Bimode-Plus分岐予測器の提案

吉瀬謙二[†,††] 片桐孝洋[†]
本多弘樹[†] 弓場敏嗣[†]

命令発行幅の増大と命令パイプライン長の増大により，プロセッサ性能に与える分岐予測器の重要性が増している．予測精度を向上させるために，過去数十年の間に様々な分岐予測器が提案されている．本稿では，極端な偏りのある分岐命令の特徴を利用する新しい分岐予測方式を提案する．本方式を Bimode 分岐予測器に組み込む新しい分岐予測器として Bimode-Plus 予測器を提案する．SpecFP，SpecINT，マルチメディア，サーバの領域を含むベンチマークプログラムを用いた評価結果から，Bimode-Plus 分岐予測器の優れた予測精度を確認する．また，SPEC CINT2000 を用いて，Bimode-Plus 予測器によるアウトオブオーダ実行プロセッサの速度向上を評価し，大規模な構成のプロセッサモデルを想定する場合に，平均で 3.2%，最大で 8.5%の速度向上を達成できることを明らかにする．

The Bimode-Plus Branch Predictor

Kenji Kise,[†,††] Takahiro Katagiri,[†] Hiroki Honda[†]
and Toshitsugu Yuba[†]

Accurate branch prediction has come to play an important role for modern microprocessors. In order to improve its prediction accuracy, many branch predictors have been investigated in the past few decades. In this paper, Bimode-Plus branch predictor is proposed as an enhanced version of Bimode branch predictor. Our experimental results using benchmarks from SpecFP, SpecINT, multi-media and server area show that Bimode-Plus branch predictor gives better prediction accuracy than Bimode branch predictor. With a cycle-accurate processor simulator modeling a wide issue out-of-order processor and SPEC CINT2000 benchmark, we show that Bimode-Plus branch predictor improves IPC by 8.5% for 197.parser and 3.2% on average.

1. はじめに

近年のマイクロプロセッサは，命令発行の幅を広くして命令レベル並列性を抽出し，命令パイプラインの段数を深くすることで動作周波数を向上させている．これらの傾向は，プロセッサ内で処理される命令数を増加させるため，分岐予測ミスが発生した際にフラッシュすべき命令数，すなわち分岐予測ミスのペナルティも増加する．これらミスペナルティの影響を軽減するために，精度の高い分岐予測器が高性能プロセッサを実現するうえで不可欠となっており，過去数十年の間に様々な分岐予測器が提案されている[4),9),20)]．

本稿では，極端な偏りのある分岐命令の特徴を利用して，分岐予測の精度向上を目指す．まず，極端な偏りのある分岐命令を区別する方式を提案する．次に，Bimode 分岐予測器の改良型として，本方式を利用する Bimode-Plus 分岐予測器を提案する．SpecFP，SpecINT，マルチメディア，サーバの領域を含むベンチマークプログラムを用いて，Bimode-Plus 分岐予測器の予測精度を評価する．既存の分岐予測器の精度と比較することで，Bimode-Plus 分岐予測器の優れた予測精度を確認する．また，アウトオブオーダ実行を模倣するクロックレベルのプロセッサシミュレータを用いて，分岐予測器の精度向上とプロセッサ性能との関係を検討する．

本稿の構成を示す．2 章で関連研究を述べる．3 章で極端な偏りのある分岐命令を区別する方式と，それを利用する Bimode-Plus 分岐予測器を提案する．4 章で Bimode-Plus 分岐予測器の予測精度を評価する．5 章では，クロックレベルのプロセッサシミュレータを用いて分岐予測器の精度向上とプロセッサ性能との関

† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, University of Electro-Communications
†† 独立行政法人科学技術振興機構さきがけ
PRESTO, Japan Science and Technology Agency (JST)

係を検討する．6 章で議論し，7 章で本稿をまとめる．

2. 関 連 研 究

本稿では，分岐予測器のことを省略して予測器と呼ぶことがある．また，分岐成立のことを taken，分岐不成立のことを untaken と呼ぶことがある．

これまでに様々な分岐予測器が提案されている．分岐予測の詳細な調査はいくつかの文献[4),19),20)]に詳しい．本章では，提案する Bimode-Plus 予測器の新規性を明確にするために必要となるいくつかの分岐予測器を概観する．

McFarling[10)] により提案された Gshare 予測器は，単純な構成で高い予測精度を達成するという特徴から，いくつかの商用プロセッサに採用されている．本稿においても，比較対象の 1 つとして Gshare 予測器を利用する．Gshare 予測器の構成を**図 1** に示す．これは，2 レベル適応型[16)] の分岐予測を拡張したもので，グローバルな分岐履歴レジスタ（BHR）の一部（図では m ビットとして記述）と分岐命令のアドレスの一部（図では n ビットとして記述）との排他的論理和によりパターン履歴表（PHT）へのインデックスを作成する．PHT は 2 ビット飽和型カウンタの配列であり，選択された 2 ビットカウンタの値により分岐方向を予測する．

Gshare 予測器において，飽和型カウンタを共有した分岐命令が互いに異なる分岐結果を格納することにより予測精度が低下するという問題（破壊的競合）を緩和するために，Lee ら[9)] により提案された Bimode 予測器の構成を**図 2** に示す．同様の構成が Sgshare 予測器として野口ら[22)] により提案されている．こちらの方が投稿時期が早いことを考慮すると，本来は Sgshare 予測器と呼ぶべきである．しかしながら，Bimode 予測器として広く認知されているため，本稿では Bimode 予測器と記述する．

Bimode 予測器は Choice PHT，Taken PHT，Untaken PHT という 3 つのテーブルを利用する．ここで，Taken PHT と Untaken PHT のことを Direction PHT と呼ぶ．Choice PHT はプログラムカウンタの一部（図では s ビットとして記述）を用いてインデックスを生成する 2 ビットカウンタの配列である．グローバルな分岐履歴レジスタ（BHR）と分岐命令のアドレス（プログラムカウンタ）との排他的論理和により Direction PHT のインデックスを生成する．Choice PHT が分岐成立と予測した場合には Taken PHT を，そうでない場合には Untaken PHT の値を用いて予測を行う．

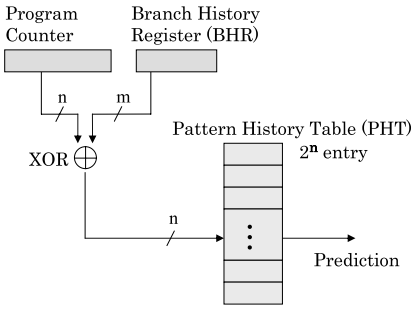


図 1 Gshare 予測器のブロック図
Fig. 1 Gshare branch predictor.

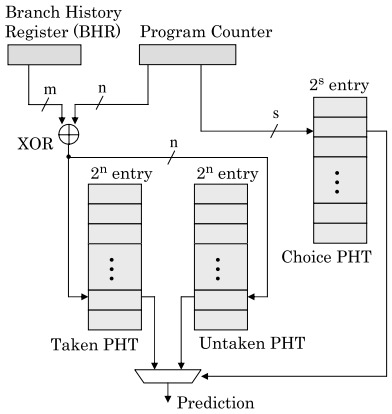


図 2 Bimode 予測器のブロック図
Fig. 2 Bimode branch predictor.

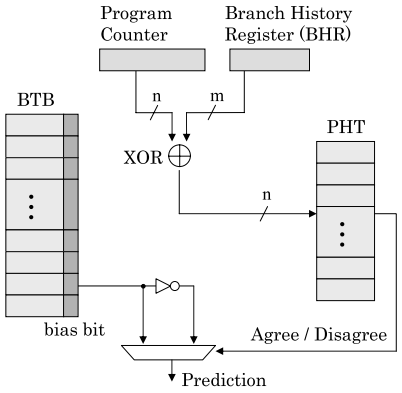


図 3 Agree 予測器のブロック図
Fig. 3 Agree branch predictor.

Sprangle ら[14)] は，Branch Target Buffer（BTB）のエントリにバイアスビットを追加することで分岐命令の偏りを記憶する Agree 予測器を提案した．この構成を**図 3** に示す．多くの分岐命令は taken あるいは untaken に偏っている．このため Agree 予測器では，分岐命令が BTB に格納された時点の結果をバイアスビットに格納する．PHT ではこのバイアスビットに

表 1 ベンチマークプログラムの特性と極端な偏りのある分岐の頻度
Table 1 Characteristics of benchmark programs and the rate of extremely biased branches.

ベンチマーク	実行命令数	動的分岐の割合	静的分岐の数	All-taken の割合	All-untaken の割合
DIST-FP-1	32 million	6.89%	444	0.15%	65.74%
DIST-FP-2	31 million	5.73%	452	0.06%	4.18%
DIST-FP-3	31 million	4.98%	810	0.72%	1.47%
DIST-FP-4	30 million	2.94%	556	6.60%	5.50%
DIST-FP-5	32 million	7.52%	243	4.82%	14.99%
DIST-INT-1	34 million	12.13%	424	1.33%	28.63%
DIST-INT-2	33 million	8.64%	1,585	1.83%	19.37%
DIST-INT-3	33 million	11.21%	989	3.30%	17.95%
DIST-INT-4	31 million	6.48%	681	1.76%	40.99%
DIST-INT-5	33 million	11.27%	441	0.40%	3.12%
DIST-MM-1	32 million	6.91%	460	1.14%	17.53%
DIST-MM-2	33 million	11.30%	2,522	6.53%	6.14%
DIST-MM-3	34 million	8.75%	1,091	35.82%	30.96%
DIST-MM-4	34 million	14.10%	2,256	2.16%	2.59%
DIST-MM-5	32 million	7.79%	4,535	13.25%	15.98%
DIST-SERV-1	35 million	10.43%	10,910	13.57%	50.89%
DIST-SERV-2	34 million	10.12%	10,560	14.16%	51.72%
DIST-SERV-3	34 million	10.94%	16,604	9.66%	44.36%
DIST-SERV-4	35 million	11.92%	16,889	17.41%	49.39%
DIST-SERV-5	35 million	11.95%	13,017	17.24%	51.25%

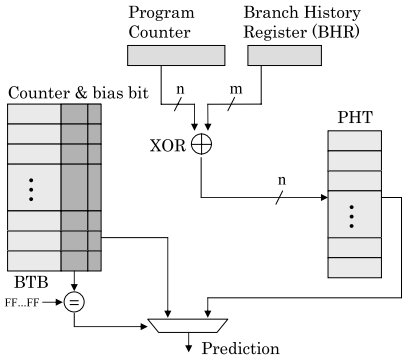


図 4 Filter 機構のブロック図
Fig. 4 Filter mechanism.

