分岐予測ミスの偏りを利用した分岐予測器の提案

孟　林[†]　小柳　滋[††]

近年のプロセッサではパイプライン段数の増加と命令発行幅の増加により分岐予測ミスのペナルティが増加するため，分岐予測の精度向上がプロセッサの性能向上にとって大きな課題となっている．本稿では，分岐予測ミスが少数の分岐命令に集中して発生していることを利用し，分岐予測ミスの集中する少数の分岐命令についてローカル履歴を利用した予測機構を従来の分岐予測器に付加する手法を提案する．提案手法を，Combining，Bimode，Bimode-Plus，Agree，Hybrid の 5 つの分岐予測器に付加して評価を行った．SPECint2000 では，平均 9%の予測ミスを減らすことができた．

A Branch Predictor Using Miss-prediction Bias

Lin Meng[†] and Shigeru Oyanagi[††]

Modern processors exploit instruction level parallelism by deeper pipeline and wider instruction issue width, which result in an increase of branch miss penalty. Hence, more accurate branch prediction is indispensable for higher performance. This article proposes a novel branch prediction mechanism by using branch miss prediction bias. This mechanism is attached to a conventional branch predictor, and utilizes local history of biased branch instructions. Experiments are done by attaching proposed mechanism to Combine, Bimode, Bimode-Plus, Agree and Hybrid predictors. The result shows that our proposal can reduce 9% of miss prediction rate to the conventional predictors at SPECint2000.

1. はじめに

分岐予測は，分岐命令を投機的に実行することにより制御依存を回避する技術であり，高精度の予測を行うことにより，大幅な性能向上が実現される．しかし，分岐予測に失敗したときは，分岐命令以降の命令実行をフラッシュし，正しい分岐先命令を新たにフェッチするため性能が低下する．これは分岐予測ミスペナルティと呼ばれている．

近年の高性能プロセッサでは，パイプライン段数の増加，命令発行幅の増加により高い命令レベル並列性を抽出しており，これに伴い分岐予測ミスペナルティは増加傾向にある．このため，分岐予測精度の更なる向上はプロセッサの性能向上にとって不可欠な課題である．

分岐予測ミスペナルティを減少させるため，様々な分岐予測器が提案されている．1981 年に最初に提案された Bimodal 予測器[1] では，各分岐命令の挙動を 2 ビット飽和型カウンタで保持し，その値に基づき分岐方向を予測する．飽和型カウンタは PHT(Pattern History Table) として構成され，分岐命令アドレスの下位ビットをインデクスとしてアクセスされる．

現在多くのプロセッサで用いられている Gshare 予測器[2] は，分岐命令のグローバル履歴と分岐命令アドレスの排他的論理和をインデクスとして PHT にアクセスして分岐予測を行う予測器である．これは分岐命令のグローバル履歴により分岐命令間の相関を利用した予測方法である．

これらの分岐予測器における予測ミスの主要な要因として破壊的競合がある．これは，異なる分岐命令が同じ PHT のエントリに分岐結果を登録することにより生じる競合である．現在の主流の分岐予測器では Gshare 予測器のように分岐命令アドレスとグローバル分岐履歴を用いて PHT をアクセスするため，破壊的競合はより複雑な様相となり，本質的には避けられない．

本論文では，従来の分岐予測器の動作を分析することにより，分岐命令の予測ミスが一部分の分岐命令に集中して発生することを示す．この知見に基づき，予測ミスが集中して発生する分岐命令のみに対処する分

† 立命館大学大学院理工学研究科
Graduate School of Science and Engineering, Ritsumeikan University
†† 立命館大学情報理工学部
College of Information Science and Engineering, Ritsumeikan University

岐予測機構を従来の予測器に付加することにより，破壊的競合を緩和して分岐予測の精度を向上させることを目指す．

本論文は以下のように構成する．2 章では，関連研究を説明する．3 章では，いくつかの分岐予測器を用いて分岐ミスが一部の分岐命令に集中して発生することを示す．4 章では，分岐ミスが集中的に発生する分岐命令をターゲットとした分岐予測付加機構を提案する．5 章では，幾つかの従来予測器に提案手法を付加した構成を詳細に評価する．6 章では，まとめと研究の展望を行う．

2. 関 連 研 究

多くの分岐予測器は一つの PHT をもつ単体予測器をベースにし，分岐命令の特性を利用して，それらの特性に適応するハードウェア機構の追加や単体予測器を組合せることにより破壊的競合を低減することを目指している．代表的な単体予測器には Bimodal，Gshare が用いられている．以下では，これらのアプローチに沿ったいくつかの分岐予測器について説明する．

