社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

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[†]

† 電気通信大学 大学院情報システム学研究科　〒 182–8585 東京都調布市調布ヶ丘 1-5-1
†† 科学技術振興事業団, 戦略的創造研究推進事業, 「情報基盤と利用環境」領域
E-mail: †{kis,katagiri,honda,yuba}@is.uec.ac.jp

あらまし　命令発行幅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．SPEC CINT95 を用いた評価の結果から，Bimode-Plus 分岐予測器の優れた予測精度を確認した．

キーワード　Bimode-Plus 分岐予測器，高性能プロセッサ，Gshare 分岐予測器，Bimode 分岐予測器

Bimode-Plus Branch Predictor

Kenji KISE[†,††], Takahiro KATAGIRI[†,††], Hiroki HONDA[†], and Toshitsugu YUBA[†]

† Graduate School of Information Systems, The University of Electro-Communications
1–5–1 Chofugaoka Chofu-shi, Tokyo, 182–8585 Japan
†† “Information Infrastructure and Applications”, PRESTO, JST
E-mail: †{kis,katagiri,honda,yuba}@is.uec.ac.jp

Abstract　Accurate branch prediction has come to play an important role for modern microprocessors. In order to improve 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 SPEC CINT95 show that Bimode-Plus branch predictor gives better prediction accuracy than Bimode branch predictor.

Key words　Bimode-Plus branch predictor, high performance processor, Gshare, Bimode

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上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．

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．SPEC CINT95 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．

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．典型的な分岐予測器の分類を図 1 に示す．図 1 では，分岐予測器の名前と発表された学会名（あるいは技術報告の名前）および年をラベルとして利用し，最近提案された予測器を右側に配置した．また，関連の深い分岐予測器を矢印で結んだ．分岐予測の詳細な調査は幾つかの文献 [2], [11], [12] に詳しい．本節では，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幾つかの分岐予測器を概観する．

