，残った Polymorphic 分岐をその他の予測テーブルを用いて予測する．Polymorphic 用の予測テーブルは各々が PC と分岐履歴を入力とするハッシュ関数を持ち，そのハッシュ値で予測テーブルは参照される．各ハッシュ関数の入力の分岐履歴長は Table-0 が最小，Table-n(図 1(b) では n=3) が最大となるように割り当てられ，それぞれのテーブルに割り当てられる履歴長は等比数列の関係になる．例えば，図中の例では Table-0 が履歴長 3，Table-1 が履歴長 6，Table-2 が履歴長 12，Table-3 が履歴長 24 といった形で割り当てられる．各テーブルの出力はカスケード接続され，後段の予測結果がより優先される．従って，出力結果は最も長い分岐履歴を用いてヒットしたエントリの予測となる．ITTAGE 分岐予測器は非常に高い予測精度を持ち，2011 年に開催された Championship Branch Prediction[1] では決勝進出の 4 方式中 3 方式が ITTAGE 分岐予測器を利用した予測方式だった．

Bimode Cascading ITTAGE (BCTAGE) 分岐予測器[4] は間接分岐のふるまいの多くが Monomorphic 分岐であるワークロードが多数あることに着目し，ITTAGE 分岐予測器の Polymorphic 分岐の予測のための資源を BTB の拡張部分として利用出来るようにしたものである．そのデータパスを図 1(c) に示す．BCTAGE 分岐予測器ではテーブル間のカスケード接続を Monomorphic 用 (BIM cascading multiplexers) と Polymorphic 用 (TAG cascading multiplexers) に多重化し，様々なワークロードに対応する．例えば，Monomorphic 分岐が多いワークロードでは Polymorphic 用のハードウェア資源を Monomorphic 用に利用する．一つの予測テーブルを Monomorphic 分岐用と Polymorphic 分岐用と振り分けることでハードウェアの利用効率を高めることができる．

3. Bimode Cascading BTB (BC-BTB)

本稿では，BCTAGE 方式に着想を得て時系列でデータパスの特性を変更し，様々なワークロードに対して適切な分岐先アドレス予測を行う Bimode Cascading BTB (BC-BTB) を提案する．BC-BTB はセットアソシアティブ方式の BTB のデータパスを変更することで Polymorphic 分岐の分岐先アドレスを効率的に予測する方式である．

BC-BTB は一つの分岐命令に対して性質の異なる 2 度の BTB 参照をして予測を行う．2 度の BTB 参照は Monomorphic 分岐に特化したものと Polymorphic 分岐に特化したものから構成される．前者の BTB 参照を **Monomorphic 参照**，後者の BTB 参照を **Polymorphic 参照**と呼ぶ．Polymorphic 参照は Monomorphic 参照で検出した分岐命令が Polymorphic 分岐であるとわかった場合にのみ実施され，そのため，Polymorphic 分岐が発生しないプログラムでは BC-BTB は通常の BTB と完全に同じ動作をする．

