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

2007－ARC－174 (34)
2007／8／3

# 実行パス履歴を有効に利用する低コスト高精度パーセプトロン分岐予測器

二ノ宮 康之† 阿部 公輝†

パーセプトロン分岐予測器は高い予測精度を示すが，構造が複雑で実装コストが大きいという欠点を持つ．これは多数の重みテーブルを利用することに起因する．本稿では新しいパーセプトロン分岐予測器を提案する．これは，(1) 重みテーブルの数を減らすことにより実装コストを削減し，(2) 詳細な実行パス履歴とグローバル履歴の一部をインデックスに利用することにより予測精度を向上させる．使用できる記憶容量を一定とすると，本手法により従来法より低い実装コストで高い予測精度を持つパーセプトロン分岐予測器が実現できる．

# A Low-Cost Highly-Accurate Perceptron Branch Predictor with Effective Use of Execution Path History

Yasuyuki Ninomiya† and Kôki Abe†

Perceptron branch predictors have been extensively studied in recent years in an attempt to reduce misprediction rates. However, it has the disadvantage that the implementation cost is high due to its complex structure. The complexity comes from a large number of weight tables they use. In this paper, we propose a new perceptron branch predictor that reduces the cost by reducing the number of weight tables, and increases the prediction rates by using detailed execution path history and part of global history as the index of weight tables. Given a constant amount of storage available, the proposed scheme enables to increase the prediction accuracy with less implementation costs compared to previous perceptron predictors.

## 1. はじめに

近年，分岐予測器はCPUの性能向上に大きな役割を果たしており，活発に研究が行われている．中でも学習理論の一種であるパーセプトロンを応用したパーセプトロン分岐予測器[1]は単体で高い予測精度を示すことで注目されている．しかし，パーセプトロン分岐予測器は複雑な構造を持つため，回路量が大きいという欠点がある．マルチコア化の進むCPUに実装するためには精度の向上とともに実装コストの削減が必要である．

本稿では新しいパーセプトロン分岐予測器を提案する．この予測器は従来のパーセプトロン分岐予測器に比べ，(1) 重みテーブルの数を減らすことにより実装コストを削減し，(2) 詳細な実行パス履歴とグローバル履歴の一部をインデックスに利用することにより予測精度を向上させる．使用できる記憶容量を一定とすると，本手法により従来法より低い実装コストで高い予測精度を持つパーセプトロン分岐予測器が実現できる．

以下，2章ではパーセプトロン分岐予測器の概要と関連研究を紹介する．3章では重みテーブルの数を削減する手法を，4章ではより長い履歴を使用する手法を提案する．5章ではこれらの提案手法を用いたパーセプトロン分岐予測器の構成例を述べ，6章で従来法との比較評価を行う．7章で実装コストとレイテンシを考察し，8章でまとめる．

## 2. 関連研究

分岐予測器はある分岐命令の分岐方向を予測する回路である．本稿ではこの予測すべき分岐命令をbranch Bと呼ぶこととする．パーセプトロン分岐予測器は，branch Bのアドレスの他に過去に実行した分岐命令のアドレスを時系列に並べた実行パス履歴，過去の分岐結果の履歴を時系列順に並べたグローバル履歴，この分岐履歴を対の情報である分岐命令のアドレスを用いて並び替えたローカル履歴などを用いて予測を行う．

† 電気通信大学 情報工学科
Department of Computer Science, The University of Electro-Communications

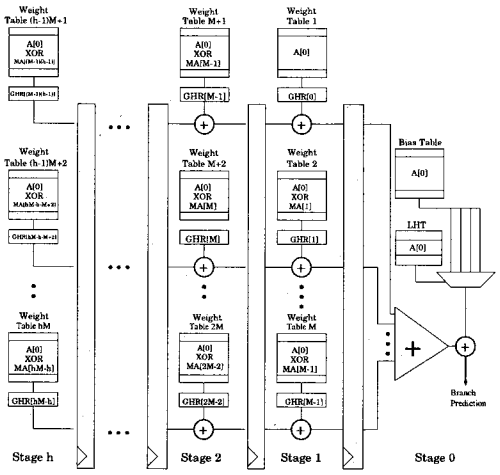


**図 1** Advanced Anti-Aliasing Perceptron Branch Predictor ($A^3$PBP) の構造

branch B のアドレスや実行パス履歴をインデックスとして重みテーブルと呼ばれる多数のテーブルからそれぞれ重みと呼ばれる整数値を読み出す．これらの重みはそれぞれグローバル履歴の特定の場所と 1 対 1 に対応している．グローバル履歴の各要素は，1 ならば分岐成立，$-1$ ならば分岐不成立を表わす．読み出された重みを $W_i$，対応するグローバル履歴の要素を $X_i$，$1 \leq i \leq n$，とし，次式を計算する．