同意するか否かという判断に基づいて予測を行う．

Chang ら[2] は，カウンタを用いて偏りのある分岐命令を選びだし，偏りのある方向を予測値とする Filter 機構を提案した．この構成を**図 4** に示す．Filter 機構は，BTB のエントリにバイアスビットとカウンタとを追加する．最初に分岐結果が得られた時点で，その分岐結果をバイアスビットに格納して，カウンタの値をリセットする．その後，分岐結果とバイアスビットが等しい場合にはカウンタの値をインクリメントする．異なる場合には，バイアスビットを反転して，カウンタをゼロで初期化する．カウンタが飽和している場合に，当該命令を偏りのある分岐命令と判断して，バイアスビットの値を予測値として利用する．

3. Bimode-Plus 予測器の提案

3.1 極端な偏りのある分岐

個々の分岐命令の挙動に注目して調査を行ったところ，プログラムの開始から終了までのすべての実行において必ず分岐成立する分岐命令，あるいは必ず不成立である分岐命令が多数存在することが判明した．これら，すべて同様の挙動を示す分岐命令のことを「極端な偏りのある分岐命令」と呼ぶことにする[17]．その中で，必ず成立する分岐命令のことを「極端な偏りのある成立分岐命令」，必ず不成立する分岐命令のことを「極端な偏りのある不成立分岐命令」と呼ぶことにする．

極端な偏りのある分岐命令の実行頻度を測定する．ベンチマークプログラムとして，Intel MRL と IEEE TC-uARCH の支援で開催されている分岐予測器のコンテスト Championship Branch Prediction[15] のために作成された 20 本のトレース（**表 1**）を利用する．これら 20 本のトレースは，SpecFP（FP），SpecINT（INT），マルチメディア（MM），サーバ（SERV）という異なる 4 つの応用分野の実行履歴を集めたものである．それぞれの分野は，5 本のトレースで構成される．個々のトレースは 3 千万命令程度の履歴で，アプリケーションの実行履歴に加えてオペレーティングシステムの実行履歴を含んでいる．このため，SimpleScalar に代表されるアプリケーションの実行履歴のみを利用する場合と比較して，現実的な評価を行う

表 2 極端な偏りのある分岐命令を区別するための 2 つのフラグ
Table 2 Two flags to distinguish extremely biased branches.

	初期値	分岐成立	分岐不成立
all_taken_flag	1	–	0
all_untaken_flag	1	0	–

ための環境を提供する．

各ベンチマークプログラムの特徴と，極端な偏りのある分岐命令の実行頻度を測定した結果を表 1 にまとめる．表の 1 列目にベンチマークの名前を示す．表の 2 列目（実行命令数）に，各ベンチマークプログラムの実行命令数を示す．ここに示す実行命令数は，算術演算などの分岐命令以外の命令を含む数である．実行命令数は，3 千万から 3 千 5 百万の間に調整されていることが分かる．表の 3 列目（動的分岐の割合）には，各ベンチマークプログラムの実行命令数に占める条件分岐命令の動的な割合を示す．表の 4 列目（静的分岐の数）には，各ベンチマークプログラムの実行における静的な条件分岐命令の数を示す．SpecFP では条件分岐命令の数が 1,000 以下と少ない，一方，サーバ系のベンチマークでは，静的な条件分岐命令の数が 1 万以上と多い．表の 5 列目（All-taken の割合）に，すべての条件分岐命令の実行回数に占める，極端な偏りのある成立分岐命令の動的な割合を示す．DIST-MM-3 の値が最も高く，その割合は 35%に達する．また，サーバ系のすべてのベンチマークで 9%以上と高い．同様に，表の 6 列目（All-untaken の割合）に，すべての条件分岐命令の実行回数に占める，極端な偏りのある不成立分岐命令の動的な割合を示す．DIST-FP-1 の場合には 65%，サーバ系のすべてのベンチマークで 44%以上と非常に高い．6 列目と 7 列目にまとめた条件分岐命令が，極端な偏りのある分岐命令として分類される．

3.2 極端な偏りのある分岐命令を区別する方式の提案

本節では，フラグを用いて極端な偏りのある分岐命令を分類する方式を提案する．

プログラムの実行中に，それぞれの分岐命令の過去の分岐方向がすべて成立または過去の分岐方向がすべて不成立であるという情報を保存する．このために，分岐命令ごとに 2 つのフラグを利用する．これらのフラグを all_taken_flag と all_untaken_flag と呼ぶことにする．

2 つのフラグの更新方法を**表 2** にまとめる．プログラムの開始時に，すべての all_taken_flag と all_untaken_flag をセット（値 1 を格納）する．分岐結果が確定した時点で，分岐不成立の場合には，当該分岐命令に対応する all_taken_flag をリセットする．このとき，all_untaken_flag の状態は変更しない．分岐成立の場合には当該分岐命令に対応する all_untaken_flag をリセットする．このとき，all_taken_flag の状態は変更しない．分岐方向を予測する際に，all_taken_flag がセットされている命令は，過去のすべての分岐方向が成立だった極端な偏りのある分岐命令として，分岐成立と予測する．all_untaken_flag がセットされている命令は，過去のすべての分岐方向が不成立だった極端な偏りのある分岐命令として，分岐不成立と予測する．

本節で提案した仕組みにより分類される割合と，表 1 にまとめた割合とが異なる点に注意する必要がある．表 1 にまとめた値は，プログラムの実行が終了した時点ですべて成立だった分岐命令の実行頻度と，すべて不成立だった分岐命令の実行頻度である．提案した仕組みでは，予測する時点までの分岐結果がすべて成立だったもの，あるいは，予測する時点までの分岐結果がすべて不成立だったものを極端な偏りのある分岐命令とする．このため，表 1 に示した割合と比較して，提案した仕組みで分類される極端な偏りのある分岐命令の割合の方が高くなる．

提案した方式は，Agree 予測器[14] と共通点がある．Agree 予測は，BTB に格納する時点の分岐方向を偏りと見なして，これに同意するかどうかで予測する．一方，提案方式は，ある分岐の過去のすべての実行履歴を考慮して，極端な偏りのあるものだけを選択できる点が大きく異なる．

提案した方式は，Filter 機構[2] とも共通点がある．Filter 機構では，BTB に追加するカウンタを用いて，一定間隔の履歴における偏りのある命令を予測する．方針は近いが，Filter 機構は，カウンタを用いるという点，一定間隔の履歴を利用するという点で提案方式と大きく異なる．Filter 機構では，カウンタが飽和するまでの学習期間における予測は PHT を利用しなければならない．このため，学習期間における Filter 機構の利得が得られない．また，Filter 機構ではカウンタが飽和するまでの閾値を適切に設定する必要がある．一方，提案手法はカウンタを用いないのでハードウェア量の点で有利である．また，提案手法は，予測時までの当該命令のすべての履歴において同様の挙動を示す分岐命令を予測の対象とする．このため，極端な偏りがあると判定された分岐命令の予測精度は非常に高い．文献 2) の例を用いて具体的に説明する．10 回のイタレーションを処理するループが多数回繰り返される例を考える．このとき，分岐成立を 1 で不成立を 0

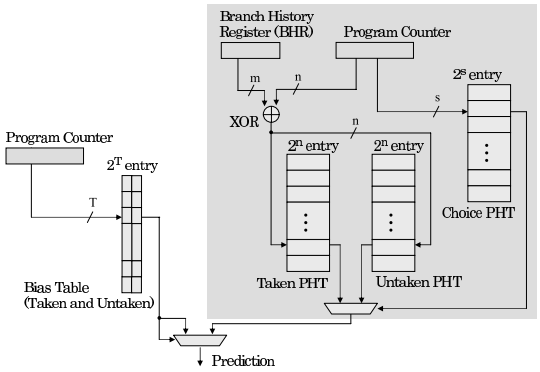


図 5 Bimode-Plus 予測器のブロック図
Fig. 5 Bimode-Plus branch predictor.

で表現すると，ループの終端の分岐命令の履歴では，1111111110 というパターンが繰り返されることになる．Filter 機構で，1 が 4 回連続したときに偏りがあると判断されると想定すると，1 イタレーションの中の残りの 6 回が Filter 機構による予測の対象となり，その中の 1 回で予測ミスが発生する．一方，提案手法では，10 個の分岐履歴を得た時点で分岐成立と不成立の両方が発生したことを検出し，以降の大部分の予測では，極端な偏りのある分岐と判定されない．

提案手法は BTB に変更を加える必要はない．また，BTB のエントリ有無を参照した分岐予測器[20] のように，予測結果を得るために BTB を参照することはない．

3.3 Bimode-Plus 予測器の提案

極端な偏りのある分岐命令を区別する方式を Bimode 予測器に組み込むことで，予測精度の向上を目指す．提案する予測器を Bimode-Plus 予測器[18] と名付ける．

Bimode 予測器が提案された後に，改良型として YAGS 予測器が提案されている．しかしながら，文献 23) で指摘されているように，YAGS 予測器の性能が Bimode 予測器より低い場合が存在することが報告されている．また，我々が YAGS 予測器を実装評価した場合にも，同様に，Bimode 予測器と比較して十分な予測精度の向上を確認することができなかった．このため，提案方式を組み込む対象から YAGS 予測器を除外した．提案方式を E-gskew 予測器や 2Bc-gskew 予測器に組み込むことも考えられるが，これらの予測器は複雑で構成パラメータが多い．これらの予測器を対象とした検討は今後の課題として，Bimode 予測器に提案方式を組み込むことにする．

Bimode-Plus 予測器の構成を**図 5** に示す．図右に示した網掛けの部分が Bimode 予測器である．左側に示した Bias テーブルは，all_taken_flag と all_untaken_flag というフラグ（2 ビット）を要素とする 2^T エントリのテーブルで，それぞれの分岐命令の過去の分岐方向がすべて成立または不成立であるという情報を保存する．

予測の方法を示す．プログラムカウンタからインデックスを生成し，予測すべき分岐命令に対応する Bias テーブルのエントリを参照する．このエントリには，all_taken_flag と all_untaken_flag というフラグが格納されている．もし，all_taken_flag がセットされている場合には，当該分岐命令は極端な偏りのある成立分岐命令であり，分岐成立と予測される．そうではなく，もし，all_untaken_flag がセットされている場合には，当該分岐命令は極端な偏りのある不成立分岐命令であり，分岐不成立と予測される．all_taken_flag と all_untaken_flag がともにリセットされている場合には，当該分岐命令は極端な偏りのある分岐命令ではない．この場合には，図 5 の右側に示した Bimode 予測器から得られる出力を用いて予測を行う．

テーブルの更新方法を示す．従来の Bimode 予測器と同様に，図 5 の右側に示した Bimode 予測器のテーブル（Choice PHT，Taken PHT，Untaken PHT）を更新する．ただし，極端な偏りのある分岐命令として予測した場合には，これらのテーブルの更新を行わない．図 5 の左側に示した Bias テーブルは，すべての条件分岐命令が処理された際に更新を行う．当該分岐命令のプログラムカウンタからインデックスを生成し，all_taken_flag と all_untaken_flag を選択する．もし，分岐結果が成立の場合には，all_untaken_flag をリセットする．もし，分岐結果が不成立の場合には，all_taken_flag をリセットする．