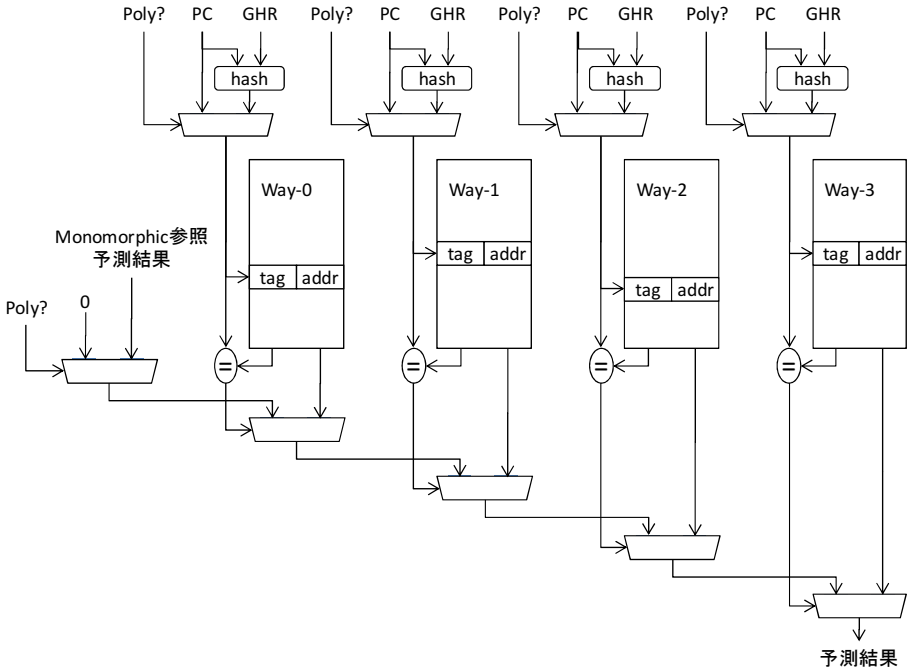


図 2 Bimode Cascading Branch Target Buffer のハードウェア構成.
Fig. 2 Block Diagram of Bimode Cascading Branch Target Buffer.

3.1 ハードウェア構成

図 2 に BC-BTB のデータパス構成を示す．BC-BTB は，従来の BTB と同じく複数のウェイを構成する RAM，その出力のタグ一致を判定する比較論理，出力を決定する選択論理部分から構成される．BC-BTB は以下の 3 点の変更を従来の BTB に加える．(1) 各エントリに対して Polymorphic 分岐であることを示す P ビットを追加．(2) 各ウェイに対して専用のハッシュ関数を付与．(3) 各ウェイの結果を選択する論理を優先度付きセレクタに変更．これらの変更は小規模な改造であり TTC のような大幅なハードウェア追加を必要としない．

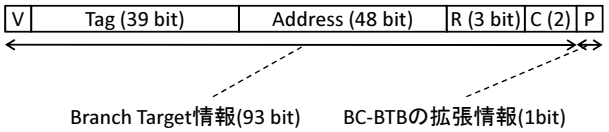


図 3 BC-BTB のエントリ構成.
Fig. 3 Table Entry in the Bimode Cascading BTB.

各エントリのビットフィールドを**図 3** に示す．各エントリは 1 ビットの有効ビット (V フィールド)，タグ，分岐先アドレス，3 ビットの置き換え情報 (R フィールド)，2 ビットの信頼度情報 (C フィールド) を持ち，さらに Polymorphic 分岐を示す P ビットを持つ．実質的なハードウェア量の増加は各エントリに対する P ビットの 1 ビットの追加のみである．48 ビットアドレスを前提とした 4096 エントリ 8 ウェイの BC-BTB では各エントリの容量は 94 ビットであり，ハードウェア量の増加は 1.1%となる．

各ハッシュ関数は分岐履歴と PC を排他的論理和を用いて折りたたみハッシュ値を生成する．得られたハッシュ値は Polymorphic 参照の場合に限りセレクト論理に選択され RAM の入力として利用される．ITTAGE 分岐予測器と同様に各ハッシュ関数には異なる履歴長の分岐履歴を関連付ける．関連付ける履歴長は各ウェイの関係が等比数列となるように割り当てる．なお，Monomorphic 参照では PC がセレクト論理によって選択され，従来の BTB と同じ方式で予測を行う．

優先度付きセレクタはウェイ m とウェイ n (n > m) の両方にヒットした場合に，ウェイ n を選択するという論理を構成する．この構成変更によるセレクト論理の深さは変更がないため，パイプラインステージを変更することなく BTB を BC-BTB で置き換えることが可能である．

BTB の置き換えポリシーは Re-reference interval prediction (RRIP)[5] を用いる．すなわち，初期化時に R フィールドに 1 が書きこまれ，そのエントリに参照があると R フィールドの値は最大値の 7 へと変化する．置き換えの際には R フィールドの値が最小のエントリが選出される．置き換え対象エントリが選択されると置き換え対象エントリの R フィールドの値の分だけ他の置き換え候補の R フィールドが減算される．