$$y = W_0 + \sum_{i=1}^{n} W_i X_i \qquad (1)$$

$W_0$ はバイアスであり，重みと同様に読み出される．求められた $y$ の値が正ならば分岐成立として 1 を，負ならば分岐不成立として $-1$ を出力する．重み $W_i$ の更新規則は，分岐結果を $t \in \{1, -1\}$ とすると次式によって表される．通常，$\alpha = 1$ とする．

$$W_i = W_i + \alpha t X_i \qquad (2)$$

これまで基本的な構造による Global/Local Perceptron Branch Predictor，予測処理のパイプライン化を可能とした Path-based Neural Predictor[2]，重み読み出しのインデックスを工夫した Piecewise Linear Branch Predictor[3][4]，重み読み出しの並列化を行った PTBP[5]，および，Advanced Anti-Aliasing Perceptron Branch Predictor ($A^3$PBP)[6] などのパーセプトロン分岐予測器が知られている．以下，これらの中で最も予測精度のよい $A^3$PBP を紹介する．

**図 1** に $A^3$PBP の構造を示す．$A^3$PBP は実行パス履歴の情報をより効率的に利用する構造を持ち，ローカル履歴を重み読み出しのインデックスに利用することにより重みの負の衝突を削減し予測精度を向上させた．実行パス履歴は重み読み出しに次のように利用される．Stage $n$ における $m$ 番目の重みテーブル Table $i = (n-1)M + m$ のインデックス index[$i$] を A[0] $\oplus$ A[$i-n$] で生成する ($1 \leq n \leq h, 1 \leq m \leq M$)．A[$s$] は branch B より $s$ 個前の分岐命令のアドレスを表す．A[0] はそのステージで得られうる最新のアドレスである．さらに，実行パス履歴を構成するアドレスを**図 2** に示す多重化アドレス (Multiple Address, MA) で置き換える．これはアドレスの一部分を削り，そこにローカル履歴ビットを付加したものである．すなわち，

$$index[i] = A[0] \oplus MA[i-n] \qquad (3)$$

とする．

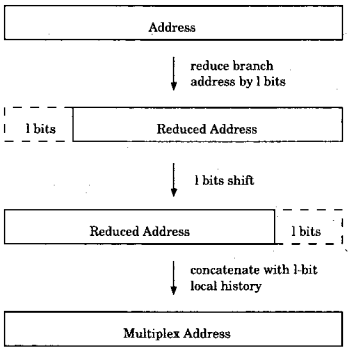


**図 2** 多重化アドレス（MA）の生成法

しかし，$A^3$PBP は他のパーセプトロン分岐予測器と同様，予測に多数の重みテーブルを使用する．重みテーブルの数が大きいと実装コストが増加する．

## 3. 重みテーブル数の削減

パーセプトロン分岐予測器において，重みテーブルの数（予測計算に使う重みの数と等しい）が増加すると累算に要するハードウェアコストや計算レイテンシが上昇する．全体の記憶容量を一定とすれば，テーブル数が多いと 1 つあたりのテーブルのエントリ数は相対的に小さくなる．エントリー数の不足は使用する重みの衝突を引き起こし予測精度低下の主因となりうる．このようにテーブル数を削減することは実装コストと計算レイテンシの低減，および重みの負の衝突の回避に役立つ．テーブル数を削減するには，使用するグローバル履歴長も短くする必要がある．branch B に近いグローバル履歴ビットほど branch B との関連が高いことは明らかであるので，重みテーブルと対応するグローバル履歴ビットは branch B と近い方から

