社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2007－ARC－174 (35)
2007／8／3

小規模・省電力コアのための省資源分岐予測方式

石井康雄† 平木敬††

本稿では, 実装時の遅延と回路規模を考慮した分岐予測器である, MIN-TAGE 分岐予測器を提案する. この手法ではパターン履歴テーブルのインデックス計算を Enhanced Folded Index 手法を利用して計算することにより, 既存手法と比較して分岐予測にかかる遅延とパイプライン化に伴うコストを削減する. このインデックス計算により効率的な分岐予測器のパイプライン化を実現する.

MIN-TAGE 分岐予測器を 2nd Championship Branch Prediction Infrastructure で評価した. その結果,MIN-TAGE 分岐予測器は TAGE 分岐予測器から構成コストを削減した上で, 分岐予測ミスを 2.6%削減した.

Complextiy-Effective Branch Prediction for Area & Power Efficient Processor

Yasuo Ishii† and Kei Hiraki††

In this paper, we propose MIN-TAGE branch predictor which realizes a complexity-effective implementation and a feasible prediction latency. MIN-TAGE branch predictor employs enhanced folded indexing registers as hash function. The enhanced folded index reduces prediction latency from conventional branch predictors and saves external costs for ahead pipelining. This feature realizes an effective ahead pipelining structure.

We evaluate MIN-TABE branch predictor in 2nd Championship Branch Prediction Infrastructure. MIN-TAGE branch predictor reduces miss predictions per 1000 instructions by 2.6% and saves implementation costs from a costly conventional ahead pipelined TAGE branch predictor.

1. はじめに

マイクロプロセッサでは, 命令間の制御依存によりパイプライン実行が滞ることのないように分岐予測を採用する. 精度の高い分岐予測手法を採用することは, 分岐予測のミスペナルティの処理性能への影響を軽減し, プロセッサの性能を向上させるために不可欠である. この目的のため, さまざまな分岐予測手法が提案されてきた. しかし, 多くの従来手法は大規模なコア上に実装することを想定しており, 十分なメモリと長い予測遅延を許容できることを前提としていた. 近年, プロセッサ設計ではマルチコア化などの影響で小規模かつ電力効率の高い設計が要求され, 従来の分岐予測器はこの要求に適合しなくなっている. また, 組み込み用途の小規模で省電力が要求されるプロセッサにおいても高性能化が進み, パイプライン長が伸び, 分岐予測ミスのペナルティは増加する傾向にある. 従来手法はこれらの要求に応えるものではない.

本稿では, 従来の巨大で複雑な分岐予測器の問題点を解決した MIN-TAGE 分岐予測器を提案する. この手法では分岐予測器の効率的なパイプライン化を実現するようにアルゴリズムを設計する. まず, インデックス計算に Enhanced Folded Index 手法を用いて, 必要な資源と遅延を削減する. 同時に,Enhanced Folded Index 手法の特性を利用した効果的な分岐予測器のパイプライン化を提案する. 以上より, 分岐予測精度を犠牲にすることなく分岐予測器をパイプライン化し, 現代のプロセッサに適した分岐予測器の構成を実現する.

本稿の構成は以下の通りである. 2 章で関連研究に関して述べその問題点を指摘する. 3 章では提案手法である MIN-TAGE 分岐予測器に関して説明をする. 4 章でその評価を行い,5 章でまとめる.

2. 関連研究

2.1 パス情報を利用した分岐予測器

動的分岐予測には古くから過去の分岐の **Taken / NotTaken** の 1 ビット情報のベクトルである分岐履歴情報を用いてきた. しかし, より高い予測精度を実

† 日本電気株式会社 コンピュータ事業部
Computers Division, NEC Corporation
†† 東京大学情報理工学系研究科
Graduate School of Information Science and Technology, the University of Tokyo

現するため, 追加の分岐履歴情報としてパス情報を利用する分岐予測器が注目されている.

Daniel らはパーセプトロンモデルを利用した分岐予測器に過去に実行した分岐命令の命令アドレスを利用する Path-Based Perceptron を提案した[3]. この手法では過去に実行した分岐命令の命令アドレスをパス情報として利用していた. これを改良した Path Trace Branch Predictor(PTBP) では, より詳細なパス情報を用いることで予測精度の向上を実現した[2]. また, 二宮らの提案した A^3PBP 予測器は PTBP で用いたパス情報にローカル分岐履歴を加えることで PTBP を超える分岐予測精度を達成した[6]. これらパーセプトロンモデルに基づく分岐予測器では分岐履歴の各ビットに対して独立したメモリが必要になる. このコストが障害となり, 現在までのところ, 商用のマイクロプロセッサには採用されていない.

