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

2007－ARC－172 (42)
2007－HPC－109 (42)
2007／3／2

Cellプロセッサの分岐ペナルティを軽減するソフトウェア分岐予測の可能性検討

吉瀬謙二† 佐々木豊††

今日のパイプライン段数の多い高性能プロセッサにおいては，分岐予測ミスのペナルティが大きいために分岐予測の精度がその性能を左右する．しかし最近では，回路面積の削減などのためにハードウェアでおこなう分岐予測を簡素化した高性能プロセッサが市場に出荷されている．そのような簡素なハードウェア分岐予測しか持たないプロセッサでは，分岐予測ミスが多発することで性能を低下するおそれがある．本稿では，簡素なハードウェア分岐予測しか持たないプロセッサを対象として，従来はハードウェアでおこなっていた分岐予測をソフトウェアで実現するソフトウェア分岐予測の枠組みを提案する．また，Cellプロセッサに含まれるSynergistic Processor Elementにおけるソフトウェア分岐予測の可能性を検討する．バブルソートをベースにして飽和型2ビットカウンタ方式のソフトウェア分岐予測を実装する場合に，予測精度の向上および分岐予測ミスペナルティの削減が可能であること，最大で17%の性能向上を得られることを確認する．

A software branch prediction to reduce the branch penalty of the Cell Broadband Engine Processor

KENJI KISE† and YUTAKA SASAKI††

Accurate branch prediction is important for modern high performance processors. In order to improve the prediction accuracy, many hardware branch predictions have been investigated. On the other hand, a processor with very simple hardware branch prediction is appearing in a market. In this paper, we introduce the framework of software branch prediction that predicts branch outcome by software with minimal hardware support. We evaluate the framework on a Synergistic Processor Element of the Cell Broadband Engine Processor. Our preliminary experimental results using a bubble sort program show that a software branch prediction of a two-bit saturating counter gives better prediction accuracy and achieves the maximum performance gain of 17%.

1. はじめに

プロセッサが条件分岐命令をフェッチする際には，その分岐命令の結果 (分岐成立あるいは不成立) を予測して，その予測が正しいという仮定のもとで投機的に処理を進めていく．この仕組みは，分岐命令をフェッチした場合にもパイプライン実行を滞りなくおこなうために，多くの高性能プロセッサに備わっている．ここで，条件分岐命令をフェッチする際に，分岐結果を予測するハードウェア回路を分岐予測器と呼ぶ．本稿では，分岐予測器をソフトウェアにより実現する方式を提案するが，これとの混乱を避けるために，ハードウェアを利用して分岐結果を生成する一般的な枠組みを**ハードウェア分岐予測**と呼ぶことにする．

† 東京工業大学 大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology
†† 東京工業大学 工学部情報工学科
School of Engineering, Tokyo Institute of Technology

高性能プロセッサの多くは，命令発行の幅を広くして命令レベル並列性を抽出するとともに，命令パイプラインの段数を深くすることで動作周波数を向上させている．これらの傾向により，プロセッサ内で処理される命令数が増加するため，分岐予測のミスが発生した際にフラッシュすべき命令数，すなわち分岐予測ミスのペナルティが増加する．これらミスペナルティの影響を軽減するために，高性能プロセッサを実現する上で精度の高い分岐予測が不可欠となっており，過去数十年の間に洗練された様々なハードウェア分岐予測が提案され，実用化されている．

高い抽象度のレベルで記述した一般的なハードウェア分岐予測の構成を**図1**に示す．ハードウェア分岐予測は，予測すべき分岐命令のアドレス (Program Counter) を入力として受け取り，これと分岐履歴などの情報を保持するための記憶領域 (Storage) からのデータとの演算 (Computation) により，分岐方向 (成立あるいは不成立) を予測する．また，予測した分岐命

