(1997. 10. 29)

大規模データパス・プロセッサにおけるフェッチ機構の検討

辻　秀典† 中村友洋† 吉瀬謙二†
安島雄一郎† 田中英彦†

命令レベル並列性を利用したプロセッサにおいて、分岐命令を効率良く処理することは重要な課題である。このような制御依存性の影響の回避は、大規模データパス・プロセッサにおいて、より重要である。そのための効率的な分岐制御を行う機構をもち、大規模に選択的な投機的フェッチを行う。その機構には多段の分岐命令をまたいだ実行パス予測が必要である。コントロールフロー・パスを利用した分岐予測に基づき、複数の予測パスをあげる手法について提案する。さらに、提案する手法によってパス予測成功率が向上することを確認した。

A study of the fetch mechanism for Very Large Data Path processor

HIDENORI TSUJI,† TOMOHIRO NAKAMURA,† KENJI KISE,†
YUICHIRO AJIMA† and HIDEHIKO TANAKA†

It is important for instruction level parallelism microprocessors to handle branch instructions efficiently. Avoiding these control dependency is more important for VLDP processor. For that purpose, it has the efficient branch control mechanism fetch instructions speculatively on a large. There is a need of a path prediction over multiple branches for the mechanism. We propose a prediction method which selects multiple paths by branch prediction based on control flow path. And we show that this method improves path prediction hit rate.

1. はじめに

今日、計算機の性能向上はめざましい。これを支えているのは計算機の中心構成要素であるマイクロプロセッサであり、1971年に4004が誕生して以来、マイクロプロセッサが常に進化を続けてきた結果でもある。命令レベル並列性を利用したマイクロプロセッサが一般的となった現在、さらなる性能向上を求めて次世代のプロセッサに関する研究が盛んに行われている。

これまでのような手法による命令レベル並列性の利用に限界が見えはじめている[3)5)]ことを背景に、チップ上にある程度の規模のプロセッサを複数載せ並列実行を目的とするオンチップ・マルチプロセッサ[8)1)]、プロセッサとメモリシステムやその他のシステムとの融合[13)10)7)]等、様々なアーキテクチャが研究されている。これらの研究は、性能向上を目指すだけでなく、設計、製造プロセスの観点からプロセッサの複雑化を避けつつより高い集積度を利用するための新しいアプローチでもある。

また、IBM/360やCDC-6600[2)]等に代表される大型計算機の時代から、計算機の基本アーキテクチャに変化がないことが、マイクロプロセッサの性能向上の限界の原因の一つであるという観点に立ち、新しいアーキテクチャを提案する[6)]アプローチも存在する。これに基づいて提案されたアーキテクチャが大規模データパス・アーキテクチャ[11)]である。

2. 大規模データパス・アーキテクチャ

大規模データパス・アーキテクチャ(以降VLDPアーキテクチャ)はマイクロ・プロセッサ単体の性能向上を目指すアーキテクチャである。大規模なハードウェア資源を利用できることを前提として、多くの演算器とそれを接続するための大規模データパス、そしてそれらを効率よく利用するための命令列制御機構をもつ。さらに、性能向上のみにとどまらず、ユーザの立場からの実行コード互換性に関する強い要望に対しても対応が考慮出来るようなアーキテクチャである。図1にVLDPアーキテクチャのブロック図を示す。VLDPアーキテクチャは以下の主要な3機構により構成されている。

(1) コントロールフロー先行展開
(Control Flow Pre-Construction)

† 東京大学大学院 工学系研究科
Graduate school of Engineering, The University of Tokyo

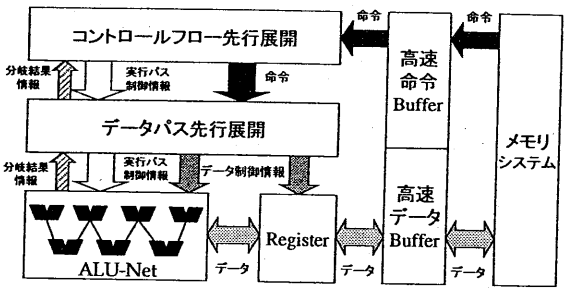

図1　VLDP アーキテクチャのブロック図
Fig. 1　block chart of VLDP architecture

(2)　データパス先行展開
(Data-Path Pre-Construction)