パーセプトロンモデルを利用しない分岐予測器でも, パス情報を分岐予測に用いることで高い予測精度を実現する予測器が提案されている. 開発がキャンセルされた Alpha EV8 に搭載される予定だった分岐予測器では分岐命令の命令アドレスの下位の数ビットを数命令分記憶しておき, パス情報として利用する設計だった[8].

2.2 PPM 分岐予測器

Pierre は従来のパターン履歴テーブル (PHT) のインデックス用のハッシュ値とは独立にハッシュ値を計算し, それをタグ情報として利用する分岐予測器を提案した[5]. これは Andre らにより TAGE 分岐予測器として改良されている[7].

TAGE 分岐予測器では複数の予測テーブルを保持し各々が異なる履歴長で 2 つの異なるハッシュ値を生成する. 生成された 2 つのハッシュ値は一方は PHT の読み出しに使い, もう一方は PHT から読み出したタグと比較する. タグの一致が見られた場合にそのテーブルの予測が有効であるとする. 得られた有効な予測結果のうち最長の履歴長でタグが一致したテーブルの予測結果を出力する. この予測手法で TAGE 分岐予測器はパーセプトロンモデルに基づく分岐予測器よりもさらに高い予測性能を実現した.

しかし,TAGE 分岐予測器は回路の遅延が大きくプロセッサへの実装が困難である. この問題を回避するためにパイプライン化などの対策をとる必要があるが, 効率的なパイプライン化の手法がない. 遅延が長くなる原因のひとつは利用する履歴長が長いため, 他方式と比較してハッシュ関数が大きい論理回路で実現されることである. 他の原因は PHT のインデックス用のハッシュ値の計算に命令アドレスを利用していることである. 命令アドレスは分岐予測の直前まで手に入らない情報であり, パイプライン化の障害となる.

2.3 分岐予測時の遅延を考慮した分岐予測手法

Pierre は分岐予測に長い分岐履歴を用いる際に, そのハッシュ関数が大きい論理回路で実現されることに注目し, ハッシュ値の計算を小規模かつ省資源な回路で実現する Folded Index 手法を提案した[5]. 従来は分岐履歴情報のみのハッシュ値の計算に利用されていたが,Enhanced Folded Index 手法でパス情報を含むように拡張された[1].

Andre らは複雑な分岐予測器をパイプライン化することにより, 分岐予測器の遅延がプロセッサのクリティカルパスに入らないようにする Ahead Pipelining 手法を提案した[9]. しかし, 分岐予測器の予測結果を計算する回路の複製と複製を保持するためのパイプラインレジスタを要求するため回路規模を増大させる. また, パイプライン化のために分岐予測を前倒して開始するため, 分岐予測の予測精度を下げてしまうという問題もある.

3. MIN-TAGE Branch Predictor

前章で示したとおり, 高い予測精度を実現するためには,TAGE 分岐予測器のように複雑で遅延の大きい分岐予測手法を採用する必要があった. クリティカルパスに入ることを回避するために単純な Ahead Pipelining を用いると, 予測精度が下がり, 結果の複製やパイプラインレジスタにより回路規模も巨大になる. パイプライン化による回路規模の増大と遅延の悪化の原因は分岐予測アルゴリズムがパイプライン化を前提としていないことにある.

本稿では, この問題を解決する MIN-TAGE 分岐予測器を提案する. MIN-TAGE 分岐予測器の全体構成を**図 1** に示す. この予測器は 4 つのテーブルと 6 つの Enhanced Folded Index レジスタ, タグ比較回路, および, セレクタからなる. TAGE 分岐予測器はテーブルの数が多いほど予測精度を高めやすいが, 多数のテーブルを持つことは最終段の遅延増加と制御回路の増加を招く. 本稿ではテーブル数の増加を考慮して, 分岐予測器の RAM 数を 4 で制限した. この 4 つの RAM のうち 3 つ (PHT) では分岐履歴を利用した分岐予測結果の生成を行い, 残りのひとつ (BIM) では分岐命令の命令アドレスのみから予測を行う. 以下では MIN-TAGE 分岐予測器の構成を述べる.