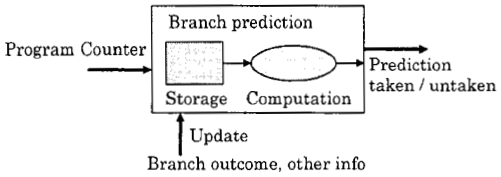

図 1　高い抽象度のレベルで記述した一般的な分岐予測の構成

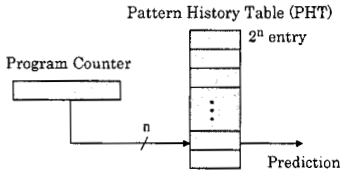

図 2　プログラムカウンタを利用して 2 ビット飽和型カウンタを選択する bimodal 分岐予測

令の結果が得られた時点で☆，その結果に基づいて分岐履歴などの情報を更新 (Update) する．ハードウェア分岐予測の例として，bimodal 分岐予測[1] の構成を**図 2** に示す．プログラムカウンタの一部をインデックスとしてパターン履歴表 (Pattern History Table, PHT) を参照する．PHT は 2 ビット飽和型カウンタの配列として実現される．選択された 2 ビットカウンタの値が 3 あるいは 2 の時に分岐成立，カウンタの値が 1 あるいは 0 の時に不成立と予測する．分岐結果が成立の場合には利用したカウンタをインクリメントし，そうでない場合にはデクリメントすることで情報を更新する．

近年，洗練された分岐予測を搭載して予測精度の向上をねらうという方向ではなく，回路面積や複雑さの削減やソフトウェアからの透過性の向上を狙い，ハードウェア分岐予測を簡素化した高性能プロセッサとして Cell プロセッサ (Cell Broadband Engine)[2] が市場に出荷されている．このような簡素なハードウェア分岐予測しか持たないプロセッサでは，分岐予測ミスが多発することで性能が低下するおそれがある．

本稿では，ハードウェア分岐予測を簡素化した高性能プロセッサを対象として，ソフトウェアで分岐予測を行うことで処理を高速化する手法を提案する．この手法を**ソフトウェア分岐予測**と呼ぶことにする．また，Cell プロセッサが搭載する Synergistic Processor Element(SPE)[3] を対象に，ソフトウェアで飽和型 2 ビットカウンタを実装・評価することでソフトウェア分岐予測の可能性を検討する．

本稿では，分岐成立あるいは不成立の 1 ビットの予測を必要とする条件分岐命令のみを予測の対象とする．

☆ 履歴は予測した結果を用いて投機的に更新することもある．

飛び先アドレスをレジスタに格納して，そのアドレスに分岐するレジスタ間接分岐命令や，常に分岐する無条件分岐命令を予測の対象には含めない．分岐成立のことを taken，分岐不成立のことを untaken と記述することがある．

以降，2 章でソフトウェア分岐予測を提案する．3 章ではバブルソートの例を用いてソフトウェア分岐予測の可能性を検討する．4 章にて議論し，5 章でまとめる．

2. ソフトウェア分岐予測の提案

2.1 ソフトウェア分岐予測

ハードウェアによる高度な分岐予測を備えていないプロセッサにおいて，ハードウェア分岐予測と同様の機能をソフトウェアによって提供する仕組みとしてソフトウェア分岐予測を提案する．精度の高い分岐予測を行う命令群をプログラムに追加することで，分岐ミスの回数と分岐ミスペナルティを削減し，プログラムの高速化を達成することがソフトウェア分岐予測の目的である．

ソフトウェア分岐予測では，予測のアルゴリズムをソフトウェアによって実現する．このため，演算により得られた結果を分岐予測として利用する基本的な枠組みが提供されていれば，ハードウェア構成を変更することなく，予測のアルゴリズムを変更できるという利点がある．このため，アプリケーションプログラムに適した予測アルゴリズムを選択すること (**アプリケーション毎の適用性**) により高い予測精度を達成できる可能性がある．加えて，プログラム内の個々の分岐命令に対して適した予測アルゴリズムを適用すること (**分岐命令毎の適用性**) により高い予測精度を達成できる可能性がある．

2.2 ソフトウェア分岐予測の枠組み

ソフトウェア分岐予測を実現するためには，ソフトウェアが生成した予測をプロセッサに伝えるメカニズムが必要となる．これに適した仕組みとして SPE における分岐ヒント命令 (hbr 命令)[4] がある．この動作を次のアセンブラコードを用いて説明する．

```
1    ila $40, .LT # set the branch taken addr
2    hbr .LB, $40 # hint branch instruction
3 .LB:            # label of the branch
4    brz $2, .LT  # branch if $2==0
5 .LN:
```

4 行目の brz 命令を予測対象の条件分岐命令とする．この分岐命令ではレジスタ$2 の値がゼロの場合にラベル.LT に分岐する．3 行目のラベル.LB は，分岐命令を指定するために利用する．2 行目の分岐ヒント (hbr) 命令の最初のオペランドが分岐ヒントの対象となる分岐 (.LB) を指定し，2 番目のオペランド ($40) が分岐

