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

2006－ARC－169 (6)
2006／8／1

実行パスとローカル履歴を重み選択に利用した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実行パス履歴を重み選択に用いることでその予測精度を高められることが最近の研究から明らかになっている．本稿ではまず，パイプライン構造のパーセプトロン分岐予測器において，実行パス履歴の情報を効率的に利用する手法を提案する．次に，パーセプトロン分岐予測器の重み選択のインデックスにローカル履歴情報を付加することで予測精度を向上させる手法を提案する．この2つの手法を適用したAdvanced Anti-Aliasing Perceptron Branch Predictor(A^3PBP)は，従来法のPipelined PTBPと比較し，予測計算に要するレイテンシは同程度でありながら，予測ミス率を最大で6.6%軽減できることをシミュレーションにより示す．

Path Traced Perceptron Branch Predictor Using Local History for Weight Selection

Yasuyuki Ninomiya† and Kôki Abe†

Recent studies on perceptron branch predictors reveal that using path history for weight selection realizes high prediction rate. In this paper, we first present a method of utilizing path information efficiently in pipelined perceptron branch predictors. Next we describe a new method of using local history as part of index for weight selection to improve the prediction accuracy. Using the proposed methods, we present a new perceptron branch predictor, Advanced Anti-Aliasing Perceptron Branch Predictor(A^3PBP). A^3PBP was evaluated by simulation and found to reduce the misprediction rates of previously reported Pipelined PTBP by 6.6% without increasing the calculation latency.

1. はじめに

プロセッサのパイプラインはさらなる深化の傾向を見せており，パイプラインが深くなるにつれ制御ハザードに起因する性能低下はより深刻なものとなっている．そのような状況の中で近年注目されているのがニューラルネットワークの1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]である．この分岐予測器は2ビットカウンターを用いて予測を行う従来のgshare分岐予測器[2]などに比べ高い予測精度をもつ．しかし，回路構造が複雑で，予測処理の時間コストが大きく，実装は現実的には不可能である．この問題を解決し，さらに予測精度を向上させるために様々な改良が提案されてきたが，それらは各々新たな問題を生み出している．

本稿ではまずはじめにパーセプトロン分岐予測器の中から最も基本的な2種類を紹介し，その構造と問題点を述べる．そして，2つの改良型パーセプトロン分岐予測器を紹介し，その改良手法と問題点を検討する．次にその問題点を改善するために，実行パス履歴の情報をより効率的に利用する構造を提案する．さらに，分岐命令ごとの分岐履歴（ローカル履歴）を利用して予測精度を向上させる手法を提案し，この構造に適用する．この新型のパーセプトロン分岐予測器は従来法に比べて高い予測精度をもつことを既存の分岐予測器と比較することにより示す．

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

2. 関連研究

2.1 Global/Local Perceptron

パーセプトロン分岐予測器は予測すべき分岐命令のアドレスであるターゲットアドレスとプログラムの分岐結果を時系列に並べたグローバル履歴またはローカル履歴を入力として用いる．まずターゲットアドレスをインデックスとして重みテーブル(Weight Table)から重みベクトルを読み出す．この重みベクトル

は各要素が分岐履歴の特定の場所と1対1に対応付けされている．分岐履歴の各要素は，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，対応する分岐履歴の要素を $X_i(1 \leq i \leq n)$ とし，$y = W_0 + \sum_{i=1}^{n} W_i X_i$ を計算する．W_0 は閾値であり，重みベクトルと一緒に読み出される．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 $t \in \{1, -1\}$ とすると基本的に $W_i = W_i + tX_i$ によって表される．本来，パーセプトロンモデルはパターンの判別が正しかった場合には重みの更新は行わない．しかし，パーセプトロン分岐予測器では更新閾値を超えない場合にのみ予測が正しかった場合でも更新を行う．

Global/Local Perceptron は予測精度は高いが，予測に必要となる計算量が多いためターゲットアドレスが入力されてから分岐予測結果が得られるまでのレイテンシが大きく，CPU などへの実装は現状では事実上不可能とされている．