3.1 PHT の構成

従来の TAGE 分岐予測器のクリティカルパスは PHT のインデックス用のハッシュ値の計算,PHT へのアクセス, および, 予測結果の演算回路からなる. 現代の長いパイプライン構造を採用したプロセッサでは, この遅延を 1 サイクルに収めることは困難であり, パイプライン化を施す必要がある.

MIN-TAGE 分岐予測器の PHT はプロセッサ上での実装を想定して設計する. この指標として,PHT に用いる RAM はレジスタファイルと同程度かそれ以下の複雑さのものであることを想定する. この RAM に

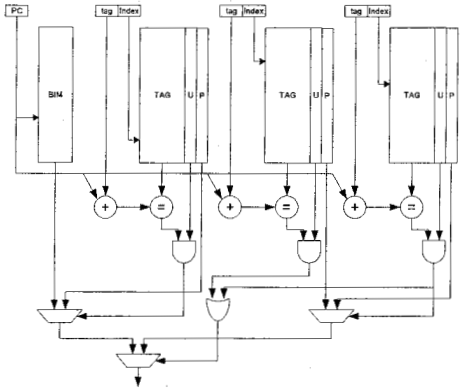

図 1　MIN-TAGE 分岐予測器
Fig. 1　MIN-TAGE Branch Predicor

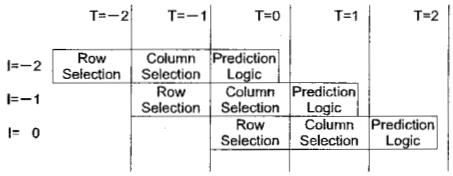

図 2　MIN-TAGE のタイミングチャート
Fig. 2　Timing Chart of MIN-TAGE Branch Predicor

対して**図 2** に示すような行選択と列選択に分割された 2 ステージのアクセスを行う．これを**図 3** に示した回路構成でパイプライン化する．以降では PHT のパイプラインの構成要素に関して詳しく述べる．

3.1.1　パイプライン構造

MIN-TAGE 分岐予測器において各 PHT は 1KB 程度の RAM で実装される．この RAM の読み出しの遅延は大きいため現実的な遅延で分岐予測器を構成するためにはパイプライン化を施す必要がある．

MIN-TAGE 分岐予測器では効率的なパイプライン化を実現するため，この PHT 参照を行選択と列選択に分割する．パイプラインでは第 1 ステージで行選択を行い PHT から 16 組程度の値を選択する．第 2 ステージで列選択を行い行選択で得られた値 (図 3 の Row Data) から読み出す値を決定する．たとえば,PHT が 1024 エントリの 1 ポートの RAM であった場合，行選択で 64 セレクトを行い 16 組の値を選択し，次のステップで 16 セレクトをする．これは MIN-TAGE 分岐予測器で指標とするプロセッサのレジスタファイルの遅延と比較して難しくない．

最終段で得られた分岐予測結果は Enhanced Folded Index レジスタに対して Forwarding される．Forwarding された値は次回以降の分岐予測のために利用される．なお,Alpha EV8 にみられるような予測結果

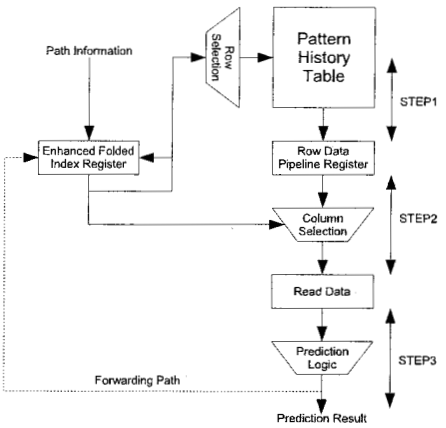

図 3　PHT のパイプライン構成
Fig. 3　Pipeline of PHT Access

の演算回路の複製は MIN-TAGE 分岐予測器では行わない．なぜならばタグ比較が入るため回路の複製による回路規模の増大のデメリットが他の分岐予測器と比較して大きいためである．