McFarling ら [4] により提案された Gshare 予測器は，シン

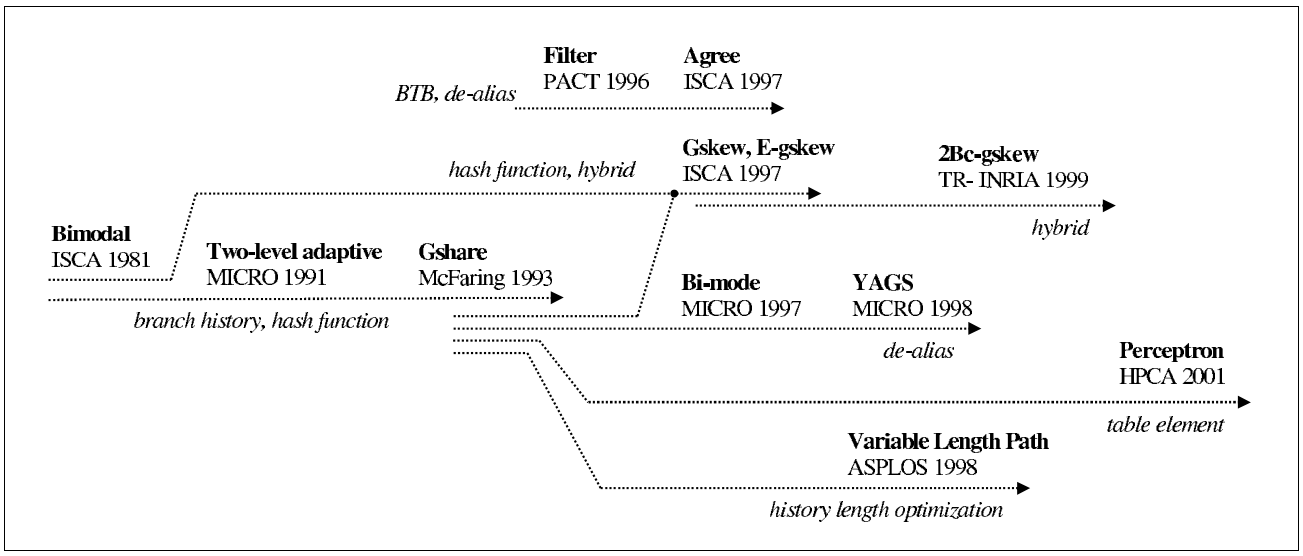

図 1　典型的な分岐予測器の分類．予測器の名前と会議の名前をラベルとして利用し，新しく提案された分岐予測器を右側に配置した．

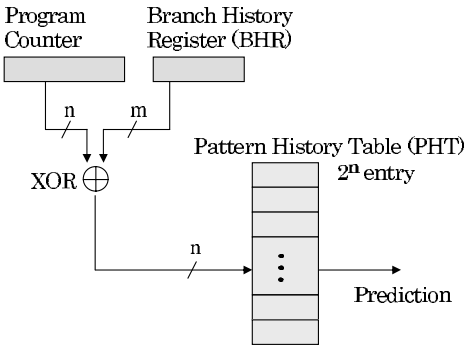

図 2　Gshare 予測器のブロック図

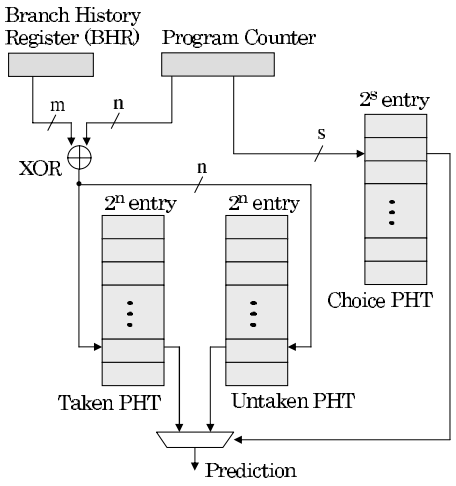

図 3　Bimode 予測器のブロック図

プルな構成で高い予測精度を達成するという特徴から，幾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，比較対象の一つとして Gshare 予測器を利用する．Gshare 予測器の構成を図 2 に示す．これは，2 レベル適応型 [8] の分岐予測を拡張したもので，グローバル分岐履歴レジスタ (BHR) と分岐アドレス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 予測器の破壊的競合を緩和するために，Lee ら [3] により提案された Bimode 予測器の構成を図 3 に示す．同様の構成が Sgshare 予測器として野口ら [13] 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によりインデックスを生成する 2 ビットカウンタの配列であり，Bimodal 予測器と同じ構成を持つ．グローバル分岐履歴レジスタ (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おこなう．

Sprangle ら [7] 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 4 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同意するかという判断で予測をおこなう．

Chen ら [1] 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．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

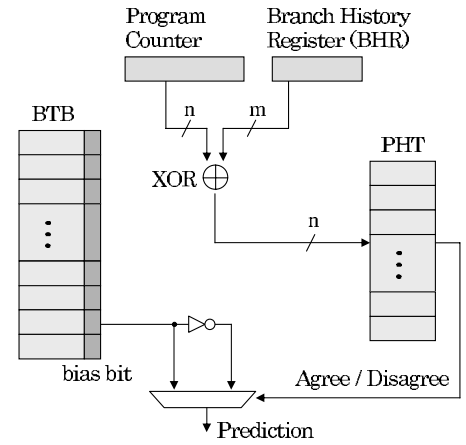

図 4　Agree 予測器のブロック図

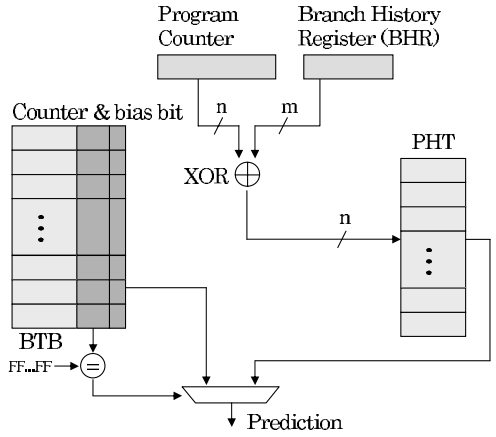

図 5　Filter 機構のブロック図

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，当該命令を偏った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おこなったところ，プログラムの開始から終了までの全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する分岐命令が多数存在することが判明した．これら，全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．

