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

野口良太[†] 森 敦司[††] 小林良太郎[††]
安藤秀樹[††] 島田俊夫[††]

命令レベル並列性を利用する高性能プロセッサにおいては，分岐予測機構の予測精度は性能に大きな影響を与える．分岐予精度を制限する主たる要因は，競合であることが知られている．これまで競合への対策として，その発生そのものを減少させる試みがなされてきた．しかしこの方策だけでは予測精度を改善するには限界がある．これに対して我々は，競合の中でも予測に悪影響を与える破壊的な競合の削減に注目し，競合が発生しても予測精度を大きく低下させないアプローチをとる．このアプローチに基づきsgshareと呼ぶ新しい予測機構を提案する．sgshareでは，パターン履歴テーブルを2つに分離し，分岐方向の偏りが同一の分岐を一方のテーブルにのみマッピングすることにより，破壊的な競合を無害な競合に変換する．IBS-Ultrixベンチマークを用いたシミュレーションにより評価をした結果，sgshareは従来のgshareと比較して，約8Kバイトのハードウェア・コストにおいて，0.05～0.89%予測精度が向上することを確認した．また，この予測精度の改善により，プロセッサの性能は今日のスーパースカラ・マシンにおいて平均7.19%，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ースカラ・マシンにおいて平均14.6%速度が向上することを確認した．

A Branch Prediction Scheme that Reduces Destructive Aliasing Using Branch Direction Bias

Ryota Noguchi,[†] Atsushi Mori,[††] Ryotaro Kobayashi,[††]
Hideki Ando[††] and Toshio Shimada[††]

Branch prediction accuracy makes a tremendous impact on modern high-performance microprocessors that exploit instruction-level parallelism. Aliasing is a known major factor that limits prediction accuracy. A several studies have attempted to reduce the total rate of aliasing. Yet, only this approach is not enough for further improvement of prediction accuracy. In this paper, we focus on reducing *destructive aliasing* that has a negative impact on prediction accuracy. We propose a new branch prediction scheme, called *sgshare*, which modestly affects prediction accuracy even when aliasing occurs. Sgshare divides the pattern history table into two parts, and maps branches with the same direction bias to only a single part of the table. This mechanism converts destructive aliasing into harmless aliasing. Our evaluation results show that sgshare achieves 0.05–0.89% better prediction accuracy than the conventional gshare with approximately 8 K-byte hardware cost. This improvement of the prediction accuracy contributes to the performance of a current superscalar processor by 7.19%, and that of a future superscalar processor with a wider instruction issue and a deep pipeline by 14.6%.

1. はじめに

近年の高性能プロセッサ[1),2)]は，パイプラインを深することによりクロック速度の向上を図るとともに，時発行命令数を増加させプログラムに内在する命令ベルの並列性をより多く引き出すことにより性能の上を図っている．このようなプロセッサにおいては，分岐による命令流の分断は非常に大きな雖能低下を引き起こす．このため分岐命令の結果が分かる前にその結果を予測し，命令を連続的に供給するという機構を備えるのが一般的である．分岐の予測が正しい間は，深いパイプラインにおいても分岐によるバブルは生じない．また，分岐が解決する前にその分岐より後方の命令を実行する投機的実行も成功するため，命令レベルの並列性を多く引き出すことができる．

これに対して分岐の予測が誤っていた場合には，投機的に実行したすべての結果を取り消し，予測を誤った分岐以前のプロセッサ状態に戻した後，新たに正し

† 株式会社日立製作所汎用コンピュータ事業部
General Purpose Computer Division, Hitachi, Ltd.
†† 名古屋大学大学院工学研究科
Graduate School of Engineering, Nagoya University

い分岐方向の制御流に従い実行を再開しなければならない．現在においても，パイプライン段数と命令発行幅を考えれば，誤った予測により破棄される命令数は非常に多く，性能に与えるペナルティは重大である．プロセッサは今後もより高いクロック速度と命令レベル並列性を追及する方向にあり，パイプライン段数はより深く，命令発行幅はより広くなると考えられる．このような傾向のもとで，分岐予測の正確さへの要求はいっそう強くなっている．

これまでさまざまな分岐予測機構が提案されてきた．これらはすべて過去の分岐方向の履歴と将来の分岐方向の間に相関があることを利用している．初期の機構[3]は，予測する分岐自身の履歴を要約するために2ビットの飽和型カウンタを用いた．この方式を**2ビット・カウンタ方式（2bc）**と呼ぶ．2bcでは，カウンタを多数並べたテーブルを設け，分岐命令のアドレスをインデクスとして参照し，過去の履歴を知り予測する．

これに対して，より高い予測精度を達成するために，より多くの履歴情報を元に予測する方式が提案された[4)~7)]．これらの方式が新たに利用している履歴情報とは，主として履歴のパターンや予測する分岐以外の他の分岐の履歴である．これらの方式を総称して**2レベル適応型方式**と呼ぶ．2レベル適応型方式では，分岐命令のアドレスと新たに加えた履歴情報の組に対して，2ビット・カウンタで構成されるテーブルのエントリを対応付ける．

2レベル適応型方式は2bcに比べて優れているものの，予測精度は**競合（aliasing）**という現象によって制限されていることが最近分かってきた[8)]．競合とは，異なる分岐命令アドレスと分岐履歴の組が同一カウンタに対応付けられることをいう．競合により，異なる組に対する予測情報を区別して記録することができなくなり，予測精度が低下する．最近の研究では，カーネルの振舞いも含めた実際的な使用環境においては競合が頻発し，その結果，予測精度が大きく低下するという問題が注目されている[9),10)]．

競合の問題に対して，分岐アドレスと分岐履歴を2ビット・カウンタ・テーブルに対応付けるマッピング関数を工夫し競合を減少させることで対応しようとした研究がある[10),11)]．このアプローチは，テーブルのエントリを有効に利用することを促進するが，これだけによる性能改善には限界がある．なぜならば，現実的なハードウェアの制約のもとで分岐命令アドレスと分岐履歴のとりうるすべての組合せに対応できる巨大なテーブルを保有することが不可能であるからである．つまり，競合を回避するいかに優れたマッピング手法を用いても競合は不可避である．

我々は競合の問題に対して，競合自体を削減するのではなく，競合が生じても予測精度が低下しない方策を考えた．具体的には，分岐命令アドレスと履歴の組に対し，競合が生じたとしても分岐方向の偏りが同一のものが競合するように予測機構を構成する．こうすることで，競合により予測が誤る確率を下げる．本論文ではこの考えに基づき，**sgshare予測機構**[12)~14)]と呼ぶ新しい機構を提案する☆．

本論文の構成は次のとおりである．2章では分岐予測機構および競合に関する過去の研究について述べる．3章では競合の問題に対する我々のアプローチについて述べる．4章で本アプローチに基づく予測機構としてsgshare予測機構を提案する．5章では同機構の性能の評価を行い有効性を検証する．6章では本機構によりプロセッサの性能がどの程度向上するかについて調査する．7章で本論文をまとめる．