Combining[2]：Combining 予測器は，Bimodal 予測器と Gshare 予測器を組み合わせたものである．さらに，二つの単体予測器の予測結果を選択するため，2 ビット飽和型カウンタより構成される Selector を設ける．Gshare 予測器はグローバル履歴を用いるが，Bimodal 予測器はグローバル履歴を用いない．すべての分岐命令がグローバル履歴に関連するとは限らない[9] という主張もなされており，Bimodal 予測器と Gshare 予測器を組み合わせることにより，グローバル履歴を使い分けて予測精度の向上を目指している．

Bimode[3]：分岐命令には，分岐方向が Taken に偏った分岐命令と，NotTaken に偏った分岐命令が存在する．これらの特性が異なる分岐命令が同一 PHT エントリを使用するとき破壊的競合が頻発する．Bimode 予測器は，Taken 用の Gshare 予測器と NotTaken 用 Gshare 予測器を用いることにより，破壊的競合を緩和することを目指した提案である．二つの Gshare 予測器の予測結果を選択するため，ChoicePHT を備えている．

Bimode-Plus[6]：分岐命令にはプログラムが終了するまで，常に Taken あるいは NotTaken をとる極端な偏りのある分岐命令が存在する．Bimode-Plus は分岐命令の極端な偏りの特性を利用する．Bimode-Plus は Bimode 予測器に加えて，Taken flag と NotTaken flag をもつ Bias Table を用意し，極端な偏りのある分岐命令を検出する．検出された極端な偏りのある分岐命令は Bimode 予測器を用いずに予測する．また，極端な偏りのある分岐命令を Bimode 予測器に登録しないことにより，破壊的競合を緩和し予測精度の向上を目指している．

Agree[5]：Agree 予測器は BTB の各エントリに情報を付加して，ベース予測器の予測ミスをチェックしている．こちらは 2 ビットのアップダウンカウンタで，BTB の各エントリに対するベース予測器の成功・不成功情報を記録する．これをベース予測器の結果と XOR することで，不成功が多い場合は予測方向を反転する．

Hybrid[4]：ローカル履歴を利用する分岐予測器とグローバル履歴を利用する分岐予測器をハイブリッドする分岐予測器である．Hybrid 予測器は 2bc 予測器，Gshare 予測器，pshare 予測器，GAS 予測器と AVG 予測器により構成される．また，BTB の各エントリに各予測器の予測状況を保持し，予測するときにこの情報を用いることにより，使用する予測器を決める．それにより，複数の分岐予測器を同時に使用することができ，予測精度を高めることができる．

PPM-Like L-TAGE[7),8)]：分岐命令の分岐方向には，一定のパターンを持つ特性も挙げられる[10]．その特性を利用するものが PPM-like 予測器と L-TAGE 予測器である．二つの予測器は PPM(prediction by partial matching) の手法を使用する．グローバル履歴 (GBHR) の異なる部分を用いて複数の PHT にアクセスし，予測を行う．PHT にタグを格納することにより，競合を防いでいる．複数の PHT を持つことにより，GBHR に含まれるパターンを利用することができ，予測精度を高めている．

3. 予測ミスの偏り

本章では，従来の分岐予測器において，予測ミスが少数の分岐命令に集中して発生することを示す．まず SimpleScalar Tool Set[14] を用いて，Combining 予測器と Bimode 予測器を実装して評価を行う．ベンチマークには SPECint2000 から bzip，gcc，gzip，mcf，parser，twolf，vpr，vortex の 8 本を使用する．プロセッサの仕様を表 1 に示す．命令セットは SimpleScalar PISA を用いる．

これらのベンチマークにおいて，最も多く予測ミスが発生する上位 8 個と 16 個の分岐命令を抽出し，これらの予測ミスが全体の予測ミスに占める割合を調べる．

図 1 と図 2 では，予測ミスが最も多い上位 8 個と 16 個の分岐命令の予測ミスが全体の予測ミスに占める割合を示す．なお，予測ミスはプログラムの実行状

況に応じて変化するため，調べ方については，20M 命令ごとに最もミスの多い上位 8 個（あるいは 16 個）の分岐命令を抽出し，それらのミスが全体ミスに占める割合を調べ，これを 5 回繰り返して 100M 命令まで評価する．

予測器のハードウェア量は，8KB，16KB，32KB の 3 種類について評価を行う．具体的には，Combining 予測器について，文献 2) では Gshare 予測器のサイズが Bimodal 予測器のサイズの 2 倍になるときより良い性能が得られると報告されている．そのため，Combining 予測器では，Bimodal 予測器と Selector のエントリ数を 8K，16K，32K，Gshare 予測器のエントリ数を 16K，32K，64K と設定する．Bimode 予測器については，Gshare 予測器のエントリ数を 8K，16K，32K とし，ChoicePHT のエントリ数を 16K，32K，64K と設定する．