このパイプライン化を実現するためには行選択のために効率的に 1 ステージ前倒して読み出せるハッシュ関数が必要になる．以下では，このハッシュ関数に関して述べる．

3.1.2　インデックス計算手法

PHT のインデックス計算に用いるハッシュ関数は**図 4** に示した Enhanced Folded Index レジスタを用いて実現する．この手法ではハッシュ値の出力は常にレジスタの出力なので従来のハッシュ計算にかかっていた遅延を削減できる．図中の Enhanced Folded Index レジスタは 15 ビットの分岐履歴と 10 ビットの実行パス履歴をハッシュする構成例である．

MIN-TAGE では列選択時のハッシュ値を利用して予測を行う．つまり，図 2 の I=0 の予測の場合には T=1 時点でのハッシュ値で PHT の参照をする．Enhanced Folded Index 手法の提案論文ではこれを分岐命令アドレスと排他的論理和 (XOR) をとるようにしていたが，十分なパス情報を含む場合には分岐予測対象の分岐命令の命令アドレスが無くても高い分岐予測精度を達成できることが知られているため，MIN-TAGE 分岐予測器では分岐命令アドレスをハッシュ関数の入力として扱わない．

次に行選択用のアドレスと列選択用のアドレスの選別方法を図 2 の I=0 の分岐命令と図 4 の Enhanced Folded Index レジスタを例に説明をする．まず，この例では PHT から読み出される値は列選択時 (図 2 の T=1) の $R_{\{0,1,2,3,4,5,6,7\}}$ の 8 bit の値で決定する．このときに T=1 の $R_{\{1,3,4,5,6\}}$ の値は 1 サイクル手前

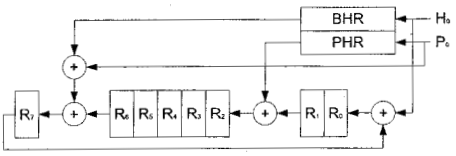

図 4 Enhanced Folded Index レジスタ
Fig. 4 Enhanced Folded Index Register

(図 2 の T=0) で既に $R_{\{0,2,3,4,5\}}$ として決定している. そこで,T=0 の段階で $R_{\{0,2,3,4,5\}}$ の 5 bit を利用して行選択を行い,T=1 で $R_{\{0,2,7\}}$ の 3 bit を用いて列選択を行うと自然な形で PHT の参照を 1 サイクル前倒しできる. このときに $R_{\{0,2,3,4,5\}}$ の情報を T=0 での行選択に利用せずに $R_{\{1,3,4,5,6\}}$ として T=1 の列選択に利用することは出来る. しかし,$R_{\{0,2,7\}}$ の情報は前倒しが出来ないため, 必ず列選択に用いる必要がある.

上記のインデックス計算手法により PHT のアクセスを予測精度を損なうことなく列選択の時点でのハッシュ値で予測の生成ができるようになる. ハッシュ値の決定を後のステージにずらすことにより, よりターゲットの分岐命令に近い地点での情報が手に入るため予測精度に好影響を与える. こうして,Enhanced Folded Index レジスタの特性を利用することで MIN-TAGE 分岐予測器は予測精度を損なうことなくアクセスの 1 サイクルの前倒しを実現している.

N 種類のビット列をハッシュした場合には XOR の出力となるレジスタの個数が $N+1$ 個となるため, 列選択で選択するデータは最低 2^{N+1} 組程度の規模になる. 列選択の規模が大きくなりすぎると遅延悪化の原因となることから,MIN-TAGE 分岐予測器では 3 種類程度のビット列しかハッシュは出来ない. しかし, ハッシュ値の計算に利用するパス情報は既存研究から 2〜3 ビット程度で十分であることが知られている. 以上から,MIN-TAGE 分岐予測器では 1 ビットの分岐履歴情報と 2 ビットの実行パス履歴情報のハッシュ値を RAM のインデックスとして用いる.

3.2 BIM へのアクセス

前節での議論はタグ一致を含む PHT のパイプライン化であり,BIM に関しては異なるアプローチをする. BIM は第 2 ステージからアクセスを始める. このときにはその時点で得られている分岐命令のアドレス, すなわち 1 命令手前の分岐命令アドレスで行選択を行う. 第 3 ステージでは得られた分岐命令アドレスで列選択を行い RAM からの値の読み出しを終わるようにする. この場合に, 第 3 ステージのタグ一致の終了と同時に BIM の値が使えるようにタイミングを調整すると, BIM から読み出した値がタグ一致の終了まで待ち合わせるという状況を回避できる. また,BIM の結果は PHT 以上に命令アドレスやパス実行履歴が重要になるため, BIM のアクセスを分岐予測対象の分岐命令の近くまで待ち合わせることは分岐処理の性能向上に大きく影響する.