Bias テーブルと Choice PHT のインデックスの生成方法について検討する．Bias テーブルを参照して極端な偏りのある分岐命令と判断された場合に，当該分岐命令は Bias テーブルの情報のみから予測可能となり，Choice PHT を参照する必要がなくなる．もし，Bias テーブルと Choice PHT でまったく同じインデックスを用いるとすると，極端な偏りのある分岐命令と判定される場合に，対応する Choice PHT のエントリが利用されなくなる．これにより，Choice PHT の利用効率が低下する恐れが生じる．これを回避するためには，プログラムカウンタの異なる領域を用いて Bias テーブルと Choice PHT のインデックスを生成すればよい．

3.4 極端な偏りのある不成立分岐命令（成立分岐命令）のみを利用する方式

表 1 に示したように，極端な偏りのある成立分岐命令の頻度と比較して，極端な偏りのある不成立分岐命

令の頻度の方が多い傾向がある．また，極端な偏りのある成立分岐命令と，不成立分岐命令の挙動が異なる可能性がある．このことを明確にするために，極端な偏りのある成立分岐命令（あるいは不成立分岐命令）のみを利用する Bimode-Plus 予測器を提案する．これを実現するためには，図 5 に示した Bias テーブルから，all_taken_flag（あるいは all_untaken_flag）に対応するフィールドを削除すればよい．

極端な偏りのある成立分岐命令のみを利用する場合には，all_untaken_flag を削除する．予測に関しては，all_taken_flag がセットされている場合のみ，分岐成立と予測する．そうでない場合には，図 5 の右側に示した Bimode 予測器から得られる予測を利用する．テーブルの更新に関しては，all_untaken_flag が削除されたために，その更新が必要なくなった点を除いて，変更点はない．同様に，all_taken_flag を削除することで，極端な偏りのある不成立分岐命令のみを利用する Bimode-Plus 予測器を構成することができる．

極端な偏りのある成立分岐命令のみを利用する Bimode-Plus 予測器のことを Bimode-Plus (Taken) と記述する．同様に，極端な偏りのある不成立分岐命令のみを利用する Bimode-Plus 予測器のことを Bimode-Plus (Untaken) と記述する．極端な偏りのある成立分岐命令および不成立分岐命令を利用することを明示する場合には，Bimode-Plus (Taken & Untaken) と記述することがある．

3.5 分岐履歴レジスタ（BHR）の更新方式の提案

Direction PHT のインデックスを生成する際に，プログラムカウンタと分岐履歴レジスタ（BHR）との排他的論理和を利用する．分岐履歴レジスタはシフトレジスタになっており，レジスタの内容を 1 ビット右にシフトした後に，分岐予測結果の 1 ビットをレジスタの最上位ビット（Most Significant Bit）に格納する．

ここで，極端な偏りのある分岐命令は，つねに一定の分岐結果を生成することに注目する．Bimode 予測器と同様に，極端な偏りのある分岐命令の分岐結果を含むすべての条件分岐命令の結果を履歴レジスタに格納する方式を考える．このとき，特に，極端な偏りのある分岐命令が頻繁に出現する場合に，分岐履歴レジスタに格納されたビットのパターンが，前回の実行におけるビットのパターンと同様の傾向を示すようになり，グローバルなパス情報としての役割を果たさない可能性が生じる．このことから，次に示す，BHR_ALL 方式と BHR_NOB 方式という 2 つの分岐履歴レジスタの更新方式を提案する．

- **BHR_ALL（update by all conditional branch）**：すべての条件分岐命令と一部の無条件分岐命令☆の結果を用いて分岐履歴レジスタを更新する方式．Bimode 予測器と同様の分岐履歴レジスタの更新方式．
- **BHR_NOB（update by no-biased conditional branch）**：極端な偏りのある分岐命令を除く条件分岐命令と一部の無条件分岐命令の結果を用いて分岐履歴レジスタを更新する方式．

BHR_NOB 方式を用いる場合には，分岐方向を予測する際に，極端な偏りのある分岐命令として予測されたことを表す 1 ビットの情報を記憶しておく．分岐履歴レジスタの更新のときに，このビットを用いて極端な偏りのある分岐命令として予測されていた場合には分岐履歴レジスタの更新を行わないように制御すればよい．1 ビットの情報を記憶することが困難な場合には，Bias テーブルの更新頻度が高くないことを考慮して，分岐履歴レジスタの更新のときに，再度，Bias テーブルを参照し，極端な偏りのある分岐命令と判断された場合に更新を行わないように制御すればよい．これらを実現するために，複雑なハードウェアを必要とすることはない．

3.6 Bimode-Plus 予測器による予測精度の向上

3.6.1 パターン履歴表（PHT）の競合緩和

本項では，パターン履歴表の競合を緩和することで，Bimode-Plus 予測器の予測精度が向上する理由を説明する．

Bimode-Plus 予測器において，パターン履歴表が競合される様子を**図 6** に示す．図 6 (a) は，従来手法の Bimode 予測器において競合が発生する様子を，図 6 (b) は，提案手法の Bimode-Plus 予測器により競合が解消される様子を示している．

従来手法の Bimode 予測器の場合には，極端な偏りのある分岐命令（Choice PHT のエントリ A を参照する分岐命令）に関しても，それ以外の分岐命令と同様に，Choice PHT と Direction PHT を参照して予測結果を生成する．プログラムカウンタと分岐履歴レジスタとの排他的論理和によりインデックスを生成して，Direction PHT を参照するために，極端な偏りのある分岐命令であっても，複数の Direction PHT のエントリを参照し，当該エントリの内容を更新する．このため，別の分岐命令（Choice PHT のエントリ B を参照する分岐命令）が利用している Direction PHT

☆ 本稿における Bimode および Bimode-Plus 予測器の標準的な構成では，条件分岐命令に加えて，CALL 命令が処理された場合に分岐履歴レジスタを更新する．これは，CALL 命令における更新を加えた方が予測精度が高くなるためである．

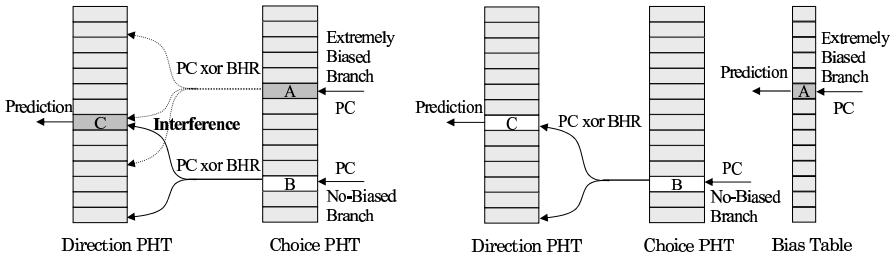


図 6 Bimode-Plus 予測器による履歴テーブルへの参照競合の緩和
Fig. 6 Mitigation of the PHT access interference with Bimode-Plus branch predictor.

のエントリとの間の参照競合が生じる可能性がある．

提案手法の Bimode-Plus 予測器の場合には，極端な偏りのある分岐命令は Bias テーブルのみを用いて予測を行う．この場合には，Choich PHT と Direction PHT の内容を参照，更新することはない．このため，極端な偏りのある分岐命令とそれ以外の命令とが，Choice PHT と Direction PHT の参照において競合することはない．すなわち，提案手法の Bimode-Plus 予測器においては，極端な偏りのある分岐命令に関する履歴を Choice PHT，Direction PHT に格納しないことで，これらの履歴テーブルの参照競合を緩和することができる．

3.6.2 分岐方向の学習時間の短縮

提案手法の Bimode-Plus 予測器は，分岐方向の学習時間を短縮することで，極端な偏りのある分岐命令の予測精度を改善する．この点に関して，従来手法と比較する．

Filter 機構を用いて，極端な偏りのある分岐命令を予測することを考える．Filter 機構では，カウンタを用いて，n 回連続して分岐方向が一致した場合に，偏りがあると判断する．このため，n+1 回目に異なる挙動を示す分岐命令のように，少ない頻度で異なる挙動を示す分岐命令において予測ミスが発生する．極端な偏りのある分岐命令を正しく判定するためにはカウンタの最大値 n を増やせばよいが，これは，ハードウェア量の増加と，偏りがあると判断するまでの n 回の学習期間の増大につながる．また，学習期間の増大は，Filter 機構を用いて予測される回数の削減となるため好ましくない．

Agree 予測器では，分岐命令が BTB に格納された時点の結果をバイアスビットに格納し，このバイアスビットに同意するか否かという判断で予測を行う．ただし，プログラムカウンタと分岐履歴レジスタの排他的論理和によるインデックスで得られた PHT のエントリの内容を用いて，バイアスビットに同意するか否かを判断する．このため，1 つの極端な偏りのある分岐命令に関連する複数の PHT のエントリの学習が正しく行われ，かつ，他の分岐命令による破壊的競合が発生しない場合に，極端な偏りのある分岐命令の予測を正確に行うことができる．

提案手法の Bimode-Plus 予測器は，プログラムカウンタのみによって生成されるインデックスを用いて Bias テーブルを参照する．また，すべての Bias テーブルのエントリは，極端な偏りのある分岐命令と判定するように初期設定される．このため，Filter 機構や Agree 予測器にみられる学習期間を必要としない．

4. 分岐予測器の予測精度

本章では，機能レベルのプロセッサシミュレータを用いて，分岐予測器の予測精度を議論する．

4.1 評 価 方 法

トレース駆動のシミュレータを構築して Gshare 予測器，Bimode 予測器，提案手法の Bimode-Plus 予測器を実装し，その性能を評価する☆．

分岐予測器は，分岐の成立あるいは不成立のどちらか（1 ビットの出力）を予測する条件分岐命令のみを予測の対象とする．ただし，分岐履歴レジスタの更新には，無条件分岐命令の結果を利用することがある．

ベンチマークプログラムとして，Intel MRL と IEEE TC-uARCH の支援で開催されている分岐予測器のコンテスト Championship Branch Prediction のために作成された 20 本のトレース（3.1 節と表 1 にまとめたもの）を利用する．

☆ 極端な偏りのある分岐命令を区別する方式と Filter 機構との間には共通点がある．このため，Bimode-Plus 予測器の評価において，Filter 機構を採用する Bimode 予測器を比較対象の 1 つに加えることが好ましい．しかしながら，我々が Filter 機構を採用する Bimode 予測器を実装して評価したところ，Filter 機構を追加することによる予測精度の改善を得ることができなかった．Bimode-Plus 予測器と Filter 機構を採用する Bimode 予測器との定量的な比較は今後の課題とする．

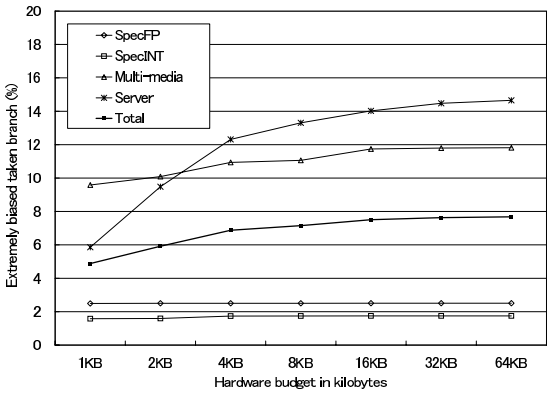


図 7 極端な偏りのある成立分岐命令の頻度（Bias テーブルのハードウェア量を 1 KB から 64 KB に設定）
Fig. 7 Rate of extremely biased **taken** branches.

