値予測を利用した分岐予測機構

片山清和[†] 戸田聡[†,☆] 中村幸司[†,☆☆]
布施裕基[†,☆☆☆] 安藤秀樹[†] 島田俊夫[†]

命令の実行結果を予測する値予測は，将来のスーパスカラ・プロセッサには必須の技術となると思われる．本論文では，値予測を分岐予測に利用する機構を提案する．これにより，値予測はデータ依存の緩和だけでなく，制御依存の緩和にも利用できる．本機構は，分岐命令のオペランドとオペコードを予測し，分岐方向を予測するものである．本機構は従来の分岐予測機構とのハイブリッドで用い，従来の分岐予測がとらえている相関に加え，値予測がとらえることができる相関を利用して予測精度を改善する．SPECint95 ベンチマークを用いた評価の結果，4 K バイトの gshare に対し，最大 2.55%ポイント，平均 0.84%ポイント予測精度を改善できることを確認した．また，この予測精度の改善により，今日のスーパスカラ・プロセッサの性能は最大 5.95%，平均 2.43%向上し，より広い命令発行幅とより深いパイプラインを持つ将来のスーパスカラ・プロセッサでは最大 14.14%，平均 6.09%の速度向上が得られることを確認した．

A Branch Prediction Mechanism via Value Prediction

Kiyokazu Katayama,[†] Satoru Toda,[†,☆] Kouji Nakamura,[†,☆☆]
Yuki Fuse,[†,☆☆☆] Hideki Ando[†] and Toshio Shimada[†]

Value prediction that predicts the results of instructions will be an essential technique for the future superscalar processors. This paper proposes a branch prediction mechanism that uses value prediction. This approach allows value prediction to be used to remove control dependences as well as data dependences. Our mechanism predicts operand values and opcode of a branch, and then predicts its outcome. Our mechanism is incorporated in a hybrid predictor with conventional branch predictors. This organization improves prediction accuracy by exploiting the correlations value prediction can capture in addition to the correlations conventional branch prediction can capture. Our evaluation results show that our hybrid branch predictor improves prediction accuracy by a maximum amount of 2.55% point or an average of 0.84% point over 4 K-byte gshare on SPECint95 benchmarks. This improvement of the prediction accuracy contributes to the performance of a current superscalar processor by a maximum of 5.95% or by an average 2.43%, and that of a future superscalar processor with a wider instruction issue and a deeper pipeline by a maximum of 14.14% or by an average 6.09%.

1. はじめに

近年の高性能プロセッサは深いパイプラインやスーパスカラに代表される複数命令の同時発行機構により命令レベル並列性を引き出し，性能を向上させている．分岐予測精度は，こうした命令レベル並列性を利用するプロセッサの性能に大きな影響を与える．今後も命令発行幅はより広く，パイプラインはより深くなる傾向にあり，分岐予測精度に対する要求はさらに強まっていく[1]．

これまでに提案された分岐予測方式の多くは，過去の分岐の振舞いと将来の振舞いとの間に相関があることを利用している．なかでも **2 レベル分岐予測機構**[2)~4)]は，過去の分岐の履歴パターンと将来の分岐方向の間に存在する相関を利用し，高い予測精度を達成している．そのため，最近の多くの高性能プロセッサに採用されている（たとえば，Compaq Alpha 21264[5)]，AMD Athlon[6)]）．

分岐予測は，制御依存を緩和し，命令レベル並列性を増加させ，性能の向上を図る技術である．一方，データ依存を緩和し，命令レベル並列性を増加させる技術

† 名古屋大学大学院工学研究科
Graduate School of Engineering, Nagoya University
☆ 現在，株式会社メルコ
Presently with MELCO INC.
☆☆ 現在，沖電気工業株式会社
Presently with Oki Electric Industry Co., Ltd.
☆☆☆ 現在，日本電気株式会社
Presently with NEC Co.

が近年さかんに研究されている．この技術の 1 つに，命令の実行結果を予測する**値予測**がある[7)~11)]．これまでに提案されたほとんどの値予測は，分岐予測と同様に，過去の値の振舞いと将来の値との間に相関があることを利用するものであり，測定の結果，20%〜80%もの値を正しく予測できている[11)]．現在のプロセッサでは値予測を用いることによる性能向上は小さいものの[9)]，命令のフェッチバンド幅や発行バンド幅が拡大する将来のプロセッサでは，約 50%もの大きな性能向上が期待できる[12)]．このことは，値予測が将来のプロセッサには必須の技術として搭載されることを示唆している．

値予測はこれまで，データ依存を緩和することによって命令レベル並列性を増加させることを目的として研究されてきた．これに対して本論文では，値予測を分岐予測に利用する機構を提案する[13),14)]．この機構は，従来の分岐予測器と値予測を利用した分岐予測器とを組み合わせて用いる．これにより，従来の分岐予測では予測が困難であった分岐の予測を，値予測を利用した分岐予測器に割り当て，最終的な分岐予測精度を改善する．

本論文の構成は以下のとおりである．2 章で関連研究についてまとめる．3 章で値予測を用いた分岐予測機構について説明する．4 章で予測精度の評価を行う．5 章で本機構によりプロセッサ性能がどの程度向上するかについて評価を行い，6 章で本論文をまとめる．

2. 関 連 研 究

ほとんどの分岐予測方式は，過去の分岐の振舞いと将来の振舞いの間に相関があることを利用するものである．初期の方式は，分岐命令アドレスをインデクスとする表を用いて過去の振舞いを記録し，予測に利用するものである．この表の各エントリは 2 ビットの飽和カウンタからなり，対応する分岐が成立すれば 1 増加させ，不成立ならば 1 減少させ，最近の分岐方向の偏りを測定する．予測においては，命令アドレスで表を参照し，2 以上なら成立と予測し，1 以下なら不成立と予測する．この方式を **2 ビット・カウンタ方式 (2bc)**[15)]と呼ぶ．

より高精度な方式として，**2 レベル分岐予測方式**がある[2)~4)]．この方式は，分岐の履歴パターンと将来の分岐方向との間の相関を利用するものである．分岐命令アドレスと分岐履歴の組に対応するエントリを持つ表を用意する．この表を**パターン履歴表**（PHT: Pattern History Table）と呼ぶ．各エントリは 2 ビットの飽和カウンタからなり，分岐結果に応じて 2bc と同様にカウンタを増減させ，予測に用いる．

2 レベル分岐予測方式では，使用する履歴の種類や命令アドレスと履歴の組を PHT にどうのようにマッピングするかで，多くのバリエーションが考えられてきた．その中でも **gshare**[2)]と呼ばれる方式は，きわめて高精度な予測が可能な方式として知られている．gshare は，ある分岐の結果はその分岐以外の履歴と相関があることを利用する方式の 1 つである．履歴を記録するために 1 本のシフト・レジスタを用意し，分岐が実行されるたびに順に記録していく．PHT へのマッピング関数としては XOR を用いる．つまり，命令アドレスと分岐履歴の XOR をとった値を PHT へのインデクスとする．

性質の異なる 2 つの分岐予測器を組み合わせた予測器がある．これを**ハイブリッド 分岐予測器**[2),16),17)]と呼ぶ．どちらの予測結果を出力するかを決めるために，分岐命令アドレスをインデクスとする表を用意する．各エントリは通常 2 ビット飽和カウンタよりなる．対応する分岐に対し，最近 2 つの予測器のどちらが正しい予測をより多く行ったかをカウンタを増減させることにより記録する．正しい予測をより多く行った方の予測器の出力を予測とする．

これまで報告されている分岐予測器は，すでに 80%から 100%に近い予測精度を達成しているが，わずかな改善もプロセッサの性能に大きく貢献する[1)]ことから精度向上は今なお重要な課題である．

値予測は，命令の実行結果を予測するものである．正しく予測できれば，命令間のデータ依存関係を削除でき，並列性を増加させることができる．これまで多くの値予測器が提案されているが，ほとんどは過去の値の振舞いと将来の値の間に相関があることを利用するものである．

値予測の中で最も単純な手法は，同一の値が繰り返されると予測する**最終値予測**[8),9)]である．この予測手法では，命令アドレスをインデクスとする**値履歴表**（VHT: Value History Table）と呼ばれる表を用意する．各エントリには対応する命令の最近の実行結果を保持する．予測時には単純にこの値を読み出し，予測値とする．

一定の差分を持った値が繰り返されると予測する手法がある[7),11),18)]．この差分のことを**スト ライド**と呼び，この手法を**スト ライド 値予測**と呼ぶ．この予測手法においても，最終値予測と同様，命令アドレスをインデクスとする VHT を用意する．VHT の各エントリには，対応する命令の最近の実行結果，ストライド，および，ストライドを検出したかどうかの状態を保持

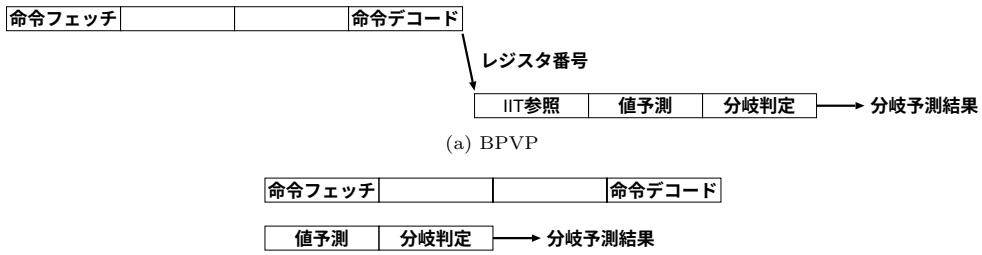


図 1 分岐予測のタイミングの比較
Fig. 1 Comparison with BPVP in timing of branch prediction.