表 1　インデックス生成関数

| アドレスの種類 | インデックス生成関数 $index[i]$ |
|---|---|
| 2 種類 | $A[0] \oplus MA[i-n]$ |
| 3 種類 | $A[0] \oplus MA[i-n] \oplus MA[3i-n]$ |
| 4 種類 | $A[0] \oplus MA[i-n] \oplus MA[3i-n] \oplus MA[5i-n]$ |
| 5 種類 | $A[0] \oplus MA[i-n] \oplus MA[3i-n] \oplus MA[5i-n] \oplus MA[7i-n]$ |
| 6 種類 | $A[0] \oplus MA[i-n] \oplus MA[3i-n] \oplus MA[5i-n] \oplus MA[7i-n] \oplus MA[11i-n]$ |

順に使用する．

しかし，予測精度の向上を実現するためには従来法と同程度の履歴情報を予測に反映させる必要がある．$A^3$PBP は実行パス履歴を重み読み出しのインデックス生成に使用した．テーブル数を削減し，テーブルあたりのエントリー数を増やすには，$A^3$PBP で使用した実行パス履歴情報と比較し，より詳細な情報を重み読み出しのインデックスに反映させる必要がある．

**表 1** は Table $i$ が使用するインデックス生成関数 $index[i]$ の例である．テーブルによって使用する MA を少しづつずらすために，$index[i]$ を実行パス履歴上の $p$ 個 ($p = 1,\ldots,5$) の多重化アドレス MA[$c_1 i-n$],...,MA[$c_p i-n$] を用いて生成している．実行パス履歴情報を予測に反映させるために $p$ 個の MA は，テーブル間で極力重複しないように選択する必要がある．そのため，係数 $c_1 = 1$，$1 < c_2 < \cdots < c_p$ を素数としてある．(なお，$p = 1$ の場合は $A^3$PBP のインデックス生成関数を表している．) これらの素数を調整し，branch B から遠くなるにつれて使用する MA の間隔が徐々に開くようにした．こうすることで，branch B と強い相関を持つ近距離の分岐履歴に重点を置きつつ，予測が遠距離の特定の区間の履歴に強く依存することを極力避けることができる．

詳細な実行パス履歴情報を重み読み出しのインデックスに反映させることにより，テーブルあたりのエントリー数を増やすことができるため，予測精度は向上するが，過剰に詳細なパス情報の利用は重みの負の衝突を誘発し，予測精度が低下する．すなわち，インデックス生成に使用するアドレスの種類数には最適値が存在することが予想される．そこで，この最適値を実験により求めた．

実験は SPEC INT2000 の 12 種類のベンチマークを SimCore シミュレータ[7] で 1 億命令程度実行した際にトレースしたデータを用いた．このデータを 8 本鎖 2 段パイプラインの $A^3$PBP をシミュレートしたプログラムで処理し，重み読み出しのインデックス生成に使用する多重化アドレスの数と予測精度の関係を測定した．Stage 0 のテーブルを除いたテーブルの個数は 8 個である．記憶容量は総計 32KB とした．

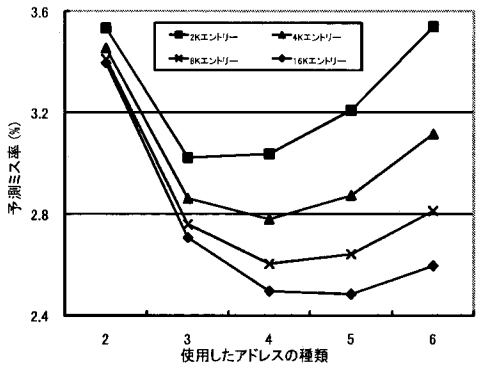


**図 3**　インデックス生成に使用するアドレスの種類と予測ミス率の関係

結果を**図 3** に示す．図からインデックス生成に使用する最適なアドレスの種類数はテーブルのエントリー数と関係があることが分かる．使用するアドレスの種類が少ないと予測ミス率が下がらない，つまり効率的にテーブルを使用できない．逆に過剰にアドレスの種類を増やすと予測ミス率は上昇してしまう．また，エントリ数が増えるに従いより詳細なパス情報を使用することによって予測精度が向上している．最適な MA の種類はエントリー数が 2K から 8K エントリの場合は 4，16K エントリの場合は 5 であった．