図 1 と図 2 の横軸は命令実行数であり，縦軸はミスの多い分岐命令の予測ミスが全体ミスに占める割合である．図において，各折れ線グラフの前の数字はミスの多い分岐命令の数，後の数字は予測器のハードウェア量である．

図 1 と図 2 より，Combining 予測器と Bimode 予測器では，8 個の分岐命令の予測ミスが，全体のミスの 70% 以上を占めることが分かる．さらに 16 個の分岐命令の予測ミスが，全体のミスの 80% 以上を占めることが分かる．

次に，Championship Branch Predicton[11] に公開された予測器 L-TAGE を用いて，分岐ミスの偏りの特性を確認する．文献 11) で提供されたトレースコードを利用し，実行命令数は 10M である．実験の結果，大半のベンチマークでは，上位 8 個のミスの多い分岐命令の予測ミスが全体予測ミスの 50% 以上を占め，幾つかのベンチマークでは 90% 以上を占めた．これにより，L-TAGE 予測器においても分岐ミスの偏りの特性があることが確認できる．

表 1 プロセッサの構成
Table 1 Processor configuration

Pipeline	5 stages: 1 Fetch, 1 Decode, 1 Execute 1 Memory Access, 1 Commit
Fetch, Decode, Dispatch	4 instructions
Issue	Int: 4, fp: 2, mem: 2
Window	Dispatch queue:256, Issue queue:256
BTB	2K-entry 4-way associative BTB, 32-entry RAS
Memory	64KB, 4-way associative, 1-cycle instruction and date caches 2MB, 8-way associative, 10-cycle L2

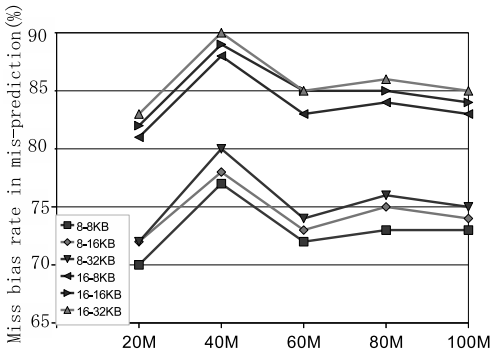

図 1 Combining 予測器における予測ミスの偏り
Fig. 1 Miss prediction bias rate in Combining predictor

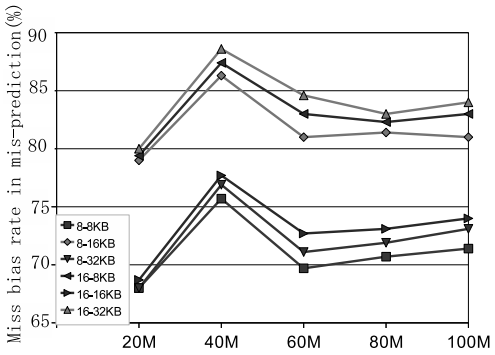

図 2 Bimode 予測器における予測ミスの偏り
Fig. 2 Miss prediction bias rate in Bimode predictor

4. ローカル履歴を用いた分岐予測付加機構

3 章では，分岐ミスが少数の分岐命令に偏るという特性をもつことを示した．我々はこの特性を利用し，予測ミスの多い分岐命令について，そのローカル履歴を用いて分岐成否を予測し，この機構をベース予測器に付加するハードウェア機構を提案する

図 3 に，提案するハードウェア機構のブロック図を示す．ベース予測器に付加される部分は MBP(Miss Bias Predictor) と LHBP(Local History Branch Predictor) により構成される．MBP は予測ミスの多い分岐命令を検出し，そのローカル履歴などの情報を保存する機構である．LHBP は検出された予測ミスの多い分岐命令のローカル履歴を利用して，分岐予測を行う機構である．

予測の流れとしては，まず MBP では，拡張された BTB 機構（EBTB: Extended BTB）を利用し，予測ミスの多発する分岐命令を検出する．そして，検出された分岐命令のローカル履歴などの情報を MBB(Miss