する．予測においては，命令アドレスで VHT を参照し，ストライドを検出していた場合，読み出した最終値とストライドを加え予測値とする．

2 レベル分岐予測方式と同様の考え方で，過去の値の振舞いのパターンに対する相関を利用する方式としては，**2 レベル値予測方式**がある[11]．この方式では，第 1 レベルの表として VHT を用意し，各エントリには対応する命令の最近数回の実行結果と出現の履歴パターンを記録する．第 2 レベルの表として，履歴パターンをインデクスとする**パターン履歴表**（PHT: Pattern History Table）を用意する．各エントリには VHT の 1 つのエントリに格納する値の数だけのカウンタを用意し，対応する履歴パターンに対し VHT 内のどの値が次に現れるかの頻度を記録し，予測に用いる．同様の考えによる予測機構として，**コンテキスト・ベース値予測**と呼ばれる方式も提案されている[10]．

最初に述べたように，値予測は命令の実行結果を予測することによりデータ依存関係を削除し，プロセッサの性能を向上させるものである．しかし，これまで提案されている値予測はいずれも，非常に高い予測精度を示すというわけではない．予測に失敗すると，予測値を使用した命令およびその後続命令を再実行する必要があり，性能向上が制限される．予測失敗を防ぐために，予測結果が信頼できるかどうかの判断を行い，信頼できない場合は予測結果を使用せず，信頼できる場合のみ使用するという方法がよく採られる（たとえば文献 19)）．この方法は，予測対象の命令を制限してしまうが，予測失敗によるペナルティを被る頻度を下げることができる．予測の**信頼性**の計測法は種々あるが，**リセッティング・カウンタ**[20]により測定する方法が一般的である．このカウンタは，予測が成功した場合 1 増加させ，失敗した場合 0 にリセットするものである．カウンタの値があらかじめ定めた閾値以上になれば，予測は信頼できるとする．

Sazeides らは，分岐命令に対して値予測と分岐予測の相関を定量的に評価し，分岐予測を誤る分岐の半分以上で値予測が成功していることを示しており，値予測を用いることによる分岐予測精度向上の可能性を示唆している[21]．

González らは，我々と同時期に値予測を分岐予測に利用する BPVP という機構を提案している[22]．BPVP では，分岐命令の各レジスタに対応し，それらを生成する命令の PC を保持する表（IIT）を用意する．命令デコードの後にレジスタ番号で IIT を参照し，得られた PC によってさらに値予測器を参照し，分岐予測を行う．この方式では，分岐を予測するのは，命令デコード後にしかできないため，たとえ正しい予測結果が得られたとしても，深いパイプラインを持つ最近のプロセッサでは大きなペナルティを被る．これに対し，我々の方式はそのような大きなペナルティを被ることはない．**図 1** に BPVP と我々の方式での分岐予測のタイミングの違いを示す．例として命令フェッチからレジスタ番号が得られるまでに 4 サイクル要する場合を考える．BPVP では，レジスタ番号が得られた後，IIT 参照，値予測，分岐判定の 3 サイクル程度消費した後，分岐予測結果が得られる．これに対して，以後の章で説明するが，我々の方式では，命令フェッチと同時に値予測を開始し，次のサイクルで予測結果を得ることができる．この点で我々の方式は優れている．

3. 値予測による分岐予測

本章では最初に，値予測による分岐予測器を構成するにあたり検討すべき事項をあげ，議論する．次にこの議論に基づき予測器を提案する．

3.1 検 討 事 項

以下の 4 点について順に議論する．

- 分岐命令のソース・オペランド値の予測
- 分岐命令のオペコードの予測
- 値予測を用いた分岐予測器の適用方法
- ハイブリッド分岐予測器に必要な選択器

3.1.1 分岐命令のソース・オペランド値の予測

通常，値予測器は命令の実行結果であるデスティネーション・オペランドの値を予測する目的で使用される．この使用目的では，命令の実行結果が得られたときに，値予測器の表のその命令に対応するエントリを更新する．これに対して，値予測による分岐予測では，分岐命令のソース・オペランド値を予測する必要がある．これに対応するため，分岐命令がソース・レジスタを読み出したときに，エントリを更新するように修正する必要がある．なお，本方式はソース・オペランド値を予測するため，コンペア&ブランチ方式の命令セットに対して有効である．

3.1.2 分岐命令のオペコードの予測

分岐方向の決定には，ソース・オペランドに対して，比較・条件判定を行う必要がある．このためにはオペコードが必要である．一方，分岐予測は通常，パイプラインの最も早い段階，すなわち，命令フェッチ時に行われる．これは，できるだけ分岐によるペナルティを小さくするためである．予測のためには，命令のデコード以前にオペコードを得る必要があるので，オペコードも予測する必要がある．

オペコードの予測は，分岐先アドレスの予測と同様の方法により簡単に予測することができる．つまり，分岐命令アドレスをインデクスとする表を用意し，エントリにオペコードを格納するようにすればよい．

3.1.3 適用方法

値予測器を用いた分岐予測の精度は，値予測器の精度で抑えられると考えられる．一方，値予測器の予測精度は，現在実現されている分岐予測器の精度に比べるとかなり低い．したがって，値予測による分岐予測器を単体で使用し，従来の分岐予測器の精度を上回る精度を達成することは不可能である．

しかしこのことが，値予測による分岐予測が，分岐予測精度の改善に対し無力であるということを意味するわけではない．従来の分岐予測器と組み合わせるハイブリッド構成にすることにより予測精度改善が期待できる．これは以下の2つの理由による．第1の理由は，予測に使用する情報量を記憶する場所が，値予測を使うことにより増加することである．つまり，容量性の分岐予測ミス[23]を減少させることができる．値予測は将来のプロセッサでは，データ依存を緩和する目的で搭載されるので，分岐予測を目的に使用する際の余分なコストは小さいことに注意されたい．第2の理由は，値予測器が利用する相関と従来の分岐予測器が利用する相関とは異なるので，分岐予測のために新たに別の相関を利用することができるということである．

3.1.4 ハイブリッド分岐予測器の選択器

ハイブリッド分岐予測器では，どちらの予測結果を出力するかを決める選択器が必要である．通常，最近どちらの予測器が正しい予測をより多く行ったかによって選択する方法が採られる．本ハイブリッド予測器でもこの方法は有効であると考えられる．しかしその実現方法として，従来の2ビット飽和カウンタによる表が最適かどうかは疑問である．その理由の1つは，2つの分岐予測器の予測精度に大きな開きがあるということである．値予測による分岐予測の精度は従来の分岐予測器に比べて低いと考えられるので，選択のリスクを回避する慎重さが要求される．また，分岐予測と値予測が利用している相関の違いを利用することが重要なので，従来の分岐予測では予測困難な分岐であり，かつ，分岐命令のオペランド値の予測が確実な分岐を正確に選択する選択器が必要である．以上の要求を満たす選択器として，次の3つを提案する．これらは4章で評価する．

(1) **リセッティング・カウンタ方式**

選択器の表（以下，**選択表**と呼ぶ）の各エントリをリセッティング・カウンタとする．リセッティング・カウンタは，値予測による分岐予測器が正しい予測を行い，従来の分岐予測器が誤った予測を行ったとき，1増加させ，従来の分岐予測の正誤にかかわらず値予測による分岐予測が誤りであったとき，0にリセットする．その他の場合は変化させない．カウンタの値があらかじめ定めた閾値以上であれば，値予測による分岐予測の結果を選択する．そうでなければ，従来の分岐予測の結果を選択する．

