情報処理学会論文誌　コンピューティングシステム　Vol. 1　No. 1　96–104 (June 2008)

CPBP：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[†1]　　阿　部　公　輝[†1]

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) 1 つの分岐命令の予測に使用する重み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．

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

Yasuyuki Ninomiya[†1] and Kôki Abe[†1]

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

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．なか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) 1 つの分岐命令の予測に使用する重み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 のアドレスのほか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べ替えたローカル履歴などを用いて予測を行う．

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テーブルから重みと呼ばれる複数の整数値を読み出す．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．

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

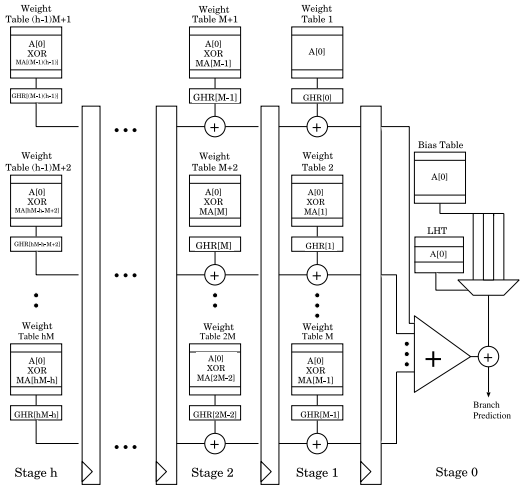

図 1　Advanced Anti-Aliasing Perceptron Branch Predictor（A^3PBP）の構造
Fig. 1　Structure of Advanced Anti-Aliasing Perceptron Branch Predictor (A^3PBP).

W_0 は閾値であり，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．

$$W_i = W_i + \alpha t X_i \tag{2}$$

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^3PBP）[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，これらの中で最も高い予測精度を示す A^3PBP を紹介する．

図 1 に A^3PBP の構造を示す．A^3PBP 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．

A^3PBP は同時刻に各ステージが異なる分岐命令の予測を行う．たとえば Stage 1 では次回にアドレスが入力される分岐命令の予測を投機的に行っている．ステージ内には多数のテーブルがあり，分岐命令が入力された場合に各テーブルでは並列にインデックス計算と重みの読み出し，加減算が行われる．1 つのテーブルから読み出される重みは 1 個である．

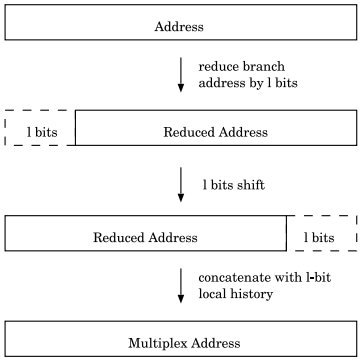

図 2　多重化アドレス（MA）の生成法
Fig. 2　An Formation of index MA (multiple address).

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] $\oplus$ MA[$i-n$] で生成する（$1 \le n \le h$，$1 \le m \le M$）．MA[s] は branch B より s 個前の分岐命令のアドレスの一部分を変化させた多重化アドレス（Multiple Address，MA）を表す．多重化アドレスの生成手法は後述する．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．たとえば，Stage 1 の Table 2 の重み読み出しのインデックス index[2] は A[0] $\oplus$ MA[1] で計算される．読み出された各重みは対応付けられたグローバル履歴のビットの値が 0 の場合にビット反転処理が行われ，前段の Stage 2 から送られてきた値と加算する．計算された値は後段の Stage 0 に送るためにパイプラインに格納される．

Stage 0 では分岐方向の予測を完了させる．Bias Table と Local History Table，LHT から閾値の候補群とローカル履歴を並列に読み出し，ローカル履歴の値によって閾値の候補群の中から 1 つの閾値を選択する．これと並行して前段のステージから送られてきた途中計算値の累算を行う．最後にこの累算された値と閾値を加算し，その結果が 0 以上であれば分岐成立，それ以外であれば分岐不成立と予測する．

図 2 に示す手法に従い A[0] のアドレスの一部を削り，そこに LHT から読み出されたローカル履歴を付加する．これを多重化アドレスとして保持し，上流のパイプラインで使用する．

A^3PBP 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

A^3PBP 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要因となりうる．したがって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．