2.2 Path-Based Perceptron

Path-Based パーセプトロン分岐予測器[3] は Global/Local パーセプトロン分岐予測器の大きなレイテンシを隠蔽するためのパイプライン構造を持つ．重みを読み出すためのインデックスに実行パス履歴を用い，パイプライン化による複数の予測計算を同時に行う．しかし，過去の実行パス履歴を用いているために，プログラムが辿る実行パスが変化した場合には，無関係な値によって予測計算がなされてしまう．このため，履歴長を伸ばしても予測精度が向上しにくい．

2.3 Piecewise Linear Branch Predictor

Piecewise Linear 分岐予測器[4] はインデックスにターゲットアドレスと実行パス履歴の両方を使用する．特に Ahead Pipelined Piecewise Linear Branch Predictor[5] はターゲットアドレスのビット長を制限し，その全ての場合に対して並列でパイプライン化された予測演算を行うことにより，時間的には実装可能とする構造である．しかし，パイプライン化のトレードオフとして，Ahead Pipelined Piecewise 予測器は従来法よりも大量の加算器が予測演算に必要となる．また，予測精度向上の効果も従来法に比べて限定的である．

2.4 Path Trace Branch Predictor

Path Trace Branch Predictor(PTBP)[6] は，数種類のアドレス履歴を組み合わせたインデックスで重みを読み出すパーセプトロン分岐予測器の1種である．

Pipelined PTBP は各パイプライン（鎖と呼ぶ）の長さを h とすると1鎖目の $n(n \leq h)$ 番目のグローバル履歴に対応する重みは n 番目の実行パス履歴をインデックスとし，2鎖目の $n+h$ 番目のグローバル履歴に対応する重みは n 番目のインデックスと $n+h$ 番目の実行パス履歴をローテートして排他的論理和を取った値をインデックスとする．このように h 個毎にアドレス履歴の排他的論理和を取り，それをインデックスとすることにより，ターゲットの分岐命令までに辿ったパス情報を予測に反映することができ，この結果，Piecewise 予測器よりも高い予測精度を示す．

プログラム中にはグローバルな分岐履歴の一部が異なっても同じふるまいをする分岐命令が多数存在する．このような場合，予測に関係のない履歴によってテーブルの読み出し位置の一部が変化し予測に悪影響を与える．パーセプトロン分岐予測器では，この予測に関係のない履歴によって生成されるインデックスで読み出される重みは予測に相関のある重みと異なるものとなってしまう．PTBP はその構造上，1つのアドレス履歴が従来のパーセプトロンに比べて多くの重みのインデックスに関わることになり，この問題が及ぼす PTBP への影響は他の分岐予測器に比較して大きい．その結果として予測精度の向上を妨げてしまうという問題点を持つ．

3. ターゲットアドレスと実行パス履歴を効率的に利用するパーセプトロン分岐予測器

前章で述べたとおり，Pipelined Piecewise 予測器は回路量の制限から．ターゲットアドレスの一部しか使用できず，ターゲットアドレスの持つ情報量の利用効率が悪い．よって，レイテンシに比較的影響を及ぼさない閾値読み出し以外では，ターゲットアドレスを使用することは現実的ではない．プログラムの局所性から，ターゲットアドレスの直近の実行パス履歴はターゲットアドレスの部分的な情報を持つと考えられる．本手法も PTBP と同様に閾値以外の重みの選択にターゲットアドレスの代わりに実行パス履歴を用いる．

実行パス履歴の情報をパイプライン構造のパーセプトロン分岐予測器で最大限に生かすため，まず，その時点で得ることができる最新の分岐命令アドレスと実行パス履歴をインデックスに反映する必要がある．次に，予測に関係のない分岐命令アドレスの悪影響を最小限に抑えるために，連鎖的にインデックスとアドレスの排他論理和を取ることは望ましくない．さらに，グローバル履歴の更新が未完了のまま予測結果を用いて投機的に次の予測を行うと，データの不正確さから予測精度は低下してしまう．投機的予測を可能な限り