この選択方法では，値予測を用いた分岐予測が選択されるためには，値予測を用いた分岐予測が連続して正しく予測し続ける必要がある．これにより値予測による分岐予測を使用するリスクを回避する．また，値予測を用いた分岐予測が最後に予測を誤ってから，次に誤る間に，従来の分岐予測器が閾値の回数だけ予測を誤る必要があるので，従来の分岐予測では予測困難な分岐であり，かつ，分岐命令のオペランド値の予測が確実な分岐を正確に選択することができる．

(2) **2ビット飽和カウンタと値予測器の信頼性を用いる方式**

選択表の各エントリは，従来のハイブリッド分岐予測器で一般に用いられている2ビット飽和カウンタとする．更新の方法も同じである．ただし，選択表により値予測を用いた分岐予測が選択された場合，使用した値予測の信頼性を確認する．信頼性があらかじめ定められた閾値よりも高いときのみ，最終的に値予測を用

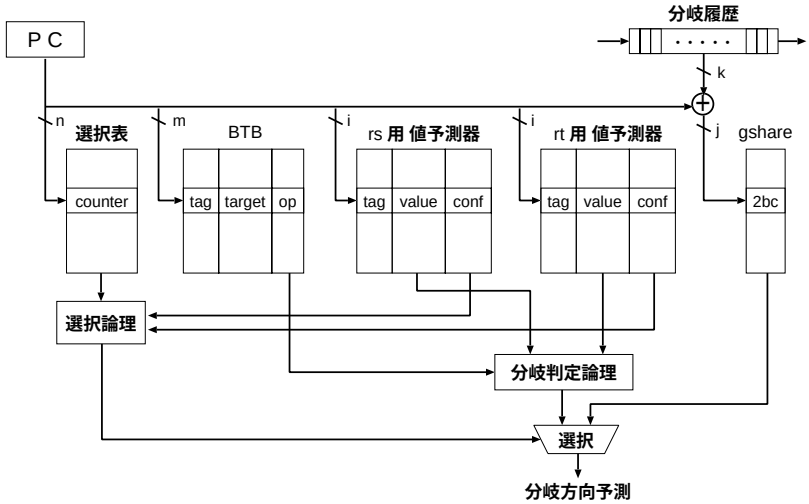


図 2 値予測を利用した分岐予測器と gshare 分岐予測器とのハイブリッド分岐予測器
Fig. 2 Hybrid branch predictor of the branch predictor with value prediction and the gshare branch predictor.

いた分岐予測の結果を選択する．信頼性が低い場合は，値予測を用いた分岐予測を選択せず，従来の分岐予測を選択する．なお，分岐命令のソース・オペランドは 1 つの場合と 2 つの場合があるが，2 つの場合は信頼性の低い方を判断に用いる．信頼性カウンタはリセッティング・カウンタを用いる．つまり，予測が成功した場合 1 増加させ，失敗した場合 0 にリセットする．この選択方法と (1) の方法の違いは，次の点である．(1) の方法では分岐予測の結果のみで選択を判断しており，値予測が正解かどうかは判断に含まれていない．不一致や大小比較を行う分岐では，値予測を誤っても，分岐予測は偶然正しい結果となる場合が多くあると思われる．このような場合を予測に利用することは利益もあるが，同時にリスクもある．(1) の方法はこのような場合が利用されるが，(2) の方法では利用されない．なぜなら，選択表によって値予測による分岐予測が選択されても，値予測の信頼性を見ることによって確認するからである．

(3) **リセッティング・カウンタと値予測の信頼性を用いる方式**

前述の (1) と (2) を組み合わせた方式で，両者の特徴を備えている．値予測による分岐予測を慎重に選び，かつ，値予測が正しく行われたことにより分岐予測が正しく行われたということを確認するものである．

3.2 値予測を用いた分岐予測器

図 2 に提案する分岐予測器の構成例を示す．従来の分岐予測器として gshare[2] を用い，値予測器には最終値予測器を用いている．他の予測器を用いる場合も同様に構成できる．

分岐命令のソース・オペランドは 1 つあるいは 2 つなので，2 つの値予測器を用意する．ソース・オペランドが 2 つの分岐命令に対しては，両方の値予測器を用い，1 つの分岐命令に対しては，左の値予測器しか用いない．各値予測器の VHT は 2^i のエントリを持つ．各エントリは命令に対応し，命令を識別するためのタグ（図では tag），最終値（図では value），および，予測の信頼性を表すリセッティング・カウンタ（図では conf）からなる．この値予測器は，命令の実行結果を予測する通常の用途にも使用され，そのときは 2^{i+1} のエントリを持つ 1 つの表としてアクセスされる．

分岐命令のオペコードを予測するために，分岐先バッファ（BTB: Branch Target Buffer）のエントリにオペコードを格納するフィールドを追加する．入れ換えの方法は通常の BTB と同じである．

分岐判定論理は，値予測器から得られるオペランド予測値と BTB から得られるオペコードを使い，分岐するかどうかを判定する．

どちらの分岐予測の結果を出力とするかの選択器は，2^n のエントリを持つ選択表と，最終的な決定を行う選択論理からなる．選択論理は，選択表から読み出された値と，BTB のヒット情報（図では省略している）と，(必要なら) 値予測器の信頼性カウンタの値を入力とし選択する．選択表のエントリのカウンタは 3.1.4 項で述べた方式により異なる．BTB ミスの場合はオペコードが得られないので，無条件に gshare を選択する．ヒットの場合は，3.1.4 項で述べた各方式にお

表 1 ベンチマーク・プログラム
Table 1 Benchmark programs.

プログラム	入力	静的分岐数	動的分岐数の90%を占める静的分岐数	動的分岐数
compress95	30000 e 2231	527	31	10.5 M
gcc	genoutput.i	16560	3657	13.0 M
go	6 9 2stone9.in	6039	1175	9.2 M
ijpeg	specmun.ppm	1496	50	23.9 M
li	train.lsp	677	59	24.2 M
m88ksim	ctl.in	1062	89	12.6 M
perl	scrabble.in	2134	175	10.4 M
vortex	vortex.in	1428	389	8.8 M

ける論理に従って選択する．

4. 評　　価

最初に，評価環境について述べる．次に値予測を用いた分岐予測器単体の予測精度を測定する．その後に，ハイブリッド予測器を構成し，3.1.4 項で述べた選択器を変えて予測精度を測定する．最後に，ハイブリッド予測器が計算機性能に与える影響を測定する．

4.1 評価環境

SimpleScalar Tool Set [24] Version 3.0a の中の，インオーダ実行をする命令レベルの機能シミュレータを基に，本機構を組み込んだシミュレータを作成し，測定を行った．命令セットは MIPS R10000 [25] を拡張した SimpleScalar/PISA である．ベンチマーク・プログラムは，SPECint95 の全 8 種類を使用した．ベンチマーク・プログラムのバイナリは，GNU GCC Version2.7.2.3（コンパイルオプション：-O6 -funroll-loops）を用いて作成した．

表 1 に各ベンチマーク・プログラムの入力，静的条件分岐数，動的分岐数の 90%を占める静的分岐数，動的条件分岐数を示す．シミュレーション時間が過大にならないようにするために，関数の出現頻度をほぼ維持しつつ，入力のパラメータを調整している．また，ijpeg では最初の 50 M 命令をスキップした後，450 M 命令を実行し，vortex では最初の 100 M 命令をスキップした後，80 M 命令を実行した．ijpeg，vortex 以外のベンチマーク・プログラムでは，命令のスキップを行わず，最後まで実行した．なお，以後のグラフでのベンチマーク・プログラム名の表記において，compress95，m88ksim，vortex は，それぞれ，comp，m88k，vort と表すこととする．なお，測定では値予測によるデータ投機は行わないものとした．