命令の次にフェッチすべき命令のアドレスを指定する．この例では，1 行目の ila(immediate load address) 命令でレジスタ$40 に分岐成立のアドレスを格納し，分岐ヒント命令でこのレジスタを指定していることから，4 行目の分岐命令は分岐成立として.LT に制御が移動することとしてプロセッサは処理を進めることができる．

アセンブラコードにおいて，2 行目の分岐ヒント命令の 2 番目のオペランドとしてレジスタ番号を指定していることに注目する必要がある．この例では 1 行目の代入で分岐成立のアドレス (.LT) を設定したが，ここを不成立のアドレス (.LN) とすることで，分岐不成立と予測することを選択できる．この自由度があるため，ソフトウェア分岐予測では，何らかの演算により分岐方向を予測し，それに対応するアドレスをレジスタに設定して分岐ヒント命令を実行する．これにより，ソフトウェアが生成した予測をプロセッサに伝えることが可能となる．

ハードウェア分岐予測の構成を図 1 に示した．これと同様に，分岐履歴などの情報を保持するための記憶領域を確保し，予測のための演算処理，履歴情報の更新処理を追加することでソフトウェアによる予測を実現できる．分岐履歴などの情報を保持するための記憶領域としては，プログラムが使用していないレジスタあるいはメモリを利用する．

ソフトウェア分岐予測の実装においては，できる限り，追加する命令の中に条件分岐命令が含まれないように工夫する必要がある．例えば，複雑な予測アルゴリズムを実現するために多数の分岐が必要になる場合には，追加された分岐によるペナルティが増大して効果が得られないことが推測される．このため，成立あるいは不成立の予測によってレジスタの値を設定するような場合にはマスク命令などを利用して分岐命令を生成しないように対処する必要がある．

2.3 ソフトウェア分岐予測のオーバヘッド

ソフトウェア分岐予測を実現するためには，アプリケーションプログラムの中に，分岐予測のための幾つかの命令を埋め込む必要がある．これら予測の演算処理のためにプロセッサの機能ユニットなどのハードウェア資源を割り当てなければならない．またレジスタあるいはメモリといった資源も割り当てなければならない．これらの割り当てによりハードウェア資源の競合が発生する場合に，アプリケーションの処理時間が増加する．

これらのオーバヘッドを考慮すると，プログラムに含まれるすべての分岐命令に対して予測のコードを追加することは現実的ではない．プログラムのソースコードに含まれる情報やプロファイル情報などを利用して，予測が困難な分岐命令を選択して予測をおこなうことが現実的となる．例えば，過去の研究[5] から極端な偏りのある分岐の割合が多いことが示されている．これらの分岐については実行時の情報を利用して分岐方向を予測する必要はない．

```
 1 bubble() {
 2   int i,j,t;
 3
 4   for(i=MAX-1; i>=0; i--)
 5     for(j=1; j<MAX; j++ )
 6       if ( a[j-1] > a[j] ) {
 7         t = a[j-1];
 8         a[j-1] = a[j];
 9         a[j] = t;
10       }
11 }
```

図 3 Stanford-integer ベンチマークに含まれるバブルソートのソースコード

予測をおこなうことで分岐予測ミスのペナルティを削減できることが期待できる場合にソフトウェアによる分岐予測を適用するが，この場合にも，命令を追加することで実行サイクルが増加しないように，できるかぎり資源競合が発生しないように命令列をスケジューリングすることが好ましい．

3. ソフトウェア分岐予測の可能性検討

本章では，Cell プロセッサに含まれる Synergistic Processor Element(SPE) を対象として，ソフトウェア分岐予測の可能性を検討する．

3.1 SPE

Cell プロセッサは 1 個の Power Processor Element と 8 個の SPE を搭載する非対称型のチップマルチプロセッサである．本評価では，単一の SPE を対象として，ソフトウェア分岐予測の可能性を検討する．

SPE は洗練されたハードウェア分岐予測を備えていない．このため，通常は，分岐命令の結果は不成立として (branch not always) 処理を継続する．このように洗練されたハードウェア分岐予測がない代わりに，2.2 節で述べたように，分岐のヒントを与える命令 (hbr 命令) が存在する．この分岐ヒント命令を適切なタイミングで実行して，予測が当たった場合にはペナルティを受けることなく分岐命令を実行できる．