2. 関連研究

本章では最初に，2レベル適応型分岐予測方式の概要について述べる．その中でも我々の提案するsgshareの基本となるgshareについて詳しく説明する．次に，予測精度を低下させる競合に関するこれまでの研究について述べる．

2.1 2レベル適応型分岐予測方式

図1に2レベル適応型分岐予測方式[4)~7)]の概要を示す．この方式ではまず，各エントリが2ビットの飽和型カウンタよりなるテーブルを持つ．このテーブルのことを**パターン履歴テーブル（PHT: Pattern History Table）**と呼ぶ．分岐命令アドレスと分岐履歴の組は，各機構特有のマッピング関数によりPHT

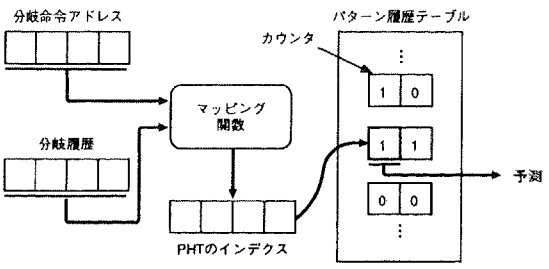


図1 2レベル適応型分岐予測方式
Fig. 1 Two-level adaptive branch prediction schemes.

☆ 同一の機構が文献15)に提案されているが，我々の研究は彼らとは無関係に独立に行われたものである．文献12)にあるように我々の発表の方が彼らの発表より早い．

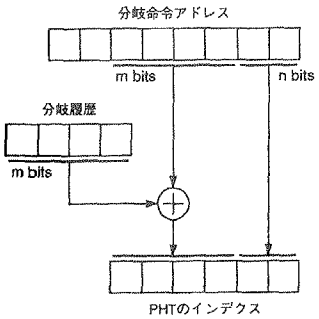


図 2 gshare における PHT のインデクスの生成
Fig. 2 Index generation of PHT in gshare.

のエントリに対応付けられる．分岐が実行されると，分岐履歴および対応するカウンタを更新する．予測は対応するカウンタの値を参照することによって行う．

2 レベル予測方式では，使用する履歴や PHT へのマッピングの方法等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**[7] と呼ばれる方式は，きわめて高精度な予測を行う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．図 2 に gshare における PHT のインデクスの生成方法を示す．履歴を記録するために m ビットのシフト・レジスタを用意する．分岐が実行されたら，シフト・レジスタを前にシフトし分岐結果を最後尾に書き込む．このようにして m ビットのシフト・レジスタに最近の過去 m 個の動的分岐の履歴を保持する．

PHT のインデクスは次のようにして生成する．分岐命令アドレスの第 $(m+n-1)$ ビット目から第 n ビット目までの m ビットの部分と分岐履歴の m ビットの XOR をとる．これを分岐命令アドレスの下位 n ビットと結合し PHT のインデクスとする．以下，分岐命令アドレスの下位 n ビットの部分を PHT のインデクスへの**追加アドレス**と呼ぶこととする．

2.2 競　　合

分岐命令アドレスと履歴の組が，PHT の同一カウンタに対応付けられる現象を**競合**（**aliasing**）と呼ぶ．カーネルの振舞いを含めた実際的な環境においては，PHT での競合が予測精度に大きな影響を与えることが分かり[9),10)]，近年この現象に対する関心が高まっている．

こうした中で，Young ら[8)] は分岐予測に与える影響の観点から競合を以下の 3 種類に分類した．

- **破壊的競合**（**destructive aliasing**）　競合の結果，競合がないときと異なる予測となり，かつ，その予測が誤っている場合における競合
- **建設的競合**（**constructive aliasing**）　破壊的競合と同じく競合がないときと異なる予測となったが，結果的にその予測が正しかった場合における競合
- **無害な競合**（**harmless aliasing**）　競合がないときと同じ予測となった場合における競合

競合は予測に対して必ず悪い影響を与えるというわけではない．上記分類では，無害な競合は影響を与えないし，建設的競合は予測精度を改善しさえする．予測精度を低下させるのは破壊的競合だけである．しかし Young らの調査では，破壊的競合の方が建設的競合に比べ圧倒的に多いという結果を得ている．つまり，競合は総合的に見れば予測に悪い影響を与えるということができる．

一方 Michaud ら[11)] は，競合の生じる原因によって以下の 3 種類に分類した．

- **初期競合**（**compulsory aliasing**）　初めてテーブルを参照した際に発生する競合
- **容量競合**（**capacity aliasing**）　テーブルの容量不足に起因する競合
- **対立競合**（**conflict aliasing**）　マッピングに起因する競合

彼らはこのうち対立競合を低減する**gskewed** と呼ぶ方式を提案した．gskewed では，異なるマッピング関数を持つ 3 つの予測器を用意する．予測においては，これらの予測器を同時に参照し，得られた予測結果の多数決によって最終的な予測結果とする．予測器のマッピング関数として，ある 1 つの予測器で競合が生じたとしても，残りの 2 つの予測器では起きにくいものを用いる．これにより，対立競合による予測精度低下を減少させた．3 つの予測器を用いるので，与えられたコストの下では 1 つの予測器を用いる場合に比べて容量競合が増加するが，対立競合減少による効果の方が大きいという測定結果を示した．

3. 破壊的競合の削減

競合という問題に対するこれまでのアプローチは主にマッピング関数を工夫して，競合の発生頻度を減少させるというものであった[10),11)]．このアプローチでは，対立競合の削減に効果があるものの，初期競合や容量競合の発生頻度を削減することはできない．そのためこのアプローチのみでは，競合による性能低下を取り去るには不十分である．

しかし，初期競合と容量競合の出現頻度を削減することは難しい．初期競合は初めて PHT が参照される際には必ず発生するので出現を妨げることはできない．

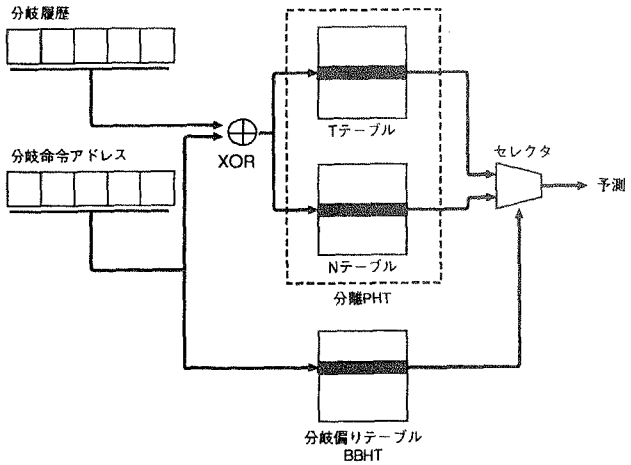


図 3 sgshare 予測機構
Fig. 3 Sgshare predictor.