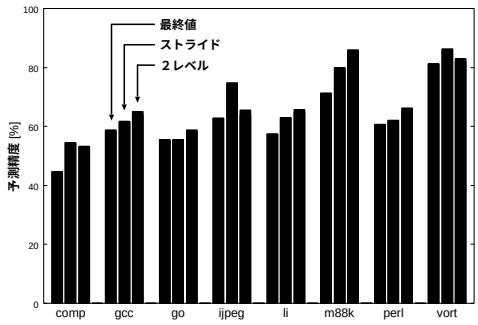
4.2 値予測による分岐予測精度

本節では，値予測を用いた分岐予測器単体の予測精度を測定する．値予測として，最終値予測，ストライド値予測，2 レベル値予測を用い，それぞれの場合について測定した．これら値予測器の動作は基本的に 2 章で説明したとおりであるが，次の点を補足しておく．いずれの値予測器も，値を予測する分岐が登録されていなければ静的予測を行う．具体的には，0 と予測する．これは分岐命令のオペランド値が 0 である確率が高いことが知られているからである [26]．ストライド値予測器では，予測する分岐の最終値は保持されているがストライドが検出されていない状態のとき，保持している最終値を出力する．いずれの予測器についても，信頼性を測定するリセッティング・カウンタを VHT の各エントリに設けた．

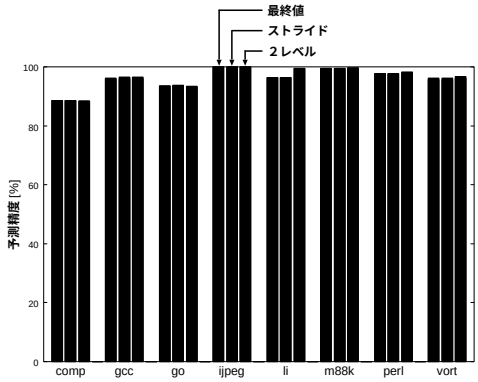
1 つの値予測器の VHT のエントリ数は 1 K とした．値予測器は 2 つ必要なので，合計で 2 K エントリである．BTB は連想度 4 で 2 K エントリの構成とした．ここで設定したエントリ数はほぼ十分であり，エントリでの衝突による予測精度低下は小さい．2 レベル値予測器では，保持する履歴数は 4 とした．この数は文献 11) を参考にして決めた．なお，この構成における最終値予測器，ストライド値予測器，2 レベル値予測器のハードウェア量はそれぞれ，13.0 KB，21.5 KB，41.5 KB である．

図 3 にソース・オペランド値の予測精度測定結果を示す．同図 (a) は分岐命令の第 1 ソース・オペランド（RS）の値予測精度であり，(b) は第 2 ソース・オペランド（RT）の値予測精度である．各ベンチマーク・プログラムにつき 3 本の棒グラフがある．左から，最終値予測，ストライド値予測，2 レベル値予測を使用した場合の測定結果である．

図 3 (a) より分かるように，RS の値予測精度（動的予測のみ）は，最終値で 45〜81%（平均 61%），ストライドで 54〜86%（平均 67%），2 レベルで 53〜86%（平均 67%）と，この順で高い精度を示している．過去の論文の測定結果と比較すると，最終値予測ではかなり高い値となっている．たとえば文献 10) では 23〜61%



(a) 第 1 ソース・オペランド（RS）



(b) 第 2 ソース・オペランド（RT）

図 3 ソース・オペランド値の予測精度
Fig. 3 Value prediction accuracy of the source operands.

（平均 40%）．ストライド値予測でもやや高い（38〜80%（平均 56%））．測定の際のバイナリや入力が異なるので一概にいえないが，分岐命令のオペランド値が他の命令のオペランド値に比べて局所性[8]が高いことが推測できる．

一方，同図 (b) より分かるように RT に関しては，90%を超す予測精度を達成しており，予測器による差はほとんどない．第 2 オペランドを持つ分岐は，MIPS 命令セットでは，`BEQ`（branch on equal）と `BNE`（branch on not equal）であるが，定数との比較が多いためと思われる．

図 4 に分岐予測精度を示す．各ベンチマーク・プログラムにつき 4 本の棒グラフがある．左から，最終値予測，ストライド値予測，2 レベル値予測をしようした場合の測定結果である．最後の 1 本は比較のためのもので，4 KB の PHT を持つ gshare の予測精度である．履歴長は，平均予測精度が最も高い 9 とした．これは測定によって求めた．

左 3 本の棒グラフは 2 つの部分に分かれている．下の黒い部分が「真」に正しく予測された部分であり，上の白い部分が「幸運」により正しく予測された割合

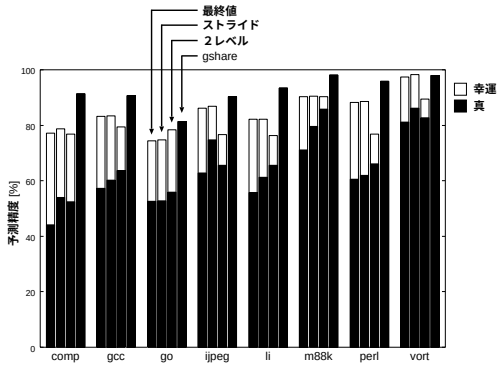


図 4 値予測による分岐予測精度
Fig. 4 Branch prediction accuracy with value prediction.

である．ここで，真に正しく予測された分岐とは，オペランド値が正しく予測され，その結果，分岐方向が正しく予測された分岐のことをいう．幸運で正しく予測された分岐とは，オペランド値は正しく予測されなかったが，偶然正しい予測結果が得られた分岐のことをいう．分岐判定が不一致や大小比較の場合，値が誤っていても正しい結果が得られる場合が多くあると考えられるので，幸運な分岐の割合はランダムに偶然が発生する割合より多い．

図 4 より，真の予測精度は，最終値で 44〜81%（平均 61%），ストライドで 53〜86%（平均 66%），2 レベルで 52〜86%（平均 67%）であった．どの予測器でもほぼ RS の値予測精度で抑えられている．ただし，幸運による予測精度増分は大きく，加えると予測精度は 77〜98%に達する．しかし依然として，ストライド値予測を用いた vortex の場合を除いたすべての場合において，gshare に勝る精度を達成することはできない．

4.3 ハイブリッド分岐予測器の予測精度

本節では，gshare とのハイブリッド予測器の性能を測定する．値予測による分岐予測器と gshare は，前節で述べたものと同一である．選択表のエントリ数は 2 K とした．この表のエントリに使用するカウンタと選択論理は，3.1.4 項で述べた 3 つの方法である．また比較のために，従来のハイブリッド分岐予測器で使用されている 2 ビット飽和カウンタによる方法についても測定した．

以下では最初に，値予測を用いた分岐予測により予測精度がどの程度改善する可能性があるか上限を調べる．その後，選択表を付加し実際の予測器の性能を測定する．

4.3.1 予測精度改善の上限

選択表が理想的に正しく選択した場合，本ハイブ

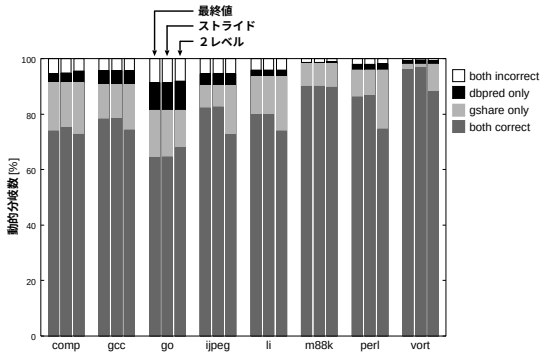


図 5 理想的な選択器による動的分岐の分布
Fig. 5 Dynamic branch distribution under the ideal selector.

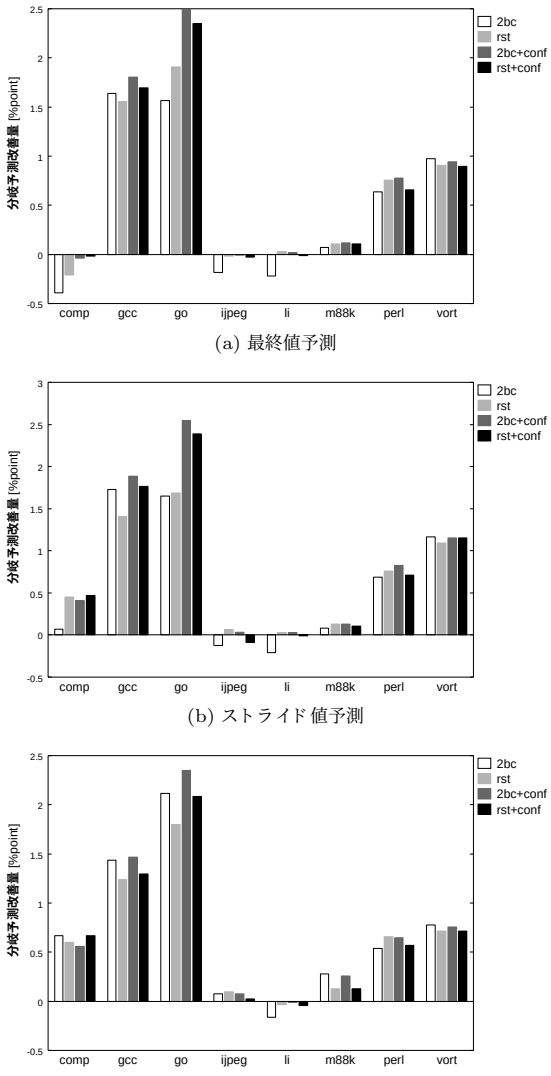
リッド分岐予測器が予測精度をどこまで改善できるかを調べる．動的分岐を以下の 4 つに分類し，その数を調べた．