SPE のパイプライン段数は 26 段と長い．このため分岐ミスペナルティは 18 サイクルと大きくなっている．このような構成から，SPE では分岐ミスを削減することで大きな性能向上を期待することできる．

3.2 飽和型 2 ビットカウンタ方式の実装

Stanford-integer ベンチマークに含まれるバブルソートを SPE をターゲットとしてコンパイルしてアセンブリコード得る．このアセンブリコードを手作業で編集することで飽和型 2 ビットカウンタ方式のソ

```
 1    il   $40, 2              # init 2BC=2
 2    il   $41, 2              # constant
 3    ila  $42, .LT            # taken addr.
 4    ila  $43, .LU            # untaken addr.
 5                             #
 6    cgti $44, $40, 1         # $44 = (2BC > 1)
 7    selb $44, $43, $42, $44  # set pred addr
 8    bg   $45, $40, $41       # $45 = 1
 9    hbr  .LB, $44            # hint branch
10                             #
11 .LB:                        #
12    brz  $2,.LT              # target branch
13 .LU:                        #
14    cgti $45, $40, 0         # $45 = -1
15                             #
16 .LT:                        #
17    a    $40, $40, $45       # update 2BC
```

図 4 2 ビットカウンタ方式を実装したバブルソートのアセンブリコードからの抜粋

フトウェア分岐予測を含むコードを作成する．バブルソートのソースコードを**図 3** 示す．6 行目に示す配列の隣り合う要素の大小を比較する部分で分岐予測ミスが多発する．この分岐命令を予測の対象とする．ここで実装する飽和型 2 ビットカウンタ方式は，1 つの分岐命令に対して 1 つのカウンタを割り当てることができる．このため，予測能力としては，図 2 に示した bimodal 分岐予測に近い．

2 ビットカウンタ方式を実装したバブルソートのアセンブリ命令列から，分岐予測のための命令列と予測対象の分岐命令を抜粋したコードを**図 4** に示す．分岐予測の対象となる命令は，12 行目に示す brz 命令である．この分岐が成立の場合にはラベル.LT に分岐する．

1 行目から 4 行目の命令ではレジスタの初期化をおこなっている．上から順番に，2 ビットカウンタ，飽和させるための定数，分岐先のアドレス，分岐しない場合のアドレスを設定する．

飽和型 2 ビットカウンタ方式では，カウンタの値が 0 あるいは 1 の時には不成立，2 あるいは 3 の時には成立と予測する．この予測は 6 行目と 7 行目の命令でおこなっている．6 行目の cgti(Compare Greater Than Word Immediate) 命令は，カウンタ ($40) の値が 1 より大きい場合に$44 のすべてのビットを 1 に設定し，そうでなければ 0 に設定する．7 行目の selb(Select Bits) 命令によって，$42 の分岐先アドレスあるいは $43 の不成立のアドレスがレジスタ$45 に格納される．8 行目が分岐ヒント命令で，予測されたアドレス ($45) を利用して分岐命令のヒントを与える．

残りの 8 行目，14 行目，17 行目がカウンタの更新の処理である．8 行目の bg(Borrow Generate) 命令では分岐成立を想定して 2 ビットカウンタが飽和していない場合に$45 に更新値 (+1) を代入しておく．14 行目の命令は分岐不成立の時のみ実行されるので，ここで，飽和していない場合に不成立の場合の更新値 (-1) で$45 を上書きする．最後に，17 行目でカウンタの値と$45 との和をとることでカウンタの更新を完了する．

このように実装した飽和型 2 ビットカウンタでは 4 本のレジスタを必要とする．また，命令数は，レジスタの初期化に 4 命令 (これはループの外側に配置するので実行回数は 1 回と性能に影響しない)，分岐予測の実行に 3 命令とカウンタの更新に 3 命令を必要とする．レジスタの初期化の部分はバブルソートの実行の隙間に隠すことはできなかったが，他の命令はすべて SPE の実行の隙間に挿入することができた．このため，命令列が増加したことによる実行サイクル数の増加はそれほど大きくなっていない．

表 1 分岐予測方式の違いによるバブルソートの評価結果

N	予測	total cycle	miss penalty	hit rate
100	BNA	550×10^3	97.6×10^3	45.2%
	BA	528×10^3	80.5×10^3	54.8%
	2BC	473×10^3	6.7×10^3	96.2%
1000	BNA	55.0×10^6	9.08×10^6	49.5%
	BA	54.3×10^6	8.90×10^6	50.5%
	2BC	47.0×10^6	0.71×10^6	99.6%