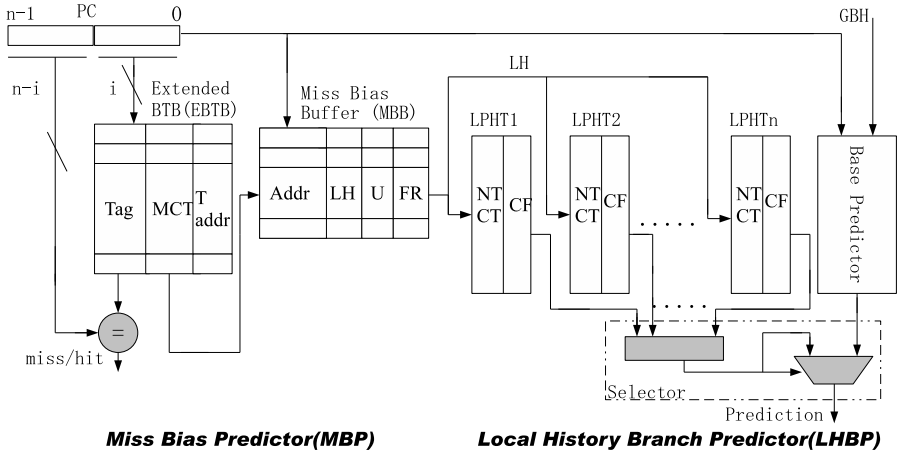

図 3 提案する分岐予測器のブロック図
Fig. 3 Block diagram of proposed branch predictor

Bias Buffer) に記憶する．さらに，LHBP では，予測ミスの多発する分岐命令について，該当する分岐命令のローカル履歴を用いて分岐成立かどうかを予測する．最後に， Selector で，LHBP により予測された結果とベース予測器により予測された結果のいずれかを選択する．

4.1 MBP による予測ミスの多発する分岐命令の発見

EBTB は，従来の BTB を利用して各エントリに飽和カウンタの MCT(Miss Counter) を追加し，Base Predictor の予測ミスを数える．EBTB では従来の BTB と同じように，タグ (Tag) と分岐先のアドレス (T addr:Target address) が設けられている．

MBB は，予測ミスの多発する分岐命令を格納するもので，Addr，LH，U，FR により構成される．Addr は分岐命令アドレス，LH は分岐命令のローカル履歴，U は該当するエントリが使用されているかどうかを示すビットであり（1 が使用中，0 が未使用），FR(Failure Rate) は LHBP の予測ミスの数とベース予測器の予測ミスの数の差を保持し，このエントリの有効性を示すために用いられる．

前章により，多くのベンチマークでは 8 個あるいは 16 個の分岐命令に予測ミスが集中的に発生しているため，MBB のサイズは 8 あるいは 16 に設定する．MBP の具体的な動作を以下で説明する．

MBB への登録: 分岐命令がコミットされるときに，分岐命令のアドレスを用いて EBTB を検索する．EBTB がヒットし，かつ予測ミスの場合は，EBTB の MCT をインクリメントする．EBTB がヒットしない場合は，従来の BTB と同じように Tag の更新を行い，予測ミスした時に MCT を 1 にし，予測成功した時に MCT を 0 にする．EBTB エントリの MCT が閾値に到達すると，MBB に分岐命令のアドレスを登録する．そして，該当する EBTB のエントリの MCT をリセットする．

MBB への登録は，以下のように行われる．まず登録しようとした分岐命令のアドレスが MBB に存在するかどうかをチェックする．このアドレスが MBB に存在しない場合には，MBB の U が 0 のエントリが存在するならば，そのエントリに登録し，U を 1 にする．もし MBB の U がすべて 1 の場合は，LRU(Least Recently Used) ロジックを利用して，最も最近利用されていないエントリを選択し，登録する．なお，LRU ロジックにおいて，LHBP が正しく予測でき，かつベース予測器が正しく予測できない場合が MBB を利用したものと判定する．

MBB の更新: 分岐命令がコミットされるとき，当該分岐命令が MBB に登録されているときは MBB の更新が行われる．LH には，ローカル履歴が更新される．FR の更新については，LHBP での予測が失敗，かつベース予測器が成功の場合はインクリメントし，LHBP での予測が成功，かつベース予測器が失敗の場合はデクリメントする．これにより，FR に LHBP とベース予測器の予測の差を格納できる．FR は 7 ビットの飽和カウンタであり，この値が閾値になると LHBP の予測ミスがベース予測器より多く発生するために LHBP による予測が有効でないと判断し，MBB のエントリをリセットする．

4.2 LHBP によるローカル履歴を用いた分岐予測

LHBP は予測ミスの多発する分岐命令の分岐成否

を予測するものである．予測ミスの多発する分岐命令毎に個別のローカル履歴に基づく予測器 LPHT が用意されている．LPHT の個数は，MBB のエントリ数と同じである．たとえば，MBB の一番目のエントリの分岐命令は，LPHT1 を用いて予測する．これにより，LHPT において破壊的競合は生じない．

LPHT のエントリ数は，使用するローカル履歴の長さにより決まる．すなわち，n ビットのローカル履歴に対して 2^n のエントリ数をもつ．LPHT の NTCT は 2 ビット飽和カウンタで，対応する分岐命令の分岐結果によりインクリメント／デクリメントされる．CF は予測の信頼性を示す，2 ビット Miss Resetting Counter[12),13)] を使用し，閾値が 3 と設定する．Miss Resetting Counter は PVN（Predictive Value of a Negative Test）を高める有効手段であるため[12)]，ローカル履歴のみを用いた予測の誤動作を緩和することを狙っている．