極端な偏りのある分岐命令の実行頻度を測定した．SPEC CINT95 の結果を表 1 にまとめる．全ての条件分岐命令の実行回数の中で，全て分岐成立する分岐命令の実行頻度を表の 4 列目 (all-taken) に示す．同様に，全て不成立する分岐命令の実行頻度を表の 5 列目 (all-untaken) に示す．4 列目 (all-taken) の値には反映されていないが，Alpha AXP アーキテクチャにおける無条件分岐の BR(Unconditional Branch) 命令と BSR(Branch to Subroutine) 命令も，極端な偏りのある分岐として分類される．このため，明らかに極端な偏りのある命令と分類される BR 命令と BSR 命令を加えると，all-taken の割合は更に増加する．

表 1　ベンチマークプログラム SPEC CINT95 の特性

Program	総実行命令	分岐	all-taken	all-untaken
099.go	138 million	12.4%	3.1 %	6.2 %
124.m88ksim	127 million	10.2%	19.3 %	13.6 %
126.gcc	150 million	15.5%	10.7 %	18.8 %
129.compress	142 million	9.3%	3.7 %	8.2 %
130.li	208 million	15.8%	11.9 %	18.3 %
132.ijpeg	172 million	6.1%	6.1 %	7.3 %
134.perl	153 million	14.4%	25.2 %	19.9 %
147.vortex	184 million	12.3%	20.6 %	55.0 %

表 1 の結果から，vortex においては，極端な偏りのある分岐命令の実行頻度が 75%に達することがわかる．また，多くのプログラムにおいて数割の頻度で極端な偏りのある分岐命令が存在することを確認できる．これら，極端な偏りのある分岐命令の特徴を利用する方式と，これを利用して予測精度の向上を目指す新しい分岐予測器として Bimode-Plus 予測器を提案する．

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，2 つのフラグを用いて極端な偏りのある分岐命令を区別する方式を提案する．

- **All-taken**: 過去の実行結果が全て成立だった分岐
- **All-untaken**: 過去の実行結果が全て不成立だった分岐
- **Both**: 成立と不成立の両方が起こった分岐

プログラムの実行中に，個々の分岐命令を上の 3 つに分類する．このために，分岐命令毎に 2 ビットのフラグを用意する．これらのフラグを taken_flag と untaken_flag と呼ぶことにする．

表 2　極端な偏りのある分岐を区別するための 2 つのフラグ

	初期値	**All-taken**	**All-untaken**	**Both**
taken_flag	0	1	0	1
untaken_flag	0	0	1	1

フラグの値と分類の関係を表 2 にまとめる．初期値として，taken_flag と untaken_flag をリセットする．分岐結果が確定した時点で，分岐が成立した場合には taken_flag をセットする．分岐が不成立だった場合には untaken_flag をセットする．

分岐方向を予測する際に，taken_flag のみがセットされている命令は，過去の全ての分岐結果が成立だった極端な偏りのある分岐命令 (**All-taken**) として，分岐成立と予測する．untaken_flag のみがセットされている命令は，過去の全てが不成立だった極端な偏りのある分岐命令 (**All-untaken**) として，分岐不成立と予測する．

taken_flag と untaken_flag の両方がセットされている場合には，極端な偏りのない分岐命令 (**Both**) である．これらは，何らかの手法により分岐方向を予測する．taken_flag と untaken_flag がリセットされている場合は，一度も実行されたことのない分岐命令である．これらは，過去の履歴が利用できないので，通常は分岐成立と予測する．

本節で提案した方式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全て成立だった分岐命令の実行頻度と，全て不成立だった分岐命令の実行頻度である．提案した方式では，予測する時点までの分岐結果が全て成立だったもの，あるいは，予測する時点までの分岐結果が全て不成立だったものを極端な偏りのある分岐命令と判断する．このため，表 1 に示した割合と比較して，提案した方式で分類される極端な偏りのある命令の割合の方が高くなる．

提案した方式は，Agree 予測器 [7] 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，ある分岐命令の過去の全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 [1] と共通点がある．Filter 機構では，カウンタを用いて一定の履歴において偏りのある命令を予測する．提案方式と方針は近いが，カウンタを用いるという点が大きく異なる．Filter 機構ではカウンタが飽和するまでの学習期間で正しい予測を生成することが困難となる．また，カウンタの閾値を適切に設定する必要がある．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，極端な偏りのある分岐を対象とするために予測精度が高い．提案手法は BTB に変更を加える必要はない．