近年，ハイブリッド型分岐予測器の研究がさかんである[7)–9)]．ハイブリッド型分岐予測器は，予測の性質が異なる複数の小規模の分岐予測器を備え，予測のたびに使用する分岐予測器を動的に切り替えることで予測精度を向上させる手法である[10)]．ハイブリッド型分岐予測器の要素としての応用を考える場合，単体で精度が高いことに加え低コストであることが望ましい．

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^3PBP 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数を増やすには，A^3PBP 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 を少しずつずらすために，$index[i]$ 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^3PBP 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．

図 3 は $p + 1 = 4$ のときの Table i，$i = 1, \ldots, 7$ のインデックス生成関数が利用する実行パス履歴 MA$[s]$ の分布を表す（MA$[s]$ は branch B より s 個前の分岐命令の多重化アドレス）．図から，branch B と強い相関を持つ近距離の分岐履歴に重点がおかれ，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分かる．

表 1　図 1 の Table i のインデックス生成関数 $index[i]$．$p + 1$ は使用するアドレスの数（p は MA の数）
Table 1　Index generation function $index[i]$ for Table i in Fig. 1. $p + 1$ is the number of addresses used for the generation. (p is the number of the MAs).

$p + 1$	$index[i]$
2	$A[0] \oplus MA[i - n]$
3	$A[0] \oplus MA[i - n] \oplus MA[3i - n]$
4	$A[0] \oplus MA[i - n] \oplus MA[3i - n]$ $\oplus MA[5i - n]$
5	$A[0] \oplus MA[i - n] \oplus MA[3i - n]$ $\oplus MA[5i - n] \oplus MA[7i - n]$
6	$A[0] \oplus MA[i - n] \oplus MA[3i - n]$ $\oplus MA[5i - n] \oplus MA[7i - n] \oplus MA[11i - n]$

MA[s]	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	*		*		*																				
Table 2		*				*				*															
Table 3			*						*						*										
Table 4				*								*								*					
Table 5					*										*										*
Table 6						*												*							
Table 7							*														*				
ALL	*	*	*	*	*	*	*		*	*		*			*			*		*	*				*

MA[s]	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																									
Table 2																									
Table 3																									
Table 4																									
Table 5																									
Table 6					*																				
Table 7										*															
ALL					*					*															

図 3　$p + 1 = 4$ のとき Table i，$i = 1, \ldots, 7$ のインデックス生成関数が利用する実行パス履歴 MA$[s]$ の分布
Fig. 3　Distribution of MAs used by $index[i]$, $i = 1, \ldots, 7$, in case of $p + 1 = 4$.

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，テーブル数を削減することにより，相対的にテーブルあたりのエントリ数を増やすことができる．このため，予測精度は向上するが，逆に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ことがある．すなわち，インデックス生成に使用するアドレスの数 $p+1$ 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[11)] 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^3PBP 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

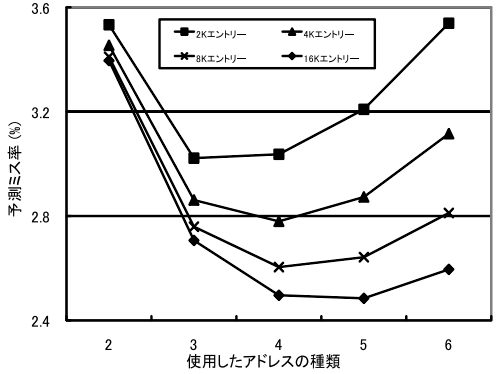

図 4　インデックス生成に使用するアドレスの数 $p+1$ と予測ミス率の関係
Fig. 4　Misprediction rate vs. $p+1$.

MA[s]	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	*	*	*	*	*	*	*		*	*		*			*					*					*

MA[s]	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 5　$p+1=4$ のとき Table i，$i=1,\ldots,7$ のインデックス生成関数が利用する実行パス履歴 MA[s] の分布（LDI を使用した場合）
Fig. 5　Distribution of MAs used by $index[i]$, $i=1,\ldots,7$, in case of $p+1=4$ when LDI is partly applied.