また，容量競合の出現頻度を下げるには，単純にテーブルの容量を大きくするか，インデクスに用いる分岐アドレスと分岐履歴の組合せ総数を減少させるかである．前者はコストが増加し，後者は予測に利用する情報の減少によって予測精度が低下する．したがって，現実的なハードウェアの制約の下では競合の発生は本質的に不可避であるということができる．

このように競合の発生がある程度は避けられないものならば，より高精度な予測のためには，たとえ競合が起きても予測精度を下げない仕組みが必要である．つまり，予測機構の**競合への耐性**が必要である．このためには，より具体的にいえば，競合が破壊的となりにくい仕組みを構築する必要がある．

ところで，多くの分岐は taken か not-taken のどちらかに偏っている（たとえば文献 16) 参照）．したがって，競合する 2 つの分岐に着目すると，多くの場合それらは，互いに異なる方向に偏った分岐であるか，または，同じ方向に偏った分岐となっている．互いに異なる方向に偏った分岐の間で競合を起こした場合，対応付けられたカウンタは taken の状態と not-taken の状態の間を激しく振動することになり正確な予測を行うことができない．これは破壊的競合である．逆に，同じ方向に偏っている分岐が競合を起こした場合には，カウンタは同じ状態へ遷移するため，競合が発生しない状態と同じく正確な予測を行うことができる．以上より，我々は，競合を引き起こす分岐の偏りの方向が同じになるような仕組みを設ければ，予測精度を悪化させる破壊的競合を削減できると考えた．

4. sgshare 予測機構

前章では，競合を起こす分岐の偏り方向をそろえることで，破壊的競合が削減できることを述べた．これを実現するために，我々は PHT を分岐方向が taken に偏った分岐用のテーブルと，not-taken に偏った分岐用のテーブルに分離し別々に管理する方式を提案する．以下この方式を**分離型 PHT 方式**（**Separate PHT**）と呼ぶ．PHT を taken に偏った分岐用と not-taken に偏った分岐用とに分離することにより，仮に競合が発生したとしても，同じ分岐方向どうしの競合，すなわち無害な競合となる可能性を高め，逆の分岐方向どうしの情報の競合，つまり破壊的競合となる可能性を下げる．

gshare 予測機構にこの手法を適用したものを我々は**sgshare 予測機構**と呼ぶ．以下では，この予測機構の仕組みの詳細について述べる．

4.1 sgshare 予測機構の構成

図 3 に sgshare 予測機構の構成を示す．同図に示すように sgshare は，**T テーブル**，**N テーブル**，および**分岐偏り履歴テーブル**（**BBHT: Branch Bias History Table**）の 3 つのテーブルで構成する．

T テーブルと N テーブルは，これまでの 2 レベル予測機構における単一の PHT を，taken に偏った分岐用のものと，not-taken に偏った分岐用のものとに分離したものである．どちらも図 2 に示した通常の gshare と同じく，分岐命令アドレスと分岐履歴の XOR をとったものをテーブルのインデックスとする（図 3 では，簡単のため追加アドレスに関しては省略している）．

BBHT は分岐ごとの分岐方向の偏り情報を保持す

る．各エントリはカウンタよりなり，分岐結果に応じて，takenの場合1増加させ，not-takenの場合1減少させる．BBHTより得られる偏り情報は，Tテーブルと Nテーブルのどちらの予測結果を使用するかを選択するために用いる．

予測結果は以下の手順で得る．まず，3つのテーブルを同時に参照する．次に，BBHTの出力結果によりTテーブルの出力か Nテーブルの出力を選択し，これを最終的な予測結果とする．BBHTより takenに偏った分岐であることが分かった場合，Tテーブルの出力を予測とする．逆に，not-takenに偏った分岐であることが分かった場合，Nテーブルの出力を予測とする．

テーブルの更新は次のようにして行う．BBHTは，予測の際に参照されたカウンタを分岐結果に従い，前述のように増減させる．PHTは，Tテーブルと Nテーブルの両方のカウンタが参照されているが，予測の際に BBHTが選択したテーブルのカウンタのみ更新する．これは，Tテーブルに takenに偏っている分岐の情報だけを保持し，Nテーブルに not-takenに偏っている分岐の情報だけを保持するようにするためである．

4.2 sgshare予測機構のコスト配分

sgshareを実現する際，与えられたハードウェア・コストを BBHTと PHTの間で最適に配分する構成を見出す必要がある．すべての場合を尽くして測定を行い最適な構成を見つけるには，測定に非常に時間がかかる．そこで本節では，測定方法の指針を得るために，どのようにコストを配分するのがよいかを定性的に議論する．

まず我々は，BBHTには PHTに優先してコストを配分すべきと考えた．これは，BBHTは PHTでの破壊的競合を抑えるための分岐のマッピングを行うので，BBHTで競合が起きると，マッピングが意図したように行われない頻度が増加し，破壊的競合抑制の効果が低下すると考えられるからである．

次に BBHTの構成について考える．BBHTのコストは，エントリ数とカウンタ・ビット数の積によって決まる．したがって，これらの間にトレードオフが存在する．これに関しては，カウンタ・ビット数よりエントリ数の方を優先しコストを割り当てるべきと考えられる．なぜなら，先に述べたように BBHTでの競合は sgshareの破壊的競合を抑えるという効果を著しく低下させると考えられるので，競合が生じないようにエントリ数は十分でなければならないからである．

分岐方向の偏りをとらえるには，カウンタ・ベースの分岐予測の研究[3]から推測すると，2ビット程度あればよいと考えられる．しかしコスト制約が非常に厳しく，十分なエントリ数が得られない場合は，カウンタのビット数を1ビットにする選択肢もある．逆にコスト制約が緩く，十分なエントリ数が確保できるならば，カウンタのビット数を多くし，より精度の高い分岐偏り情報を得るという選択肢も存在する．

5. 性能評価

本章では sgshareを評価する．最初に評価環境について述べる．次に，構成のためのパラメータを変化させ性能を評価する．

sgshareに関するパラメータを以下に示す．

- BBHTのエントリ数
- BBHTのカウンタ・ビット数
- 分岐履歴長
- 追加アドレス長

4.2節での議論に従い，まず BBHTに関するパラメータを変化させ評価し，BBHTの構成を決定する．そのうえで，分岐履歴と追加アドレスの長さを変化させ，従来の gshareと比較することにより PHTを分離したことの効果を評価する．

5.1 評価環境

トレース駆動シミュレーションにより評価を行った．ベンチマーク・プログラムとして IBS-Ultrix[17]を用いた．**表1**に各ベンチマークの説明と，これらに含まれる動的な分岐数と静的な分岐数を示す．これらのベンチマークの各トレースは DECstation （MIPS R3000）にハードウェア・モニタをつなぐことで，OS（Ultrix 3.1）の実行トレースも含めて採取された．そのため，このベンチマークは従来用いられてきた SPEC等から採取したアプリケーションのみの実行トレースに比べ，より実際の環境に近い実行トレースを提供できるとされている．このため，最近の分岐予測の研究で多く用いられている[9)~11),15)]．