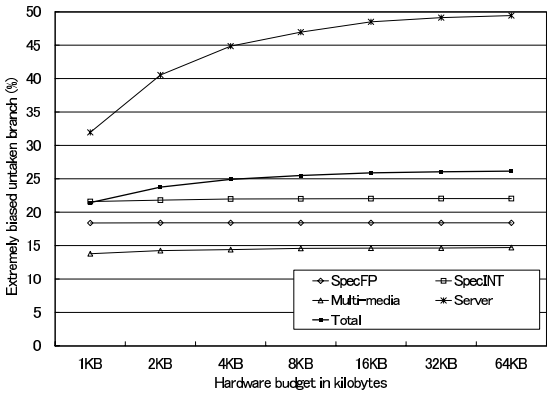


図 8 極端な偏りのある不成立分岐命令の頻度（Bias テーブルのハードウェア量を 1 KB から 64 KB に設定）
Fig. 8 Rate of extremely biased **untaken** branches.

分岐予測器の予測精度を示す尺度として，Championship Branch Prediction や文献 12) などで用いられている 1000 命令あたりの分岐予測ミスの回数（Mispredictions Per Kilo Instructions）を利用する．

SpecFP，SpecINT といった各分野の予測精度，あるいは 20 本のベンチマーク全体の予測精度を計算する場合には，Championship Branch Prediction の評価方法と同様に，算術平均を利用する．

4.2 極端な偏りのある分岐命令の頻度

Bimode-Plus 予測器の挙動を把握するために，Bimode-Plus 予測器が極端な偏りのある分岐命令と判断する頻度を測定する．極端な偏りのある成立分岐命令の分類結果を**図 7** に，不成立方向に極端な偏りのある分岐の分類結果を**図 8** に示す．横軸は Bimode-Plus 予測器における Bias テーブルのみのハードウェア量を示し，1 KB から 64 KB の範囲で変化させる．Bias テーブルのインデックスとして，プログラムカウンタの下位ビットを利用する．縦軸は，極端な偏りのある分岐命令と判断された頻度（条件分岐命令の実行回数に占める極端な偏りのある分岐命令と判断された条件分岐命令の実行回数）を表している．

図 7 の結果から，極端な偏りのある成立分岐命令の頻度は，Bias テーブルのハードウェア量を 64 KB と大きくした場合に，平均（Total）で 7.6%に達することが分かる．表 1 に示したように，SpecFP，SpecINT の静的な分岐命令の数は 243 から 1,585 と少ない．1 KB の Bias テーブルは 8,192 エントリを持つので，SpecFP，SpecINT のように静的な分岐命令の数が少ない場合には，1 KB のテーブルで多くの極端な偏りのある分岐命令の分離に成功する．一方，サーバ系のベンチマークでは，静的な分岐命令の数が 1 万以上と多いので，Bias テーブルのハードウェア量を増やすに従って，分類される分岐命令の割合が増加する．マルチメディア系のベンチマークは，ハードウェア量の増加に従って，分類される分岐命令の割合が緩やかに増加する．Bias テーブルのハードウェア量を 64 KB と大きくした場合に，マルチメディア系のベンチマークで平均 11.8%，サーバ系のベンチマークで平均 14.6%の条件分岐命令が極端な偏りのある成立分岐命令として識別される．

同様に，図 8 の結果から，極端な偏りのある不成立分岐命令の頻度は，Bias テーブルのハードウェア量を 64 KB と大きくした場合に，平均で 26.1%に達することが分かる．この値は，極端な偏りのある成立分岐命令の 3.4 倍と非常に高い．分野別の結果では，64 KB の Bias テーブルを用いた場合に，サーバ系のベンチマークの平均で不成立分岐命令の頻度が 49.4%と極端に高い．また，サーバ系のベンチマークでは，Bias テーブルのハードウェア量を大きくすることによる利得が大きいことが分かる．

4.3 BHR_ALL 方式を採用する Bimode-Plus 予測器の予測精度

Gshare 予測器，Bimode 予測器，提案手法の Bimode-Plus 予測器の予測精度を評価する．本節の評価では，Bimode 予測器との比較を明確にするために，分岐履歴レジスタの更新方式を Bimode 予測器と同様の BHR_ALL 方式に設定する．それぞれの予測器のハードウェア量が 8 KB，16 KB，32 KB，64 KB となるよう，以下のようにパラメータを調整する．

Bimode 予測器の場合には，Choice PHT のエントリ数を Direction PHT のエントリ数の倍（図 2 において $s = n + 1$）に設定する．たとえば，8 KB の Bimode 予測器では，Choice PHT 4 KB，Taken PHT 2 KB，Untaken PHT 2 KB の合計 8 KB となる．

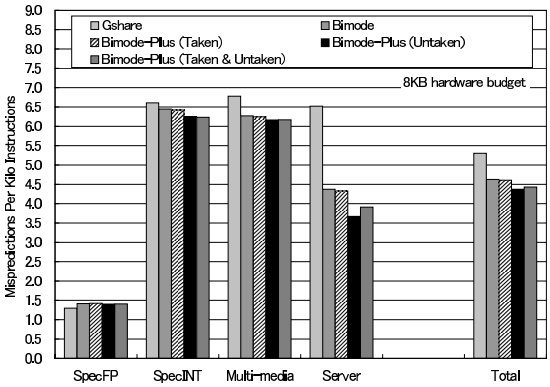


図 9 BHR_ALL 方式の Bimode-Plus 予測器の予測精度（8 KB）

Fig. 9 Evaluation result of Bimode-Plus branch predictor with BHR_ALL policy of 8 KB hardware budget.

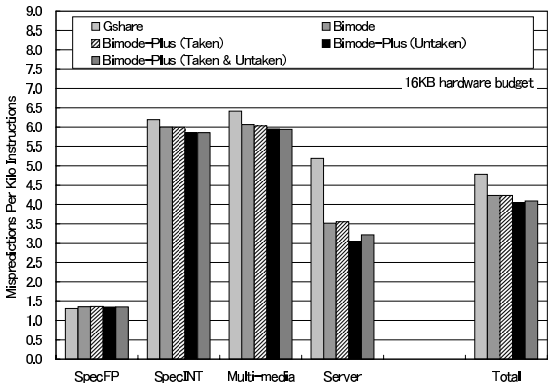


図 10 BHR_ALL 方式の Bimode-Plus 予測器の評価結果（16 KB）

Fig. 10 Evaluation result of Bimode-Plus branch predictor with BHR_ALL policy of 16 KB hardware budget.

分岐成立と不成立の両方の偏りを利用する Bimode-Plus 予測器の場合には，Choice PHT のエントリ数，Direction PHT のエントリ数，Bias テーブルのエントリ数を等しく設定する．たとえば，8 KB のハードウェア量の場合には，Choice PHT 2 KB，Taken PHT 2 KB，Untaken PHT 2 KB，Bias テーブル 2 KB の合計 8 KB となる．

単一の方向の偏りを利用する Bimode-Plus 予測器の場合には，Bias テーブルにおけるエントリあたりのハードウェア量が半分になる．このため，Choice PHT のエントリ数と Direction PHT のエントリ数を同一に，Bias テーブルのエントリ数をこれらの倍に設定する．この場合にも，Choice PHT 2 KB，Taken PHT 2 KB，Untaken PHT 2 KB，Bias テーブル 2 KB の合計 8 KB となる．

図 9 に，ハードウェア量を 8 KB に設定した Gshare 予測器，Bimode 予測器，Bimode-Plus 予測器の予測精度をまとめる．Bimode-Plus 予測器では，極端な偏りのある成立分岐命令を利用するもの Bimode-Plus (Taken)，極端な偏りのある不成立分岐命令を利用するもの Bimode-Plus (Untaken)，両方を利用するもの Bimode-Plus (Taken & Untaken) の 3 つの構成の結果を示す．SpecFP，SpecINT，マルチメディア，サーバという異なる分野の算術平均と，20 本のトレース全体の算術平均（Total）を示す．

図 9 の結果から，Bimode と Bimode-Plus (Taken) の予測精度は同程度であることが分かる．しかし，5 つの分岐予測器の比較では，Bimode-Plus (Untaken) が最も高い予測精度を達成し，続いて，Bimode-Plus (Taken & Untaken) が高い予測精度を達成する．8 KB のハードウェア量の構成では，Bimode と比較して，Bimode-Plus (Untaken) がサーバ系のベンチマークで 16.0%，すべてのベンチマークプログラムの平均で 5.4%の予測ミスを削減する．

図 10 に，ハードウェア量を 16 KB に設定した Gshare 予測器，Bimode 予測器，Bimode-Plus 予測器の予測精度をまとめる．全体的な傾向は，図 9 に示した 8 KB の構成に近い．5 つの分岐予測器の比較では，Bimode-Plus (Untaken) が最も高い予測精度を，続いて，Bimode-Plus (Taken & Untaken) が高い予測精度を達成する．16 KB のハードウェア量の構成では，Bimode と比較して，Bimode-Plus (Untaken) がサーバ系のベンチマークで 13.5%，すべてのベンチマークプログラムの平均で 4.3%の予測ミスを削減する．

ハードウェア量を 8 KB，16 KB，32 KB，64 KB に設定した Gshare 予測器，Bimode 予測器，Bimode-Plus 予測器の予測精度を**図 11** にまとめる．それぞれの構成の予測器に関して 20 個のベンチマークプログラムの算術平均の値を示す．図 11 では，分岐予測器の間の相違が明確になるように，縦軸の予測精度の範囲を調整している．

図 11 の結果を議論する．Bimode と Bimode-Plus (Taken) の比較では，ハードウェア量が 8 KB の構成で，わずかに Bimode-Plus (Taken) が高い予測精度を示すが，16 KB よりハードウェア量が大きい領域で優劣が逆転する．5 つの構成の比較では，すべてのハードウェア量の設定で，Bimode-Plus (Untaken) が最も高い予測精度を示す．続いて，Bimode-Plus (Taken & Untaken) が高い予測精度を示す．

これら，Bimode-Plus (Untaken) が最も高い予測精度を示すという評価結果から，次節では，Bimode-

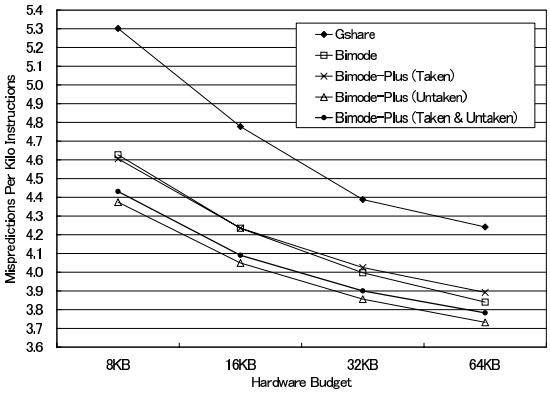


図 11 ハードウェア量を変化させた場合の BHR_ALL 方式の Bimode-Plus 予測器の予測精度
Fig. 11 Evaluation result of Bimode-Plus branch predictor with BHR_ALL policy (8 KB, 16 KB, 32 KB and 64 KB hardware budget).