3.2 予 測 方 式

BC-BTB では Polymorphic 分岐の予測を Monomorphic 参照と Polymorphic 参照の 2 つのフェーズに分割することで，大きなハードウェアの追加なしで効率的に分岐先アドレスの予測を実施する．

Monomorphic 参照は通常の BTB 参照と同じく，命令キャッシュ参照などの他のパイプラインステージと同期して行われる．Monomorphic 参照では PC が各テーブルの入力として用いられる．このときにヒットしたエントリが P=0 の場合には，Monomorphic 分岐と判断し結果をパイプライン中に出力して分岐先アドレス予測を終了させる．エントリが P=1 の場合には，後続のフェッチ命令を破棄して Polymorphic 参照を行う．

Polymorphic 参照では ITTAGE 分岐予測器などで採用されるマルチステージ型カスケード予測を行う．全てのウェイで，そのウェイと関連するハッシュ値を用いてテーブル参照を行い，それぞれのテーブルから予測結果を得る．ここで得られた予測のうちで最長の分岐履歴を用いたウェイの結果を最終的な予測結果として採用する．もしも，全てのウェイでミスが発生した場合には，Monomorphic 参照で得られた分岐先アドレスを予測結果として採用する．また，マルチステージ型カスケード予測では R=0，かつ，C=0 のケースで最長の履歴長を用いた予測結果が最適でない場合があることが知られている[10]．このような場合には 2 番目に長い履歴長を用いてヒットしたエントリから得られた予測結果を最終的な予測結果とし出力する．もしも，Polymorphic 参照でヒットしたエントリが 1 つだけだった場合には，Monomorphic 参照の結果を 2 番目に長い履歴長を用いた予測結果として使用する．この予測の上書きを行うかは IITAGE 分岐予測器と同じく飽和カウンタを用いて決定する．予測に用いた情報はエントリの更新に利用するため，その他の分岐予測情報とともに記憶しておく．

3.3 更 新 方 式

分岐命令のリタイア時にハードウェア資源の更新を行う．この節では分岐先アドレス予測に成功した場合，分岐先アドレス予測に失敗した場合に分けて BC-BTB の動作を説明する．

3.3.1 予測に成功した場合

分岐先アドレス予測に成功した場合には，予測に用いたエントリの置き換え情報と信頼度カウンタの更新を行う．まず，全てのケースにおいて Monomorphic 参照でヒットしたエントリの R フィールドの値を RRIP ポリシーに従って更新する．

もしも，Polymorphic 参照を行いヒットしたエントリがあった場合には，そのエントリの C フィールドと R フィールドの更新を行う．C フィールドは予測に成功したためインクリメントし，R フィールドは 2 番目に長い履歴長を用いた予測結果が正しい予測結果を生成できなかった場合にのみインクリメントする．

もしも，Polymorphic 参照を行わなかった，あるいは，Polymorphic 参照を実施したがヒットしたエントリがなかった場合には，Monomorphic 参照を行ったエントリの C フィー

ルドのインクリメントを実施する．

3.3.2 予測に失敗した場合

分岐先アドレス予測に失敗した場合には，予測に用いたエントリの分岐先アドレス情報，R フィールド，および，C フィールドの更新を行う．まず，全てのケースにおいて Monomorphic 参照でヒットしたエントリの R フィールドを RRIP ポリシーに従って更新する．

もしも，Monomorphic 参照でヒットしたウェイがない場合には，Monomorphic 参照のエントリを確保する．このときには，RRIP ポリシーに基づいて追い出しエントリを決定する．新しく確保したエントリは R=1，C=0，および，P=0 として初期化する．

もしも，Monomorphic 参照でエントリが発見できていたが，P=0 で Polymorphic 参照を行わなかった場合には Monomorphic 参照でヒットしたエントリの更新を行う．C フィールドが 0 の場合には分岐先アドレスを更新エントリに書き込み，そうでない場合には C フィールドのデクリメントを行う．さらに，次回以降の予測で Polymorphic 参照を行うために P=1 に変更し，新しいエントリを Polymorphic 参照のために確保する．エントリの確保に関しては 3.4 節で説明する．