5.2 BBHTの構成

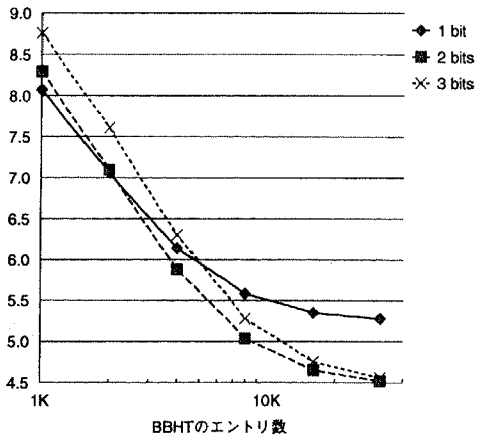
本節では，PHTに関するパラメータを固定し，BBHTのエントリ数とカウンタ・ビット数を変化させ性能に与える影響を調査する．これにより BBHTの最適なコスト配分を決定する．

図4に BBHTのエントリ数とカウンタ・ビット数を変化させたときの分岐予測ミス率の全ベンチマークでの平均を示す．BBHTが予測精度に与える影響が大きく出るように，競合が頻繁に起こる状況下で測定した．具体的には，PHTの履歴を10ビット，追加アドレスを0ビットとした．

図4からエントリ数が2K以下とそれほど多くない

表 1 IBS-Ultrix ベンチマーク・プログラム
Table 1 IBS-Ultrix benchmark programs.

プログラム	分岐数		説明
	動的	静的	
groff	11,568,181	5,634	nroffのGNU C++版 (version 1.09)
gs	14,288,742	10,935	Ghostscript (version 2.4.1) テキストと画像を含む1ページの表示
jpeg_play	20,926,069	6,716	xloadimage (version 3.0) 2ページのJPEG画像の表示
mpeg_play	8,109,029	4,752	mpeg_play (version 2.0) 圧縮ファイルからの85フレームの表示
nroff	21,368,201	4,480	Ultrix 3.1版nroff
real_gcc	13,940,672	16,716	GNU Cコンパイラ (version 2.6)
sdet	5,221,321	4,583	マルチプロセス性能評価 SPEC SDMベンチマークの1つ
verilog	5,692,823	3,918	Verilog-XL (version 1.6b) マイクロプロセッサのシミュレーション
video_play	5,175,630	3,977	mpeg_playの修正版 圧縮されていないファイルからの610フレームの表示



BBHT のエントリ数とカウンタ・ビット数の予測精度に与える影響

Impact on prediction accuracy with the number of BBHT entries and counter bits.

には，2ビットや3ビットのカウンタを用いるよ
1ビットのものを用いた方が性能が良いことが
る．この理由は，ビット数が少ないほうがBBHT
ォーム・アップ時間が短いためである．つまり，
エントリにそれまで割り当てられていた分岐とは
る分岐が割り当てられたとき，そのエントリのカ
タが新しく割り当てられた分岐に関して十分な情
蓄えるまでの時間は，ビット数が少ないほど速い．
えば，3ビット場合，カウンタ値が0のときに逆
りを持つ分岐が新しく割り当てられたとすると，
り値を超えるまでにはその分岐は少なくとも4回
回）実行されなければならない．これに対して
1ビットなら，1回ですむ．エントリ数が2K以下のときは，カウンタのビット数を増加させることによる情報の確かさの向上より，競合からのウォーム・アップ時間が重要となり，ビット数が少ない方が性能が良いという結果となっている．

BBHTを1ビット・カウンタで構成する場合は，8Kエントリ程度で性能が飽和する．このときのBBHTのハードウェア量は1Kバイトである．

図4の測定結果より，BBHTにより多くのハードウェアが割けるならば，BBHTを2ビット・カウンタで構成し，より正確な情報を蓄えることが性能改善に貢献することが分かる．この場合には，16Kエントリ程度で性能は飽和する．この時のBBHTのハードウェア量は4Kバイトである．

BBHTのカウンタ・ビット数を3ビットに増やしても，測定した32Kエントリまででは2ビットで構成したものの性能を上回ることができなかった．32Kエントリのときのハードウェア量は12Kバイトにもなることから，3ビット以上のカウンタを用いることは良い選択ではないことが分かる．

以上のことから，低コストの予測機構においては1ビットのBBHTを，大規模な予測機構では2ビットのBBHTを用いることが適当であることが分かった．以降の性能評価では，BBHTとして8Kエントリで1ビット・カウンタの構成と，16Kエントリで2ビット・カウンタの構成についてのみ評価する．