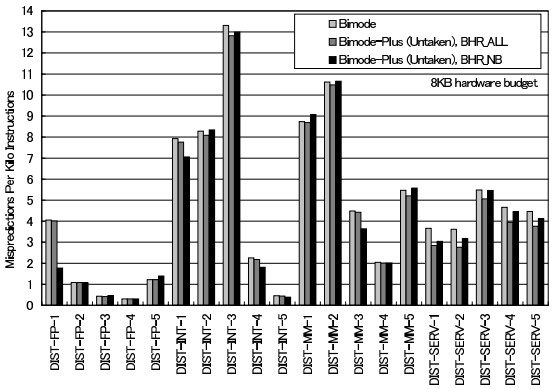


図 12 ベンチマークごとの Bimode-Plus (Untaken) の予測精度（8 KB）
Fig. 12 Evaluation result of the 8 KB Bimode-Plus (Untaken) branch predictor.

Plus (Untaken) に絞って，分岐履歴レジスタの更新方式の影響を評価する．

4.4 分岐履歴レジスタの更新方式

本節では，分岐履歴レジスタの更新方式として，BHR_ALL 方式と BHR_NOB 方式を用いた Bimode-Plus (Untaken) の予測精度を比較する．

図 12 に，ハードウェア量を 8 KB に設定した場合の評価結果をまとめる．横軸は 20 本のベンチマークを，縦軸は予測精度を示している．それぞれのベンチマークにおいて，左から，Bimode，BHR_ALL 方式を採用する Bimode-Plus (Untaken)，BHR_NOB 方式を採用する Bimode-Plus (Untaken) の 3 本の結果を示す．

図 12 の結果から，BHR_ALL 方式を用いる Bimode-Plus (Untaken) の予測精度と比較して，DIST-FP-1，DIST-INT-4，DIST-MM-3 の各ベンチマークで，BHR_NOB 方式を採用することによる大幅な予測精度の改善を確認できる．特に，SpecFP において予測精度の悪い DIST-FP-1 では，BHR_ALL 方式を用いる場合と比較して，BHR_NOB 方式を採用することで 55.7%の予測ミスを削減する．一方，サーバ系のベンチマークとその他のいくつかのベンチマークでは，BHR_ALL 方式と比較して，BHR_NOB 方式を採用することで予測精度が低下する．

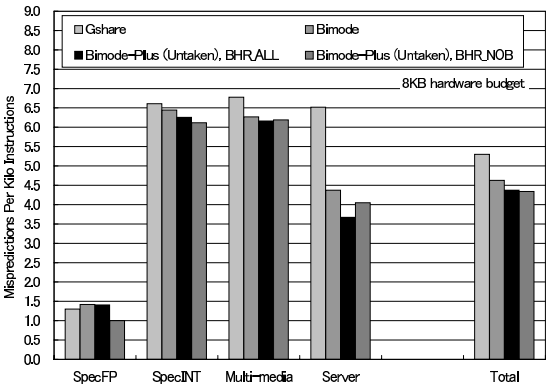


図 13 Bimode-Plus (Untaken) の予測精度（8 KB）
Fig. 13 Evaluation result of the 8 KB Bimode-Plus (Untaken) branch predictor.

図 12 に示した 8 KB の Bimode-Plus 予測器の予測精度から，分野と全体の平均を計算した結果を**図 13** にまとめる．SpecFP と SpecINT には BHR_NOB 方式が適している，一方，マルチメディア系とサーバ系には BHR_ALL 方式が適している．全体の平均では，BHR_NOB 方式を用いた Bimode-Plus (Untaken) の予測精度が最も高い．ハードウェア量を 8 KB に設定した構成では，Bimode 予測器と比較して，BHR_NOB 方式を用いた Bimode-Plus (Untaken) が 6.2%の予測ミスを削減する．

図 14 に，ハードウェア量を 16 KB に設定した場合の Bimode-Plus (Untaken) の予測精度の評価結果を示す．全体的な傾向は，図 12 に示した 8 KB の構成の結果と同様である．DIST-FP-1，DIST-INT-4，DIST-MM-3 の各ベンチマークでは，BHR_NOB 方式を採用することで大幅な予測精度の改善を達成できることが分かる．BHR_ALL 方式を用いる Bimode-Plus (Untaken) の予測精度と比較して，BHR_NOB 方式を採用することで，DIST-FP-1 が 57.8%，DIST-INT-4 が 20.8% DIST-MM-3 が 21.9%の予測ミスを削減する．

図 14 に示した 16 KB の Bimode-Plus 予測器の予測精度から，分野と全体の平均を計算した結果を**図 15** にまとめる．8 KB の構成と同様に，SpecFP と SpecINT

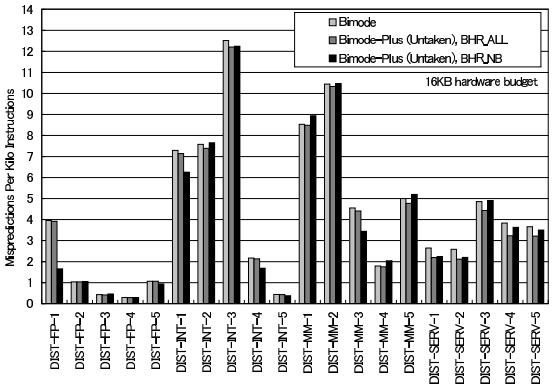


図 14 ベンチマークごとの Bimode-Plus (Untaken) の予測精度（16 KB）
Fig. 14 Evaluation result of the 16 KB Bimode-Plus (Untaken) branch predictor.

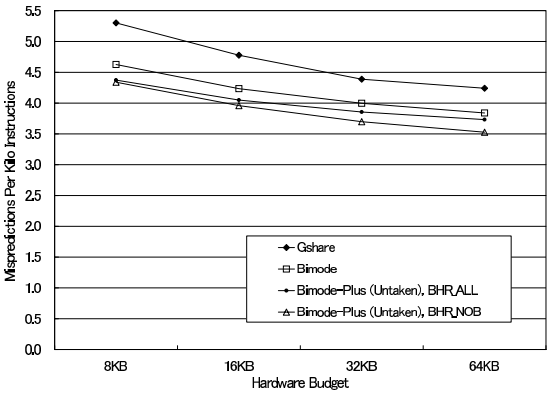


図 16 Bimode-Plus (Untaken) の予測精度
Fig. 16 Evaluation result of Bimode-Plus (Untaken) branch predictor with 8 KB, 16 KB 32 KB and 64 KB hardware budget.

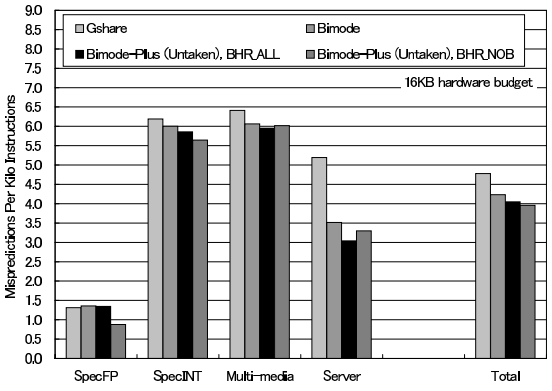


図 15 Bimode-Plus (Untaken) の予測精度（16 KB）
Fig. 15 Evaluation result of the 16 KB Bimode-Plus (Untaken) branch predictor.

には，BHR_NOB 方式が適している，一方，マルチメディア系とサーバ系では BHR_ALL 方式が適している．全体の平均では，BHR_NOB 方式を用いた Bimode-Plus (Untaken) の予測精度が最も高い．ハードウェア量を 16 KB に設定した構成では，Bimode 予測器と比較して，BHR_NOB 方式を用いた Bimode-Plus (Untaken) が 6.4%の予測ミスを削減する．

図 16 に，ハードウェア量を 8 KB，16 KB，32 KB，64 KB に設定した，Gshare 予測器，Bimode 予測器，BHR_ALL 方式を採用する Bimode-Plus (Untaken)，BHR_NOB 方式を採用する Bimode-Plus (Untaken) の予測精度をまとめる．すべてのハードウェア量の構成において，BHR_NOB 方式を利用する Bimode-Plus (Untaken) の精度が最も優れている．続いて，BHR_ALL 方式を利用する Bimode-Plus (Untaken)，Bimode，Gshare の順番となる．BHR_ALL 方式と比較して，BHR_NOB 方式を利用する Bimode-Plus (Untaken) は，ハードウェア量が 32 KB，64 KB と大きいところで大幅な予測精度の向上を達成できる．一方，BHR_ALL 方式を利用する Bimode-Plus (Untaken) の予測精度は，ハードウェア量が増えるに従って Bimode と同様の精度まで低下する．

従来手法の Bimode との比較では，8 KB，16 KB，32 KB，64 KB の BHR_ALL 方式を利用する Bimode-Plus (Untaken) が，それぞれ 5.4%，4.3%，3.5%，2.8%の予測ミスを削減する．また，8 KB，16 KB，32 KB，64 KB の BHR_NOB 方式を利用する Bimode-Plus (Untaken) は，それぞれ 6.2%，6.4%，7.4%，8.1%の予測ミスを削減する．

BHR_ALL 方式では分岐予測器の容量を増やすと Bimode の予測精度に近付いていく．Bimode-Plus 予測器の構成はハードウェア量が 8 KB から 16 KB の現実的な部分を想定して最適化を施している．このため，64 KB のハードウェア量の BHR_ALL 方式では Bias テーブルの容量が 8 KB（65,536 エントリ）と必要以上に大きくなり，無駄が生じる．また，分岐予測器の容量を大きくした場合には，分岐命令間の履歴テーブル利用の競合が低減し，Bimode 予測器でも極端な偏りのある分岐命令を正しく予測できる割合が向上すると考えられる．これらの理由から，ハードウェア量を増やしたときに，BHR_ALL 方式の Bimode-Plus 予測器の精度が Bimode 予測器の精度に近付くと考えられる．

図 16 と同じ構成の予測器の結果を，縦軸を予測ミス率としてまとめたグラフを**図 17** に示す．8 KB のハードウェア量のとき，従来手法の Bimode 予測器での予測ミス率 4.33%を，提案手法の BHR_NOB 方式を利用する Bimode-Plus (Untaken) により 4.04%へと改

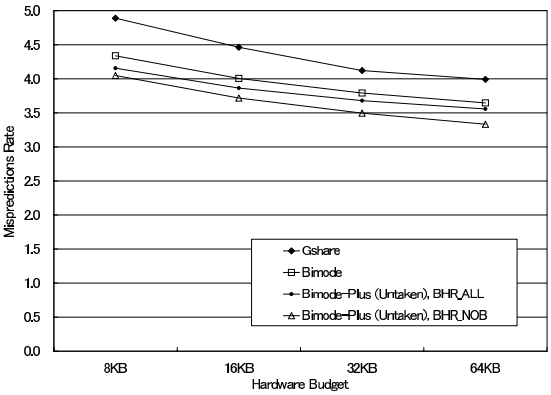


図 17 Bimode-Plus (Untaken) の予測失敗率
Fig. 17 Misprediction rate of Bimode-Plus (Untaken) branch predictor with 8 KB, 16 KB 32 KB and 64 KB hardware budget.