3.3 考　　察

MIN-TAGE 分岐予測器では Enhanced Folded Index 手法を用いることにより, 従来のクリティカルパスの始点となっていた予測対象の分岐命令の命令アドレスをハッシュ関数の入力から外す. このハッシュ計算方法により, クリティカルパスを短縮し低遅延での予測を達成している. この実装の影響を予測精度と実装可能性の 2 つの観点から考察した.

3.3.1 予測精度

MIN-TAGE 分岐予測器の予測精度に関する懸念はハッシュ関数の入力にターゲットの分岐命令の命令アドレスを利用していないことである. 予測対象となる分岐命令の命令アドレスは分岐予測器の予測精度にとって重要な要素のひとつであるが, PTBP を始めいくつかの分岐予測器では十分なパス情報を用いることでこの影響を最小限に抑えている.

また、比較対象のタグは PHT からの RAM 読み出しが終わるまでに準備が出来ていれば良いので, こちらでターゲットの分岐命令の命令アドレスを組み込める. TAGE 分岐予測器の場合にはタグの精度が十分であれば, タグの一致によってハッシュ値の精度を保障できる. 従って,MIN-TAGE 分岐予測器は TAGE 分岐予測器と比較して遜色ない予測制度を実現できる.

3.3.2 実装可能性

MIN-TAGE 分岐予測器では PHT のパイプライン化で遅延に余裕がある. さらに, 第 2 ステージの列選択は 16 セレクト程度で実現できるため第 3 ステージの論理を前倒しできる可能性がある. 第 3 ステージでの遅延は最大で 10 ビット程度の値比較と 4 つのデータのセレクトとなる. BIM からの値の読み出し結果は第 2 ステージから始めた場合には第 3 ステージのタグの一致の間に追いつくことが出来るため効率的になる. 遅延の短縮によるパイプライン段数の短縮に加え, 従来の Ahead Pipelining で必要だった回路の複製も存在しないため省規模化も実現できる.

4. 評　　価

MIN-TAGE 分岐予測器を 2nd Championship Branch Prediction Infrastructure によって評価する. この環境では SPEC から選出した 20 のベンチマークに関して各 300 万命令ずつ実行する. このときに得られた分岐予測ミスの回数によって MPKI 値 (1000 命令あたりの分岐予測ミス回数) を計算し, 評価をする.

比較対象として Gshare 予測器, Bimode++予測器[4], 2BC-gskew 予測器, GEHL 予測器, TAGE 予測器を用いた. 5 つの分岐予測器のうち,Gshare 予測器, Bimode++予測器, 2BC-gskew 予測器の 3 方式は

表 1　最適化した各予測器の記憶容量別の履歴長
Table 1　Tuned History Length for Each Predictor

Budget	1KB	2KB	4KB	8KB	16KB
Gshare	12	13	14	15	16
Bimode++	11	12	13	14	15
2BC-gskew	22	24	28	32	36
GEHL	40	48	56	64	72
TAGE	30	40	60	80	120

実現可能性が高い分岐予測器として評価をおこなう．残りの 2 つの手法は，過去の Championship Branch Prediction において，Best Practice Award を獲得するなど高い予測精度を示すことが知られている分岐予測器である．これらは有限のメモリ量をいかに有効に利用できているかの指標として評価する．

以下では予測器の構成に関して説明する．MIN-TAGE 分岐予測器と TAGE 分岐予測器では BIM と 3 つの PHT という形式を採用した．各 PHT は 2 ビットの予測カウンタを持つ．1KB,2KB, および,4KB の構成では 3 つの PHT に用いるタグ長を 3, 5, 8 とした．8KB, および,16KB の構成では 3 つの PHT に用いるタグ長を 13, 14, 15 とした．本来ならば，総メモリ量を連続的に変化させタグ長を少しずつ伸ばすほうが良い結果が得られる傾向にあるが，離散的なメモリ容量において評価を実施するためにこのような値を採用している．