5.3 PHT 分離の効果

この節では，PHTをTテーブルとNテーブルに分離することで，予測精度と競合にどのような効果が

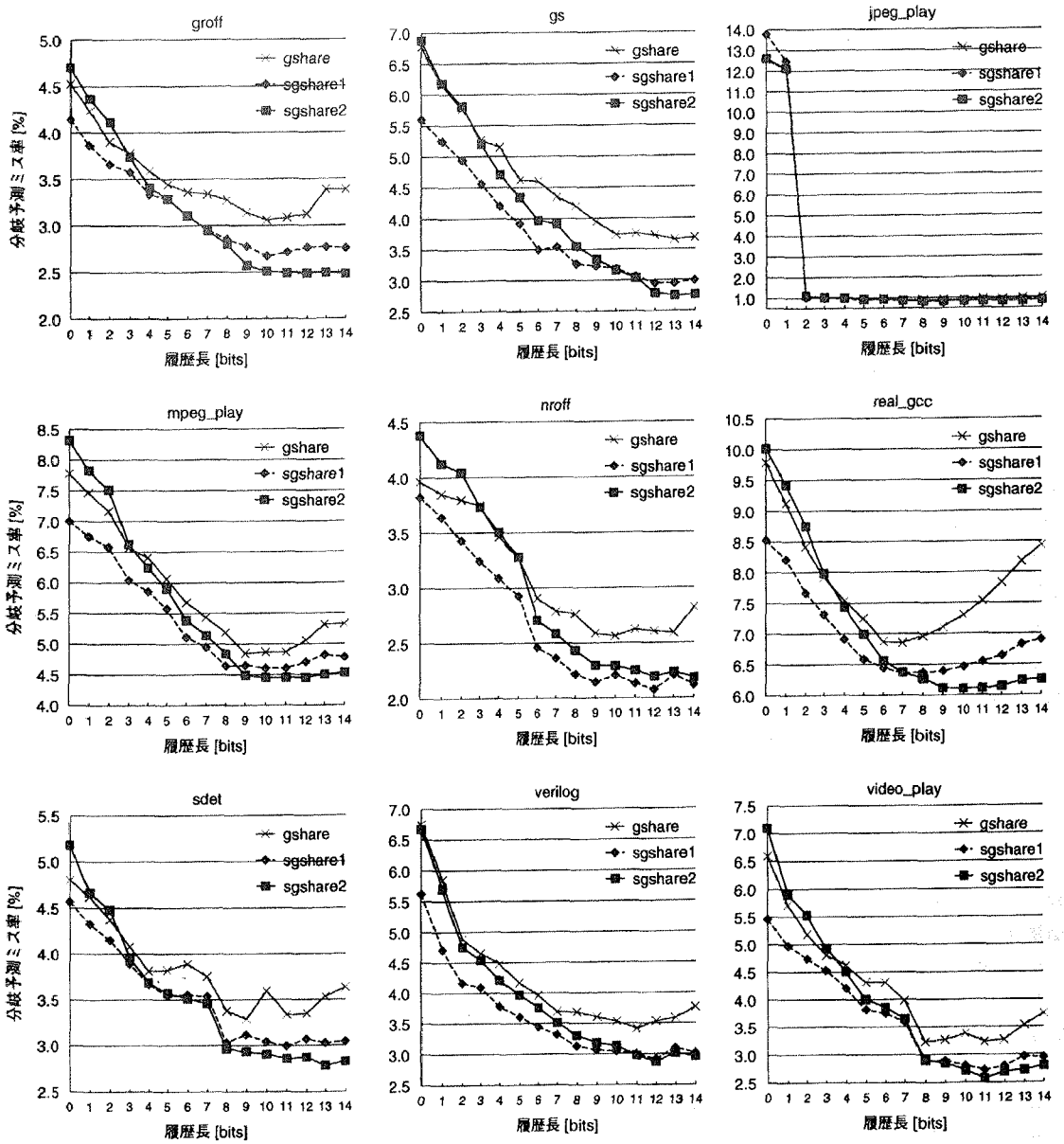


図 5 32 K エントリの PHT での履歴長と予測精度
Fig. 5 Prediction accuracy vs. history length with 32K-entry PHT.

現れるかを調査する．PHT の大きさを 32 K エントリとし，以下の 3 つのモデルについて測定した．

- **gshare**：gshare 予測機構
- **sgshare1**：BBHT を 1 ビット × 8 K エントリで構成した sgshare 予測機構
- **sgshare2**：BBHT を 2 ビット × 16 K エントリで構成した sgshare 予測機構

5.3.1 履歴長と予測精度の関係

図 5 に，履歴長を変化させ各予測機構について予測精度を測定した結果をベンチマークごとに示す．横軸は履歴長で，縦軸は予測ミス率である．PHT のエントリ数は 32 K で固定なので，履歴長を 1 増加させた場合，追加アドレス長は 1 減らしている．

まず第 1 に，sgshare1，sgshare2 ともに，履歴長が非常に短い場合を除いて，すべてのベンチマークで gshare より良い予測精度を示している．jpeg_play はわずかな履歴を用いるだけで，99%という非常に高い予測精度を得ることができるので改善の余地がほとんどない．しかしその他のベンチマークでは，各予測機構での最善の履歴長と追加アドレス長の組合せにおいて，

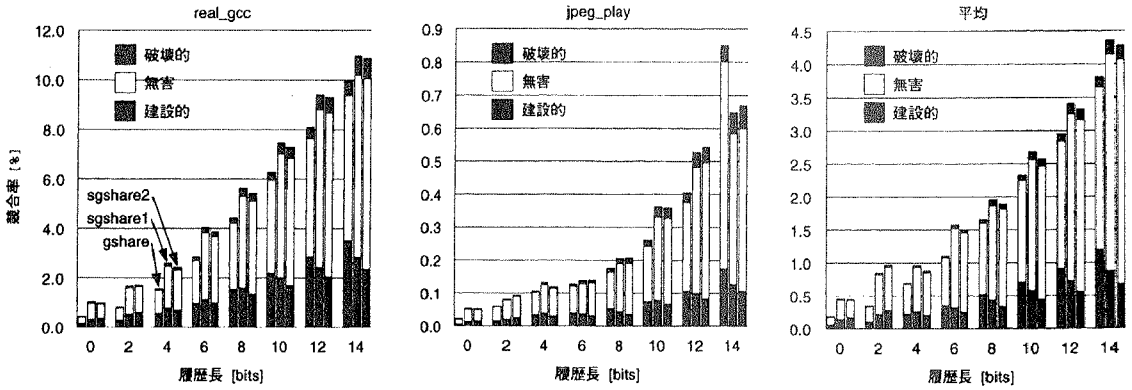


図 6 競合率とその内訳
Fig. 6 Aliasing rate and its breakdown.

gshare に比べ sgshare1 は 0.24～0.71%，sgshare2 は 0.37～0.91%性能が良い☆．

どの予測機構も履歴長を延ばせば，予測に利用する情報が増加するためあるところまでは予測精度が改善する．しかし gshare の場合，あまり長くするとほとんどのベンチマークで予測精度が悪化する．これは履歴を延ばすことにより，1 つの静的分岐に対して対応付けられる PHT のエントリ数が増加することにより競合が増加するからである．履歴を長くし過ぎると，競合による負の効果が，多くの情報を予測に用いる正の効果を上回り，結果として予測精度が低下する．

この現象は real_gcc において著しい．図 5 を見れば分かるように履歴を 7 ビットより長くすると急速に予測精度が悪化する．real_gcc は表 1 に示したように他のベンチマークに比べて静的分岐数が非常に多い．そのため PHT での競合が起きやすく，競合による負の効果が顕著に現れる．

このように競合により性能が著しく低下するという性質は予測機構として望ましくない．なぜならば，プロセッサに予測機構を組み込む際，何らかの評価の下に妥当なパラメータに決めたとしても，設計段階で想定したプログラム（これは評価に使ったプログラムの平均であろうが）より静的分岐数の非常に多いプログラムが実行された場合，著しく性能が低下してしまうからである．望ましい性質は，たとえ設計段階で想定しなかったような静的分岐数の多いプログラムが実行されたとしても，予測精度が大きく下がらないことである．つまり，予測機構には**競合への耐性**が要求される．

図 5 から分かるように，sgshare は gshare と異なり競合への耐性が非常に強い．履歴を長くし競合が頻発する状況となっても，予測精度に与える負の影響は小さい．たとえば，real_gcc では，gshare は履歴長を最適なものから 14 ビットに増加させると予測精度は 1.58%も悪化するが，sgshare1 では 0.54%，sgshare2 ではわずか 0.15%しか悪化しない．

5.3.2 競合内容

図 6 は，各予測機構の**競合率**とその種類の内訳を示したものである．競合率とは，競合の発生回数を実行した分岐の数で割って算出したもので，競合がどのような頻度で発生したかを示す．図 6 の縦軸はその競合率を示す．横軸は履歴長である．0 から 14 までの各履歴長に対する 3 つの棒グラフは，左から gshare，sgshare1，sgshare2 に関するものである．各棒グラフは競合の種類で 3 つの部分に分解している．それぞれ下から，破壊的競合，無害な競合，建設的競合の発生頻度である．競合の分類は，実際の予測機構と同一履歴長で PHT の大きさが無限で競合のない予測機構による予測結果を，実際の予測機構の予測結果と比較することにより行った．図 6 には紙面を節約するため，最も競合を起こす real_gcc と，最も競合を起こさない jpeg_play と，全ベンチマークの平均の 3 つの結果を示した．他のベンチマークの結果は，これらと類似した傾向を持ち，数値は real_gcc と jpeg_play 間のものとなる．