- *both correct*：gshare と値予測を用いた分岐予測の両方が予測正解
- *gshare only*：gshare のみが予測正解
- *dbpred only*：値予測を用いた分岐予測のみが予測正解
- *both incorrect*：gshare と値予測を用いた分岐予測の両方が予測不正解

図 5 に各分類に属する分岐の割合を示す．各ベンチマーク・プログラムにつき 3 本の棒グラフがある．左から，最終値予測，ストライド値予測，2 レベル値予測を使用した場合の測定結果である．選択器が理想的に分岐予測器を選択すれば，gshare 単体の場合に比べて *dbpred only*の分だけ予測精度が向上する．分岐予測精度が 100%に近い m88ksim を除くプログラムでは，値予測を用いた分岐予測により予測精度が改善する可能性は 1%ポイント以上ある．特に，gshare による予測精度が低い compress95，gcc，go では，予測精度改善に大きな可能性があることが分かる．compress95 では 3.2%ポイント〜4.1%ポイント，gcc では 5.0%ポイント〜5.2%ポイント，go では 10.0%ポイント〜10.6%ポイントもの改善可能性が存在する．全ベンチマーク平均では 3.6%ポイント〜3.9%ポイントの改善可能性が存在する．

4.3.2 選択器と予測精度

図 6 にハイブリッド分岐予測器の予測精度の評価結果を示す．同図 (a)〜(c) は値予測としてそれぞれ，最終値予測，ストライド値予測，2 レベル値予測を用いた場合である．各グラフの縦軸は予測精度の改善量を示す．すなわち，ハイブリッド分岐予測器の予測精度から gshare 単体の予測精度を引いた値である．選択



(a) 最終値予測

(b) ストライド値予測

(c) 2 レベル値予測

図 6 ハイブリッド分岐予測器の予測精度
Fig. 6 Prediction accuracy improvement of the hybrid predictors.

器の違いで，各ベンチマーク・プログラムにつき 4 本の棒グラフがある．最も左の棒グラフは，従来のハイブリッド分岐予測器で採られている方式で，選択表を 2 ビット飽和カウンタで構成する方式（2bc）である．その右 3 本は 3.1.4 項で述べた方式であり，それぞれ，リセッティング・カウンタ方式（rst），2 ビット飽和カウンタと値予測器の信頼性を用いる方式（2bc+conf），リセッティング・カウンタと値予測器の信頼性を用いる方式（rst+conf）である．選択表と値予測器の信頼性のリセッティング・カウンタの閾値は，測定により最適な値を求めた．**表 2** にその値をまとめる．また，このときの値予測器と選択表を合わせたハードウェア

表 2 最適な閾値
Table 2 Optimum threshold values.

選択方式	最終値		ストライド		2 レベル	
	選択表	信頼性	選択表	信頼性	選択表	信頼性
rst	2	-	3	-	2	-
2bc+conf	-	3	-	3	-	3
rst+conf	1	4	1	4	1	5

表 3 値予測器と選択表のハードウェア量
Table 3 Hardware cost of both value predictor and selection table.

選択方式	ハードウェア量 [KB]			
	選択表	最終値	ストライド	2 レベル
2bc	0.50	13.0	21.5	41.5
2bc+conf	0.50	13.5	22.0	42.0
rst	0.50	13.0	21.5	41.5
rst+conf	0.25	13.8	22.3	42.3

量を**表 3** に示す．表中において選択表の欄の数値は選択表のみのハードウェア量を表し，最終値，ストライド，2 レベルの欄の数値は値予測器のハードウェア量を表している．表 3 より選択表に必要なハードウェア量は小さいことが分かる．将来のプロセッサでは値予測器はデータ依存を緩和するために搭載されるため，本手法を実現するためのハードウェア量は小さいといえる．

図 6 より次のことが分かる．

(1) 図 5 より分かるように，gcc と go は，理想の選択器での予測精度の改善が大きく，実際の選択器を使った場合でも予測精度が大きく改善された．

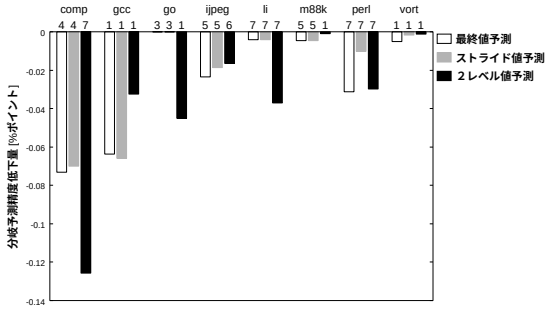
(2) m88ksim と vortex は理想の選択器での予測精度の改善が小さい．このため，m88ksim では実際の選択器を使った場合にはわずかな予測精度の改善にとどまっている．一方，vortex では図 4 の真の予測精度を見れば分かるように値予測精度がきわめて高く，選択を誤る確率が小さいため予測精度が改善した．

(3) compress95，ijpeg，li，perl は，理想選択器での予測精度の改善が中程度である．そのため多くの場合で compress95 と perl の実際の選択器での改善量は gcc や go と m88ksim のそれとの中間となっている．ただし最終値予測で 2bc を選択器を使った compress95 では場合の性能が低下している．これは最終値予測の予測精度が非常に低く，2bc では，値予測を用いた分岐予測が選択されるリスクを回避できないためと思われる．また，ijpeg，li は実際の選択器を使った場合，予測精度は改善しないか，改善したとしても非常にわずかであった．これは今回の測定データからは推測できない．理想の選択器では改善が認められるので，選択器の改良が必要であると思われる．

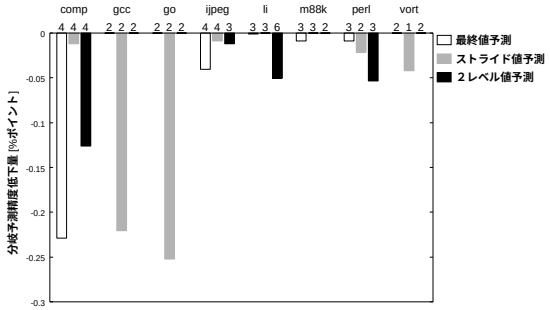
(4) 選択器は，どの値予測器の場合でも，改善量の平均値で比べると，2bc+conf が最も良い性能を示した．これは，2bc では値予測を用いた分岐予測が選択されるリスクが回避できず，rst や rst+conf では値予測を用いた分岐予測の選択を過度に制限するためであると考えられる．

(5) 2bc+conf を使った場合で，最終値予測で最大 2.50%ポイント，平均 0.77%ポイント，ストライド値予測で最大 2.55%ポイント，平均 0.88%ポイント，2 レベル値予測で最大 2.35%ポイント，平均 0.76%ポイント予測精度を改善できた．このようにストライド値予測が最大の改善量を得た．

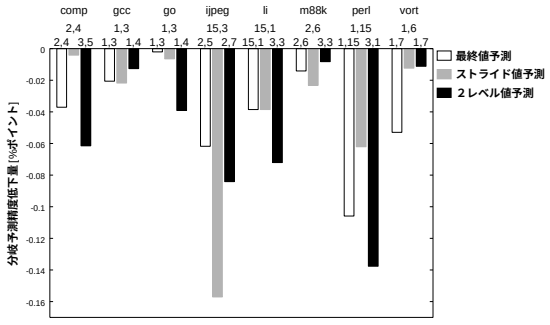
次に予測精度が，選択表と値予測の信頼性のリセッティング・カウンタの 2 つのパラメータによって，どの程度影響を受けるかについて考察する．このためには，各ベンチマークごとに閾値を変化させ，そのときの予測精度の変化を図示すればよいが，誌面節約のため，ここでは，各ベンチマーク・プログラムについて最適な閾値の場合に対し，ベンチマークの平均で最適な閾値の場合，どの程度予測精度が低下するかを測定した結果を**図 7** に示す．同図 (a)〜(c) は選択方式としてそれぞれ，2bc+conf，rst，rst+conf を用いた場合である．横軸はベンチマーク・プログラムで，縦軸は各ベンチマーク・プログラムごとの最適な閾値からの予測精度低下量である．各ベンチマーク・プログラムにつき 3 本の棒グラフがあり，左から最終値予測，ストライド値予測，2 レベル値予測の場合である．横軸上の数字や数字の組は，そのベンチマークにおける最適な閾値を表す．つまり，2bc+conf では最適な信頼性カウンタの閾値，rst では最適な選択表の閾値，rst+conf では選択表と信頼性カウンタの閾値をそれぞれ意味している．同図より分かるように，ベンチマークについて最適化しても，ほとんどの場合変化は小さい．大きくても，0.26%ポイントである（rst のストラ



(a) 2bc+conf



(b) rst



(c) rst+conf

図 7 ベンチマークごとの最適パラメータからの予測精度低下量
Fig. 7 Prediction accuracy degradation from optimum parameters case.