ブルの個数は 8 個である．記憶容量は総計 32 KB とした．

結果を**図 4** 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数 $p+1$ はテーブルのエントリ数と関係があることが分かる．使用するアドレスの数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数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．最適なアドレスの数 $p+1$ はエントリ数が 2 K エントリの場合は 3，4 K，8 K エントリの場合は 4，16 K 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アドレスの数はエントリ数に応じて 3 から 5 という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．たとえば 4 種類のアドレス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[5i-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 [12] 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歴と長いグローバル履歴を利用する手法を提案する．

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．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，表 1 における 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 から遠くなるに従い徐々に開くように値を選択する必要がある．たとえば 4 種類のアドレスを使用する場合（$p=3$），表 1 の式

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

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．

$$A[0]\oplus MA[i-1]\oplus MA[5i-1]\oplus MA[7i-1] \quad (4)$$

このときの MA の分布を**図 5** に示す．

式 (4) 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 (3) 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

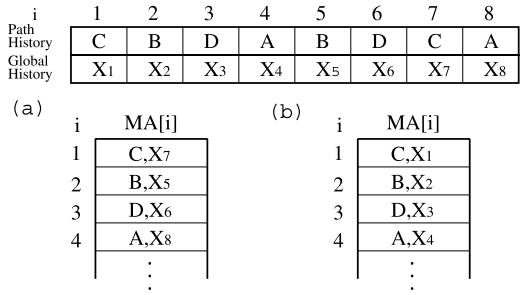

図 6　MA の内容：(a) MA の更新を行わない場合と (b) MA の更新を行った場合
Fig. 6　Contents of MA: (a) without updating. (b) with updating.

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 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 6** はローカル履歴が 1 ビットの場合，（A^3PBP 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 MA で構成されるパス履歴の中に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．

このように 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．

5. CPBP の構造

以上の提案手法に基づく新しいパーセプトロン分岐予測器，CPBP（Compact Perceptron Branch Predictor）と呼ぶ．**図 7** は CPBP の構造である．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．次章でパラメータ M および I を変えて評価を行う．

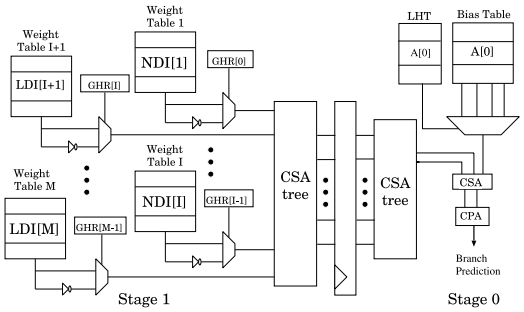

図 7　CPBP の回路構成
Fig. 7　Structure of CPBP.

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^3PBP 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閾値の重みを加算し，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

CPBP 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．NDI は式 (3)，LDI は式 (4) を用いた．結果を**図 8** 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 5 ないし 6 ビットとする．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．

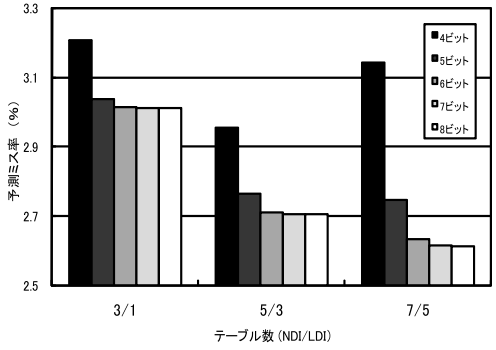

図 8　テーブル数を変えたときの重みのビット長と予測ミス率の関係
Fig. 8　Misprediction rate vs. bit length of weights with the number of tables varied.

CPBP 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^3PBP を用いた．O-GEHL branch predictor はパーセプトロン分岐予測器ではないが，テーブル数が CPBP に近いことと各テーブルから読み出された値を加算することから比較対象に加えた．低い面積コストをもたらす小さな記憶容量から比較的大きな記憶容量の広い範囲で予測精度を評価するため，記憶容量は 8 KB から 128 KB までの 5 段階とする．CPBP は NDI の異なる 2 種類（CPBP 3x，CPBP 4x）を用意した．CPBP 4x の NDI は式 (3)，LDI は式 (4) を，CPBP 3x の NDI は以下の式 (5)，LDI は式 (4) 用いた．

$$A[0] \oplus MA[i-1] \oplus MA[5i-1] \tag{5}$$

表 2 は各分岐予測器のパラメータを示す．O-GEHL は文献 12)，Pipelined PTBP は文献 5)，A^3PBP 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/LDI を使用するテーブルの数/Bias Table の数すなわち，$I/(M-I)/1$ を表す．テーブルの総数はこれらの和 $M+1$ である．A^3PBP の Bias Table は他のテーブルの 4 倍，CPBP 4x 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いずれも 2 bit とした．閾値シフトは Bias Table から読み出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 桁としたとき，$2.1 \times (T + 2^q)$ とした．

図 9 は記憶容量の変化による各予測器の平均予測ミス率の変化をグラフにしたものである．提案手法はすべ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

表 2　実験に用いた予測器のパラメーター
Table 2　Parameters used in experiment.