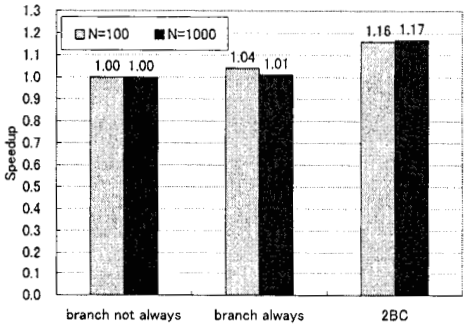

図 5 ソフトウェア分岐予測によるプロセッサ性能の向上

3.3 2 ビットカウンタ分岐予測の評価結果

ソフトウェア分岐予測の評価結果を**表 1** にまとめる．比較のために，実装した 2 ビットカウンタ方式 (2BC) に加えて，1 個の分岐ヒント命令を加えて作成した branch always 方式 (BA) と，何も手を加えていない branch not always 方式 (BNA) の 3 つの結果を示す．バブルソートの問題サイズ N は 100 もしくは 1000 とする．ソートの入力はランダムな値で初期化している．IBM Cell BE SDK1.0 に含まれる Full System Simulator for the Cell Broadband Engine Processor を利用して，各プログラムの実行サイクル数などを計測する．単一の SPE をターゲットとして実行ファイルを生成しているので，スレッドを起動するための処理は含まれない．4 列目に示した miss

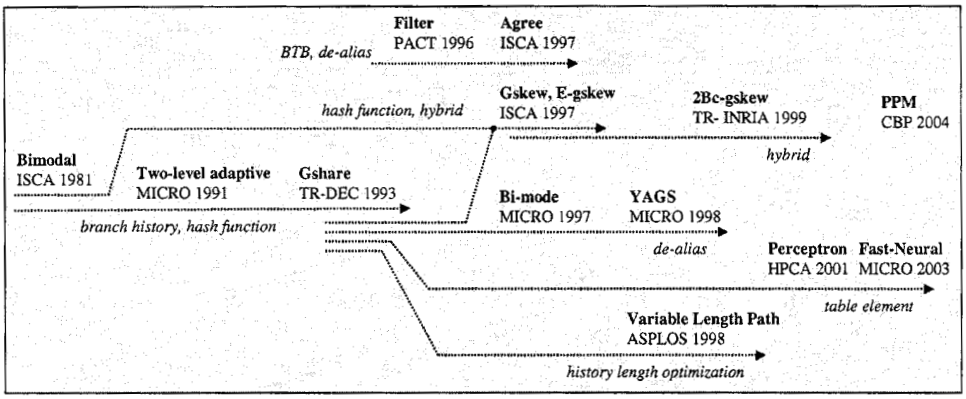

図 6　提案された年代と方式の相関を考慮して描いた代表的なハードウェア分岐予測の分類

penalty は，予測の対象となった分岐命令における分岐ミスペナルティのサイクル数である．5 列目に示した hit rate は，予測の対象となった分岐命令の予測成功率である．3 列目に示した total cycle は，プログラムの処理に必要となったサイクル数である．

表 1 の結果を検討する．hit rate から明らかなように，branch not always 方式および branch always 方式では高々55%の予測成功率しか達成できていない．一方，2 ビットカウンタ方式を用いたソフトウェア分岐予測の場合には 95%以上の高い分岐予測のヒット率を達成できている．特に，N=1000 と問題サイズを大きくした場合には，99%を超える高いヒット率を達成している．次に miss penalty を比較すると，ソフトウェア分岐予測では，分岐成功率の向上に伴い，分岐ミスペナルティが他の 2 つの方式の 10 分の 1 以下と大幅に削減できていることがわかる．また，分岐ミスペナルティの削減にともなって，実行サイクル数が減少する．

次に，ソフトウェア分岐予測によるプロセッサ性能の向上を**図 5** にまとめる．ここでは，branch not always 方式の性能を 1 とした相対性能で示している．ここから，バブルソートの問題サイズ N=100 の時に 16%，N=1000 の時に 17%の性能向上を達成することがわかる．