LPHT の更新:分岐命令がコミットされるときに，分岐命令のアドレスを用いて，MBB を検索する．MBB に存在するとき MBB のローカル履歴を利用し，対応する LPHT のエントリの NTCT を更新する．さらに，CF の更新も行う．すなわち予測が成功した場合は CF をインクリメントし，ミスした場合はリセットする．

分岐成否の予測：分岐命令がフェッチされるときに，フェッチされた分岐命令のアドレスを用いて，MBB を検索する．MBB に存在する場合は，MBB 内の LH に対応する LPHT のエントリの NTCT と CF を得る．得られた情報を用いて，LHBP で予測を行う．さらに，Selector において，LHBP の予測結果とベース予測の予測結果のいずれかを選択する．具体的には，NTCT を利用して Taken と NotTaken を判断し，かつ CF が閾値になる場合に LHBP の予測結果を採用し，その他の場合はベース予測の予測結果を利用する．

5. 評　　価

5.1 最適な構成法の評価

提案手法を評価するためには，最適な構成のパラメータを決める必要がある．具体的には，以下のパラメータがある．

- MCT のサイズ
- MBB のエントリ数 (LPHT の個数)
- LPHT のエントリ数 (LH の長さ)

本節では，実験により本提案の最適な構成法を求め，それによる性能評価について説明する．実験に関しては，提案手法を SimpleScalar Tool Set[14)] の上に実装し，シミュレーションにより評価を行った．プロセッサの仕様は表 1 と同一である．ベンチマークには SPECint2000 から bzip，gcc，gzip，mcf，parser，twolf，vpr，vortex の 8 本と，CommBench[12] から drr，reed_dec，reed_enc，rtr，zip_enc の 5 本を用いた．命令セットは SimpleScalar PISA を用いる．

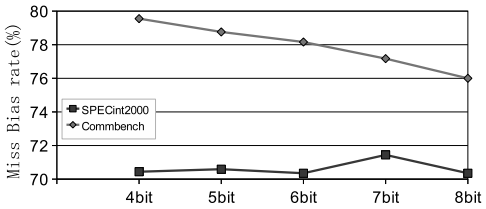

図 4 MCT のサイズに対するミスの偏り
Fig. 4 Miss bias rate to MCT size.

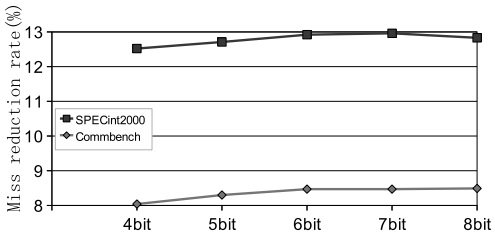

図 5 MCT サイズに対するミスの削減率
Fig. 5 Miss reduction rate to MCT size.

5.1.1 MCT のサイズの評価

まず，EBTB の MCT のサイズについて議論する．MCT は EBTB 内の各命令のミス数を数える飽和カウンタである．MCT が飽和すると，予測ミスが集中的に発生する分岐命令として判定し，MBB に登録する．ベース予測器には 16KB の Bimode 予測器を用い，MCT のサイズが 4bit から 8bit までについて，MBB のサイズを 8，LPHT のエントリ数を 1024 として実験を行った．

図 4 はミスの多発する命令の予測ミスが全体のミスに占める割合を示す．横軸が MCT のサイズで，縦軸がミスの割合 (SPECint2000 平均値と CommBench の平均値) である．図 5 は，予測ミスの削減率を示す．横軸が MCT のサイズで，縦軸がベース予測器に対して提案手法が削減した予測ミスの割合 (SPECint2000 平均値と CommBench の平均値) である．

図 4 より，設定した MCT のサイズで SPECint2000 において約 70%の予測ミスを検出でき，CommBench において約 75%以上の予測ミスを検出できることが示される．図 5 より，設定した MCT のサイズで SPECint2000 において約 13%の予測ミスが減少し，

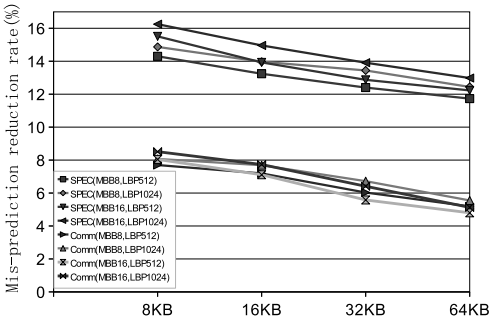

図 6 MBB と LPHT サイズに対する予測ミス削減率 (Combining)
Fig. 6 Miss prediction reduction rate to MBB and LPHT size (Combining)