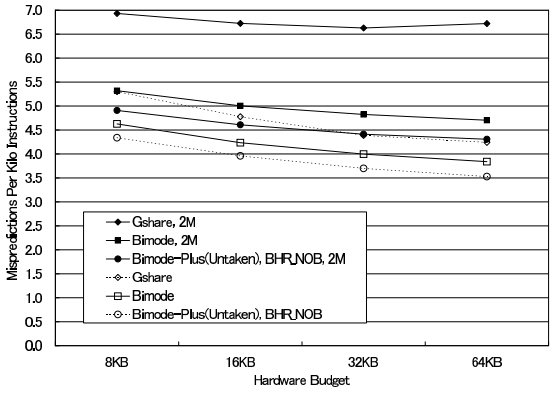


図 18 コンテキストスイッチの影響（間隔を 200 万命令に設定）
Fig. 18 Misprediction rate of Bimode-Plus (Untaken) under the 2 M instruction context switching environment.

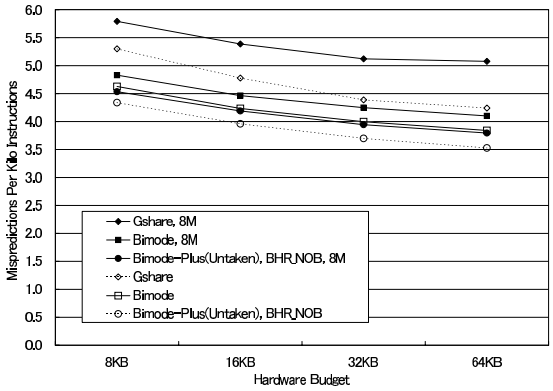


図 19 コンテキストスイッチの影響（間隔を 800 万命令に設定）
Fig. 19 Misprediction rate of Bimode-Plus (Untaken) under the 8 M instruction context switching environment.

善する．同様に，提案手法は，16 KB のハードウェア量のとき 4.00%を 3.71%へ，32 KB のハードウェア量のとき 3.79%を 3.49%へ，64 KB のハードウェア量のとき 3.64%を 3.33%へと予測ミス率を改善する．

4.5 コンテキストスイッチの影響

高性能マイクロプロセッサは，複数のプロセス（あるいはスレッド）を切り替えながら処理を進めていく．本節では，このように，複数のタスクを同時実行する環境が分岐予測精度に与える影響を議論する．

本節では，コンテキストスイッチが発生する間隔を 200 万命令と 800 万命令に設定して分岐予測器の精度を測定する．オペレーティングシステムが 1 ミリ秒の間隔でコンテキストスイッチを行い，プロセッサの動作周波数を 2 GHz，サイクルあたりの平均処理命令数を 1 と想定すると，200 万命令の単位でコンテキストスイッチが発生する．動作周波数を 4 GHz，サイクルあたりの平均処理命令数を 2 とする高性能プロセッサを想定すると，800 万命令の単位でコンテキストスイッチが発生する．

これらの設定では，オペレーティングシステムがタスクを切り替える間隔を 1 ミリ秒としている．一方，現在の一般的なオペレーティングシステムでは，この間隔が 10 ミリ秒程度に設定されることが多い．このため，先に示した，200 万命令と 800 万命令という間隔は，コンテキストスイッチが頻繁に行われる最悪の場合の評価ととらえることができる．

コンテキストスイッチが発生したときの分岐履歴への対処として，すべての履歴テーブルの情報を初期化する方式と，履歴テーブルの情報に変更を加えることなく上書きする方式を考えることができる．本節では，文献 5) などで利用されている，前者の，すべての履歴テーブルの情報をコンテキストスイッチの際に初期化する方式を採用する．

200 万命令の単位でコンテキストスイッチが発生すると想定した場合の予測精度を**図 18** に，800 万命令とした場合の予測精度を**図 19** にまとめる．図には，Gshare 予測器，Bimode 予測器，BHR_NOB 方式を採用する Bimode-Plus (Untaken) 予測器に関して，コンテキストスイッチを行う場合（実線で表示）とそうでない場合（破線で表示）の 6 本のデータを示した．横軸はハードウェア量，縦軸が予測精度である．

図 18 の 200 万命令の単位でコンテキストスイッチが発生すると想定した場合の結果を議論する．実線と破線との比較から，すべての予測器でコンテキストスイッチによる精度の低下を確認できる．しかしながら，同一の条件の Bimode 予測器と Bimode-Plus 予測器を比較すると，コンテキストスイッチの有無にかかわ

表 3 SPEC CINT2000 の特性と極端な偏りのある分岐の頻度
Table 3 Characteristics of SPEC CINT2000 and the rate of extremely biased branches.

ベンチマーク	実行命令数	動的分岐の割合	静的分岐の数	All-taken の割合	All-untaken の割合
164.gzip	596 million	10.19%	858	0.03%	2.11%
176.gcc	551 million	12.35%	17,593	10.54%	18.88%
181.mcf	188 million	11.25%	841	1.30%	2.57%
186.crafty	800 million	8.80%	2,506	4.34%	6.68%
197.parser	611 million	13.41%	2,473	5.03%	15.55%
252.eon	94 million	6.54%	1,522	6.79%	28.84%
254.gap	800 million	9.92%	3,334	27.08%	17.61%
255.vortex	147 million	10.11%	6,179	19.51%	53.78%
256.bzip2	800 million	9.31%	806	20.96%	2.03%
300.twolf	91 million	8.12%	2,724	9.20%	27.24%

らず，Bimode 予測器に対して，Bimode-Plus 予測器は 6.2%から 8.5%の予測ミスを削減できることが分かる．

図 19 の 800 万命令の単位でコンテキストスイッチが発生すると想定した場合の結果を議論する．コンテキストスイッチの頻度が減少したために，図 18 に見られるほどの精度低下はおこらない．この場合も，同一条件の Bimode 予測器と Bimode-Plus 予測器を比較すると，コンテキストスイッチの有無にかかわらず，Bimode 予測器に対して，Bimode-Plus 予測器は 6.1%から 8.1%予測ミスを削減する．

これらの評価結果から，コンテキストスイッチが発生する実際のプロセッサの環境においても，Bimode-Plus 予測器は，Bimode 予測器に対して，6.1%から 8.5%の予測ミスを削減できることが分かる．

5. 分岐予測器の精度向上とプロセッサ性能向上との関係

本章では，アウトオブオーダ実行を模倣するクロックレベルのプロセッサシミュレータ（SimpleScalar [3] リリース 3.0d に含まれる sim-outorder）を用いて，分岐予測器の予測精度向上とプロセッサの処理性能向上との関係を議論する．

5.1 評 価 環 境

3.1 節にまとめた分岐予測器を評価するためのトレースはバイナリとして提供されているわけではないので，SimpleScalar の入力として利用することはできない．このため，SPEC CINT2000 の 10 本のプログラムを利用してプロセッサ性能との関係を評価する．SPEC CINT2000 のバイナリは，SimpleScalar のサイトからダウンロードしたものを利用する．CINT2000 には 12 本のプログラムが含まれるが，シミュレータの動作検証の問題から 175.vpr と 253.perlbmk を除く 10 本を利用する．186.crafty，252.eon，254.gap を除く 7 本のプログラムに関しては，University of Minnesota が提供している縮小入力セット MinneSPEC [1] 利用する．186.crafty，252.eon，254.gap に関しては，SPEC CINT2000 が提供する test 入力セットを参考にシミュレーションの命令数を抑えるように入力パラメータを調整したものを利用する．シミュレーション時間の制約から，個々のベンチマークプログラムの実行命令数の上限を 8 億命令とする．ベンチマークプログラムの特徴を**表 3** にまとめる．表 3 の形式は，表 1 と同じである．

SimpleScalar に，Gshare 予測器，Bimode 分岐予測器，Bimode-Plus 予測器を実装する．これらのハードウェア量は 16 KB とし，分岐予測器のバラメータは 4.3 節で定義したものを利用する．SimpleScalar の構成として次に示す BASE，WIDE という 2 つのプロセッサモデルを利用する．

BASE モデル サイクルあたり 4 命令のフェッチ，デコード，完了を行うプロセッサ．Register Update Unit（RUU）のサイズを 16 エントリとする．2048 エントリ，4 ウェイのセット・アソシアティブ方式の BTB とする．高性能プロセッサが採用する深いパイプライン段数を考慮して，分岐予測ミスのペナルティを 10 サイクルとする☆．これは，分岐予測に関する部分を除いて SimpleScalar のデフォルトのパラメータを用いた保守的な構成である．

WIDE モデル サイクルあたり 8 命令のフェッチ，デコード，完了を行う大規模なプロセッサを想定する構成．RUU のサイズを 512 エントリとする．2048 エントリ，4 ウェイのセット・アソシアティブ方式の BTB とする．分岐予測ミスのペナルティを 10 サイクルとする．32 KB，4 ウェイのセット・ア

☆ SimpleScalar の起動パラメータは次のとおり．
-bpred:spec_update ID -bpred:btb 2048 4 -fetch:mplat 10

表 4 アウトオブオーダ実行のプロセッサで測定した予測精度（単位は mispredictions per kilo instructions）
Table 4 Prediction accuracy measured on a out-of-order processor simulator.