Polymorphic 参照でヒットしたエントリがあった場合には，Polymorphic 参照で最長の履歴長でヒットしたエントリの更新を行う．更新エントリの C フィールドが 0 の場合には分岐先アドレスを更新エントリに書き込み，そうでない場合には C フィールドのデクリメントを行う．R フィールドは 2 番目に長い履歴長の予測結果が正しかった場合にのみデクリメントする．さらに，3.4 節に示す手順で新しいエントリの確保を行う．

3.4 新しいエントリの確保

もしも，Monomorphic 参照でヒットしたエントリがあり，かつ，分岐先アドレス予測が失敗した場合には，その分岐命令が Polymorphic 分岐であることがわかる．この場合，Polymorphic 参照を行う為に新しいエントリの確保を行う．

新しいエントリの確保は，前回の予測で利用した履歴長よりも長い履歴長を利用するテーブルからエントリの確保を行う．たとえば，Polymorphic 参照でヒットしたエントリがない場合には，全てのウェイが置き換え対象のウェイとなり，Polymorphic 参照でウェイ 1 にヒットした場合にはウェイ 2 以降が置き換え対象のウェイとなる．

置き換え対象の候補となったウェイで Polymorphic 参照時に R フィールドが 0 となっているエントリから置き換えエントリをランダムに選択する．R フィールドが 0 のエントリがない場合には，置き換え候補のエントリの R フィールドをデクリメントする．

確保されたエントリはタグ，分岐先アドレスに現状の分岐履歴や分岐結果から得られたタグと分岐先アドレスを書き込み，他のフィールドは R=0，C=0, および，P=0 として初期化する．

表 1 シミュレーション環境の構成
Table 1 Simulation Parameter

環境	CBP3 Framework (トレースシミュレータ)
分岐方向予測器	GEHL 分岐予測器 16 テーブル， 4096 エントリ/テーブル， 6 ビット/エントリ 最大履歴長:200， 最小履歴長:5， Geometrical History Length
分岐先アドレス予測	Branch Target Buffer (4096 エントリ， 8 ウェイ) リターンアドレススタック (64 エントリ)
VPC 予測	最大で 12 回の BTB アクセス
BC-BTB	最大履歴長:120， 最小履歴長:5， Geometrical History Length

4. 評　　価

4.1 評価環境

評価環境は CBP3 Framework[1] を用いて提案した BC-BTB を評価する．評価対象のベンチマークは CBP3 Framework と同時に配布されたトレースから容量無制限の BTB で予測した際に分岐先アドレス予測ミスが多い 10 本を選出した．今回の評価では，BC-BTB による予測精度改善によるプログラムの実行時間の評価 (IPC による評価) は行わない．Cycle accurate な環境での評価は今後の課題としたい．

表 1 に今回評価に用いた分岐予測器の構成を示す．ベースラインの分岐先アドレス予測として 4096 エントリ，8 ウェイの BTB を採用した．また，BTB とあわせて 64 エントリのリターンアドレススタックも利用する．分岐方向予測器は GEHL 分岐予測器[9] を採用した．比較対象として BTB と資源を共用する VPC 予測[6] を用いる．VPC 予測は BTB に対して複数の異なるアドレスでアクセスし，分岐方向予測を用いてその中から一つの分岐先アドレスを選出するという方式である．VPC 予測は BTB を Polymorphic 分岐のための分岐先アドレス予測にも用いる点で BC-BTB と特性が近い．BC-BTB は 8 ウェイの BTB 上に実装されるため 8 つのハッシュ関数を用いる．BC-BTB が用いる分岐履歴情報は分岐方向予測と共用化するため，前章で述べた以上のハードウェア量の増加はない．

BC-BTB と VPC 予測は Polymorphic 分岐の予測では複数回 BTB に対してアクセスを行う方式である．今回の評価では Cycle accurate シミュレーションを実施しないため，複数回のアクセスを BTB に対して行う場合の性能への影響は計測できない．しかし，文献[6]