MIN-TAGE 分岐予測器は前章で示したパイプライン構成，つまり,2 サイクルの PHT へのアクセスと 1 サイクルの演算からなる 3 段のパイプライン構成をとる．これに対して従来の TAGE 分岐予測器は 4 段のパイプライン構成をとる．これは PHT のインデックスに利用するハッシュ値の計算に 1 サイクル必要になるためである．

各分岐予測器で利用した分岐履歴長を**表 1** に示す．この履歴長は既存研究で利用された値から取得している．

4.1　評 価 結 果

4.1.1　分岐予測精度

各分岐予測器の予測精度を**図 5** に示す．全ての構成で MIN-TAGE 分岐予測器が従来手法の予測精度を上回ることが分かった．特に，今回ターゲットとする小規模な構成において高い予測精度を示し，1KB 構成の MIN-TAGE 分岐予測器で 16KB の Bimode++予測器と同程度を分岐予測精度を実現し，8KB の 2BC-gskew 予測器や GEHL 予測器と同程度の予測精度を実現した．2KB 構成の MIN-TAGE 分岐予測器は同規模の TAGE 分岐予測器と比較して 2.6%の予測精度向上を実現した．

また,2KB 構成での各ベンチマークごとの分岐予測精度を**図 6** に示す．こちらでも全てのベンチマークで提案手法が既存手法を上回る予測精度を示している．

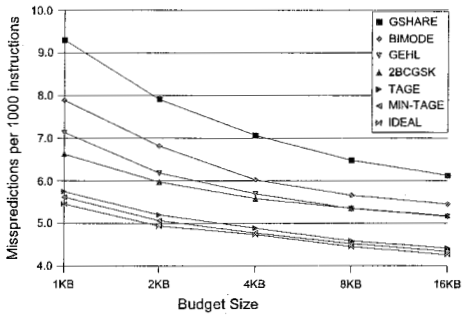

図 5　各分岐予測器の予測ミス率
Fig. 5　Miss prediction rate of each predictor.

4.1.2　実行パス履歴の予測精度への影響

MIN-TAGE 分岐予測器では予測対象の分岐命令の命令アドレスをインデックス計算に用いないことで分岐予測レイテンシを削減する．その代わりに実行パス履歴情報を付加することで高い分岐予測精度を実現する．ここで実行パス履歴情報のビット数を操作した場合の分岐予測精度を**図 7** に示す．このグラフで **No Path** はパス情報を含まない場合の分岐予測精度，**1 bit**,**2 bit** ではそれぞれ 1 bit / 2 bit のパス情報を含んだ場合の分岐予測精度を示す．また,**Address** は MIN-TAGE 分岐予測器で予測対象の分岐命令の命令アドレスをハッシュ関数の入力に利用した場合の予測精度である．

この結果から，小規模な構成では実行パス履歴の予測精度への影響が大きいことが分かる．MIN-TAGE 分岐予測器では 2 ビットの実行パス履歴情報を利用しており，パス情報を利用しない構成から 2KB 構成で予測精度を 3.7%改善する．

次に，実装上の制約を無視し分岐命令の命令アドレスを用いた TAGE 分岐予測器 (**Address**) と MIN-TAGE 分岐予測器 (**2 bit**) で予測精度を比較する．その結果，高々0.5%しか予測精度が悪化しないことが分かった．十分なパス履歴情報を分岐命令の命令アドレスの代替とすることは有効であるといえる．

5.　お わ り に

本稿では，小規模・省電力プロセッサのための省資源な実装を持つ分岐予測器である MIN-TAGE 分岐予測器を提案した．MIN-TAGE 分岐予測器は TAGE 分岐予測器に Enhanced Folded Index 手法を適用することで効率的なパイプライン構成を実現したものである．

MIN-TAGE 分岐予測器では，パイプライン化に伴い各 PHT の読み出しに用いるインデックス計算の入力から分岐予測対象の分岐命令の命令アドレスを除去する．このことでハッシュ値の計算の遅延を削減し