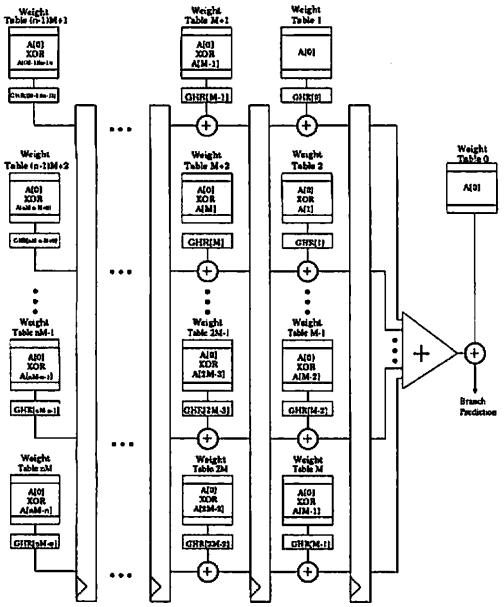

図 1　パス履歴を効率的に利用するパーセプトロン分岐予測器

避けるためには，予測時に更新未完了のグローバル履歴の個数を減らす必要がある．そのためには分岐命令の処理時間を稼ぐ必要があり，ターゲットアドレスに近い実行履歴ほどパイプラインの後方ステージで使用されることが重要である．

以上のことに留意し，**図 1** に示すパイプライン構造をつパーセプトロン分岐予測器を提案する．パイプライン段数を $h+1$，パイプライン 1 段で読み出す重みの数を M とすると，グローバル履歴長は hM である．ここでは，ある重み読み出しのインデックス生成に使用するのは，ターゲットアドレスではなく使用し得る最新のアドレスと実行パス履歴である．たとえばテーブル M では，このパイプラインステージで使用し得る最新のアドレス $A[0]$ と実行パス履歴 $A[M-1]$ の排他的論理和を取り，重み読み出しのインデックスとする．また，ターゲットアドレスに近い実行履歴ほどパイプラインの後方で使用するため，例えば図 1 の後ろから 2 段目のパイプラインではターゲットアドレス至近の履歴 0 から履歴 $M-1$ までのグローバル履歴と実行パス履歴を用いて予測計算を行っている．また，閾値は加算の前に定数と乗算する．これは閾値の予測における影響力を強めるために行う．乗算はビットシフトにより行うため，余分な回路の追加は必要ない．

4. ローカル履歴の付加

ローカル履歴は以前から分岐予測器で使用されている入力であり，Global/Local パーセプトロン分岐予測器ではローカル履歴を使用することにより，グローバル履歴のみを用いた場合よりも予測精度が向上する．しかしながら，パイプライン化不能という構造上の問題で Path-Based 予測器など，実装に配慮しているパーセプトロン分岐予測器でローカル履歴が利用された例はない．

ここではパイプライン化が可能なローカル履歴の利用手法を提案する．ローカル履歴長を l とし，ターゲットアドレスの下位 l ビットを削り，そこにローカルヒストリテーブル (LHT) から読み出したローカル履歴を付加したものを多重化アドレス (Multiplex Address, MA) と呼ぶことにする．本手法では，閾値の読み出しに多重化アドレスを利用する．また，多重化アドレスをアドレスの代わりに実行パス履歴として記憶しておき，重みの読み出しにも使用する．このような手法でパイプライン化を実現する．ローカル履歴は直接的に予測演算に使用されることはないが，インデックスに加えることで 1 つのアドレスからローカル履歴長のべき乗通りのインデックスが作成可能になる．これにより，たとえば 1 つの分岐命令が両方の分岐方向を持つ場合に，提案手法はローカル履歴によって使用する重みを変更することができる．すなわち，従来法では不可能であった重みの負の衝突を回避することができる．したがって，ハードウェアコストの増加を最小限に抑えつつ予測精度を向上できる可能性がある．

この手法を実装する場合の問題点として，ローカル履歴を読み出した後に閾値を読み出す必要があることがあげられる．しかし，ローカル履歴の読み出しと平行してターゲットアドレスで l 個の閾値群を一括して読み出した後そのうちの 1 つをセレクタで選択すれば，セレクタ分のみのレイテンシの増加で，ローカル履歴と閾値の読み出しを平行して行うことができる．また，LHT にはローカル履歴の値をあらかじめデコードして保持することにより，閾値選択に必要なレイテンシを抑えることができる．

提案したパーセプトロン分岐予測器を Advanced Anti-Aliasing Perceptron Branch Predictor(A^3PBP, A3PBP) と名付ける．**図 2** は A^3PBP の回路模式図，**図 3** は擬似ソースコードである．