によると，再度アクセスするオーバーヘッドは小さいとされる．

4.2 各方式の予測精度

予測精度として 1000 命令あたりの分岐予測ミス回数 (MPKI) を測定する．また，それぞれの予測器の特性を明らかにするために発生した分岐予測ミスの要因ごとに採取する．分類に用いるミス要因は，(A) 分岐先アドレス予測が成功し，分岐方向予測が失敗した回数，(B) 分岐方向予測が失敗し，分岐先アドレス予測も失敗した回数，(C) 分岐方向予測が成功し，分岐先アドレス予測が失敗した回数，(D) 無条件分岐で，分岐先アドレス予測が失敗した回数，(E) 間接分岐 (Monomorphic 分岐) で，分岐先アドレス予測が失敗した回数，(F) 間接分岐 (Polymorphic 分岐) で，分岐先アドレス予測が失敗した回数，の 6 種類を用いる．

(A)，(B) は分岐方向予測の失敗が原因による分岐予測ミスである．これは分岐先アドレス予測を改善しても減ることはない．しかし，VPC 予測では分岐方向予測器を分岐先アドレス予測に用いる方式で，分岐方向予測に対する影響が発生する．従って，VPC 予測ではこの分岐予測ミス回数が変化すると考えられる．(C)，(D)，(E) は BTB によって予測可能な Monomorphic 分岐の分岐先アドレス予測が原因の分岐予測ミスである．BC-BTB や VPC 予測は Polymorphic 分岐を予測する為に 1 つの分岐命令に対して複数の BTB エントリを関連付けるため，Monomorphic 分岐のためのエントリ数が減らされ性能悪化の可能性がある．(F) は Polymorphic 分岐が原因の分岐予測ミスである．これは BTB の巨大化によって削減することはできず，この分岐予測ミスを減らすことが BC-BTB や VPC 予測の目的である．

図 4 に各ベンチマークに関して上記 6 分類での 1000 命令あたりの予測ミス回数を示す．まず，BC-BTB が全てのワークロードで性能の向上をしていることが確認できる．特に，CLIENT04 CLIENT06，CLIENT11，INT04，および，MM06 では BTB と比較して Polymorphic 分岐による分岐先アドレス予測ミスを 10%以下に削減しており，その予測精度の効率の良さが確認出来る．また，全てのベンチマークで VPC 予測の予測精度を上回り，VPC 予測から平均で 21.3%の予測ミスの削減を達成出来ることがわかった．

VPC 予測は INT04 のように予測ミスを 1%以下に削減できるケースもある．しかし，CLIENT05 や CLIENT12 のように Polymorphic 分岐での分岐先アドレス予測ミスを削減するが，それ以上に分岐方向予測ミスの増加がそれを上回る場合がある．これは VPC 予測が分岐方向予測を利用して分岐先アドレス予測を行うためで，そのトレーニングが分岐方向予測の精度を低下させていると考えられる．対して，BC-BTB は通常のタグ比較論理を用いて分岐先アドレス予測を行うため，分岐方向予測への影響無く Polymorphic 分岐の予測精度を高めることができる．結果として，VPC 予測は Monomorphic 分岐でも 12.7%の予測ミスの増加がみられるのに対して，BC-BTB の Monomorphic 分岐での予測ミスの増加は 0.5%に留まっている．