図 6 から分かるように，real_gcc では競合は平均の倍もの頻度で起こっている．14 履歴のときの競合率は約 10%もあり，競合が予測精度に大きく影響している

☆ 一般にパーセントで 2 つの数値を比較するときは，それらの比をいう．本論文でも断りのないかぎり数値をパーセントで表現するときは，比をいうこととする．ただし，予測精度の比較では，たとえば，予測精度 A_1%の機構は予測精度 A_2%の機構より A_3%良いと表現する場合は，予測精度の差 $(A_1 - A_2)$ が A_3 であることを意味する．

ことを裏付けている．一方 jpeg_play では，競合は平均の 1/5～1/10 しかない．14 履歴のときでも 1%もなく，競合による予測への影響は小さいことが確認できる．

履歴長にかかわらず sgshare は，競合の発生頻度そのものは，ベンチマーク平均で 0.3～0.5%程度増加している☆．これは，sgshare では BBHT の出力する分岐の偏り情報が変化することによってまったく同じアドレスと分岐履歴を持つものであっても両方のテーブルへ対応付けられることがあるためである．PHT に 2 重に情報を蓄えれることがあるので，競合の総数は増加する．

競合自体は増加しているものの，予測に悪影響を与える破壊的競合については，ベンチマーク平均で見ると，履歴長 6 ビット以上ではむしろ減少している．特に履歴が長く，競合の発生頻度が高くなったときの減少率は大きい．履歴長 14 ビットのとき，sgshare1 は gshare での破壊的競合の 27.2%を，sgshare2 は 43.4%を削減している．

履歴長が 6 ビット以下と少なく競合自体があまり発生することのない場合は，sgshare では破壊的競合はわずかに増加している．しかし，前節，図 5 での予測性能の結果から見て分かるように，履歴がきわめて短いときは高い予測精度を達成することができないので，このような結果は重要ではない．

また，予測に好影響を与える建設的競合は，sgshare ではすべての履歴長において gshare より高い．これは競合全体の発生頻度の増加によるものである．

以上より，PHT を分割したことで破壊的競合を無害な競合に変えるという仕組みは有効に機能していることが分かる．

5.4 コスト性能比の評価

本節では，sgshare と gshare のコスト性能比について調査する．前節と同様に，gshare，sgshare1，sgshare2 について測定を行う．

PHT の大きさを変化させ，各 PHT に対し全ベンチマークでの平均予測精度が最も高くなる履歴と追加アドレスの配分を測定によって求めた．**図 7** に，この最適な配分での各ベンチマークにおける予測精度を示す．ただし，紙面の節約のため jpeg_play を省略し，代わりに全ベンチマークでの平均値を載せた．jpeg_play は図 5 に示したように，わずかな履歴を予測に用いるだけで非常に高い予測精度を示し，コスト依存性がほとんどない．このため本研究にとっての興味はほとんどない．ただし平均の測定結果には含めている．

図 7 において，横軸のハードウェア・コストとは，各予測機構を構成するテーブルの大きさ（単位：K バイト）である．つまり，gshare の場合 PHT の大きさであり，sgshare1 および sgshare2 の場合，PHT と BBHT の大きさの合計である．

図 7 から，ハードウェア・コストが 2 K バイト以下のときには，多くのベンチマークで sgshare1 は同規模の gshare の性能に劣っている．同様に，ハードウェア・コストが 6 K バイト（sgshare2 の測定での最少のコスト）のときには，半数のベンチマークで sgshare2 は同規模の gshare の性能に劣っている．8 K エントリの 1 ビット・カウンタで構成した BBHT は 1 K バイト，16 K エントリの 2 ビット・カウンタで構成した場合にはこれだけで 4 K バイトのハードウェア量を要する．このため，小規模な構成のときでは，破壊的競合削減の効果よりも BBHT を加えたことによるコスト増加の影響が大きく，同一コストでは予測精度が向上しない．

逆に費やすことのできるハードウェア・コストが大きくなると，BBHT を追加することによるコスト増は相対的に小さなものとなる．この場合，破壊的競合が削減された効果により同一コストにおいても予測精度が向上する．ほぼ同等のコストで比較すると，たとえば，8 K バイトの gshare と 9 K バイトの sgshare1 では，最大 0.89%（gs），ベンチマーク平均で 0.45%予測精度が向上する．コストの大きなところでは sgshare2 がやや有利となり，128 K バイトの gshare と 132 K バイトの sgshare2 では，最大 0.82%（real_gcc），ベンチマーク平均で 0.41%予測精度が向上する．

表 2 に，図 7 の「平均」のグラフの各測定点における履歴長と追加アドレス長の組合せを示す．この表から，sgshare は同程度の予測精度を持つ gshare と比較して，追加アドレス長を短くできることが分かる．たとえば，予測ミス率約 3.2%を実現するのに gshare，sgshare1 ともに 11 の履歴長を要しているが，追加アドレス長は gshare が 5 であるのに対し，sgshare1 では 2 ビット少ない 3 で同程度の予測精度を達成している．同様に予測ミス率約 3.0%を達成するには，gshare，sgshare2 ともに 12 の履歴長を要しているが，追加アドレス長は gshare では 5，sgshare2 では 2 である．これは，sgshare では破壊的競合が削減された結果，gshare と同じ履歴長を利用して同程度の予測精度を達成するために必要とする PHT のハードウェア・コストが 1/4～1/8 で済むことを意味する．このため前述の 2 つの例では，それぞれ 43.8%（(1 − 9 Kbytes/16 Kbytes)×100），