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 予測器と名付ける．

極端な偏りのある分岐命令を区別する単純な方法は，bimode 予測器の Choice PHT に taken_flag と untaken_flag という 2 つのフラグを追加することである．この方法は単純であるが，Choice PHT のエントリ毎に 2 ビットのハードウェアを追加する必要がある．より洗練された方法として，Choice PHT を構成する 2 ビットカウンタを taken_flag と untaken_flag として共有することで，ハードウェアの増加を 1 ビットに抑える方式を提案する．Bimode-Plus 予測器はこの方式を採用する．

極端な偏りのある分岐命令の場合には，Choice PHT の 2 ビットを taken_flag と untaken_flag として利用する．このように 2 ビットを利用している状態を Plus モードと呼ぶことにする．Plus モード以外は，Choice PHT の 2 ビットは 2 ビット飽和型カウンタとして利用される．こちらの状態を 2BC モードと呼ぶことにする．Plus モードと 2BC モードの 2 つを区別するために，1 ビットのフラグを用意する．このフラグがセットされた時に，2 ビットカウンタが有効になるという意味で有効ビット (valid bit) と呼ぶことにする．有効ビットを追加した 2 ビットカウンタの状態遷移を図 6 に示す．各状態の括弧内に，2 ビットの状態を 2 進数で記述した．左に示す 4 つの状態が通常の 2 ビット飽和型カウンタであり，右に示す 3 つの状態が追加された状態 (Plus モード) である．初期状態は右端の有効ビットを含む 3 ビットが全て 0 の状態である．分岐結果 taken が続く場合には，All Taken の状態に留まる．一方，分岐結果 untaken が続く場合には，All Untaken の状態に留まる．分岐

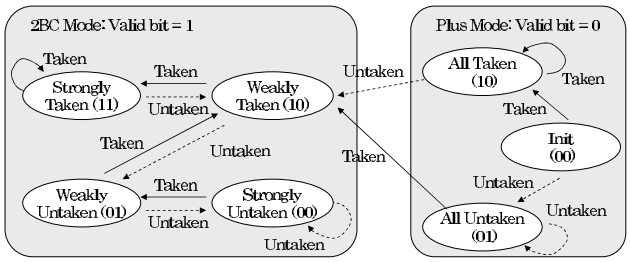

図 6　2 ビットカウンタプラスの状態遷移

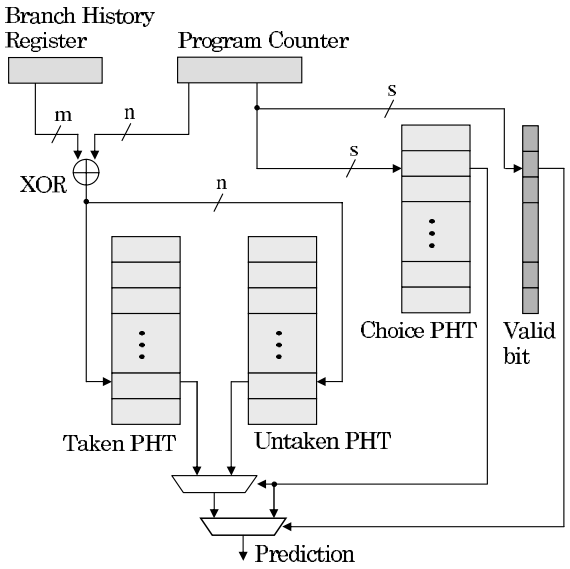

図 7　Bimode-Plus 予測器のブロック図

成立と不成立の両方が実行された場合に Plus モードから 2BC モードに移行する．図 6 に示した構造を 2 ビットカウンタプラスと呼ぶことにする．

Bimode 予測器に有効ビットを追加した Bimode-Plus 予測器の構成を図 7 に示す．有効ビットが 0 で Plus モードの時は Choice PHT の値から予測方向を得る．そうでない場合には，Bimode 予測器と同様に Direction PHT の出力から予測値を得る．有効ビットが 0 で Plus モードの時には，Direction PHT の値は更新しない．有効ビットが 1 で 2BC モードの時には Bimode 予測器と同様に Partial Update 戦略に基づいて Direction PHT の値を更新する．分岐の結果が得られた時点で，図 6 の状態遷移に従って Choice PHT のカウンタと有効ビットを更新する．