## 4. より長い履歴の利用

前章で効率的なインデックス生成に必要な MA の数はエントリ数に応じて 4 種類から 5 種類ということことを実験的に示した．しかし，表 1 のインデックス生成法では従来法よりも短い履歴しか使用できない．例えば 4 種類の MA を用いてインデックス生成を行う場合を考えると，使用される MA のうち branch B から最も遠いものは MA[5$i$-1] である．$i = 8$ の場合，これは MA[39] (branch B よりも 40 個前に処理された分岐命令の MA) となり，従来法で使用する履歴長の半分程度となる．

また，O-GEHL branch predictor[8] で知られているように予測精度の向上のためには長いグローバル履歴を用いることが必要である．従来法ではグローバル履歴中の 1 つの分岐結果は 1 つの重みテーブルに対応する．したがって，テーブル数を削減すると利用できるグローバル履歴も削減される．

以下では，少ないテーブル数の下で長い実行パス履

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Table 1 (NDI) | * | | * | | * | | | | | | | | | | | | | | | | | | | | |
| Table 2 (NDI) | | * | | | | * | | | | * | | | | | | | | | | | | | | | |
| Table 3 (NDI) | | | * | | | | | | * | | | | | | * | | | | | | | | | | |
| Table 4 (NDI) | | | | * | | | | | | | | * | | | | | | | | * | | | | | |
| Table 5 (LDI) | | | | | * | | | | | | | | | | | | | | | | | | | | * |
| Table 6 (LDI) | | | | | | * | | | | | | | | | | | | | | | | | | | |
| Table 7 (LDI) | | | | | | | * | | | | | | | | | | | | | | | | | | |
| ALL | * | * | * | * | * | * | * | | * | * | | * | | | * | | | | | * | | | | | * |

| | 26 | 27 | 28 | 29 | 30 | 31 | 32 | 33 | 34 | 35 | 36 | 37 | 38 | 39 | 40 | 41 | 42 | 43 | 44 | 45 | 46 | 47 | 48 | 49 | 50 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Table 1 (NDI) | | | | | | | | | | | | | | | | | | | | | | | | | |
| Table 2 (NDI) | | | | | | | | | | | | | | | | | | | | | | | | | |
| Table 3 (NDI) | | | | | | | | | | | | | | | | | | | | | | | | | |
| Table 4 (NDI) | | | | | | | | | | | | | | | | | | | | | | | | | |
| Table 5 (LDI) | | | | | | | | | | * | | | | | | | | | | | | | | | |
| Table 6 (LDI) | | | | | * | | | | | | | | | | | | * | | | | | | | | |
| Table 7 (LDI) | | | | | | | | | | * | | | | | | | | | | | | | | * | |
| ALL | | | | | * | | | | | * | | | | | | | * | | | | | | | * | |

**図 4** 使用する MA の分布

歴と長いグローバル履歴を利用する手法を述べる.

### 4.1 より長い実行パス履歴の利用

より長い実行パス履歴を利用するために, 前章で述べた重みテーブルのインデックス生成法に加え, より過去の MA を用いるインデックス生成法も用意する. その際, 予測器全体で使用する MA の間隔を考慮しなければならない. 一般に branch B に近い履歴ほど分岐結果と強い相関を持っているので近距離の履歴に重点を置き, かつ, 遠距離の履歴も満遍なく利用することが必要である. よって, $i$ の係数 $c_1, \cdots, c_p$ は, 予測器全体で使用する MA の間隔が branch B から遠くなるにしたがい徐々に開くように値を選択する必要がある. 例えば 4 種類のアドレスを使用する場合 ($p = 3$), 表 1 の式

$$A[0] \oplus MA[i-1] \oplus MA[3i-1] \oplus MA[5i-1] \quad (4)$$

を用いる Table 1,2,3,4 に加え, 次式をインデックスとする Table 5,6,7 を用意する.

$$A[0] \oplus MA[i-1] \oplus MA[5i-1] \oplus MA[7i-1] \quad (5)$$

この時の MA の分布を**図 4** に示す.

式 (5) のように命令間距離の長い MA を使用して生成したインデックスを Long Distance Index(LDI) と呼び、式 (4) のような通常のインデックス (Nomal Distance Index, NDI) と区別する. 各テーブルでは NDI と LDI のどちらか一方を使用する. 上の例では $i$ の値が小さいものから NDI を, 残りは LDI を用いた. こうすることで, より長い実行パス履歴を使用することができる.