これらの評価結果より，実装した 2 ビットカウンタ方式のソフトウェア分岐予測により，予測精度の向上および分岐予測ミスペナルティの削減を達成し，最大で 17%という高い性能向上を得られる可能性があることを確認できた．

4. 議　　論

本稿では，従来はハードウェアでおこなっていた洗練された分岐予測をソフトウェアで実現するソフトウェア分岐予測の枠組みを提案した．また，その可能性を検討するために，バブルソートを例として，手作業にて 2 ビットカウンタ方式を実装して性能を評価することで，その可能性を検討した．

図 6 に，提案された年代と方式の相関を考慮して描いた代表的なハードウェア分岐予測の分類を示す．このように，様々な洗練されたハードウェア分岐予測が提案されている．また，それに伴って予測精度の向上が達成されている．一方，本稿にて検討した飽和型 2 ビットカウンタ方式は，図では左端の Bimodal に対応しており，古典的な構成をもつハードウェア分岐予測をソフトウェアで実装したに過ぎない．

今後の課題を列挙する．

- 図 6 に示したように様々なハードウェア分岐予測が提案されている．ソフトウェア分岐予測の視点から，これらの予測アルゴリズムの得失を検討する必要がある．
- ソフトウェア分岐予測における予測と更新のアルゴリズムはハードウェア分岐予測のそれを参考にすべきであるが，それに捕らわれる必要はない．ソフトウェア分岐予測に適した方式を検討する必要がある．
- プログラムに含まれる全ての分岐を予測の対象とすることは現実的ではない．プロファイル情報などを利用してソフトウェア分岐予測を適用する分岐命令を選択するための仕組みを開発する必要がある．
- ソフトウェア分岐予測が持つアプリケーション毎の適用性，あるいは分岐命令毎の適用性を利用することで予測精度を大幅に改善できる可能性がある．これらを利用する枠組みの開発が必要である．
- ソフトウェア分岐予測を実現するためには多数の命令を追加する必要があるとともに，追加による

実行時間の増大を抑えたい，適切なタイミングで予測を得る必要があるといった制約を満たす必要がある．これらの作業を自動でおこなうスケジューラの開発が必要である．

- 本稿では1つの分岐命令を対象として，その予測と更新をおこなう命令列を追加する枠組みを提案した．一方，ソフトウェアで予測することによるオーバヘッドを削減するためには，複数の分岐命令の予測をまとめておこなうといった最適化が考えられる．これらを考慮しながらソフトウェア分岐予測の枠組みを拡張することが必要となる．

5. ま と め

本稿では，簡素なハードウェア分岐予測しか持たないプロセッサを対象として，従来はハードウェアでおこなっていた洗練された分岐予測をソフトウェアで実現するソフトウェア分岐予測の枠組みを提案した．また，Cellプロセッサに含まれるSynergistic Processor Element(SPE)におけるソフトウェア分岐予測の可能性を検討した．

Stanford-integerベンチマークに含まれるバブルソートのアセンブリコードをベースに，手作業により飽和型2ビットカウンタ方式のソフトウェア分岐予測を実装し，その性能を評価した．この結果から，ソフトウェア分岐予測を追加することで，予測精度の向上および分岐予測ミスペナルティの削減が可能であること，最大で17%という高い性能向上を得られる可能性があることを確認した．

謝　　辞

本研究の一部は，科学研究費補助金若手研究(B)課題番号18700042「投機技術を積極的に利用するチップマルチプロセッサに関する研究」の助成による．

参 考 文 献

1) James E. Smith: A study of branch prediction strategies, *Conference proceedings of the eighth annual symposium on Computer Architecture*, pp. 135–148 (1981).
2) H. Peter Hofstee: Power Efficient Processor Architecture and The Cell Processor, *The Eleventh International Symposium on High-Performance Computer Architecture*, pp. 258–262 (2005).
3) J.A.Kahle, M.N.Day, H.P.Hofstee, C.R.Johns, T.R.Maeurer and D.Shippy: Introduction to the Cell multiprocessor, *IBM J. Research and Development*, Vol. 49, No. 4/5, pp. 589–604 (2005).
4) Sony Computer Entertainment Inc.: Synergistic Processor Unit 命令セット・アーキテクチャ.
5) 吉瀬謙二，片桐孝洋，本多弘樹，弓場敏嗣: Bimode-Plus 分岐予測器の提案, 情報処理学会論文誌コンピューティングシステム, Vol.46, No.SIG 7(ACS 10), pp. 85–102 (2005).