図 3 に示す Bimode 予測器のハードウェア量は，Direction PHT のエントリ数を 2^n, Choice PHT のエントリ数を 2^s として，次式で計算できる．

$$2\ bit \times 2^n\ entry \times 2 + 2\ bit \times 2^s\ entry \qquad (1)$$

一方，図 7 に示した Bimode-Plus 予測器のハードウェア量は次式で計算できる．

$$2\ bit \times 2^n\ entry \times 2 + 3\ bit \times 2^s\ entry \qquad (2)$$

評価パラメータを削減するために，Direction PHT と Choice PHT のエントリ数を等しいとする ($s = n$)．この場合に先の 2

つの式は下の様に簡略化される．

$$6\ bit \times 2^n\ entry \tag{3}$$

$$7\ bit \times 2^n\ entry \tag{4}$$

これらの式から明らかなように，Direction PHT と Choice PHT のエントリ数を等しいとする時，Bimode 予測器の 7/6 倍 (117%) のハードウェア量で Bimode-Plus 予測器を構築できる．

3.4 Bimode 予測器以外の予測器への組み込み

前節では，極端な偏りのある分岐命令を区別する方式と，それを用いた Bimode-Plus 予測器を提案した．自然な構成として，極端な偏りのある分岐命令を区別する方式を他の分岐予測器に組み込むことを考える．

YAGS 予測器 [2] においては，Bimode の場合と同様に Choich PHT の要素を 2 ビットカウンタから，図 6 に示した 2 ビットカウンタプラスの構造に変更すればよい．この構成を YAGS-Plus 予測器と定義する．

E-gskew 予測器 [5] や 2Bc-gskew [6] 予測器においては，Bimodal 予測の部分の要素を 2 ビットカウンタから，図 6 に示した 2 ビットカウンタプラスの構造に変更すればよい．これらを E-gskew-Plus 予測器，2Bc-gskew-Plus 予測器と定義する．

4. 評　　価

4.1 評 価 方 法

機能レベルのシミュレータ SimAlpha [10] に 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．文献 [9] に Gshare 予測器と Bimode 予測器の実装の詳細を記述した．SimAlpha は，ライブラリを静的にリンクしたアプリケーションのコードを実行する．オペレーティングシステムのコードが評価に含まれていない点に注意する必要がある．また，YAGS 予測器 [2] の評価でおこなわれているようなコンテキストスイッチの影響は考慮していない．すなわち，一つのアプリケーションプログラムの評価が全て終了した後に，次のプログラムの評価を開始する．

シミュレーションの命令数を抑えるように入力パラメタを調整する．SPEC CINT95 のバイナリは DEC C コンパイラ，最適化オプション O4 を用いて生成する．

各ベンチマークの実行命令数を表 1 の 2 列目にまとめる．各ベンチマークの分岐命令の実行頻度を 3 列目にまとめる．Alpha AXP アーキテクチャにおける BR(Unconditional Branch) 命令，BSR(Branch to Subroutine) 命令を含む分岐命令を評価対象とする．ジャンプ命令は対象外である．

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し，Plus モードで予測をおこなった頻度を測定した．図 8 に評価結果を示す．横軸は Bimode-Plus 予測器のハードウェア量であり，縦軸は Bimode-Plus 予測器が極端な偏りのある分岐命令と判断した頻度を表している．図 8 にまとめた Plus モードで予測をおこなう頻度は，Choice PHT のエントリ数（パラメータ s) に依存し，分岐履歴レジスタ数のビット数 (パラメータ m) と Direction PHT のエントリ数 (パラメータ n) には依存しない．