最後段のパイプラインステージではターゲットアドレスをインデックスとして閾値の候補 4 個が同時に読み出される．それと平行して，LHT からローカル履歴を読み出し，この値を用いて閾値を選択する．閾値と途中計算結果を加算し，予測結果を出力する．最後

```
GHR[0...nM − n]                                  グローバルヒストリレジスタ
Index[1...n, 1...M]                              重みテーブルのインデックス
MA[0...nM − n]                                   多重化アドレス
Reg[1...n, 1...M]                                パイプラインレジスタ
W[1...nM, 0...L × 2^(LHbit)]                     重みテーブル
W0[1...L, 0...2^(LHbit)]                         閾値の重みテーブル
LHT[0...K]                                       ローカルヒストリテーブル
output := W0[A[0], LHT[A[0]]]
output := (output_is)                            s ビットシフト
for i = 1 to M do
  output := output + Reg[1, i]
end for
prediction := (0 ≤ output)
NewMA := ((A[0] >> 2) << 2) or LHT[A[0]]         多重化アドレスを生成
(GHR << 1) or Prediction
for i = 1 to n do                                i はパイプラインステージを表す
  for j = 1 to M do                              j は鎖の番号を表す
    if i = 1 and j = 1 then Index[1, 1] := A[0]  Table 1 だけはアドレスのみで読み出し
    else Index[i, j] := MA[(i − 1) * M − i + j] xor A[0]
    if GHR[(i − 1) × M − i + j] = 1 then         グローバルヒストリの値に応じて重みを加減算
      Reg'[i, j] := Reg[i + 1, j] + W[(i − 1) × M + j, Index[i, j]]
    else
      Reg'[i, j] := Reg[i + 1, j] − W[(i − 1) × M + j, Index[i, j]]
  end for
end for
Reg := Reg'                                      パイプラインレジスタを更新
MA := (MA << 1) or rot(NewMA, 1)                 多重化アドレスを更新
```

図 3 A^3PBP の予測アルゴリズム

段以外のパイプラインステージでは，最新のアドレスと実行履歴の代わりに多重化アドレスの排他的論理和を取り，重み読み出しのインデックスとする．ただし，図 2 の Table 1 は多重化アドレスを用いず，アドレス A[0] でインデックスを生成している．これは Table 1 で使用する多重化アドレスの生成のタイミングと多重化アドレスの使用のタイミングが同じだからである．このため，レイテンシを増やさないためには Table 1 の重み選択には多重化アドレスを使用せずにアドレスのみで読み出す必要がある．

読み出された重みはパイプラインの前段から送られてきた計算途中の値と加算器によって加減算される．パイプラインの最後段では M 個の重みの途中計算結果を足し合わせる必要があるが，閾値やローカル履歴の読み出しとオーバーラップさせて計算させることでその計算時間を隠蔽することができる．

5. A^3PBP の予測精度の評価

A^3PBP の予測精度をシミュレーションによって測定する．また，2 つの提案手法がそれぞれ個別に利用される場合の効果を調べるため，パス履歴を効率的に利用するパーセプトロン分岐予測器 (New Pipelined PBP と呼ぶ)，Pipelined PTBP に多重化アドレスを付加した構造 (Pipelined PTBP with MA と呼ぶ) の予測精度も測定する．提案手法との評価対象として Global/Local Perceptron 分岐予測器，Path-Based Perceptron 分岐予測器，Ahead Pipelined Piecewise Linear 分岐予測器，Pipelined PTBP を用いる．

実験環境は SPEC CINT2000 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各分岐予測器をシミュレートしたプログラムで処理し，その予測ミス率