イド値予測の go）．また，最適な閾値はベンチマークによっては，表 2 に示したベンチマーク平均で最適な閾値と大きく異なるものがある．このことから，閾値に対する予測精度の感度は小さいことが分かる．

5. 計算機性能に与える影響

本章では，gshare とのハイブリッド予測器が計算機性能に与える影響について評価を行い，次に既存手法との性能比較を行う．

5.1 評価モデル

評価には，SimpleScalar Tool Set [24] Version 3.0a の中のスーパスカラ・プロセッサ用シミュレータに本機構を組み込んだものを用いた．ベンチマーク・プログラムは前章までと同じ SPECint95 の全 8 種類を用いた．以下の 4 つのモデルについて評価を行った．

- **Current モデル**：8 命令フェッチ，分岐予測ミスペナルティ最小 10 サイクル
- **Wide モデル**：32 命令フェッチ，分岐予測ミスペナルティ最小 10 サイクル
- **Deep モデル**：8 命令フェッチ，分岐予測ミスペナルティ最小 40 サイクル
- **Wide&Deep モデル**：32 命令フェッチ，分岐予測ミスペナルティ最小 40 サイクル

Current モデルは現在のハイエンドのプロセッサに近い命令発行幅とパイプラインの深さのものを想定している．他の 3 つはいずれも将来におけるプロセッサを想定したものである．Wide モデルは，将来のプロセッサが命令発行幅の増加という方向へ進んだ場合のモデルである．一方，Deep モデルは，将来のプロセッサがパイプラインを深くし，クロック速度を向上させる方向へ進んだ場合のモデルである．最後の Wide&Deep モデルは，パイプライン段数，命令発行幅の両方が増加する方向へ進んだ場合のモデルである．

表 4 に評価したプロセッサモデルを示す．表 4 (a) は 4 つのモデルすべてに共通のパラメータであり，表 4 (b) はモデルによって変化するパラメータである．なお，表 4 (b) のパラメータは Current モデルと Deep モデルのものである．Wide モデル，Wide&Deep モデルでは命令フェッチ幅が Current モデル，Deep モデルに対して 4 倍になるので，表 4 (b) に示したパラメータも 4 倍とした．

また，ハイブリッド分岐予測器がプロセッサのサイクル時間に悪影響を与えないように，1 サイクル目で値予測器および選択表の参照までを行い，2 サイクル目で値予測を用いた分岐予測の結果を得るようにした．そのため，プロセッサは gshare の予測結果を用いて投機的にフェッチを行い，選択器が値予測を用いた分岐予測の結果を用いると判定し，その予測方向が gshare のものと異なっていた場合のみ，1 サイクルのペナルティを払って再フェッチを行うものとした．gshare の履歴情報は予測を行った直後に予測結果を用いて投機的に更新し，PHT は命令のコミット時に更新するものとした．値予測器と値予測器の信頼性カウンタと選択器の更新も命令のコミット時に行うものとした．また，値予測によるデータ投機は行わないものとした．

5.2 評価結果

gshare とのハイブリッド分岐予測器の IPC を**図 8** に，性能向上率を**図 9** に示す．選択器には最も性能

表 4 プロセッサモデル
Table 4 Processor model.

(a) すべてのモデルに共通のパラメータ

実行レイテンシ	iALU 1，iMULTI/DIV 3/20，Ld/St 1/1， fpALU 2，fpMULT/DIV/SQRT 4/12/24
分岐予測機構	インデクス長 14 ビット 履歴長 9 ビット gshare， 2048 エントリ 4 ウェイ・セットアソシアティブ BTB， 32 エントリ リターン・アドレス・スタック
命令キャッシュ	64 KB 2 ウェイ・セットアソシアティブ， ライン幅 32 バイト，ヒットレイテンシ 2 サイクル
データキャッシュ	64 KB 2 ウェイ・セットアソシアティブ， ライン幅 32 バイト，ヒットレイテンシ 2 サイクル
二次キャッシュ	統合，1 M 8 ウェイ・セットアソシアティブ， ライン幅 64 バイト，ヒットレイテンシ 10 サイクル， ミスペナルティ 32 サイクル

(b) モデルによって変化するパラメータ

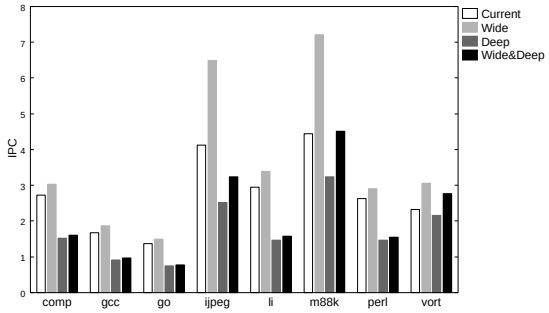
命令デコード幅	8 命令
命令発行幅	16 命令
命令コミット幅	8 命令
命令ウィンドウ	RUU (Register Update Unit) 128 エントリ LSQ (Load/Store Queue) 64 エントリ
機能ユニット数	iALU 8，iMULTI/DIV 1，Ld/St 4， fpALU 2，fpMULT/DIV/SQRT 1

向上が大きかった 2bc+conf を用いた．分岐予測器の各種パラメータは前章で示したものを用いた．それぞれの図において，(a)〜(c) はそれぞれ値予測に最終値予測，ストライド値予測，2 レベル値予測を用いた場合である．図 8 の各グラフの縦軸は IPC で，図 9 の各グラフの縦軸は，4 KB の gshare を持つプロセッサの IPC に対する性能向上率である．各ベンチマーク・プログラムにつき 4 本の棒グラフがあり，左から順に，Current モデル，Wide モデル，Deep モデル，Wide&Deep モデルでの結果である．

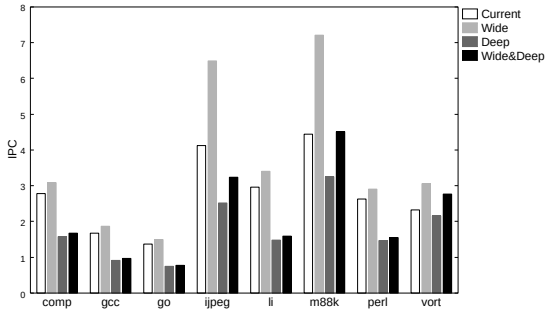
性能向上率は Current モデルでは最大 5.95%（gcc），平均 2.43%であり，Wide&Deep モデルでは最大 14.14%（vortex），平均 6.09%であった．各ベンチマーク・プログラムごとに見ると，gcc，go では分岐予測精度の改善が大きいが，全体の実行時間が長いため，分岐予測精度を改善してもその効果は小さくなる傾向にある．ijpeg，li では分岐予測精度の改善がほとんどないため，性能向上率もほとんどない．m88ksim，perl，vortex では分岐予測率の改善がそれほど高くないが，全体の実行時間が短いため，わずかな分岐予測精度の改善でもその効果は大きくなる傾向にある．compress95 では 2 レベル値予測器を使った場合のみ，予測精度が改善しているが，性能向上率が低下している．これはアウトオブオーダ実行時の分岐予測精度の改善がほとんどなくなったためであり，その原因は選択器や値予測器の予測精度の低下にあると考えられる．