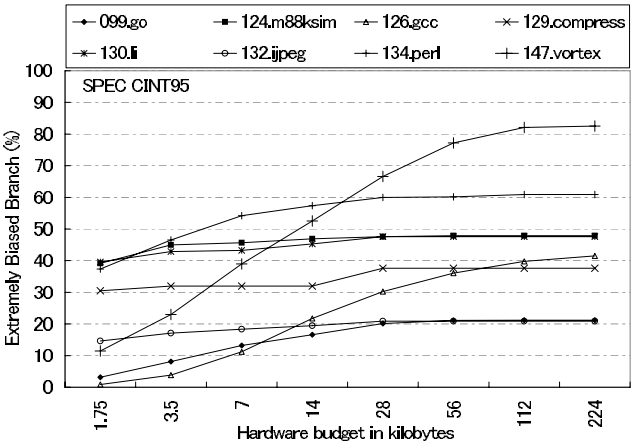

図 8　SPEC CINT95 を用いた極端な偏りのある分岐の分類結果

図 8 の結果から，ハードウェア量の増加に伴い，極端な偏りのある分岐命令と判定される割合が増加することがわかる．静的な分岐命令の数が 15550 命令と極端に多い 124.gcc と，6310 命令と 2 番目に多い 147.vortex を除いて，ハードウェア量を 28KB 付近まで大きくすることで極端な偏りのある分岐命令の割合が飽和する．ハードウェア量が 28KB のデータを見ると，極端な偏りのある分岐命令と判定される割合は最大が 147.vortex の 66%，最低が 099.go の 20%であり，8 本中の 4 本のプログラムで 40%を超えることがわかる．

4.3 予 測 精 度

Bimode-Plus 予測器と Bimode 予測器には，Direction PHT のエントリ数を指定するパラメータ n, Choice PHT のエントリ数を指定するパラメータ s, 分岐履歴レジスタのビット長 m という 3 つのパラメータが存在する．評価パラメータの数を削減するために Direction PHT と Choice PHT のエントリ数を等しいとする ($s = n$)．Bimode-Plus 予測器，Bimode 予測器，Gshare 予測器の 3 つの予測精度は，分岐履歴レジスタのビット長 m に依存する．取りうる可能性のある分岐履歴レジスタの全てのビット長の予測精度を測定し，その中で最も高い性能を示した構成を用いて比較する．これらの構成を，分岐履歴レジスタのビット長が最適という意味で，Bimode-Plus.best, Bimode.best, Gshare.best と記述する．

Bimode-Plus.best, Bimode.best, Gshare.best の予測精度を図 9 にまとめる．縦軸は予測ミス率であり下に行くほど性能が高い．横軸は対数で示したハードウェア量である．Bimode 予測器と Bimode-Plus 予測器の予測精度は 5KB 付近で交差する．ハードウェア量が 5KB 以下の構成では Bimode 予測器の性能が最も高く，ハードウェア量が 5KB を超えたところでは Bimode-Plus 予測器の予測精度が最も高い．半導体技術の進歩により，分岐予測器のために利用できるハードウェア量が増加する傾向にある．高性能プロセッサのための分岐予測器を考えた場合に，数十 KB のハードウェア量の構成が現実的な選択肢として重要となる．このような 10KB を超える重要な領域にお

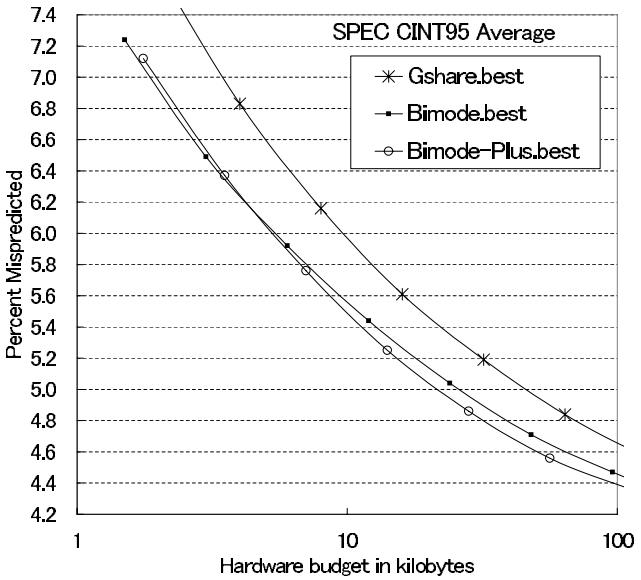

図 9　SPEC CINT95 を用いた Bimode-Plus 予測器の予測精度

いて，Bimode-Plus 予測器の優位性が明確になり，この時，約 0.1%の予測精度の向上を確認できる．

図 9 では，比較結果を明確にするために，得られたデータ間を線で結んでいる．しかしながら，Bimode 予測器では 12KB, 24KB, 48KB といったハードウェア量の構成が，Bimode-Plus 予測器では 14KB, 28KB, 56KB といった離散的なハードウェア量の構成のみが利用可能である点に注意する必要がある．