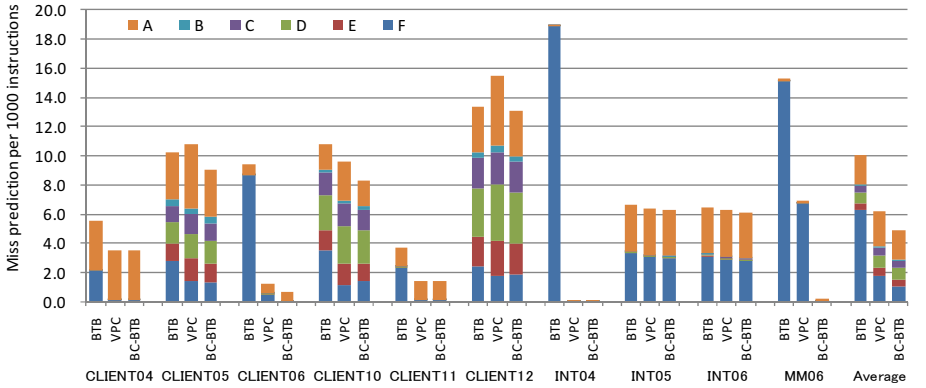


図 4 1000 命令あたりの分岐予測ミス回数とその内訳．
Fig. 4 Miss Prediciton per 1000 Instructions and Breakdown of the Miss Prediction.

以上より，BC-BTB が既存の BTB 資源を有効利用し既存手法を上回る予測精度を実現できることがわかった．

4.3 BTB 容量と予測精度

BC-BTB では BTB 資源を共有する為に分岐予測の容量と通常のターゲット予測との関係が変化することが予測される．そこで，BTB の容量を 1024 エントリから 16384 エントリまで変化させた場合の BC-BTB の予測精度の変化を計測した．なお，すべての構成で BTB の連想度は 8 を採用する．**図 5** に評価結果を示す．4.2 章と同じく全ての BTB 構成において BC-BTB が他方式の予測精度を上回ることを確認できた．以下では各予測ミス原因の詳細に関して議論する．

まず，分岐方向予測のミス回数 (A)，(B) に着目すると，VPC 予測の予測ミス回数が容量の増加に伴い増えていることがわかる．その増加量は 1024 エントリ構成から 16384 エントリの増加で予測ミス回数は 8.1%上昇する．対して BC-BTB での増加量は 0.1%と非常に小さい．

次に，Monomorphic 分岐での予測ミス回数 (C)，(D)，(E) に着目する．1024 エントリからエントリ数を増加するにしたがって Monomorphic 分岐のミスは削減され，16384 エン

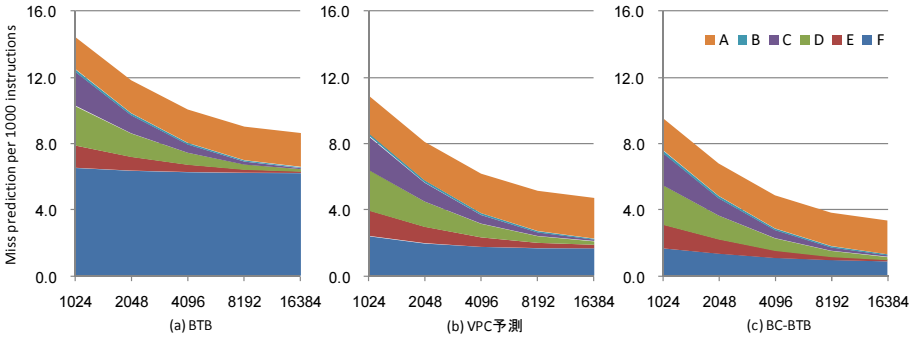


図 5 BTB 容量と予測ミス回数の相関.
Fig. 5 Performance impact of the number of BTB entries.

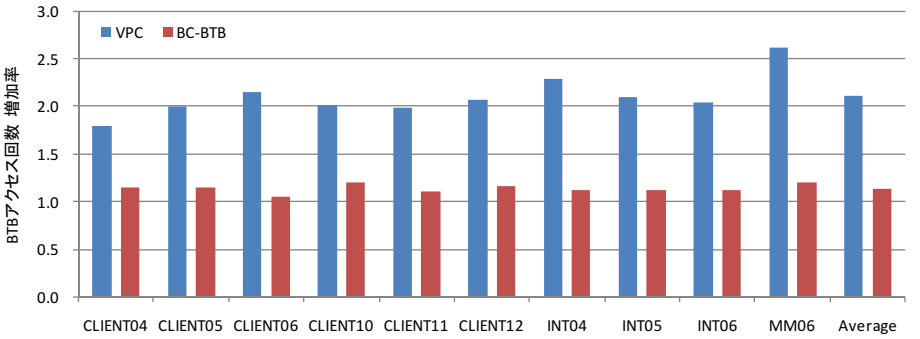


図 6 BTB の読み出し回数の増加率.
Fig. 6 Increasing Rate of BTB Access Count.