### 4.2 より長いグローバル履歴情報の利用

多重化アドレスへ分岐結果を付加することにより多重化アドレスの列 (すなわち実行パス履歴) にグローバル履歴情報を付加する. 具体的には branch B から生成された多重化アドレスのローカル履歴の部分のうち最も過去の履歴ビットを branch B の分岐方向が確定した際にその分岐結果で置換することにより MA

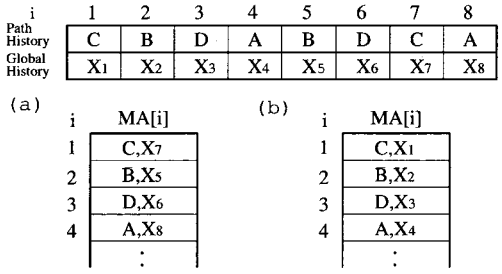


**図 5** (a)MA の更新を行わない場合と (b)MA の更新を行った場合

を更新する. これは分岐方向が確定し更新が完了したローカル履歴で多重化アドレスを生成するのと等価である. **図 5** はローカル履歴が 1 ビットの場合, ($A^3$PBP のように) 更新を行わない場合と, 更新を行った場合の MA の比較である. 更新を行った場合には MA の中にそのアドレスの最も新しい分岐結果が含まれ, MA で構成されるパス履歴の中にはグローバル履歴が含まれていることが分かる. 更新処理は信号線の繋ぎ替えだけで可能となり, また, 予測処理とは別に行うことができるため予測に必要なレイテンシに変化はない.

MA をそのアドレスの分岐結果を用いて更新することにより, MA の列 (すなわち実行パス履歴) にグローバル履歴情報を付加することができる. テーブルのインデックスとして使用される MA の間隔は branch B から遠くなるにつれて徐々に開くように設定されている. このため, MA に含まれている分岐履歴もまた branch B から遠くなるにつれて間隔が徐々に開いていき, 使用する履歴に偏りが生じない. このようにして長いグローバル履歴を利用する.

## 5. 提案手法に基づく回路構成例

以上の提案手法に基づくパーセプトロン分岐予測器

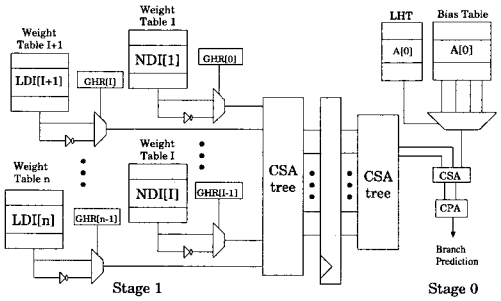


**図 6** 提案手法に基づく回路構成例

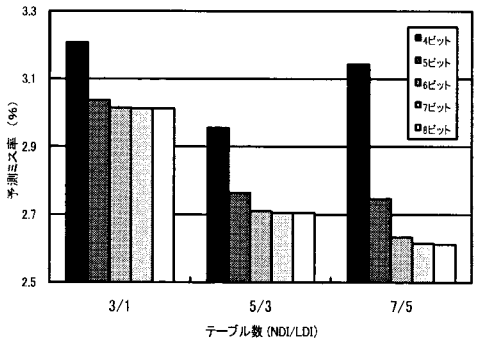


**図 7** テーブル数を変えたときの重みのビット長と予測ミス率の関係

の構成例を**図 6** に示す．この分岐予測器の重みテーブル数 $M$ は累算を 2 段のパイプラインで処理することができる程度に小さい．図において，NDI を使用するのは Table 1 から Table $I$，$(1 \leq I < M)$ までである．Table $I+1$ から Table $M$ までは LDI を使用する．NDI は式 (4)，LDI は式 (5) を用いる．次章でパラメーター $M$ および $I$ を変えて評価を行う．

読み出された重みは対応付けられたグローバル履歴ビットの値が 1 ならばそのまま，0 ならばビット反転を行う．累算のための CSA 木は 2 つのステージがバランスするように分割する．Stage 0 では $A^3$PBP と同様に branch B のアドレスを用いて Bias Table から重みの候補を読み出す．この中からローカル履歴の値によって予測に使用する重みを選択する．ローカル履歴は branch B のアドレスを用いて Local History Table から読み出す．この処理と並行して Stage 1 で読み出した重みの累算を行う．最後に累算された途中結果とバイアスの重みを加算し，その結果が正であれば分岐成立，負であれば分岐不成立と予測する．学習は従来のパーセプトロン分岐予測器と同様の方法で行う．