幾つかの構成例を用いて予測器を比較する．

ハードウェア量 12KB の Bimode 予測器のミス率 5.44%に対して，ハードウェア量 14KB の Bimode-Plus 予測器のミス率が 5.25%である．この時，2KB のハードウェア量の増加で，0.19%の予測ミス率を改善できる．

ハードウェア量 24KB の Bimode 予測器のミス率 5.04%に対して，ハードウェア量 28KB の Bimode-Plus 予測器のミス率が 4.86%である．この時，4KB のハードウェア量の増加で，0.18%の予測ミス率を改善できる．

4.4 議　　論

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 予測器に有効ビットと幾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) が増加することは無いと考えられる．

5. 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評価した．

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，提案した方式を既存の分岐予測器に組み込むための構造として 2 ビットカウンタプラスを提案し，これを用いることでハードウェア量の増加を抑えることができることを述べた．新しい分岐予測器として，2 ビットカウンタプラスを Bimode 分岐予測器に組み込んだ Bimode-Plus 予測器を提案した．

SPEC CINT95 を用いた評価の結果から，Bimode-Plus 予測器の優れた予測精度を確認した．SPEC CINT95 の平均では，ハードウェア量 24KB の Bimode 予測器のミス率 5.04%に対して，ハードウェア量 28KB の Bimode-Plus 予測器のミス率が 4.86%となることを示した．この時，4KB のハードウェア量の増加により，0.18%の予測ミス率を改善する．また，ハードウェア量が 10KB を超える領域において，Bimode-Plus 予測器が最も高い性能を示すことを確認した．

SimAlpha Version 1.0 と、Bimode-Plus 予測器の性能を評価するために変更を加えたソースコードは次の URL からダウンロードできる。

`http://www.yuba.is.uec.ac.jp/~kis/SimAlpha/`

文　　献

[1] Po-Yung Chang, Marius Evers, and Yale N. Patt. Improving branch prediction accuracy by reducing pattern history table interference. In *International Conference on Parallel Architectures and Compilation Techniques*, 1996.

[2] A. N. Eden and T. Mudge. The yags branch prediction scheme. In *Proceedings of the 31st annual ACM/IEEE international symposium on Microarchitecture*, pp. 69–77, 1998.

[3] Chih-Chieh Lee, I-Cheng K. Chen, and Trevor N. Mudge. The bi-mode branch predictor. In *Proceedings of the 30th annual ACM/IEEE international symposium on Microarchitecture*, pp. 4–13, 1997.

[4] S. McFarling. Combining Branch Predictors. Technical report, WRL Technical Note TN-36, Digital Equipment Corporation, 1993.

[5] Pierre Michaud, Andre Seznec, and Richard Uhlig. Trading conflict and capacity aliasing in conditional branch predictors. In *Proceedings of the 24th international symposium on Computer architecture*, pp. 292–303, 1997.

[6] Andre Seznec and Pierre Michaus. De-aliased Hybrid Branch Predictors. Technical report, INRIA RR-3618, February 1999.

[7] Eric Sprangle, Robert S. Chappell, Mitch Alsup, and Yale N. Patt. The agree predictor: a mechanism for reducing negative branch history interference. In *Proceedings of the 24th international symposium on Computer architecture*, pp. 284–291, 1997.

[8] Tse-Yu Yeh and Yale N. Patt. Two-level adaptive training branch prediction. In *Proceedings of the 24th annual international symposium on Microarchitecture*, pp. 51–61, 1991.

[9] 吉瀬謙二, 片桐孝洋, 本多弘樹, 弓場敏嗣. 高性能プロセッサのための代表的な分岐予測器の実装と評価. Technical Report UEC-IS-2003-2, 電気通信大学　大学院情報システム学研究科, May 2003.

[10] 吉瀬謙二, 本多弘樹, 弓場敏嗣. Simalpha: C++で記述したもうひとつの alpha プロセッサシミュレータ. 情報処理学会研究報告　2002-ARC-149, pp. 163–168, August 2002.

[11] 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, November 2001.

[12] 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, November 2002.

[13] 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, May 1999.