(3)　ALU-Net

コントロールフロー先行展開は、VLDP プロセッサにおけるフェッチ機構であり大規模な投機的フェッチを行う。これにより制御依存関係の解消された命令列を連続的に供給する。データパス先行展開は、VLDP プロセッサにおける命令解析、実行準備機構であり、制御依存関係の解消された命令列からデータパスを抽出する。抽出した情報に基づき効率的な ALU-Net の接続を行う。ALU-Net は多数の演算器とそれを接続するネットワークで構成され、その制御はデータパス先行展開により行われる。ALU-Net にデータフローが効率良くマッピングされデータが流れることにより演算が行われる。最終的に実行情報はコントロールフロー先行展開、データパス先行展開のそれぞれにフィードバックされる。

2.1　コントロールフロー先行展開

コントロールフロー先行展開は、命令をフェッチしつつコントロールフロー・パスの構築を行い、このコントロールフロー・パスに基づいた情報の管理を行う。

命令フェッチは投機的に行われるが、この投機的フェッチが分岐予測に基づく分岐確率よって選択的に行われる。結果として、図 2 に示すような投機的にフェッチされるパスとされないパスが存在するコントロールフロー・パスが構築され、これに基づき選択を行うための分岐確率情報が管理される。この図において矢印は 1 命令であり、実線が実際にフェッチされた部分、点線が命令は存在するがその時点においてフェッチされていない部分である。さらに (info) は分岐命令のための分岐確率情報であり、細い曲線の矢印はこの分岐確率情報の依存関係を表している。

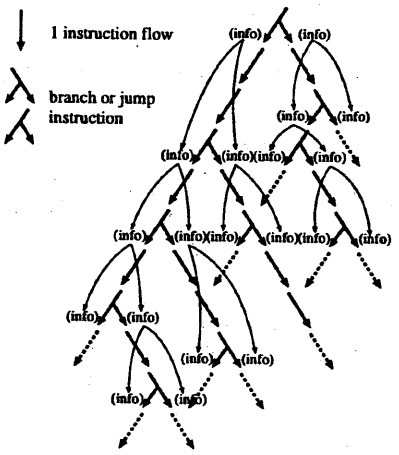

図2　コントロールフロー先行展開のイメージ
Fig. 2　image of Control Flow Pre-Construction

本稿では、そのようなコントロールフロー・ツリーに基づき実行確率の高いパスを予測する方法とその有効性について評価を行う。

なお、本稿で言及される命令セットおよびサンプルプログラム解析は、SPARC version 8[4] アーキテクチャによる。また、解析に用いたサンプルプログラムは SPEC92 等から cc1, compress, dhrystone, espresso, sc であり、それぞれのプログラムの 1 千万命令実行を評価の対象としている。

3.　コントロールフローに基づくパス予測

実行パスの予測を行うためには、多段の分岐命令をまたいだ分岐予測 (以降、マルチレベル分岐予測[9]) を行わなければならない。既存の分岐予測を多段に渡って行うことによっても実行パスの予測は可能であるが、予測候補が 1 つに限定されその予測成功率は段数分の積となる。そのような予測成功率を持つパスが 1 つ予測出来たとしても、実用的であるとは考えられない。

先に述べたコントロールフロー先行展開は、複数の予測候補をあげることにより予測成功率の向上を得ようとするものである。そして、複数の予測候補をあげるために、情報管理のためのコントロールフローを構築する。このコントロールフローはそれぞれの予測候補の実行確率を管理する。具体的には分岐命令の分岐先すべてに分岐予測に基づく重み付けを行い、その重みを用いて予測候補の実行確率を得る。その実行確率の高いものから予測パスとすることによって、複数の予測候補をあげることのできるパス予測を行う。

以降、各種分岐命令の重み付けに関して考察する。

3.1　分岐命令について

まずはじめに、分岐命令について整理する。分岐命令は 表 1 に示す 4 種類である。

- CALL 命令の分岐先は単一であり、そのアドレスは静的に決定している。
- Bicc, FBfcc, CBccc 命令の分岐先は 2ヶ所以下で

種類	飛び先	命令
無条件	即値	CALL
	レジスタ間接	JMPL,RETT
条件	即値	Bicc, FBfcc, CBccc
トラップ	レジスタ間接	Ticc