プロセッサモデル 分岐予測器	BASE Gshare	BASE Bimode	BASE Bimode-Plus	WIDE Gshare	WIDE Bimode	WIDE Bimode-Plus
164.gzip	4.52	3.92	3.83	7.82	5.16	4.64
176.gcc	11.22	8.21	7.60	17.79	10.96	9.36
181.mcf	11.76	5.10	4.46	11.14	5.01	4.86
186.crafty	5.87	5.38	5.49	9.22	6.75	6.51
197.parser	9.82	8.50	7.67	20.26	11.42	9.40
252.eon	3.97	2.52	1.91	9.13	3.25	1.71
254.gap	3.94	2.64	2.60	6.81	3.62	3.06
255.vortex	7.93	2.17	1.26	9.89	3.07	1.66
256.bzip2	9.11	8.44	8.61	11.91	9.18	9.26
300.twolf	8.23	7.34	6.42	11.26	9.04	6.73
Average	7.63	5.42	4.99	11.52	6.75	5.72

ソシアティブ方式の命令キャッシュとデータキャッシュとする☆.

5.2 アウトオブオーダ実行のプロセッサを用いて測定した予測精度

Gshare 予測器，Bimode 予測器，BHR_NOB 方式を採用する Bimode-Plus 予測器の予測精度を**表 4** にまとめる．表は，BASE モデルと WIDE モデルのアウトオブオーダ実行プロセッサの構成で測定した予測精度をまとめたものである．分岐予測器の更新は命令デコードのステージで投機的に行われる．このため，分岐予測器の精度はプロセッサモデルにより異なることに注意する必要がある．

BASE モデルの場合の，Gshare 予測器，Bimode 予測器，Bimode-Plus 予測器の予測精度（1000 命令あたりの予測ミスの回数）は，7.63，5.42，4.99 である．Bimode 予測器と比較して，Bimode-Plus 予測器は 8.0%の予測ミスを削減する．

WIDE モデルの場合の，Gshare 予測器，Bimode 予測器，Bimode-Plus 予測器の予測精度は，11.52，6.75，5.72 である．Bimode 予測器と比較して，Bimode-Plus 予測器は 15.2%の予測ミスを削減する．BASE モデルと比較して，WIDE モデルの方が Bimode-Plus 予測器の利得が大きくなっている．Bimode-Plus 予測器では，極端な偏りのある分岐命令と判定され，予測が成功した場合に，分岐履歴テーブルが更新されることはない．このため，大規模なプロセッサ構成において発生する投機的な履歴テーブルの更新による汚染の問題を緩和できると考えられる．プロセッサモデルと分岐予測器の精度に関する詳細な検討は今後の課題である．

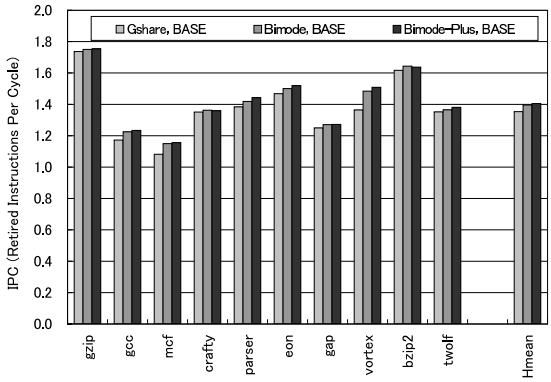


図 20 BASE モデルの分岐予測器とプロセッサ性能の関係
Fig. 20 Processor performance of the BASE model.

5.3 分岐予測器の精度とプロセッサ性能

BASE モデルのプロセッサ性能を**図 20** に，WIDE モデルのプロセッサ性能を**図 21** にまとめる．縦軸はプロセッサ性能に対応するサイクルあたりの完了命令数（Retired Instructions Per Cycle，IPC）で，IPC が高いほどプロセッサ性能が高いことになる．横軸はベンチマークプログラムの名前である．右端には，10 本のプログラムの調和平均の値を表示する．それぞれのベンチマークには，Gshare 予測器，Bimode 予測器，Bimode-Plus 予測器の 3 つの構成の結果を示す．

図 20 の BASE モデルの結果を議論する．10 本のプログラムの調和平均では，Bimode 予測器の IPC=1.39 に対して，Bimode-Plus 予測器の IPC=1.40 であり，性能向上は 0.6%と低い．BASE モデルでは，分岐予

☆ SimpleScalar の起動パラメータは次のとおり．
-bpred:spec_update ID, -bpred:btb 2048 4, -fetch:mplat 10, -fetch:ifqsize 8, -decode:width 8, -issue:width 8, -commit:width 8, -ruu:size 512, -lsq:size 128, -res:ialu 8, -res:imult 4, -res:fpalu 8, -res:fpmult 4, -mem:width 32, -cache:il1 il1:1024:64:4:l, -cache:il1lat 1, -cache:dl1 il1:1024:64:4:l, -cache:dl1lat 1, -cache:dl2 ul2:8192:64:4:l, -cache:dl2lat 6

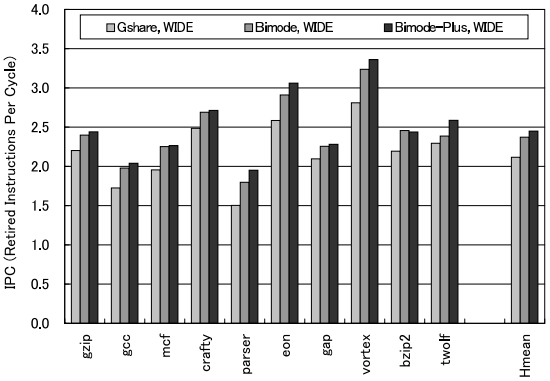


図 21 WIDE モデルの分岐予測器とプロセッサ性能の関係
Fig. 21 Processor performance of the WIDE model.

測器のほかに ALU の数といった資源の制約が厳しい．このため，分岐予測の精度向上がプロセッサ全体の性能向上に与える影響が小さくなると考えられる．

図 21 の WIDE モデルの結果を議論する．10 本のプログラムの調和平均では，Bimode 予測器の IPC=2.37 に対して，Bimode-Plus 予測器は IPC=2.45 であり，3.2%の性能向上を達成する．個別のベンチマークの結果を見る．Bimode 予測器に対して，Bimode-Plus 予測器を利用する構成では，bzip2 においてのみ IPC が 0.7%低下する．それ以外の 9 本のベンチマークでは性能が向上し，特に，197.parser の性能向上が 8.5%，300.twolf の性能向上が 8.3%と高い．

本章では，分岐予測器の予測精度向上とプロセッサの処理性能向上の関係を検討した．保守的なプロセッサ構成の BASE モデルと，大規模な構成の WIDE モデルを定義し，それぞれのモデルにおける分岐予測器の影響を評価した．評価結果から，BASE モデルにおいては Bimode-Plus 予測器の利得が小さいが，WIDE モデルでは Bimode-Plus 予測器により平均で 3.2%，最大で 8.5%の速度向上を達成できることを示した．

本章の評価では，大規模なプロセッサとして Register Update Unit（RUU）のサイズが 512 エントリの WIDE モデルを想定した．一方，次世代のプロセッサアーキテクチャとして，数千命令が処理されるプロセッサの研究[6] が進められており，このような大規模なプロセッサには精度の高い分岐予測器が必要となることが指摘されている．大規模なプロセッサモデルや近年提案されている様々な高速化技術を組み込んだプロセッサモデルの処理性能と分岐予測精度の関係の詳細な検討は今後の課題である．

6. 議　　論

6.1 低消費電力

低消費電力という視点から議論する．Bimode-Plus 予測器では，極端な偏りのある分岐命令を予測する場合に，Choice PHT と Direction PHT への参照を必要としない．Bias テーブルを確認した後に Choice PHT と Direction PHT の参照の必要性を判断できるとすれば，Choice PHT と Direction PHT への参照回数を削減できる．

4.2 節で議論したように，4 KB 程度の Bias テーブルを利用した場合に，平均で，6.8%が極端な偏りのある成立分岐命令，24.9%が極端な偏りのある不成立分岐命令に分類される．また，サーバ系のアプリケーションでは，平均で 12.3%が極端な偏りのある成立分岐命令，44.8%が極端な偏りのある不成立分岐命令に分類され，これらの条件分岐命令に関して，Bias テーブルを参照するだけで予測が可能となる．これにより，予測器全体としての消費電力を削減できる可能性がある．

2Bc-gskew 予測器[13] のように，利用するテーブル数が増加すると，分岐予測器の構成が複雑になる．その場合に，提案方式を用いて極端な偏りのある分岐命令をシンプルなハードウェアで予測して，その他の分岐命令と区別することによる利得が増大する可能性がある．これらの分岐予測器に，極端な偏りのある分岐命令を区別する方式を組み込むことは今後の課題である．

6.2 ハードウェアの複雑さ

ハードウェアの複雑さを議論する．提案した Bimode-Plus 予測器を実現するためには，Bimode 予測器に Bias テーブルといくつかの回路を追加する必要がある．これらの変更はそれほど大きなものではない．Bimode 予測器と比較して，予測のためのサイクル数（レイテンシ）が増加することはないと考えられる．

2001 年に提案された Perceptron 予測器[8] は，Bimode 予測器と比較して，8.2%の予測ミスを削減する☆．しかしながら，Perceptron 予測器は，ハードウェアが複雑になる，予測を行うためのレイテンシが長い，予測器の学習が必要となる，といった欠点がある．文献 7) によると，8 KB の Perceptron 予測器が予測結果を生成するまでのレイテンシは 4 サイクルと見積もられている．同文献 7) では，予測のレイテンシの短縮

☆ ベンチマークプログラムの違いなどにより，Perceptron 予測器の 8.2%という予測ミスの削減と，本稿で示した Bimode-Plus 予測器の削減率を直接比較することはできない．

手法などが議論されているが，本提案の Bimode-Plus 予測器と比較すると，ハードウェアが複雑となることは避けられない．

Bimode-Plus 予測器は，単純なハードウェア構成を保ちながら，Bimode 予測器と比較して，6.2 から 8.1%の予測ミスを削減する．

6.3 階層化された予測機構における有効性

命令フェッチの速度を分岐命令により低下させないためには，分岐方向を正しく予測するとともに，予測アドレスを早く送出する必要がある[21]．このため，現在のハイエンドプロセッサにおける分岐予測機構は，精度は低いが瞬時の予測が可能な予測器と，高精度の予測が可能であるが予測に時間がかかる予測器を用意する階層化されたものが主流となっている．

Alpha 21264 プロセッサ[11] の場合には，命令フェッチステージでライン予測☆を利用して，命令キャッシュ上の次のブロックを予測する．続く，パイプラインステージにおいてローカル予測器とグローバル予測器のハイブリッド分岐予測器を用いて分岐方向を予測する．ここでライン予測と分岐予測の方向が異なっている場合には，ライン予測の結果の破棄して，分岐予測の結果を用いて処理を進める．Alpha EV8 [12] の場合にも，同様に，ライン予測とそれを支援するパイプライン化された PC アドレス生成ユニットという階層化された予測機構を利用する．