	記憶容量	8 KB	16 KB	32 KB	64 KB	128 KB
Pipelined	テーブル数	22	26	32	43	50
PTBP	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
	テーブル数	16	26	36	43	63
A^3PBP	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
	テーブル数	3/2/1	3/3/1	4/4/1	5/4/1	7/4/1
CPBP 3x	重みビット長	5 bit	5 bit	5 bit	5 bit	5 bit
	閾値シフト	1 bit	1 bit	1 bit	1 bit	1 bit
	テーブル数	3/2/1	4/2/1	4/3/1	5/4/1	11/3/1
CPBP 4x	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

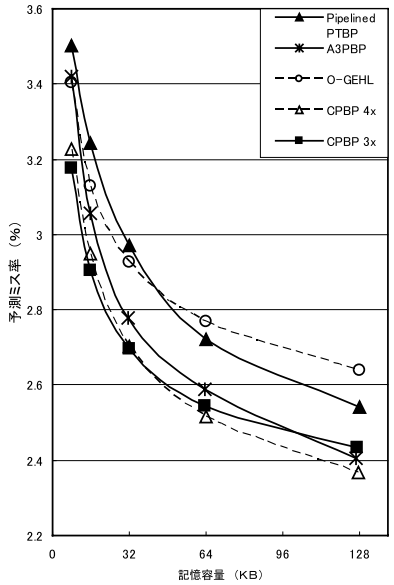

図 9　記憶容量を変えたときの予測ミス率の従来法との比較
Fig. 9　Misprediction rate of predictors.

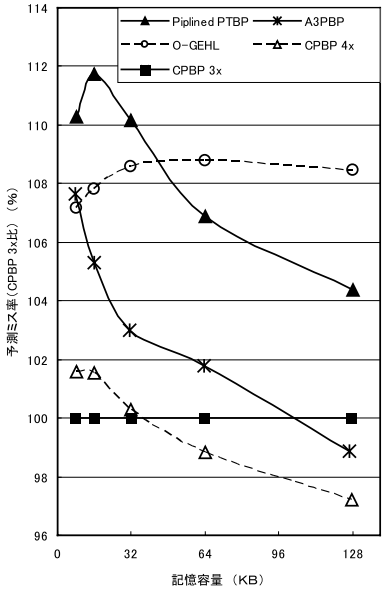

図 10　各分岐予測器の CPBP 3x との予測ミスの比
Fig. 10　Ratio of misprediction rates of predictors to CPBP 3x.

る．A^3PBP と比べて平均で 3.9%，最高で 7.1%（8 KB 構成時）予測ミス率が低い．また，32 KB 構成までは CPBP 3x が，それ以上の記憶容量を使用する場合は CPBP 4x の方が予測ミス率が低くなっている．これは図 4 の実験から得られた結果と同じ傾向を示している．よって，より大容量の記憶領域を使用する際には，インデックス生成に使用するアドレスの数を増やすことで予測精度がさらに向上することが示唆される．

図 10 は CPBP 3x の予測ミス率を 100 とした場合の，各予測器の予測ミスの比である．図から CPBP は他のパーセプトロン分岐予測器と比べて小さい記憶容量での予測精度の向上が著しいことが分かる．また，O-GEHL と比べどの記憶容量でも予測精度が高い．

また，分岐予測器の特性を比較するため，ベンチマークごとの予測精度を比較した．**図 11** は図 9 の 32 KB の記憶容量を使用した場合におけるベンチマークごとの予測ミス率の比較である．図からほとんどのベンチマークにおいて CPBP は従来法に比べて予測精度が良いことが分かる．176.gcc はプログラム内の分岐命令の数が多いため各テーブルのエントリ数が予測精度に大きな影響を与える．従来，パーセプトロン分岐予測器はこのようなプログ