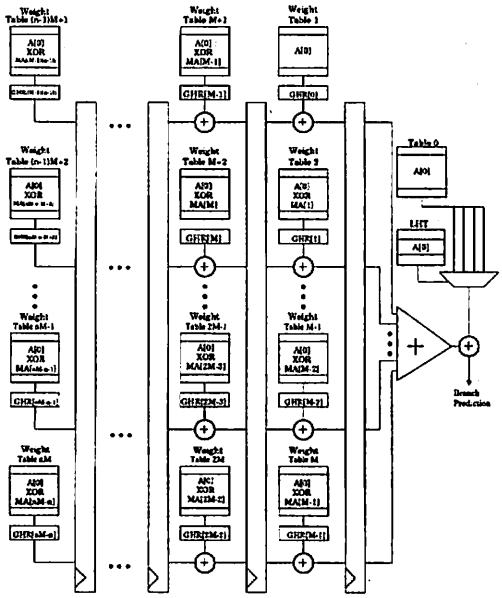

図 2 A^3PBP の構造図

表 1 各分岐予測器の履歴長

Budget size	Global Local	Path Based	Piece wise	PTBP	New Arch	A^3 PBP
8KB	34/12	32	20	22	32	16
16KB	38/14	34	24	26	36	26
32KB	40/14	34	26	32	48	36
64KB	50/18	37	43	43	52	43
128KB	60/20	40	50	50	68	63
256KB	70/20	42	51	51	72	63

を測定した．評価する分岐予測器は8KB から 256KB までの 6 段階で記憶容量を制限し，同容量での予測ミス率の違いを評価する．表 1 は各分岐予測器が各記憶容量で使用するグローバル履歴長の値である．この値は文献[5)6)] を参考にした．重みは原則 8 ビットを使用しているが，A^3PBP の 8KB，16KB，32KB に関しては 7 ビットを使用した．また，A^3PBP における閾値のビットシフトは 8KB，16KB 構成時が 2bit，その他は 3bit である．パーセプトロン分岐予測器の学習閾値は全て 2.1 × (履歴長 +1) とした．Pipelined PTBP に多重化アドレスを付加した予測器の履歴長は表 1 の Pipelined PTBP と同じである．A^3PBP の重み同時読み出し数 $M = 16$，ローカル履歴長 $l = 2$ とした．比較に用いた Pipelined PTBP の鎖数は 16 とした．

図 4 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．A^3PBP 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．16 本鎖の Pipelined PTBP と比べ，平均で 5.3%，最高で 6.6% (32KB 構成時) の予測ミスを削減することができた．以上より，A^3PBP は他の分岐予測器よりもテーブルの使用効率が高いということがいえる．また，図 5 より New Pipelined PBP と Pipelined PTBP with MA を比較すると，後者の方が予測精度が高い．このことからパイプラインアーキテクチャの改良よりも MA の付加の方が予測精度の向上の程度が大きい．

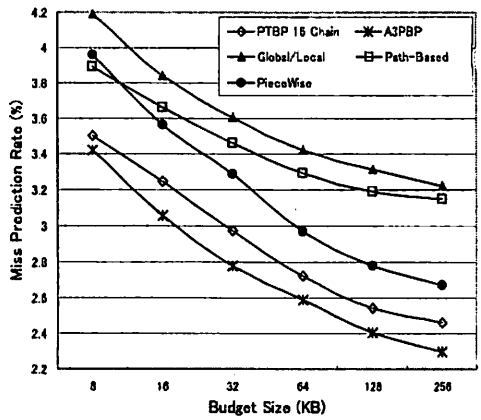

図 4 各分岐予測器の予測精度

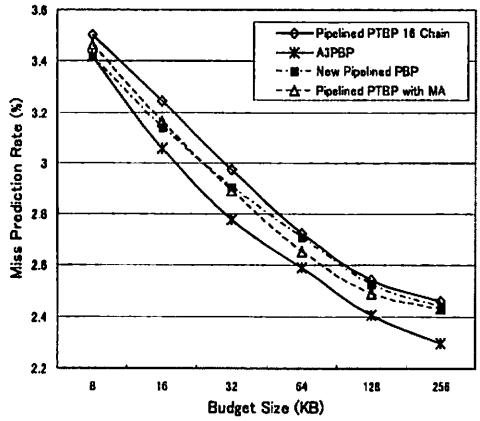

図 5 提案手法の個別利用時の予測精度

6. 提案手法の回路量とレイテンシの考察

A^3PBP に必要なハードウェア量を Pipelined PTBP と比較する．記憶量に関して A^3PBP の記憶量の増加分としては LHT1 個のみである．しかし，グローバル履歴長が $h$bit の時，LHT は重みテーブル全体の $1/2h$ でしかない．実験で使用した設定では 0.5%か

ら4%程度であり，重みテーブルと比べて非常に小さい．また，図4の実験ではA^3PBPでは付加するLHTの記憶容量を確保するために重みテーブルを同じ量だけ減らしている．このため記憶容量に差はない．