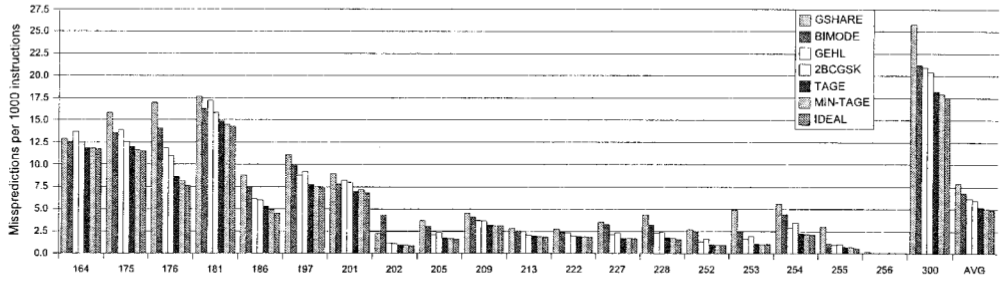

図 6　2KB 構成での各ベンチマークでの予測精度
Fig. 6　Miss prediction rate of each benchmark with a 2KB budget.

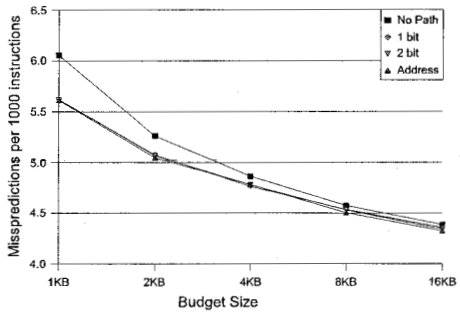

図 7　実行パス履歴情報の予測精度への影響
Fig. 7　Performance impact of path history.

PHT へのアクセスのパイプライン化を実現した．さらに,Enhanced Folded Index 手法を適用することでハードウェアを複雑化することなく実行パス履歴情報をインデックス計算に付加した．実行パス履歴を用いることで，命令アドレスを利用しないことによる予測精度の低下を抑えた．

MIN-TAGE 分岐予測器をシミュレーションを用いて評価をした．評価結果では 2KB のメモリを利用する構成で従来の Ahead Pipelining 手法を適用した TAGE 分岐予測器と比較して 2.6%の分岐予測ミス率の削減をできた．分岐予測手法として MIN-TAGE 分岐予測器を採用することにより，小規模な構成で高精度かつ低レイテンシな分岐予測を行うことが可能になる．

今後は,MIN-TAGE 分岐予測器の採用が実行性能へあたえる影響の詳細な評価や LSI 上に実装した際の消費電力と遅延の詳細な評価を行っていく予定である．

参 考 文 献

1) Yasuo Ishii. Fused two-level branch prediction with ahead calculation. In *The 2nd JILP Championship Branch Prediction Competition(CBP-2)*, 2006.
2) Yasuo Ishii and Kei Hiraki. Path trace branch prediction. In *IPSJ Transactions on Advanced Computing Systems*, volume 47, pages 58–72, 2006.
3) Daniel A. Jiménez. Fast path-based neural branch prediction. In *Proceedings of the 36th Annual IEEE/ACM International Symposium on Microarchitecture*, pages 243–252. IEEE Computer Society, 2003.
4) Kenji Kise, Takahiro Katagiri, Hiroki Honda, and Toshitsugu Yuba. The bimode++ branch predictor. In *IWIA '05: Proceedings of the Innovative Architecture on Future Generation High-Performance Processors and Systems*, pages 19–26, Washington, DC, USA, 2005. IEEE Computer Society.
5) Pierre Michaud. A ppm-like, tag-based predictor. In *The 1st JILP Championship Branch Prediction Competition(CBP-1)*, 2004.
6) Yasuyuki Ninomiya and Koki Abe. Path traced perceptron branch predictor using local history for weight selection. In *The 2nd JILP Championship Branch Prediction Competition(CBP-2)*, 2006.
7) Andre Seznec. A 256 kbits l-tage branch predictor. In *The 2nd JILP Championship Branch Prediction Competition(CBP-2)*, 2006.
8) Andre Seznec, Stephen Felix, Venkata Krishnan, and Yiannakis Sazeides. Design tradeoffs for the alpha ev8 conditional branch predictor. *SIGARCH Comput. Archit. News*, 30(2):295–306, 2002.
9) Andre Seznec and Antony Fraboulet. Effective ahead pipelining of instruction block address generation. In *ISCA '03: Proceedings of the 30th annual international symposium on Computer architecture*, pages 241–252, New York, NY, USA, 2003. ACM Press.