表 1　分岐命令の種類
Table 1　branch instruction types

PC	Status	
PC01	1	1
PC04	1	0
PC09	0	1
PC15	0	0
PC23	1	1

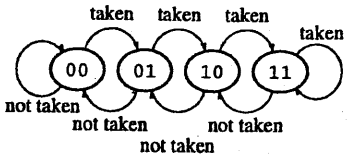

図3　BHT と状態遷移
Fig. 3　BHT and state change

あり、それらのアドレスは静的に決定している。また、分岐先は条件により動的に決定する。

- JMPL 命令の分岐先は複数であり、そのアドレスは動的に決定する。
- Ticc 命令はソフトウェアトラップを引き起こす。

以後 Bicc, FBfcc, CBCcc をまとめて Bicc、JMPL, RETT をまとめて JMPL と表現する。

評価においては CALL 命令は静的に削除できるため削除して考える。さらに Ticc 命令は数が少ないことから無視し、予測パスが Ticc 命令において分断された場合は評価の対象外とする。本稿では Bicc および JMPL 命令に限定して、実行パス予測および評価を行う。

3.2　条件分岐命令の重み付け処理

条件分岐命令の分岐先候補は静的に決定しており、条件が動的に決定する。通常の条件分岐予測では、BHT (Branch History Table) 等で条件の履歴を取ることによって予測を行う。現在、BHT を基本とした多くの予測方式が存在する。

図3 において BHT と履歴エントリの状態遷移の一例を示す。BHT は条件分岐命令のアドレスをタグとする表であり、該当する条件分岐命令の条件成立、不成立に応じて表の右の状態遷移図に従い履歴を更新する。通常は履歴の最上位ビット値をもって分岐成立、不成立の予測を行う。この履歴を値として用いることにより分岐先の重み付けを行う。例えば、履歴値を n、履歴ビット数を m とした場合、分岐成立方向の重みを n、分岐不成立方向の重みを $(2^m+1)-n$ とする。

3.3　レジスタ間接分岐命令の重み付け処理

レジスタ間接分岐命令は分岐先アドレスが動的に変化し、分岐先アドレスは 1 つのものから複数のものまでさまざまである。このように、動的に決定する複数の分岐先をもつレジスタ間接分岐命令の分岐先予測は困難である。通常 BTAC (Branch Target Address Cache) を用いて飛び先アドレスをキャッシングすることにより分岐予測を行うが、その予測成功率は高いとはいえない。さらに BTAC では複数の分岐先候補をあげることは不可能である。そのため、複数の分岐先候補をあげるために multi-BTAC が提案[12] されている。本稿では、さらにこれに重み付けを行った理想的な multi-BTAC (以降 ideal multi-BTAC) を提案し、これらを用いて評価を行う。

multi-BTAC　通常 BTAC にキャッシングできる分岐先アドレスは 1 つであるが、これを複数もつことができるように拡張したものが multi-BTAC である。図4 に示す左の表が multi-BTAC であり、分岐命令をタグとして分岐先履歴を管理する。さらに、本稿の条件では multi-BTAC の分岐先履歴数は無限で可変とする。分岐先履歴は最近分岐したアドレス順にソートされており (LRU) その順に分岐する可能性の高い分岐先候補と予測する。

ideal multi-BTAC　multi-BTAC に加え、分岐命令毎に分岐先履歴に対する表 (図の右の表) を持ち、それぞれの分岐先への分岐回数を保持している。右の表最上段 24 は PC 値 XX の分岐命令の分岐先アドレス target81 に分岐した回数である。これによって分岐先アドレスの重み付けを行い、この値の大きいほど分岐する可能性の高い分岐先候補と予測する。multi-BTAC の性能限界値を得るためのひとつとして提案した方式である。

図5 に、評価に用いるサンプルプログラムの飛び先数の分布を示す。これより、飛び先は複数存在するもののその分布は偏っており、それほど多くの分岐先に分散することは少ないことが分かる。よって JMPL