## 6. 実験結果

提案手法の予測精度を 3 章と同様の環境でシミュレートしたプログラムを用いて評価する．

はじめに最適な重みのビット長を実験的に求めるために NDI の数 $I$，LDI の数 $M-I$ を変えて予測精度を求めた．実験は $(I, M-I) = (3,1),(5,3),(7,5)$ について行った．結果を**図 7** に示す．図より重みのビット長は長いほど予測精度は良いが，6 ビット以上ではほとんど変化しないことが分かる．以下の実験では重みのビット長は 6 とする．これは従来のパーセプトロン分岐予測器で使用されてきた 8 ビットよりも短い．

提案手法と従来法の予測精度を比較する．比較対象として Pipelined PTBP，O-GEHL branch predictor，$A^3$PBP を用いた．記憶容量は 8KB から 128KB までの 5 段階とする．**表 2** は各分岐予測器のパラメーターを示す．O-GEHL は文献[8]，Pipelined PTBP は文献[5]，$A^3$PBP は文献[6] を参考にした．提案手法のテーブル数は，NDI を使用するテーブルの数 $(I)$ /LDI を使用するテーブルの数 $(M-I)$ /Bias Table の数を表す．テーブルの総数はこれらの和 $(M+1)$ である．$A^3$PBP の Bias Table は他のテーブルの 4 倍，提案手法の Bias Table は他のテーブルの 2 倍の容量とし，MA で使用するローカル履歴長はどちらも 2 bit とした．閾値シフトは Bias Table から読みだした閾値に対する右シフトの桁数である．また，更新閾値はテーブルの総数を $T$，閾値の右シフトを $q$ 桁とした時，$2.1 \times (T + 2^q)$ とした．

**表 2** 各予測器のパラメーター

| | 記憶容量 | 8KB | 16KB | 32KB | 64KB | 128KB |
|---|---|---|---|---|---|---|
| Pipelined PTBP | テーブル数 | 22 | 26 | 32 | 43 | 50 |
| | 重みビット長 | 8 bit | 8 bit | 8 bit | 8 bit | 8 bit |
| O-GEHL | テーブル数 | 8 | 8 | 8 | 8 | 8 |
| $A^3$PBP | テーブル数 | 16 | 26 | 36 | 43 | 63 |
| | 重みビット長 | 7 bit | 7 bit | 7 bit | 8 bit | 8 bit |
| | 閾値シフト | 2 bit | 2 bit | 2 bit | 2 bit | 2 bit |
| Proposed | テーブル数 | 3/2/1 | 4/2/1 | 4/3/1 | 5/4/1 | 7/5/1 |
| | 重みビット長 | 6 bit | 6 bit | 6 bit | 6 bit | 6 bit |
| | 閾値シフト | 1 bit | 1 bit | 1 bit | 1 bit | 1 bit |

**図 8** は各予測器の記憶容量別の予測ミス率をグラフにしたものである．提案手法は全ての記憶容量において，どの分岐予測器よりも低いミス率を実現している．$A^3$PBP と比べて平均で 3.2%，最高で 5.6% (8KB 構成時) 予測ミス率が低い．

また，図より低記憶容量において他の分岐予測器との予測精度の差が顕著となっている．これは，各容量で同じインデックス生成関数を用いたため，大記憶容量においてテーブルを効率的に利用できなかったこと