☆ これらの数値は競合率の差である．

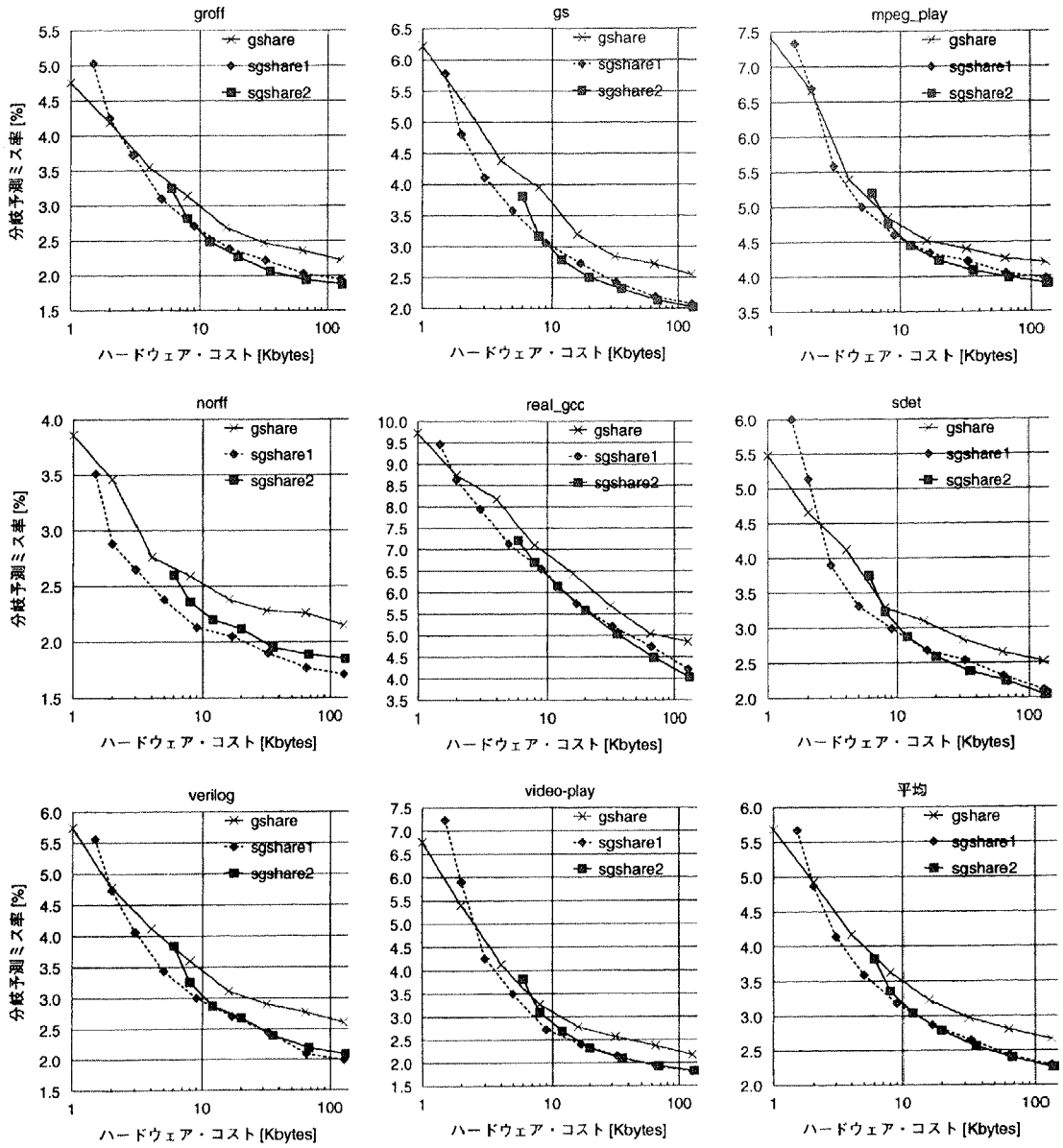


図 7 ハードウェア・コスト対予測精度
Fig. 7 Hardware cost vs. prediction accuracy.

62.5% ((1 − 12 Kbytes/32 Kbytes)×100) も少ないハードウェア量で gshare と同程度の予測精度を達成している．このことから，sgshare は非常にコスト性能比の高い予測機構であることが分かる．

6. プロセッサの性能に与える効果

本章では，分岐予測精度の向上が現在および将来のプロセッサにおいてどの程度性能向上に貢献するかについて評価を行う．

評価は，トレース駆動のシミュレーションを用いて行った．ベンチマークには IBS-Ultrix の 9 本のベンチマークのそれぞれ 1000 万命令のトレースを用いた．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 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 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 40 サイクル
- **Wide&Deep** モデル：32 命令フェッチ，分岐予

表 2 分岐履歴と追加アドレスの配分
Table 2 Distribution of the branch history and the extra address.

予測機構	PHTのインデクス		コスト [Kbytes]	予測ミス率 [%]
	履歴長	追加アドレス長		
gshare	4	8	1	5.67
	5	8	2	4.92
	9	5	4	4.17
	9	6	8	3.63
	11	5	16	3.23
	12	5	32	2.97
	12	6	64	2.81
	16	3	128	2.67
sgshare1	5	5	1.5	5.67
	6	5	2	4.87
	9	3	3	4.14
	9	4	5	3.59
	11	3	9	3.18
	11	4	17	2.87
	14	2	33	2.65
	17	0	65	2.43
	18	0	129	2.29
sgshare2	10	2	6	3.82
	11	2	8	3.36
	12	2	12	3.04
	13	2	20	2.79
	16	0	36	2.56
	17	0	68	2.40
	18	0	132	2.26

測ミスペナルティ40サイクル

Current モデルは現在のハイエンドのプロセッサ[1),2)]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．

評価した n 命令フェッチのマシンは，現在の PC から分岐予測が誤るまで最大 n 命令フェッチし，リザベーション・ステーションに登録し，実行可能になった命令から out-of-order で実行する．機能ユニットは，ALU，分岐，ロード/ストア，浮動小数点ユニット等からなり，それぞれ n 個持つ．リザベーション・ステーション，リオーダ・バッファ，キャッシュ，BTB は十分大きくし，これらによってパイプラインが停止することがないようにした．

各モデルにおいて，分岐予測機構として8Kバイトのコストの gshare を組み込んだものと，ほぼ同等のコストである9Kバイトの sgshare1 を組み込んだものとを比較した．これらの予測機構のパラメータは前節までの測定結果に基づき最適に設定した．

図8に，gshare を用いた場合に対する sgshare1 を用いた場合の各モデルの実行サイクル数による速度向上率を示す．いずれのモデルにおいても，予測精度が向上した結果，性能が向上している．jpeg_play ではすでに予測精度が 99%に達しており，sgshare による予測精度の向上は小さい．このため，どのモデルについても jpeg_play の性能向上は小さい．しかし他のベンチマークでは大きな性能向上を達成していることが分かる．Current モデルでは 4.0～13.3%性能が向上している．パイプライン段数が深い Deep モデル，Wide&Deep モデルでは，9.8～27.2%もの性能向上を達成している．

9Kバイトの sgshare1 の予測ミス率はベンチマーク平均で 3.2%である（図7参照）．この予測ミス率は十分に小さいように感じられるが，依然としてプ

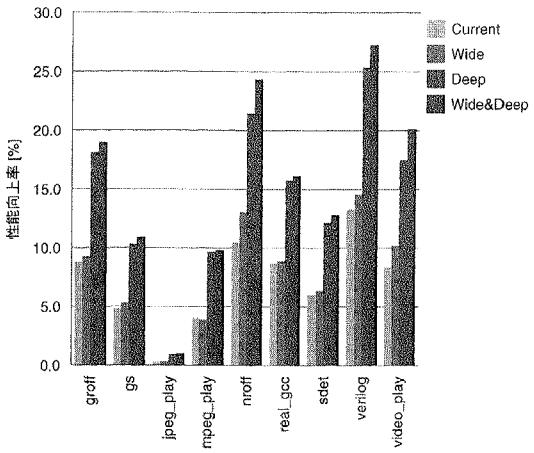


図 8 プロセッサ性能への効果
Fig. 8 Impact on processor performance.