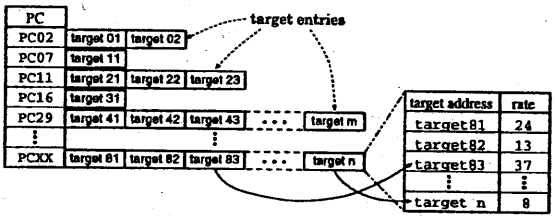

図4　multi-BTAC と理想化した multi-BTAC
Fig. 4　multi-BTAC and ideal multi-BTAC

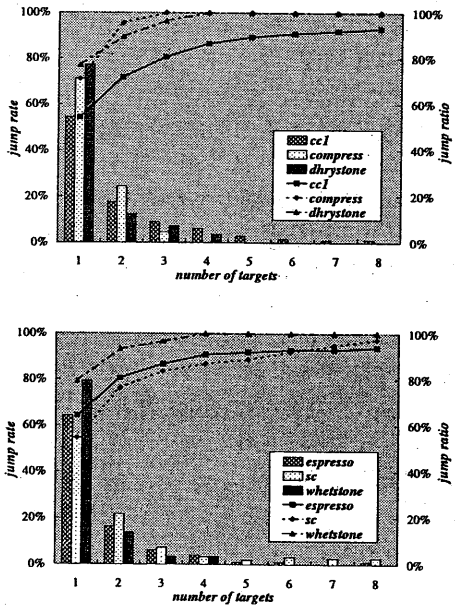

図5　分岐先分布
Fig. 5　distribution of branch targets

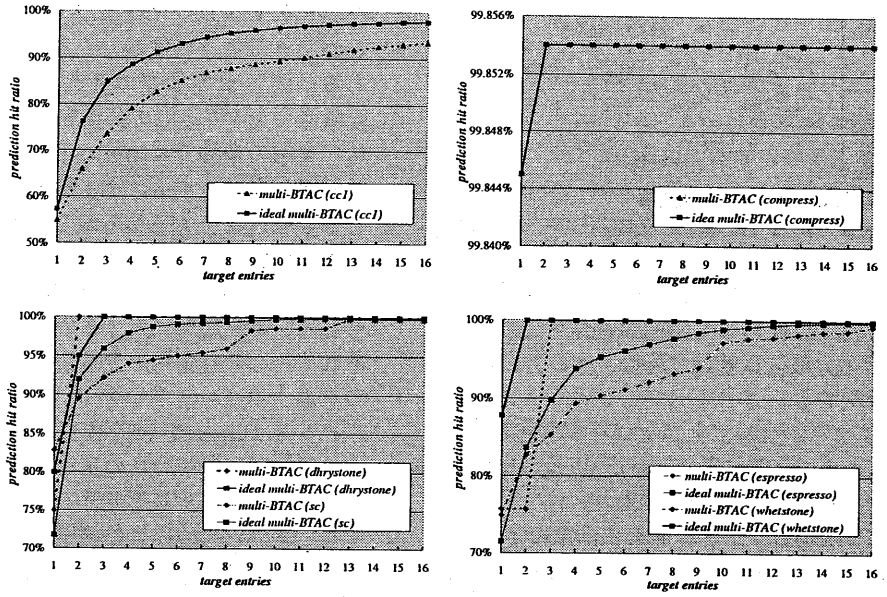

図6　multi-BTAC による分岐先予測成功率
Fig. 6　hit rate of target prediction by multi-BTAC

命令に関しても多くの分岐先候補を選択するのではなく、Bicc 命令の処理と同様に少ない分岐先候補の選択を行えばよいと思われる。

multi-BTAC および ideal multi-BTAC について分岐先予測成功率の評価を行った。その結果を 図6 に示す。用いたサンプルプログラムは既に述べた 6 プログラムそれぞれの、最初 1000 万命令実行における結果である。複数の分岐先候補をあげ、その中に実際の分岐先を含んでいた場合を予測成功とする評価である。なお、エントリ数 1 の場合の multi-BTAC は通常の BTAC と等価である。multi-BTAC (ideal multi-BTAC) によって BTAC の予測成功率を改善し、かつ、複数の分岐先候補をあげることができる。

3.4　重みを利用したパス予測方法

これまでに示した、重み付けを行う BHT と multi-BTAC を組み合わせることにより複数の Bicc および JMPL 命令を越えたパス予測を行う。

はじめに、分岐先候補の集合によってコントロールフロー・ツリーの構築を行う。Bicc 命令に関しては分岐先候補が静的に決まっているが、JMPL の分岐先候補は静的に決まっていないため multi-BTAC または ideal multi-BTAC によって予測した分岐先候補を用いる。本稿の評価においては、JMPL の候補数は 2 と限定する。このように Bicc および JMPL の予測候補を展開して構築されたツリーが 図7 である。これらの分岐先候補集合は、既に述べられた方法によって求められた重み値をもつ。これは図中の各枝に示された数値である。