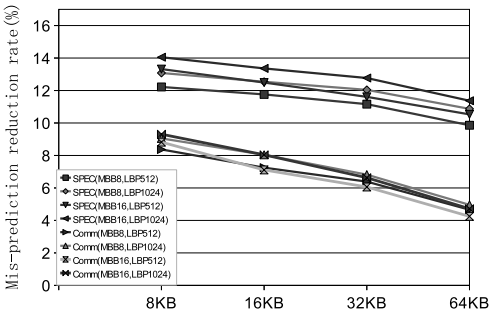

図 7 MBB と LPHT サイズに対する予測ミス削減率 (Bimode)
Fig. 7 Miss prediction reduction rate to MBB and LPHT size (Bimode)

CommBench において約 8%以上の予測ミスが減少することが示される．

二つの図をあわせて分析することにより，提案方式により予測ミスが多発する分岐命令の約 70%の検出ができ，かつ予測ミスも減らすことができることが分かった．さらに，MCT のサイズに関して，ミスの減少率では大きな差がないとも確認できた．以下の実験については，ハードウェアの使用量を考慮したうえで，MCT が最小の 4 ビットを使用する．

5.1.2 MBB と LPHT のエントリ数の評価

MBB と LPHT のエントリ数について議論を行う．MBB のエントリ数は LPHT の個数と同一であり，LPHT のエントリ数はローカル履歴長 (MBB の LH) により決まる．MBB のエントリ数を大きくすることにより，ターゲットとする予測ミスの偏る分岐命令の数を増やすことができ，LPHT のエントリ数を大きくすることにより，長いローカル履歴を用いた予測精度の向上が可能であると考えられる．実験では，MBB と LPHT のエントリ数は (8,512)，(8,1024)，(16,512)，(16,1024) の四種類と設定した．図 6〜図 8 は Combining 予測器，Bimode 予測器，Bimode-Plus 予測器をベースにした提案手法での MBB と LPHT のエントリ数の評価結果である．横軸はベース予測器のサイズ (8KB，16KB，32KB，64KB) である．縦軸はベース予測器に対する提案方式の予測ミス削減率の平均値であり，SPECint2000 および CommBench の平均値を示す．

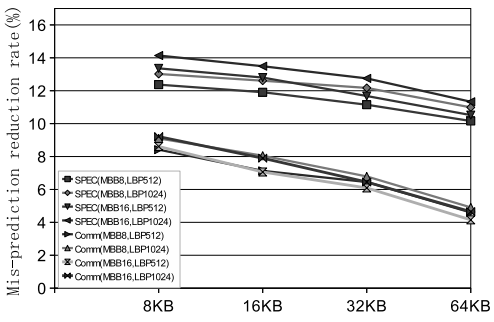

図 8 MBB と LPHT サイズに対する予測ミス削減率 (Bimode-Plus)
Fig. 8 Miss prediction reduction rate to MBB and LPHT size (Bimode-Plus)

図 6〜図 8 より，MBB のエントリ数を 16，LPHT のエントリ数を 1024 と設定した場合は予測ミス削減率が最大で，MBB のエントリ数を 8， LPHT のエントリ数を 512 と設定した場合が最小である．特に SPECint2000 において，この差が大きい．但し，図 6〜図 8 より，MBB と LPHT のエントリ数の変化に対して予測ミスの削減率が小さい．例えば，各サイズの Combining 予測器において，SPECint2000 の場合に予測ミス削減率が最大となった (MBB=16，LPHT=1024) は，予測ミス削減率が最小となった (MBB=8，LPHT=512) と僅か 2%の差であり，CommBench の場合は僅か 1%未満の差である．Bimode 予測器，Bimode-Plus 予測器においても同じ傾向である．

以上の実験により，MBB と LPHT のエントリ数を増やしても予測ミスの減少はわずかであると確認した．以降の実験について，ハードウェアの使用量を考慮したうえで，MBB のエントリ数を 8，LPHT のエントリ数を 512 に設定する．

5.2 予測ミスの削減率の評価

本節では，提案手法による予測ミスの削減率の評価を行う．ベース予測器として Combining 予測器，Bimode 予測器，Bimode-Plus 予測器，Agree 予測器，Hybrid 予測器の 5 種類を用いる．ベース予測