トリでは全ての構成で 0.5MPKI を下回る数字となっている．つまり，16384 エントリでは Polymorphic 分岐以外の分岐先アドレス予測ミスがほぼ発生しなくなることがわかる．また，この部分の性能は BTB が全てのケースで最も良い値を実現している．これは BTB が Monomorphic 分岐で最も効率が良いためである．

最後に，Polymorphic 分岐の予測ミス回数 (F) に関して着目する．1024 エントリから 16384 エントリへと増加させる間に，VPC 予測では 31.5%の予測ミスの削減がみられるのに対して，BC-BTB では 47.7%の予測ミスの削減があった．これは，VPC 予測が 1 つの分岐命令に 12 個の分岐先アドレスまでしか割り当てられないのに対して，BC-BTB は関連付けができる分岐命令の個数に上限がないため，利用できるハードウェア資源が増加することでより多くの分岐先アドレスを単一の Polymorphic 分岐に対して関連付け，高い予測精度が実現できているためと考えられる．なお，BTB ではわずか 0.6%の予測ミスの削減に留まる，これは BTB が Polymorphic 分岐を本質的にサポートできないためである．

以上より，BTB の容量にかかわらず BC-BTB が VPC 予測を上回る性能を達成していることが確認できた．

4.4 BTB へのアクセス回数

BC-BTB は VPC 予測と同じく，BTB に対して複数回のアクセスを行うことで予測精度を高める予測方式である．しかし，複数回のテーブル参照を行うと多くの電力を分岐先アドレス予測のために消費してしまう．この影響を見積もる為に BTB へのアクセス回数を評価した．**図 6** は各方式の BTB へのアクセス回数を BTB 方式のアクセス回数を 1 として正規化したものである．

結果，VPC 予測が BTB から 110.5%のアクセス回数の増加であるのに対して BC-BTB は BTB から 13.5%のアクセス回数の増加で留まることがわかった．この傾向は全てのベンチマークに関して同じであり，BC-BTB が VPC 予測から分岐予測ミスを削減させるのみでなく BTB アクセス回数を削減することで電力削減にも貢献できることが確認できた．

以上の評価より，同程度の資源をもった場合に BC-BTB は VPC 予測から平均で 21.3%の予測ミス削減を実現し BTB に対しては 51.5%の予測ミス削減を実現できた．さらに，BTB への参照回数は VPC 予測が 110.5%であるのに対して BC-BTB の参照回数増加はわずか 13.5%であり高精度で省電力な分岐先アドレス予測が実現出来ることがわかった．

5. ま と め

本稿では，BTB を Polymorphic 分岐の分岐先アドレス予測のために活用する高精度な分岐先アドレス予測方式 Bimode Cascading BTB (BC-BTB) を提案した．BC-BTB はプロセッサが持つ分岐先アドレス予測のためのハードウェア資源である BTB を間接分岐の分岐先アドレス予測に対応させることで，専用の予測機構を持つことなく分岐先アドレス予測の精度を向上させてプロセッサの性能向上を目指すものである．BC-BTB は BTB エントリに対する 1 ビットのフラグ追加，分岐履歴を用いるハッシュ関数の各ウェイへの付与，および，予測結果の選択論理に対する最小限のデータパス変更を施すことで，新たな Polymorphic 分岐の予測のための資源を追加することなく高い予測精度を実現する．Polymorphic 分岐

を予測する際には，Monomorphic 参照と Polymorphic 参照の特性の異なる 2 回の BTB 参照を行うことで効率的に分岐先アドレスの予測を行う．