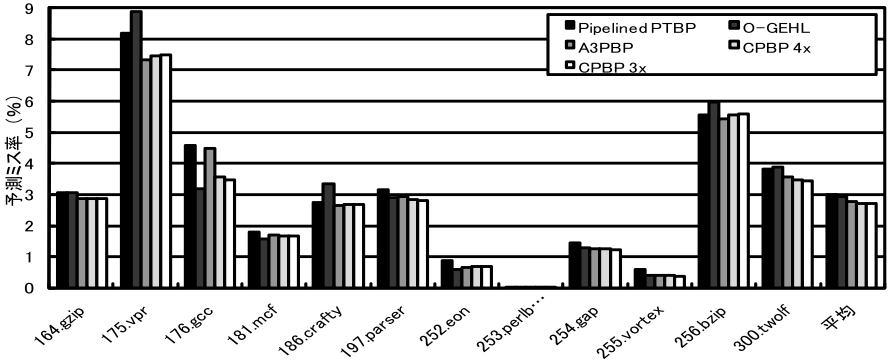

図 11　ベンチマークごとの予測ミス率：従来法との比較（32 KB 時）
Fig. 11　Misprediction rate for each benchmark program compared with conventional methods when 32 KB memory is used.

ラムに対しては予測精度が低かったが，CPBP は各テーブルのエントリ数を増加させることにより予測精度が向上し，最もエントリ数の多い O-GEHL に近い予測精度を示すようになった．175.vpr，186.crafty，256.bzip で A^3PBP よりわずかに劣るのは，CPBP のテーブル数が A^3PBP よりも少なく，細かな履歴の違いを予測に反映できない場合があるためと考えられる．

また，CPBP とは予測方式の異なる分岐予測器であり，現在知られている中で最も高精度の TAGE Branch Predictor と比較するため，TAGE の精密な評価がなされた 2nd Championship Branch Prediction（CBP-2）Infrastructure を用いて CPBP の予測精度を評価した．CPBP と公平な比較をするため，比較対象の TAGE としてパイプライン版の 8C-Ahead TAGE を採用した．記憶容量は Realistic Track の規定に従い 32 KB+256 bit 以下とした．各テーブルの設定は**表 3** のとおりであり，すべてのテーブルで重みは 6 ビットとした．NDI は式 (3)，LDI は以下の式 (6) を使用した．

$$A[0] \oplus MA[i-1] \oplus MA[5i-1] \\ \oplus MA[7i-1] \oplus MA[9i-1] \quad (6)$$

結果を**表 4** に示す．CPBP の予測ミス率は平均で 3.779%であり，TAGE の 3.552%に劣るが，いくつかのプログラム（gzip，vpr，mpegaudio，twolf）では TAGE に勝る．TAGE に平均の予測精度では及ばないが，いくつかのベンチマークでより高い予測精度を示してい

表 3　CBP-2 Infrastructure を用いた実験における CPBP のテーブル設定
Table 3　Table configration of CPBP in CBP-2 experiment.

テーブル番号	Index	エントリ数	テーブル番号	Index	エントリ数
0 (Bias)	-	8 K	6	NDI	2 K
1	NDI	4 K	7	NDI	2 K
2	NDI	4 K	8	NDI	2 K
3	NDI	4 K	9	NDI	2 K
4	NDI	4 K	10	LDI	2 K
5	NDI	4 K	11	LDI	2 K

表 4　CBP-2 Infrastructure を用いた CPBP と TAGE の予測ミス率の比較
Table 4　Comparison of misprediction rate of CPBP and TAGE using CBP-2 Infrastructure.

ベンチマーク	CPBP	TAGE	ベンチマーク	CPBP	TAGE
gzip	10.613	10.854	javac	1.20	1.195
vpr	8.928	9.384	mpegaudio	1.185	1.188
gcc	5.199	3.627	mtrt	0.802	0.571
mcf	11.403	9.688	jack	0.804	0.690
crafty	2.964	2.728	eon	0.466	0.246
parser	6.320	5.438	perlbmk	0.414	0.398
compress	6.303	6.024	gap	2.044	1.581
jess	0.518	0.406	vortex	0.200	0.169
raytrace	0.744	0.515	bzip2	0.044	0.043
db	2.457	2.439	twolf	12.980	13.868

ることから，予測精度の面からも異なる性質を持っていることが分かる．

7. 実装コストとレイテンシの考察

CPBP のハードウェアコストを A^3PBP 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^3PBP 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．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^3PBP と比べ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 単位あたりの面積コストも減少している．