次に、全てのパスに関して重み値を求める。全てのパスは分岐先候補の集合によって構成されるが、パス上に存在する分岐先候補集合のそれぞれの重みの積をパスの重み値とする。これは図中の各葉に示された数値である。この値の大きいものほど実行確率の高い候補パスとする。図ではパスの重みが最大である path1 が最も実行される可能性の高い候補となる。図中の *level* は、またいだ分岐の段数であり、*level* = 1 の場合は通常の命令単位の分岐予測と等価である。

JMPL における重み付けに関して、multi-BTAC においては分岐先の重みはすべて同じとし、ideal multi-BTAC においては既に述べた方法により分岐先候補に重み値を適用する。また、各重みの計算に用いる BHT および multi-BTAC の情報はパスの始点時の履歴を用いており、パスの構築中に更新されることはない。

4.　評　　価

これまで同様に 6 種類のサンプルプログラムの最初

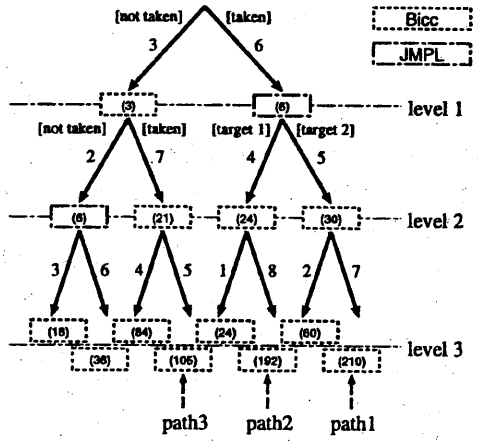

図7　コントロールフロー・ツリーによるパスの予測
Fig. 7　path prediction based on control flow tree

program	cc1	compress	dhrystone
分岐予測成功率	89.80 %	92.12 %	94.99 %
program	espresso	sc	whetstone
分岐予測成功率	89.84 %	95.58 %	99.10 %

表 2　BHT による分岐予測成功率
Table 2　BHT prediction hit rate

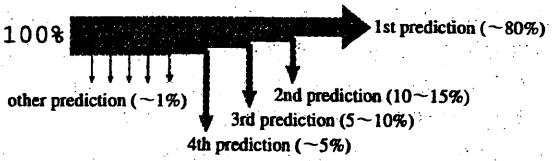

図 10　パス予測成功率の分布
Fig. 10　distribution of path prediction hit rate

1000 万命令実行を対象として、前節で述べた方法により実行パスの予測を行った。なお、実行プログラム中に含まれる全ての Bicc, JMPL 毎に、予測を行う為のツリーを構築しパス予測を行う。

図 8 ～ 図 14 にプログラム別の結果を示す。各グラフ中の横軸が予測候補数、縦軸が予測候補中に実際に実行されたパスを含んでいた割合で、これを予測成功率とする。

グラフ中の点線は JMPL 命令の分岐先予測に multi-BTAC を用いた場合、実線は JMPL 命令の分岐先予測に ideal multi-BTAC を用いた場合で、グラフ内の数字は該当する曲線の予測レベル (またいだ分岐命令の段数) である。またグラフ中の水平な線 (BHT と表記) は、BHT で単純に予測を行った場合を仮定したパス予測成功率である。既に説明されているが、この場合は予測候補は 1 種類に限定されるが、グラフ上で識別しやすくする為に直線としている。これは 表 2 に示す BHT を用いた分岐予測の予測成功率を *hit*、予測レベルを *level* としたときの hit^{level} で計算される値である。JMPL を無視した計算であるため実際の予測成功率より高い値となっている。