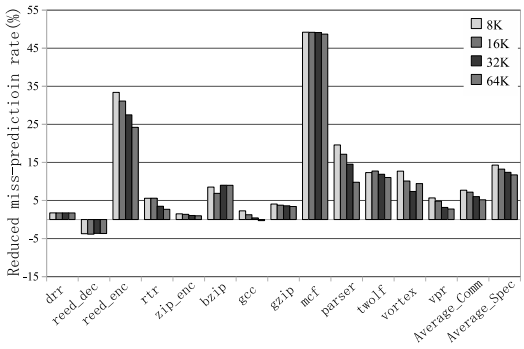

図 9 Combining における予測ミス削減率
Fig. 9 Miss prediction reduction rate to Combining

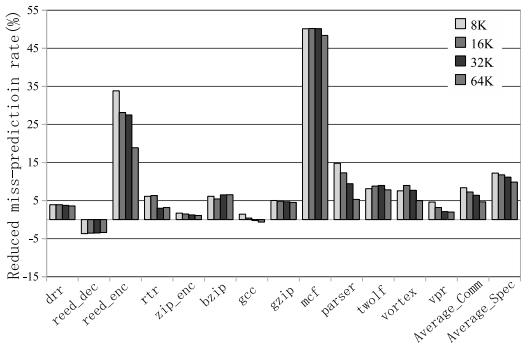

図 10 Bimode における予測ミス削減率
Fig. 10 Miss prediction reduction rate to Bimode

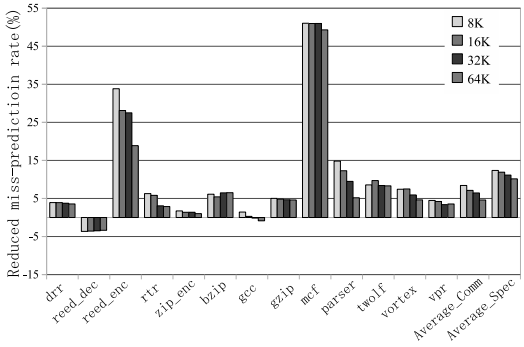

図 11 Bimode-Plus における予測ミス削減率
Fig. 11 Miss prediction reduction rate to Bimode-Plus

器 Combining, Bimode, Bimode-Plus, Agree のサイズを (8KB, 16KB, 32KB, 64KB) の 4 種類とし，Hybrid 予測器のサイズは 10.5KB, 17.75KB, 30KB, 60.5KB と設定する．

図 9〜図 11 は予測器 Combining, Bimode, Bimode-Plus をベースにした提案手法により削減した予測ミス率をベンチマーク毎に示した図である．横軸はベンチマークであり，縦軸は提案手法のベース予測器に対する予測ミス削減率である．

図 9〜図 11 より，Combining, Bimode, Bimode-Plus をベースにした提案手法は，SPECint2000 において最大 40%，以上平均 10% 以上の予測ミスを減らすことができ，CommBench において最大 30%，平均 6% 以上の予測ミスを減らすことができた．

Agree をベースにした提案手法は，SPECint2000 において最大 23%，平均 10% 以上の予測ミスを減らすことができ，CommBench において最大 30%，平均 6% 程度の予測ミスを減らすことができた．Hybrid をベースにした提案手法は，SPECint2000 において最大 45%，平均 7% 以上の予測ミスを減らすことができ，CommBench において最大 20%，平均 3% 程度の予測ミスを減らすことができた．

また先行研究の L-TAGE をベースとした提案手法の予測ミス削減率の評価も行った．評価について，文献 11) に公開された予測器 L-TAGE とトレースコードを利用し，実行命令数は 10M である．ベンチマークが異なるので他の予測器と直接比較できないが，ほとんどのベンチマークにおいて提案手法により予測ミスが削減でき，平均的には 3%の予測ミスの削減ができた．

このようにベースとなる予測器により提案手法の効果に差はあるが，いずれの場合でも本提案手法の付加が有効であることが示されている．

5.3 ハードウェア規模の検討

提案方式の主なハードウェア規模について考察する．EBTB に関しては，従来の検索ポートを利用するために，検索用のポートを追加する必要がない．また，MCT が 4 ビットのため EBTB に追加するメモリ量は 1KB である．MBB について，エントリ数は 8，Addr が 32 bit，LH が 9 bit，U が 1 bit，FR が 7 bit であるので語長は 49 bit となる．そのため，追加されるメモリ量は 392 ビットの CAM となる．LPHT について，個数が 8，エントリ数は 512，NTCT が 2 bit，CF が 2 bit であるので語長は 4 bit となり，追加されるメモリ量は 2KB である．

表 2 は予測器のサイズとそれらの MPKI(miss-prediction per kilo instruction) の関係を示す．予測器の各欄は，対応するサイズでの MPKI である．表 2 に示すように，8KB の Combining 予測器に対して 3KB 程度の提案機構を追加したものは，64KB の Combining 予測器と同等以上の性能になる．Bimode, Bimode-Plus についても同様である．これらのことより，従来予測器に本提案機構を追加することは，従

表 2 MPKI 実験結果 (平均)
Table 2 Result of MPKI (Average)