CPBP は A^3PBP に比べ，重み読み出しのインデックス生成にかかる回路は，テーブル 1 つあたりでは増加する．CPBP 4x の場合，インデックス生成に関する回路は重みテーブル 1 つに対し従来法の 3 倍必要となる．しかし，テーブルの数は従来法の 1/4 であるため，分岐予測器全体ではインデックス生成にかかる回路は減少する．

次にレイテンシを A^3PBP 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．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閾値と加算する前に 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^3PBP と比べて CSA1 個のレイテンシを軽減させることが可能である．

重み読み出しのインデックス生成にかかるレイテンシは上昇している．しかしながら，増加量は XOR 回路 1 段分と無視できる程度である．

以上から提案手法は実装コストやレイテンシの面においても A^3PBP より優れていることが分かった．

3 章で述べたように，ハイブリッド型の分岐予測器の要素は単体で小型高精度であり，各要素間で予測の性質が異なることが望ましい．本手法は単体で小さい記憶容量で高い予測精度を示し，かつ実装コストが従来のパーセプトロン分岐予測器よりも低く，TAGE とは異なる性質を持つ．したがって，TAGE や他のハイブリッド型の高精度分岐予測手法の要素の一部を本手法で置き換えることも考えられる．

8. ま と め

本稿では新しいパーセプトロン分岐予測器，CPBP 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 3.9%削減した．特に 8 KB という従来パーセプトロン分岐予測器が苦手とした低容量時の予測ミス率を 7.1%削減し大幅に改善した．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）18500048）による．

参 考 文 献

1) Jimenez, D.A. and Lin, C.: Dynamic Branch Prediction with Perceptrons, *Proc. 7th International Symposium on High-Performance Computer Architecture* (*HPCA'01*), pp.197–206 (2001).
2) Jimenez, D.A.: Fast Path-Based Neural Branch Prediction, *Proc. 36th Annual IEEE/ACM International symposium on Microarchitecture* (*MICRO-36*), pp.243–252 (2003).
3) Jimenez, D.A.: Idealized Piecewise Linear Branch Prediction, *The Journal of Instruction-Level Parallelism*, Vol.7 (2005).
4) Jimenez, D.A.: Piecewise Linear Branch Prediction, *Proc. 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，Vol.47, No.SIG3 (ACS13), pp.58–72 (2006).
6) Ninomiya, Y. and Abe, K.: A3PBP: A Path Traced Perceptron Branch Predictor Using Local History for Weight Selection, *The Journal of Instruction-Level Parallelism*, Vol.9 (2007).
7) Seznec, A.: The L-TAGE Branch Predictor, *The Journal of Instruction-Level Parallelism*, Vol.9 (2007).
8) Ishii, Y.: Fused Two-Level Branch Prediction with Ahead Calculation, *The Journal of Instruction-Level Parallelism*, Vol.9 (2007).
9) Gao, H. and Zhou, H.: PMPM: Prediction by Combining Multiple Partial Matches, *The Journal of Instruction-Level Parallelism*, Vol.9 (2007).
10) McFarling, S.: Combining Branch Predictions, WRL Technical Note TN-36 (1993).
11) 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).
12) Seznec, A.: Genesis of the O-GEHL Branch Predictor, *The Journal of Instruction-Level Parallelism*, Vol.7 (2005).

(平成 19 年 10 月 9 日受付)
(平成 20 年 1 月 31 日採録)

二ノ宮康之（正会員）

2006 年電気通信大学電気通信学部情報工学科卒業．2008 年電気通信大学大学院電気通信学研究科情報工学専攻博士前期課程修了．同年日本電気株式会社入社，また同年電気通信大学大学院電気通信学研究科情報工学専攻博士後期課程入学，現在に至る．計算機アーキテクチャ，特に分岐予測に関する研究に従事．

阿部　公輝（正会員）

1969 年横浜国立大学工学部電気工学科卒業．1971 年横浜国立大学大学院工学研究科電気工学専攻修士課程修了．1974 年東京大学大学院理学系研究科物理学専攻博士課程単位取得退学．同年電気通信大学電気通信学部電子計算機学科助手．1980〜1982 年カーネギーメロン大学客員研究員．1990 年電気通信大学電気通信学部情報工学科助教授．2007 年同准教授．2008 年同教授．現在に至る．理学博士．計算機アーキテクチャ，VLSI システム設計，コンピュータネットワーク等の研究に従事．電子情報通信学会，電気学会，IEEE 各会員．