提案した，極端な偏りのある分岐命令を区別する Bimode-Plus 予測器を利用することで，極端な偏りのある分岐命令に関しては，Bias テーブルのみを参照して分岐方向を予測することができる．Bias テーブルは 2 ビットあるいは 1 ビットのタグを持たない非常にシンプルなハードウェアであり，命令パイプラインの早い段階で分岐方向を予測することが可能である．

階層化された分岐予測機構において，極端な偏りのある分岐命令を早期に解決するフィルタの役割として提案方式を利用することで，命令フェッチの高効率化を達成できる可能性がある．

7. おわりに

命令発行幅の増大と命令パイプライン長の増大により，プロセッサ性能に与える分岐予測器の重要性が増している．本稿では，極端な偏りのある分岐命令に注目し，この特徴を利用した分岐予測の精度向上手法を提案し評価した．

☆ 文献 11) では，ライン・ウェイ予測という言葉を利用している．本稿においては，文献 12) のライン予測という言葉で統一した．

まず，極端な偏りのある分岐命令が存在することを示し，極端な偏りのある分岐命令を区別する方式を提案した．次に，極端な偏りのある分岐命令を区別する方式を Bimode 分岐予測器に組み込んだ Bimode-Plus 予測器を提案した．

SpecFP，SpecINT，マルチメディア，サーバの領域を含むベンチマークプログラムを用いた評価結果から，Bimode-Plus 分岐予測器の優れた予測精度を確認するとともに，以下の知見を得た．(1) 極端な偏りのある成立分岐命令のみを利用する Bimode-Plus (Taken)，極端な偏りのある不成立分岐命令のみを利用する Bimode-Plus (Untaken)，極端な偏りのある成立分岐命令と不成立分岐命令を利用する構成 Bimode-Plus (Taken & Untaken) の予測精度を比較し，Bimode-Plus (Untaken) が最も高い予測精度を達成する．(2) すべての条件分岐命令に関して分岐履歴レジスタを更新する BHR_ALL 方式と，極端な偏りを持たない条件分岐命令に関して分岐履歴レジスタを更新する BHR_NOB 方式を提案評価し，BHR_NOB 方式が高い予測精度を達成する．(3) 従来手法の Bimode 予測器との比較では，BHR_NOB 方式を利用する Bimode-Plus (Untaken) が，8 KB，16 KB，32 KB，64 KB のそれぞれのハードウェア量の構成において，6.2%，6.4%，7.4%，8.1%の予測ミスを削減する．

SpecFP，SpecINT，マルチメディア，サーバの領域を含むベンチマークプログラムを用いて，コンテキストスイッチが分岐予測精度に与える影響を検討した．これらの評価結果から，コンテキストスイッチが発生する実際のプロセッサの環境においても，Bimode-Plus 予測器は，Bimode 予測器に対して，6.1%から 8.5%の予測ミスを削減できることを明らかにした．

SPEC CINT2000 の 10 本のプログラムを利用して，分岐予測器の予測精度向上とプロセッサの処理性能向上の関係を検討した．保守的な構成の BASE モデルと，大規模な構成の WIDE モデルを定義し，それぞれのモデルにおける分岐予測器の影響を評価した．評価結果から，BASE モデルにおいては Bimode-Plus 予測器の利得が小さいが，WIDE モデルでは Bimode-Plus 予測器により平均で 3.2%，最大で 8.5%の速度向上を達成できることを明らかにした．

参考文献

1) KleinOsowski, A. and Lilja, D.: MinneSPEC: A New SPEC Benchmark Workload for Simulation-Based Computer Architecture Re-

search, *Computer Architecture Letters*, Vol.1 (2002).
2) Chang, P.-Y., Evers, M. and Patt, Y.N.: Improving Branch Prediction Accuracy by Reducing Pattern History Table Interference, *International Conference on Parallel Architectures and Compilation Techniques*, p.48 (1996).
3) Burger, D. and Austin, T.M.: The Simplescalar Tool Set, Version 2.0, Technical Report CS-TR-1997-1342, University of Wisconsin-Madison (1997).
4) Eden, A.N. and Mudge, T.: The YAGS Branch Prediction Scheme, *Proc. 31st Annual ACM/IEEE International Symposium on Microarchitecture*, pp.69–77 (1998).
5) Evers, M., Chang, P.-Y. and Patt, Y.N.: Using Hybrid Branch Predictors to Improve Branch Prediction Accuracy in the Presence of Context Switches, *Proc. 23rd Annual International Symposium on Computer Architecture*, pp.3–11, ACM Press (1996).
6) Galluzzi, M., Puente, V., Cristal, A., Beivide, R., Gregorio, J.-A. and Valero, M.: A First Glance at Kilo-Instruction Based Multiprocessors, *Proc. 1st Conference on Computing Frontiers*, pp.212–221, ACM Press (2004).
7) Jimenez, D.A.: Fast Path-based Neural Branch Prediction, *Proc. 36th Annual IEEE/ACM International Symposium on Microarchitecture*, pp.243–252 (2003).
8) Jimenez, D.A. and Lin, C.: Dynamic Branch Prediction with Perceptrons, *Proc. 7th International Symposium on High-Performance Computer Architectuer*, pp.197–206 (2001).
9) Lee, C.-C., Chen, I.-C.K. and Mudge, T.N.: The Bi-Mode Branch Predictor, *Proc. 30th Annual ACM/IEEE International Symposium on Microarchitecture*, pp.4–13 (1997).
10) McFarling, S.: Combining Branch Predictors, Technical report, Digital Equipment Corporation WRL TN-36 (1993).
11) Kessler, R.E.: The Alpha 21264 Microprocessor, *IEEE Micro*, Vol.19, No.2, pp.25–36 (1999).
12) Seznec, A., Felix, S., Krishnan, V. and Sazeides, Y.: Design Tradeoffs for the Alpha EV8 Conditional Branch Predictor, *Proc. 29th Annual International Symposium on Computer Architecture*, pp.295–306 (2002).
13) Seznec, A. and Michaus, P.: De-Aliased Hybrid Branch Predictors, Technical report, INRIA RR-3618 (1999).
14) Sprangle, E., Chappell, R.S., Alsup, M. and Patt, Y.N.: The Agree Predictor: A Mechanism for Reducing Negative Branch History Interference, *Proc. 24th International Symposium on Computer Architecture*, pp.284–291 (1997).
15) The Journal of Instruction-Level Parallelism: Championship branch prediction. http://www.jilp.org/cbp
16) Yeh, T.-Y. and Patt, Y.N.: Two-level adaptive training branch prediction, *Proc. 24th Annual International Symposium on Microarchitecture*, pp.51–61 (1991).
17) 吉瀬謙二：極端な偏りに基づく分岐予測器，それを組み込んだプロセッサ及びハードウェア予測器，特願 2003-271395 (2003).
18) 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，電子情報通信学会技術研究報告，CPSY-2003-10，pp.25–30 (2003).
19) 斉藤史子，山名早人：投機的実行に関する最新技術動向，情報処理学会研究報告，2001-ARC-145, pp.67–72 (2001).
20) 斉藤史子，山名早人：BTB のエントリ有無を参照した分岐予測器，情報処理学会論文誌：コンピューティングシステム，Vol.45, No.SIG11(ACS 7), pp.71–79 (2004).
21) 中西知嘉子，安藤秀樹，原　哲也，中屋雅夫：高い命令供給速度を実現するスーパスカラ・マシン向け命令フェッチ機構，並列処理シンポジウム JSPP'97 論文集，pp.213–220 (1997).
22) 野口良太, 森　敦司, 小林良太郎, 安藤秀樹, 島田俊夫：分岐方向の偏りを利用し破壊的競合を低減する分岐予測方式，情報処理学会論文誌，Vol.40, No.5, pp.2119–2131 (1999).
23) 斉藤史子，北村建志，山名早人：ハイブリッド分岐方向予測機構の性能比較，情報処理学会研究報告，2002-ARC-150, pp.89–94 (2002).

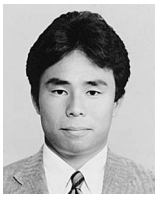
(平成 16 年 10 月 1 日受付)
(平成 17 年 1 月 26 日採録)

吉瀬　謙二（正会員）

1995 年名古屋大学工学部電子工学科卒業．2000 年東京大学大学院情報工学専攻博士課程修了．工学博士．同年電気通信大学大学院情報システム学研究科助手．計算機アーキテクチャ，並列処理に関する研究に従事．電子情報通信学会，IEEE-CS，ACM 等各会員．

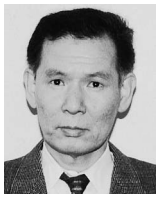
片桐　孝洋（正会員）

1973年生．電気通信大学情報システム学研究科助手．1994年豊田工業高等専門学校情報工学科卒業．1996年京都大学工学部情報工学科卒業．2001年東京大学大学院理学系研究科情報科学専攻博士課程修了．博士（理学）．2001年4月日本学術振興会特別研究員PD，2001年12月科学技術振興事業団研究者を経て，2002年6月より現職．並列計算機を用いた効率の良い行列計算アルゴリズムの研究，およびソフトウェア自動チューニングの研究に従事．2002年山下記念研究賞受賞．日本ソフトウェア科学会，日本応用数理学会，日本計算工学会，日本計算機統計学会，ACM，IEEE，SIAM等各会員．



本多　弘樹（正会員）

1984年早稲田大学理工学部電気工学科卒業．1991年同大学大学院理工学研究科博士課程修了．1987年より同大学情報科学研究教育センター助手．1991年より山梨大学工学部電子情報工学科専任講師．1992年より同助教授．1997年より電気通信大学大学院情報システム学研究科助教授．並列処理方式，並列化コンパイラ，並列計算機アーキテクチャ，グリッド等の研究に従事．工学博士．電子情報通信学会，IEEE-CS，ACM各会員．平成15年度山下記念研究賞受賞．



弓場　敏嗣（フェロー）

1966年神戸大学大学院工学研究科修士課程修了．（株）野村総合研究所を経て，1967年通商産業省工業技術院電子技術総合研究所（現在，独立行政法人産業技術総合研究所）に入所．以来，計算機のオペレーティングシステム，見出し探索アルゴリズム，データベースマシン，データ駆動型並列計算機等の研究開発に従事．その間，計算機方式研究室長，知能システム部長，情報アーキテクチャ部長等を歴任．1993年より，電気通信大学大学院情報システム学研究科教授．並列処理・分散処理の科学技術一般に興味を持つ．工学博士．情報処理学会，電子情報通信学会，各フェロー．日本ソフトウェア科学会，日本ロボット学会，ACM，IEEE各会員．