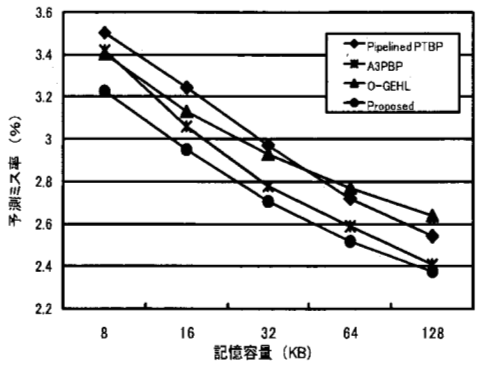


**図 8** 記憶容量を変えた時の予測ミス率の従来法との比較

が原因であると考えられる．大容量時にインデックス生成に使用する MA の種類を増やすことで予測精度がさらに向上する可能性がある．

## 7. 実装コストとレイテンシの考察

提案手法のハードウェアコストを $A^3$PBP と比較する．同じ記憶容量で比較した場合，2 つの予測器のハードウェア量の違いは重みの読み出し，加減算，更新に関する回路である．表 2 から分かるとおり，提案手法は $A^3$PBP と比較し予測に用いる重みの数が 1/4 程度である．パーセプトロン分岐予測器は 1 つの重みに対してそれぞれ 1 つの読み出し，更新回路が必要となる．これに加えて重みの累算回路の回路面積も累算する重み数と比例する．このため，提案手法は $A^3$PBP と比べてこれらの回路の数が 1/4 程度で済むことになる．また，重みのビット長も短くなっているため，加減算や更新に関する回路の 1 単位当たりの面積コストも減少している．

次にレイテンシを $A^3$PBP と比較する．提案手法は累算する重みの数が少なく，重みのビット長も短いため CPA 回路が縮小される．この結果，CPA のレイテンシは従来よりも軽減される．反対にテーブルのエントリ数の違いによりテーブル読み出しのレイテンシは増加する．ただし，その増加量はエントリ数の対数値に比例する．2 つの予測器のエントリ数の比は最大で 4 倍程度であり，この分のレイテンシの増加は小さいことが分かる．以上から提案手法のレイテンシは $A^3$PBP と同等であると言える．さらに，累算する重みの数が減少しているため累算のレイテンシも大幅に減少するので，Stage 1 で読み出した複数の重みをバイアスと加算する前に CPA を用いて累算しておくことも可能である．このようにすることで $A^3$PBP と比べて CSA1 個のレイテンシを軽減することが可能である．

以上より，提案手法は予測精度のみでなく実装コストやレイテンシの面でも $A^3$PBP より優れていることが分かる．

## 8. ま と め

本稿では新しいパーセプトロン分岐予測器を提案した．評価の結果，本手法は従来法よりも少数のテーブルを予測に使用することで重みの累算や学習に必要となる実装コストを大幅に削減できる．また，詳細な実行パス履歴とグローバル履歴の一部をインデックスに利用することにより，従来法より予測ミス率を平均で 3.2%，最大で 5.6%削減した．使用できる記憶容量を一定とすると，本手法により従来法より低い実装コストで高い予測精度を持つパーセプトロン分岐予測器が実現できる．

**謝辞** Alpha プロセッサシミュレータ SimCore を提供していただいた東京工業大学大学院情報理工学研究科の吉瀬謙二講師に感謝する．本研究は，一部，日本学術振興会科学研究費補助金 (基盤研究 (C)(2)18500048) による．

## 参 考 文 献

1) D. A. Jimenez, and C. Lin : Dynamic Branch Prediction with Perceptrons, *Proc. the Seventh International Symposium on High-Performance Computer Architecture (HPCA'01)*, pp.197-206 (2001)
2) D. A. Jimenez : Fast Path-Based Neural Branch Prediction, *Proc. the 36th Annual IEEE/ACM International symposium on Microarchitecture (MICRO-36)*, pp.243-252 (2003)
3) D. A. Jimenez: Idealized Piecewise Linear Branch Prediction, *The Journal of Instruction-Level Parallelism*, Vol.7 (2005)
4) D. A. Jimenez: Piecewise Linear Branch Prediction, *Proc. the 32nd International Symposium on Computer Architecture (ISCA'05)*, pp.382-393 (2005)
5) 石井康雄, 平木敬 : 実行パス履歴情報を利用した分岐予測手法, 情報処理学会論文誌 : コンピューティングシステム, 47 巻, SIG3(ACS13) 号, pp.58-72 (2006)
6) Y. Ninomiya and K. Abe : A3PBP: A Path Traced Perceptron Branch Predictor Using Local History for Weight Selection, *The Journal of Instruction-Level Parallelism*, Vol.9 (2007).
7) 吉瀬謙二, 片桐孝洋, 本多弘樹, 弓場敏嗣 : SimCore/Alpha Functional Simulator の設計と実装, 電子情報通信学会論文誌, Vol.J88-D-I, No.2, pp.143-154 (2005)
8) Andre Seznec : Genesis of the O-GEHL Branch Predictor, *The Journal of Instruction-Level Parallelism*, Vol.7 (2005)