5.3 値予測による分岐予測のレイテンシの影響

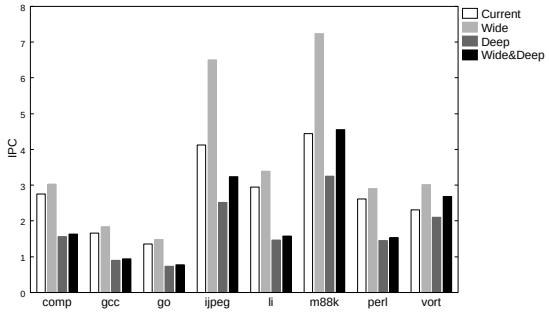
本節では BPVP [22] と本方式の大きな違いである値予測を用いた分岐予測のレイテンシがプロセッサ性能に与える影響について評価する．ハイブリッド予測器は同一のものを用い，値予測による分岐予測のレイテンシのみを変えて測定した．2 章で述べたように，BPVP では命令の参照レジスタ番号が得られて 3 サイクル程度後に予測結果が得られる．このタイミングは命令フェッチとデコードに何サイクル要するか，つまりパイプラインの深さに依存する．そこで本評価では，BPVP の分岐予測レイテンシを分岐予測ミスペナルティの半分と仮定することとした．本方式におけるレイテンシは 5.1 節で述べたとおり，2 とした．**図 10** に評価結果を示す．選択器には 2bc+conf を用い，分岐予測器の各種パラメータは前章で示したものを用いた．同図 (a)〜(c) はそれぞれ値予測に最終値予測，ストライド値予測，2 レベル値予測を用いた場合であり，左のグラフは Current モデルと Wide モデルでの結果であり，右のグラフは Deep モデルと Wide&Deep モデルでの結果である．グラフの縦軸は性能向上率で，4 KB の gshare を持つプロセッサの IPC に対する性能向上率である．各ベンチマーク・プログラムにつき 4 本の棒グラフがあり，左の図では左から順に 2 本ずつ，Current モデル，Wide モデルでの結果であり，右の図では左から順に 2 本ずつ，Deep モデル，Wide&Deep モデルでの結果である．各モデルには BPVP による性能向上率と本方式による性能向上率を示してある．



(a) 最終値予測



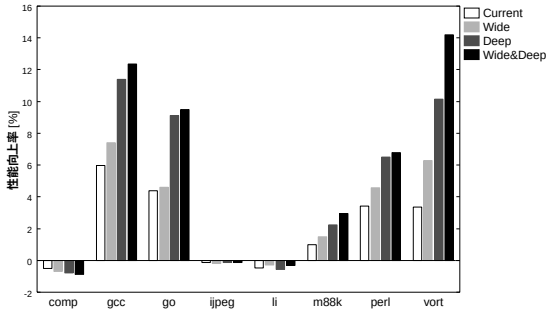
(b) ストライド値予測



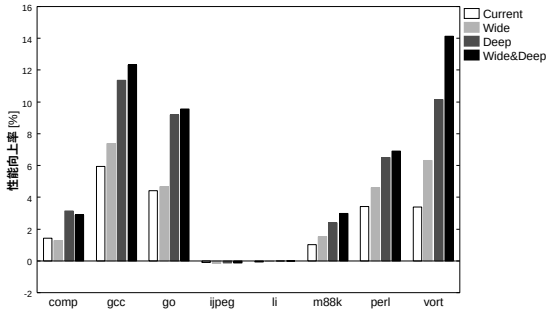
(c) 2 レベル値予測

図 8 プロセッサ性能

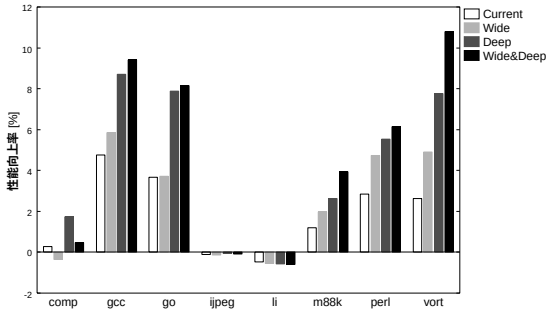
Fig. 8 Processor performance with hybrid branch predictor.



(a) 最終値予測



(b) ストライド値予測



(c) 2 レベル値予測

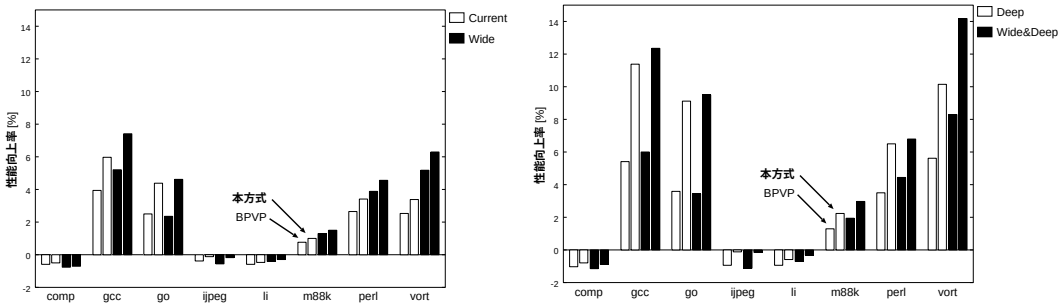
図 9 プロセッサ性能への影響

Fig. 9 Impact on processor performance.

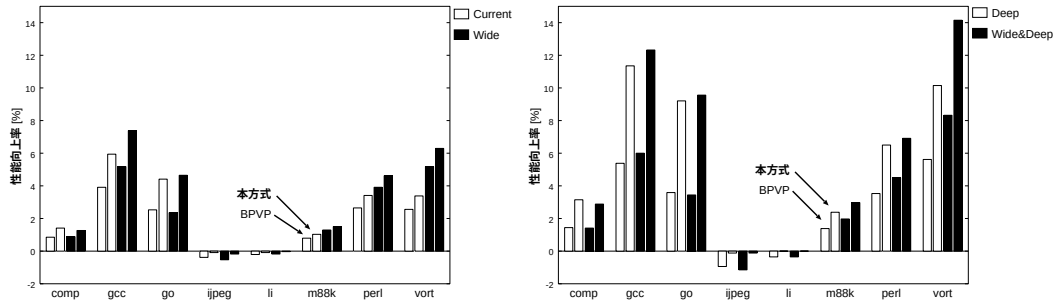
図から分かるようにすべてのモデルにおいて，本方式は BPVP よりも性能向上率が高い．当然であるが，パイプラインが深いほどその差は大きい．また，大きく性能が向上しているものほどその差が大きい．この理由を以下に示す．本方式での実行サイクル数を C_{dbpred}，本方式と BPVP において値予測による分岐予測を適用した場合のレイテンシの差を P_{hit}，正しく分岐が予測された中で，値予測による分岐予測の結果が採用された回数を V とすると，BPVP での実行サイクル数はおよそ $C_{BPVP} = C_{dbpred} + P_{hit} \times V$ という関係がある．つまり値予測による分岐予測を適用すればするほど，また値予測による分岐予測を採用するのに要するレイテンシが大きくなればなるほど，両方式での実行サイクル数の差が大きくなる．一般に，値予測を用いた分岐予測を多く行うと，大きな性能向上が得られる．したがって，大きな性能向上が得られるものほど，BPVP との性能向上率に大きな差が生じる．

6. ま と め

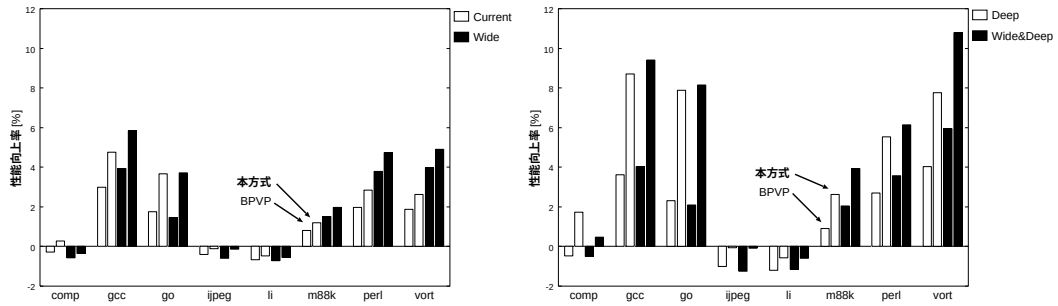
値予測は，データ依存を緩和し性能を向上させるために，将来のプロセッサには必須の技術となる．本論文では，プロセッサに搭載された値予測器を分岐予測に利用する方式を提案した．本方式は，分岐命令のオペランド値とオペコードを予測し，分岐方向を予測す



(a) 最終値予測



(b) ストライド値予測



(c) 2 レベル値予測

図 10 値予測による分岐予測のレイテンシがプロセッサ性能に与える影響
Fig. 10 Impact on processor performance with different latency of branch prediction using value prediction.