次に計算に必要な回路量を比較する．Pipelined PTBPは鎖1つに対して1つのインデックス計算でよいが，A^3PBPは重みテーブル1つに対して1つのインデックス計算が必要となる．このため，A^3PBPのインデックス計算のためのXOR回路はPTBPと比べて大きい．しかし，A^3PBPで必要とされるXOR回路は多重化アドレスのbit長×履歴長であり，予測器の大部分を占める重みテーブルに比べて非常に小さい．加算回路は同じ数の重みを使用する場合，Pipelined PTBPもA^3PBPも同じ回路量で実現できる．多重化アドレスを使用するために必要な回路は4-to-1セレクタ1つのみであり，これも予測器全体に比べて非常に小さい．このようにA^3PBPはPipelined PTBPと比べ，回路コストをほとんど増加することなく予測精度を向上させることができる．

分岐予測計算のレイテンシについて考察する．A^3PBPとPipelined PTBPはいずれもパイプラインの各ステージで重み読み出しと加算を行う構造である．重みテーブルの読み出し時間はテーブルの容量による．表1と図4から，A^3PBPはグローバル履歴長を伸ばすことによる予測精度の向上がPipelined PTBPに比べ顕著である．重みテーブル1つのエントリ数はグローバル履歴長と反比例するので，A^3PBPにおいては重みテーブルはPipelined PTBPより細かく分割され，重みテーブル1つあたりの大きさは相対的に小さい．したがってA^3PBPの重みテーブルの読み出し時間はPipelined PTBPのそれよりも小さくなると予想される．

最後段のパイプラインではTable 0からの閾値の読み出しと同時に鎖数分の累算値を加算する．テーブルからの読み出し時間は前述のようにA^3PBPの方が小さい．図4の実験条件では鎖の数は両者とも16個であり累算値のビット長は両者とも10bit程度である．累算結果は桁上げ保存形で得ればよく，桁上げ保存形の鎖数分の加算はCSA(Carry-Save Adder)6段で得られる．テーブルのサイズは少なくとも500エントリ程度であるから，CSA6段のレイテンシは閾値の読み出しのレイテンシと比較し，同等かそれ以下である．A^3PBPでは4つの閾値候補の読み出しの後にLHTから読み出された値によって閾値を選択する必要があるが，LHTは予めデコードされた値が保存されており，純粋に候補の中から1つを選ぶのみである．選択に要するレイテンシは，選択後に行われるCPA(Carry-Propagation Adder)による10bit程度の値の加算と比べると非常に小さい．また，最後段のパイプライン以外では，Pipelined PTBPはA^3PBPよりもファンアウトによるレイテンシが大きい．

以上より，A^3PBPとPipelined PTBPにおける分岐予測計算に要するレイテンシは同等といえる．

7. おわりに

本稿ではまず，実行パス履歴の情報を効率的に利用するパイプライン構造を提案した．また，ローカル履歴を重み選択のインデックスに付加することにより従来法よりも詳細なプログラムの経路把握を可能とする手法を提案した．この2つの手法を適用した新型パーセプトロン分岐予測器A^3PBPを提案し，シミュレーションにより評価した．予測ミス率は従来法の中で最もミス率が低いPipelined PTBPの16本鎖のミス率を最高で6.6%低減できることを示した．A^3PBPはPipelined PTBPと同程度のレイテンシで分岐予測計算を行うことができる．

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致します．

参考文献

1) D. A. Jimenez, and C. Lin : Dynamic branch prediction with perceptrons, In Proceedings of the Seventh International Symposium on High-Performance Computer Architecture (HPCA'01), pp.197-206(2001)
2) S McFarling: Combining Branch Predictions, WRL Technical Note TN-36,(1993)
3) D. A. Jimenez : Fast Path-Based Neural Branch Prediction, In the Proceedings of the 36th Annual IEEE/ACM International symposium on Microarchitecture (MICRO-36),pp.243-252(2003)
4) D. A. Jimenez: Idealized piecewise linear branch prediction, In The first JILP Championship Branch Prediction Competition (CPB-1),(2004)
5) D. A. Jimenez: Piecewise Linear Branch Prediction, In the Proceedings of the 32nd International Synposium on Computer Architecture (ISCA'05), pp.382-393(2005)
6) 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)
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).