予測パスが 1 種類の場合は BHT による予測成功率と近い値であるが、予測パスの数を増加させるにつれ予測成功率も増加していくことが確認できる。さらに、レベルが変化しても増加の傾向に大きな差がないことが確認できる。今回評価を行ったどのレベルに関しても、予測数 1 から 4 への予測成功率向上は大きく、実行確率が高いとされる予測候補を 4 種選択した場合の予測成功率の向上の傾向が類似している。図 10 でそれを模式的に示した。各矢印は予測候補パスを意味し、その太さはコントロールフロー・ツリーによって求められたパスの実行確率である。実行確率の高い順に 1st, 2nd, 3rd, 4th prediction とし、それ以下の実行確率の候補パスは other prediction と分類した。予測レベルを変化させた場合においてもこの 2nd, 3rd, 4th の矢印の太さに大きな変化がみうけられない。単純に考えた場合、レベル 2 の全ての候補数は $2^2 = 4$、レベル 5 の全ての候補数は $2^5 = 32$ である。このような候補数の増加は図で示される、ごく細い矢印の数が増加の場合であることが多いことが、これらのグラ

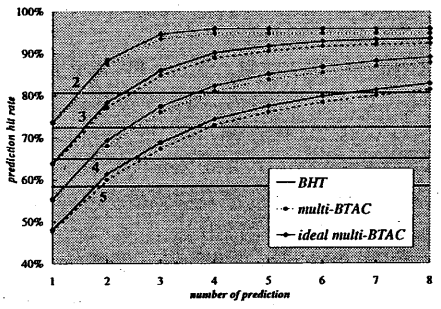

図 8　パス予測成功率 (cc1)
Fig. 8　path prediction hit rate (cc1)

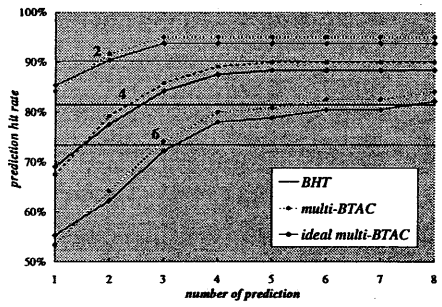

図 11　パス予測成功率 (dhrystone)
Fig. 11　path prediction hit rate (dhrystone)

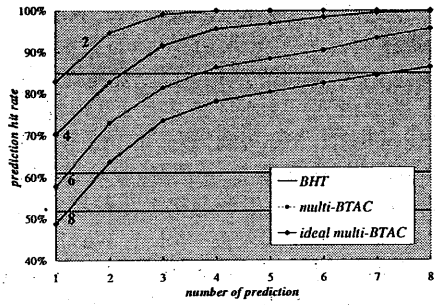

図 9　パス予測成功率 (compress)
Fig. 9　path prediction hit rate (compress)

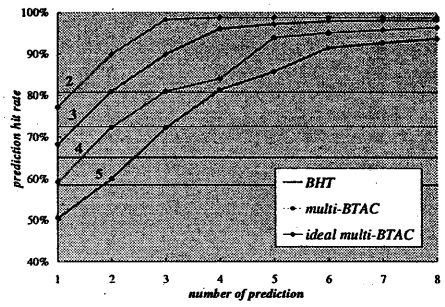

図 12　パス予測成功率 (espresso)
Fig. 12　path prediction hit rate (espresso)

フより読み取ることができる。これは、実行確率の高い典型的なコントロールフロー・パスがレベルによらず存在する傾向にあるということ示している。

また、JMPL の分岐先予測に multi-BTAC と ideal multi-BTAC を使った場合の予測成功率の差は小さい。ideal multi-BTAC の実装の複雑さを考慮に入れると、予測を行う場合 multi-BTAC で実用上の問題はないと考えられる。

5. さいごに

VLDP プロセッサにおけるフェッチ機構であるコントロールフロー先行展開について説明を行い、そこで必要となる、複数の予測候補をあげることのできるマルチレベル分岐予測ついて検討を行った。その方法として、従来の BHT および BTAC を用いた分岐予測方式に重み付けを行い、コントロールフローに基づくパス予測を行った。

これにより複数の予測候補をあげるパス予測が可能になり、単一の予測候補をあげる場合と比較して予測成功率が高くなることを示した。さらに、実行確率の高い典型的なコントロールフロー・パスがレベルによらず存在する傾向にあることが確認できた。これは、コントロールフロー先行展開で行おうとしている、確率に基づいて選択的フェッチを行うことが有効であることを示している。今回行った手法では全ての分岐候補についてパスの重みの計算を行っておりハードウェア実装という観点においてはコスト的に現実的ではない。しかし、選択的フェッチを行うことでその展開コスト削減を行うことができる。