BC-BTB を CBP3 Framework で評価した結果，従来の BTB から平均で 51.5%の分岐予測ミスを削減することが確認できた．また，既存の省資源型分岐先アドレス予測方式からも 21.2%の分岐予測ミスの削減が確認できた．さらに，Polymorphic 分岐の予測のための BTB へのアクセス回数の増加は，VPC 予測が 110.5%であるのに対して BC-BTB では 13.5%であり，高い電力効率を実現する見込みがあることがわかった．今後は，BC-BTB の予測方式の見直しを行いさらに高い予測精度が実現できる予測方式の模索を行うとともに，詳細な評価を行い BTB を再利用する分岐先アドレス予測方式の特性を明らかにしていきたい．

参 考 文 献

1) : 2nd JILP Workshop on Computer Architecture Competitions (JWAC-2): Championship Branch Prediction, http://www.jilp.org/jwac-2/.
2) Chang, P.-Y., Hao, E. and Patt, Y.N.: Target prediction for indirect jumps, *Proceedings of the 24th annual international symposium on Computer architecture*, ISCA '97, pp.274–283 (online), DOI:http://doi.acm.org/10.1145/264107.264209 (1997).
3) Driesen, K. and Hölzle, U.: The cascaded predictor: economical and adaptive branch target prediction, *Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture*, MICRO 31, pp. 249–258 (online), available from ⟨http://portal.acm.org/citation.cfm?id=290940.290993⟩ (1998).
4) Ishii, Y., Sawada, T., Kuroyanagi, K., Inaba, M. and Hiraki, K.: Bimode Cascading: Adaptive Rehashing for ITTAGE Indirect Branch Predictor, *2nd JILP Workshop on Computer Architecture Competitions (JWAC-2)* (2011).
5) Jaleel, A., Theobald, K. B., Steely, Jr., S. C. and Emer, J.: High performance cache replacement using re-reference interval prediction (RRIP), *Proceedings of the 37th annual international symposium on Computer architecture*, ISCA '10, New York, NY, USA, ACM, pp. 60–71 (online), DOI:http://doi.acm.org/10.1145/1815961.1815971 (2010).
6) Kim, H., Joao, J. A., Mutlu, O., Lee, C. J., Patt, Y. N. and Cohn, R.: VPC prediction: reducing the cost of indirect branches via hardware-based dynamic devirtualization, *Proceedings of the 34th annual international symposium on Computer architecture*, ISCA '07, pp. 424–435 (online), DOI:http://doi.acm.org/10.1145/1250662.1250715 (2007).
7) Lee, J. K.F. and Smith, A.J.: Analysis of Branch Prediction Strategies and Branch Target Buffer Design, Technical Report UCB/CSD-83-121, EECS Department, University of California, Berkeley (1983).
8) Li, T., Bhargava, R. and John, L.K.: Rehashable BTB: An Adaptive Branch Target Buffer to Improve the Target Predictability of Java Code, *Proceedings of the 9th International Conference on High Performance Computing*, HiPC '02, pp.597–608 (online), available from ⟨http://portal.acm.org/citation.cfm?id=645448.653035⟩ (2002).
9) Seznec, A.: Analysis of the O-GEometric History Length Branch Predictor, *Proceedings of the 32nd annual international symposium on Computer Architecture*, ISCA '05, Washington, DC, USA, IEEE Computer Society, pp.394–405 (online), DOI:http://dx.doi.org/10.1109/ISCA.2005.13 (2005).
10) Seznec, A. and Michaud, P.: A case for (partially) TAgged GEometric history length branch prediction, *The Journal of Instruction Level Parallelism*, Vol.8 (online), available from ⟨http://www.jilp.org/vol8⟩ (2006).
11) Sinharoy, B., Kalla, R., Starke, W.J., Le, H.Q., Cargnoni, R., VanNorstrand, J.A., Ronchetti, B.J., Stuecheli, J., Leenstra, J., Guthrie, G.L., Nguyen, D.Q., Blaner, B., Marino, C.F., Retter, E. and Williams, P.: IBM POWER7 multicore server processor, *IBM Journal of Research and Development*, Vol.55 (2011).