表 3 分岐予測率 100%による性能向上率
Table 3 Speedup with perfect branch prediction.

プログラム	性能向上率 [%]
groff	51.9
gs	48.9
jpeg_play	3.0
mpeg_play	26.8
nroff	61.5
real_gcc	75.7
sdet	39.9
verilog	66.5
video_play	47.6

ロセッサ性能に多大な影響を及ぼしている．表 3 に gshare を組み込んだ Current モデルに対し，分岐予測が 100%成功した場合の性能向上率を示す．性能向上率は jpeg_play を除いて約 26.8～75.7%もある．分岐予測精度はさらなる改善が必要であることが分かる．

7. まとめ

競合はハードウェア資源の限られた 2 レベル予測機構の性能を制限する大きな原因である．扱う分岐アドレスと分岐履歴の組合せ数に対し，現実的には小さなテーブルしか用意できないので，競合は本質的に不可避である．これに対して本文では，分岐方向の偏りを考慮した PHT の運用を行うことで破壊的競合が削減できることを示し，この考えに基づく sgshare と呼ぶ予測機構を提案した．sgshare は taken に偏った分岐と not-taken に偏った分岐を別々の PHT に対応付けることで破壊的競合を削減する．この結果，gshare に比べ競合による予測精度への悪影響を減少させることができ，同じ大きさの PHT でより長い履歴を活用することができる．

シミュレーションにより性能評価を行った結果，ハードウェア・コスト 8K バイトの gshare に対して，ほぼ同等のコストでベンチマーク平均で 0.45%，最大で 0.89%予測精度が向上することを確認した．この予測精度の向上により，今日のスーパースカラ・マシンで 0.3～13.3%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った将来のスーパースカラ・マシンでは 0.9～27.2%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，文部省科学研究費補助金基盤研究（C）「広域命令レベル並列によるマイクロプロセッサの高性能化に関する研究」（課題番号 1068034）および財団法人堀情報科学振興財団助成「広域命令レベル並列性を利用するコンピュータ・アーキテクチャとコンパイラに関する研究」の支援により行った．

参考文献

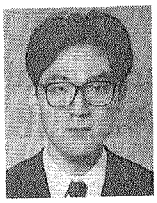
1) Gwennap, L.: Intel's P6 Uses Decoupled Superscalar Design, *Microprocessor Report*, Vol.9, No.2, pp.9–15 (Feb. 1995).
2) Gwennap, L.: Digital 21264 Sets New Standard, *Microprocessor Report*, Vol.10, No.14, pp.11–16 (Oct. 1996).
3) Lee, J.K.F. and Smith, A.J.: Branch Prediction Strategies and Branch Target Buffer Design, *Computer*, Vol.17, No.1 (Jan. 1984).
4) Yeh, T-Y. and Patt, Y.: Two-Level Adaptive Branch Prediction, *Proc. 24th Annual International Symposium and Workshop on Microarchitecture*, pp.55–61 (Nov. 1991).
5) Pan, S-T., So, K. and Rahmeh, J.T.: Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation, *Proc. 5th Int. Conf. on Architectural Support for Programming Languages and Operating Systems*, pp.76–84 (Oct. 1992).
6) Yeh, T-Y. and Patt, Y.: A Comparison of Dynamic Branch Predictors that use Two Levels of Branch History, *Proc. 20th Annual International Symposium on Computer Architecture*, pp.257–266 (May 1993).
7) McFarling, S.: Combining Branch Predictors, WRL Technical Note, TN-36, Digital Equipment Corporation (June 1993).
8) Young, C., Gloy, N. and Smith, M.D.: A Comparative Analysis of Schemes for Correlated Branch Prediction, *Proc. 22nd Annual International Symposium on Computer Architecture*, pp.276–286 (May 1995).

9) Gloy, N., Young, C., Chen, J.B. and Smith, M.D.: An Analysis of Dynamic Branch Prediction Schemes on System Workloads, *Proc. 23rd Annual International Symposium on Computer Architecture*, pp.12–21 (May 1996).
10) Sechrest, S., Lee, C-C. and Mudge, T.: Correlation and Aliasing in Dynamic Branch Predictors, *Proc. 23rd Annual International Symposium on Computer Architecture*, pp.22–32 (May 1996).
11) Michaud, P., Seznee, A. and Uhlig, R.: Trading Conflict and Capacity Aliasing in Conditional Branch Predictors, *Proc. 24th Annual International Symposium on Computer Architecture*, pp.292–303 (May 1997).
12) 野口良太，森 敦司，小林良太郎，安藤秀樹，島田俊夫：履歴情報の競合の悪影響を緩和した分岐予測機構の予備的評価，電気関係学会東海支部連合大会論文集，p.317 (Sep. 1997).
13) 野口良太，森 敦司，小林良太郎，安藤秀樹，島田俊夫：競合による予測精度低下を緩和する分岐予測機構，情報処理学会研究報告，97-ARC-127, pp.63–70 (Dec. 1997).
14) 野口良太，森 敦司，小林良太郎，安藤秀樹，島田俊夫：分離型パターン履歴表による分岐予測機構の競合耐性の改善，1998 年並列処理シンポジウム JSPP'98, pp.7–14 (June 1998).
15) Lee, C-C., Chen, I-C.K. and Mudge, T.N.: The Bi-Mode Branch Predictor, *Proc. MICRO-30*, pp.4–13 (Dec. 1997).
16) Chang, P-Y., Hao, E., Yeh, T-Y. and Patt, Y.: Branch Classification: A New Mechanism for Improving Branch Predictor Performance, *Proc. MICRO-27*, pp.22–31 (Nov. 1994).
17) Uhlig, R., Nagle, D., Mudge, T., Sechrest, S. and Emer, J.: Instruction Fetching: Coping with Code Bloat, *Proc. 22nd Annual International Symposium on Computer Architecture*, pp.345–356 (May 1995).

(平成 10 年 8 月 27 日受付)
(平成 11 年 1 月 8 日採録)

野口 良太（正会員）

1996 年名古屋大学工学部電子情報学科卒業．1998 年，名古屋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（株）日立製作所に入社，汎用コンピュータ事業部に所属．現在に至る．在学中，分岐予測の研究に従事．



森 敦司（学生会員）

1975 年生．1997 年名古屋大学工学部電子情報学科卒業．1999 年同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．同年富士通 VLSI（株）に入社．

小林良太郎（学生会員）

1995 年名古屋大学工学部電子情報学科卒業．1997 年名古屋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．現在，名古屋大学大学院工学研究科電子情報学専攻博士課程後期課程在学中．計算機アーキテクチャの研究に従事．電子情報通信学会会員．

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 年大阪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 年東京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 年より名古屋大学工学部電子情報学科教授．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 度注目発明受賞．工学博士．