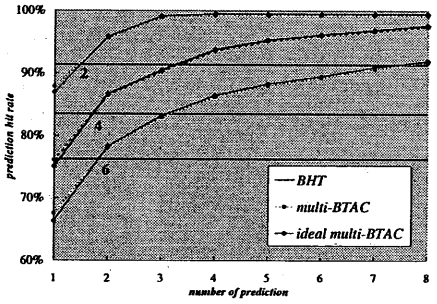

図 13　パス予測成功率 (sc)
Fig. 13　path prediction hit rate (sc)

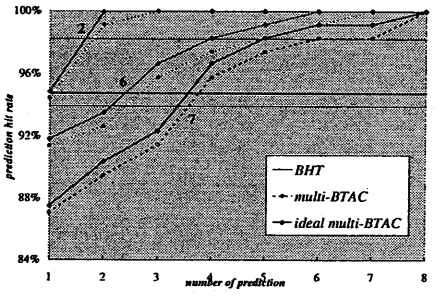

図 14　パス予測成功率 (whetstone)
Fig. 14　path prediction hit rate (whetstone)

本稿では純粋に分岐予測成功率に注目し評価を行ったが、今後はこれらに基づきフェッチコストを考慮に入れたパス予測について検討し、さらにフェッチシステム全体についての検討を行っていく。

謝辞　本研究の遂行にあたり、文部省科学研究費 (一般研究 (B) 課題番号 07458052「大規模データパスプロセッサの研究」) のご支援を頂きました。ここに感謝の意を表します。

参考文献

1) A.Nayfeh, B., Hammond, L. and Olukotun, K.: Evaluation of Design Alternatives for a Multiprocessor Microprocessor, *23rd ISCA*, pp. 67–76 (1996).
2) J.E.Thornton: Parallel Operation in the Control Data 6600, *Proceedings AFIPS Fall Joint Computer Conference, Part II*, Vol. 26, pp. 33–40 (1965).
3) Smith, M. D., Johnson, M. and Horowitz, M. A.: Limits on Multiple Instruction Issue, *3rd ASPLOS*, pp. 290–302 (1989).
4) SPARC International, I.: *The SPARC Architecture Manual Version 8*, Prentice-Hall (1992).
5) Wall, W. and D.: Limits of Instruction-Level Parallelism, *4th ASPLOS*, pp. 176–188 (1991).
6) 田中英彦: ここいらで、計算機アーキテクチャを再考しよう, 情処研究会 ARCH, Vol. 108, No. 6, pp. 33–40 (1994).
7) 宮嶋浩志, 岩下茂信, 村上和彰: 高性能システム・オン・チップ構成法に関する性能評価, 情処研究会 HPC 62-6, Vol. 96, No. 81, pp. 33–38 (1996).
8) 坂井修一: オンチップマルチプロセッシングに関する初期的検討, 情処研究会 ARCH, Vol. 122, No. 7, pp. 33–38 (1997).
9) 辻秀典, 中村友洋, 吉瀬謙二, 安島雄一郎, 田中英彦: マルチレベル分岐予測の検討と評価, 情報処理学会 第 55 回全国大会, Vol. 1, No. 3F-6 (1997).
10) 岩下茂信, 宮嶋浩志, 村上和彰: 次々世代汎用マイクロプロセッサ・アーキテクチャPPRAM の概要, 情処研究会 ARCH, Vol. 113, No. 1, pp. 1–8 (1995).
11) 中村友洋, 吉瀬謙二, 辻秀典, 安島雄一郎, 田中英彦: 大規模データパスプロセッサの構想, 情処研究会 ARCH, Vol. 124, No. 3, pp. 13–18 (1997).
12) 中村友洋, 吉瀬謙二, 辻秀典, 安島雄一郎, 田中英彦: 分岐アドレス予測機構の比較検討, 情報処理学会 第 55 回全国大会, Vol. 1, No. 3F-5 (1997).
13) 村上和彰, 吉井卓, 岩下茂信: 21 世紀に向けた新しい汎用機能部品 PPRAM の提案, 情処研究会 ARCH, Vol. 108, No. 8, pp. 1–8 (1994).