るものである．本方式による分岐予測器は，単体では従来の分岐予測器の性能を上回ることができないが，ハイブリッド構成にすることにより予測精度の改善に寄与することができる．この構成は，従来の分岐予測器が利用していた相関とは異なる相関を利用できるという長所がある．本ハイブリッド分岐予測器に潜在する性能をより引き出すためには，本論文では種々の手法を提案した．実験の結果，最近の予測成功頻度の違いにより選択する従来手法は不十分であり，予測の信頼性を検査する手法が必要であることが分かった．

シミュレーションにより性能評価を行った結果，4 K バイトの gshare とのハイブリッド構成で，最大 2.55% ポイント，平均 0.84%ポイント予測精度を改善できることを確認した．この予測精度の向上により，今日のスーパスカラ・プロセッサで最大 5.95%（gcc），平均 2.43%の速度向上が，また命令発行幅が増えパイプライン段数が深くなる将来のスーパスカラ・プロセッサでは最大 14.14%（vortex），平均 6.09%の速度向上が見込めることが分かった．

謝辞 本研究の一部は，文部科学省科学研究費補助金基盤研究（C）「分岐予測と投機的実行に関する研究」（課題番号 11680351）および財団法人大川情報通信基金の支援により行った．

参 考 文 献

1) 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機構，情報処理学会論文誌，Vol.40, No.5, pp.2119–2131 (1999).
2) McFarling, S.: Combining Branch Predictors, WRL Technical Note TN-36, Digital Equipment Corporation (1993).
3) Yeh, T.-Y. and Patt, Y.N.: Two-Level Adaptive Branch Prediction, *Proc. 24th Ann. Int'l Symp. and Workshop on Microarchitecture*, pp.51–61 (1991).
4) Yeh, T.-Y. and Patt, Y.N.: A Comparison of Dynamic Branch Predictors that use Two Levels of Branch History, *Proc. 20th Int'l Symp. on Computer Architecture*, pp.257–266 (1993).
5) Gwennap, L.: Digital 21264 Sets New Standard, *Microprocessor Report*, Vol.10, No.14, pp.11–16 (1996).
6) Advanced Micro Devices, Inc.: *AMD Athlon Processor Technical Brief* (1999).
7) Gabbay, F. and Mendelson, A.: Speculative Execution based on Value Prediction, EE Department TR #1080, Technion—Israel Institute of Technology (1996).
8) Lipasti, M.H., Wilkerson, C.B. and Shen, J.P.: Value Locality and Load Value Prediction, *Proc. 7th Int'l Conf. on Architectural Support for Programming Languages and Operating Systems*, pp.138–147 (1996).
9) Lipasti, M.H. and Shen, J.P.: Exceeding the Dataflow Limit via Value Prediction, *Proc. 29th Int'l Symp. on Microarchitecture*, pp.226–237 (1996).
10) Sazeides, Y. and Smith, J.E.: The Predictability of Data Values, *Proc. 30th Int'l Symp. on Microarchitecture*, pp.248–258 (1997).
11) Wang, K. and Franklin, M.: Highly Accurate Data Value Prediction using Hybrid Predictors, *Proc. 30th Int'l Symp. on Microarchitecture*, pp.281–290 (1997).
12) Gabbay, F. and Mendelson, A.: The Effect of Instruction Fetch Bandwidth on Value Prediction, *Proc. 25th Int'l Symp. on Computer Architecture*, pp.272–281 (1998).
13) 戸田 聡, 布施裕基, 片山清和, 安藤秀樹, 島田俊夫：値予測を用いた分岐予測，2000 年記念並列処理シンポジウム JSPP2000, pp.237–244 (2000).
14) 中村幸司，片山清和，布施裕基，安藤秀樹，島田俊夫：値予測を用いた分岐予測機構の計算機性能に与える影響，情報処理学会研究報告，2001-ARC-141, pp.59–64 (2001).
15) Lee, J.K.F. and Smith, A.J.: Branch Prediction Strategies and Branch Target Buffer Design, *IEEE Computer*, Vol.17, No.1, pp.6–22 (1984).
16) Chang, P.-Y., Hao, E. and Patt, Y.N.: Alternative Implementations of Hybrid Branch Predictors, *Proc. 28th Int'l Symp. on Microarchitecture*, pp.252–257 (1995).
17) 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 Int'l Symp. on Computer Architecture*, pp.3–11 (1996).
18) 西本晴子，勝野 昭，木村康則：ロードアドレス予測による命令並列度の向上，情報処理学会研究報告，96-ARC-119, pp.49–54 (1996).
19) Calder, B., Reinman, G. and Tullsen, D.M.: Selective Value Prediction, *Proc. 26th Int'l Symp. on Computer Architecture*, pp.64–74 (1999).
20) Jacobsen, E., Rotenberg, E. and Smith, J.E.: Assigning Confidence to Conditional Branch Predictions, *Proc. 29th Int'l Symp. Michroarchitecture*, pp.142–152 (1996).
21) Sazeides, Y. and Smith, J.E.: Modeling Program Predictability, *Proc. 25th Int'l Symp. on Computer Architecture*, pp.73–84 (1998).
22) González, J. and González, A.: Control-Flow Speculation through Value Prediction for Superscalar Processors, *Proc. Parallel Architectures and Compilation Techniques*, pp.57–65 (1999).
23) Michaud, P., Seznec, A. and Uhlig, R.: Trading Conflict and Capacity Aliasing in Conditional Branch Predictors, *Proc. 24th Int'l Symp. on Computer Architecture*, pp.292–303 (1997).
24) Burger, D. and Austin, T.M.: The SimpleScalar Tool Set, Version 2.0, Technical Report CS-TR-97-1342, University of Wisconsin-Madison (1997).
25) MIPS Technologies, Inc.: *MIPS R10000 Processor User's Manual, Version 2* (1996).
26) Hennessy, J.L. and Patterson, D.A.: *Computer Architecture: A Quantitative Approach*, 2nd edition, Morgan Kaufmann Publishing Inc., San Francisco, CA (1996).

(平成 13 年 5 月 8 日受付)
(平成 13 年 8 月 20 日採録)

片山　清和（正会員）

1994 年名古屋大学工学部情報工学科卒業．1996 年同大学大学院工学研究科情報工学専攻博士課程前期課程修了．1999 年同大学院工学研究科電子情報学専攻博士課程後期課程満了．現在，同大学院工学研究科在学中．計算機アーキテクチャの研究に従事．

戸田　　聡（正会員）

1976 年生．2000 年名古屋大学工学部電気電子情報学科卒業．同年メルコ（株）に入社．在学中，分岐予測の研究に従事．

中村　幸司（正会員）

1976 年生．1999 年名古屋大学工学部電気電子情報学科卒業．2001 年同大学大学院工学研究科電子情報学専攻博士課程前期課程修了．同年沖電気（株）に入社．

布施　裕基（正会員）

1976 年生．1999 年名古屋大学工学部電気電子情報工学科卒業．2001 年同大学大学院工学研究科電子情報学専攻博士課程前期課程修了．同年日本電気（株）に入社．

安藤　秀樹（正会員）

1959 年生．1981 年大阪大学工学部電子工学科卒業．1983 年同大学大学院修士課程修了．京都大学工学博士．1983 年三菱電機（株）LSI 研究所．ISDN 用ディジタル信号処理 LSI，第 5 世代コンピュータ・プロジェクトの推論マシン用プロセッサの設計に従事．1991 年 Stanford 大学客員研究員．1997 年名古屋大学大学院工学研究科電子情報学専攻講師．1998 年同大学助教授．1998 年東京大学大学院理学系研究科助教授併任．1998 年情報処理学会論文賞受賞．計算機アーキテクチャ，コンパイラの研究に従事．

島田　俊夫（正会員）

1968 年東京大学工学部計数工学科卒業．1970 年同大学大学院修士課程修了．同年電子技術総合研究所入所．1993 年より名古屋大学大学院工学研究科電子情報学専攻教授．人工知能向き言語，LISP マシン，データフロー計算機の研究に従事．最近はマイクロプロセッサのアーキテクチャやチップ内並列処理の研究を行っている．1988 年度市村賞，1994 年度情報処理学会論文賞，1995 年度注目発明賞受賞．工学博士．