Predictor	Specint2000				CommBench			
	8KB	16KB	32KB	64KB	8KB	16KB	32KB	64KB
Combining	5.53	5.30	5.06	4.84	7.34	7.20	6.96	6.67
Combining+提案	4.76	4.60	4.43	4.30	6.48	6.43	6.35	6.18
Bimode	5.40	5.14	4.93	4.73	7.34	7.03	6.84	6.52
Bimode+提案	4.76	4.55	4.38	4.28	6.50	6.37	6.27	6.15
Bimode-Plus	5.36	5.12	4.90	4.71	7.29	7.00	6.82	6.51
Bimode-Plus+提案	4.71	4.50	4.35	4.24	6.45	6.34	6.23	6.13

来予測器のエントリー数の増加以上の効果があるといえる．

6. おわりに

近年のプロセッサではパイプライン段数の増加と命令発行幅の増加により分岐予測ミスのペナルティが増加するため，分岐予測の精度向上がプロセッサの性能向上にとって大きな課題となっている．本稿では，分岐予測ミスが少数の分岐命令に偏っているとの特徴を利用した分岐予測器を提案した．

提案した予測器は，分岐ミスが集中的に発生する分岐命令を判定し，それらの分岐命令についてローカル履歴を利用した予測機構を従来の分岐予測器に付加することにより予測精度を向上させる．

Combining，Bimode，Bimode-Plus，Agree，Hybrid の予測器をベースにして本提案を付加した予測機構を SimpleScalar Tool Set 上に実装して評価を行った．その結果，SPECint2000 において，提案手法はベース予測器に対して平均 9%程度の予測ミスを減らすことができた．CommBench においても予測ミスを減らすことができた．

今後の課題としては，更なる改良・評価が必要である．提案手法は平均として従来手法より予測ミスを減らすことができるが，逆に従来手法より予測ミスが増えるベンチマークも存在している．そのため，それらの対策を検討し，更なる予測ミスを減らすことが課題である．また，パイプラインを拡張した大規模なプロセッサにおける詳細な評価により，IPC の向上を確認する必要がある．

参考文献

1) J.E.Smith. ”A Study of Branch Prediction Strategies”，ISCA 1981，pp.135-148，1981.
2) S.McFarling. ”Combining Branch Predictors”，Technical report TN-36，Digital Western Research Laboratory，1993.
3) Chih-Chieh Lee，I-Cheng K. Chen，Trevor N. Mudge. ”The bi-mode branch predictor”，MICRO97，pp.4-13，Dec. 1997.
4) M.Evers，P-Y.Chang，Y.N.Patt，”Using Hybrid Branch Predictors to Improve Branch Prediction Accuracy in the Presence of Context Switches”, ISCA 1996, pp.3-11, May 1996
5) Eric Sprangle, Robert S. Chappell, Mitch Alsup，Yale N. Patt. ”The Agree Predictor: A Mechanism for Reducing Negative Branch History”, ISCA 1997, pp284-291, June 1997.
6) 吉瀬謙二，片桐孝洋，本多弘樹，弓場敏嗣．”Bimode-Plus 分岐予測器の提案”，情報処理学会論文誌，ACS 10，PP.85-102，2005.
7) A. Seznec. ”The L-TAGE branch predictor”，JILP，vol. 9，May 2007.
8) M. pierre. ”A PPM-Like, tag-based predictor”，JILP，April 2005.
9) L.Porter，D.M.Tullsen，”Greating Artificial Global History to Improve Branch Prediction Accuracy”，ICS’09，PP.266-275，2009.
10) I.-C.K. Chen，J.T.Coffey，T.N.Mudge. ”Analysis of branch prediction via data compression”，ASPLOS-VII，pp128-137，Oct.1996.
11) JILP:The 2nd JILP Championship Branch Predicton Competition. http://cava.cs.utsa.edu/camino/cbp2/
12) H. Akkary，S. T. Srinivasan，R. Koltur，Y. Patil，W. Refaai，”Perceptron-Based Branch Confidence Estimation”，HPCA-10，pp.265，Feb. 2004.
13) 二ノ宮康之，阿部公輝．”パーセプトロン分岐予測器を用いた予測信頼性の動的判定に基づく電力削減 ”，SACSIS2009，pp.327-334，May. 2009.
14) D.Burger，T.M.Austin. ”The SimpleScalar Tool Set , Version2.0 ”, Technical Report，University of Wisconsin-Madison Computer Sciences Dept，July 1997.
15) T.Wolf，M.A.Franklin，“ CommBench - a Telecommunications Benchmark for Network Processors ”, ISPASS-2000, Austin, TX, pp.154-162，April 2000.

(平成？年？月？日受付)
(平成？年？月？日採録)