大規模データパス・アーキテクチャにおけるフェッチ機構

塚本泰通[†]　安島雄一郎[†]
坂井修一[†]　田中英彦[†]

大規模データパス・アーキテクチャは最大 4 つの分岐命令を含む命令ブロックを処理単位とし、複数パス実行を行うことで実効 IPC 8 を目指す。命令ブロックには最大 5 つの出口があるため、その中から確率が高い出口を予測する必要がある。本論文では、大規模データパス・アーキテクチャの複数パスフェッチ戦略と分岐予測機構を提案し、その初期評価を行った。その結果、90%を越える予測成功率が達成できることを示した。

Fetch Mechanism for Very Large Data Path Architecture

Yasumichi Tsukamoto,[†] Yuichiro Ajima,[†] Shuichi Sakai[†]
and Hidehiko Tanaka[†]

Very Large Data Path Architecture supports multipath execution of instructions in a granularity called Instruction Block to achieve IPC 8. An Instruction Block consists of four contiguous basic blocks, each of which can have up to 8 instructions. Because an instruction block has up to 5 break points, we have to predict from which break point it will exit, and arrange the processor to fetch next IB starting from that point. In this paper, we present fetch strategy and fetch mechanism for VLDP, and evaluate them. The evaluation shows that VLDP can achieve more than 90% branch prediction accuracy.

1. はじめに

現在マイクロプロセッサは、アーキテクチャの改善と半導体プロセス技術に支えられ著しい性能向上を果たしている。しかし将来利用可能なトランジスタ数は増加が見込まれる一方、現在のアーキテクチャの主流であるスーパースカラ・アーキテクチャは、その設計の複雑さから投入したハードウェア量に見当った性能が引出せないという問題がある。そこで、大規模なハードウェアを有効に利用する大規模データパス (Very Large Data Path:VLDP)・アーキテクチャが提案されている [1)]。

VLDP では、複数パス実行による大規模な投機実行、最大 32 命令からなる命令ブロックを処理単位とする高スループット、複数の実行ユニットによる命令ブロックの並列実行、レジスタを介さないデータアクセスの明示的記述などを特徴とし、実効 IPC 8 を目指す。IPC 8 を実現するには、2 桁のフェッチ能力が必要となる。また複数パス実行を行う上で、高い精度の分岐予測と無駄のないパス展開を行う必要がある。本論文では、これらを実現するフェッチ機構を提案し、その初期評価を行う。

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

2. 大規模データパス・アーキテクチャ

2.1 概　　要

VLDP が実効 IPC 8 を実現するためには、毎サイクルに 2 桁命令のフェッチスループットが必要となる。そこで、VLDP では最大 4 つの分岐命令を含む命令ブロック (IB:Instruction Block) を導入 [1)2)] し、処理単位とする。命令ブロックの構成を図 1 に示す。IB は最大 8 命令からなる field を 4 つ持ち、分岐命令は各 field の最後にのみ置かれる。分岐の区切りにより命令が埋められない場合には NOP を挿入する (図 1 中の Blank Slot)。通常各 field は 1 つの基本ブロックであるが、分岐命令が 8 命令以上の間隔で出現した場合には、その基本ブロックを複数の field に分割する。VLDP では 1 サイクルに 1IB をフェッチすることで、毎サイクル最大 32 命令のフェッチスループットを確保する。また、IB の実行を行う実行ユニット (Execution Unit:EU) を複数用意し複数の IB を並列に実行することで、命令レベルの並列性に加え IB レベルの並列性の抽出も目指す。

VLDP のブロック図を図 2 に示す。VLDP は Con-

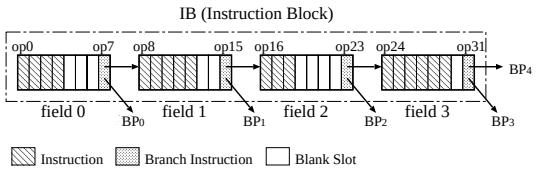

図 1 Instruction Block Structure

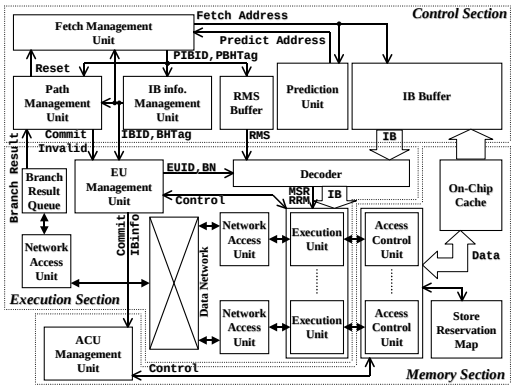

図 2 VLDP Block Diagram

trol Section(CS), Execution Section(ES),Memory Section(MS) の 3 つのセクションから構成されており、それぞれは以下の役割を持つ。

CS ：IB のフェッチ、コントロールフローの管理
ES ：IB のデコード・実行、EU の管理
MS ：メモリ・アクセスの管理

CS では、実行確率が高い IB を予測して、1 次命令キャッシュに相当する命令ブロックバッファ(IB Buffer)からフェッチを行う。フェッチした IB には各 IB の親子関係を示す IBinfo と呼ばれるタグが付けられる。VLDP では複数パス実行を行うため、IBinfo を用いてパスのコミットや無効化といった実行パス管理を CS で行う。

ES では、CS によりフェッチされた IB をデコードし EU への割当てを行った後、実際に命令を実行する [3]。またパスのコミット、無効化情報作成の為、IB の分岐結果 (Break Point:BP) が判明した時点で BP を CS に送信する。

MS では、複数パスに渡る out of order 実行を行う EU からのロード/ストアリクエストを in order で処理する [4]。階層メモリアクセス制御をはじめ、ロード/ストア命令に必要な処理はすべて MS で行うため、EU はアドレス計算のみで済み EU の負担が軽減される。

2.2 フェッチ機構の機能

CS の機能は大きく分けて、IB のフェッチと実行パス管理の 2 つがある。本節ではフェッチ機構の概要を述べる。

図 1 から分かるように IB の BP は最大 5 つ存在する。Eager Execution と呼ばれる 5 つすべての BP からパスを広げるモデルの場合、管理する IB の数が爆発的に増加しパス管理が複雑になる。そこで VLDP では、実行確率の高いパスを予測する機構を設け、予測確信度の高い間は予測結果に従って単一パスのみを伸ばし、低い場合に複数のパスから IB をフェッチすることで効率的なパス展開を行う。

予測確信度が低い場合は 1 つの IB からフェッチすべき候補が複数存在することになる。しかし、1 サイクルにフェッチ可能な IB は 1 つであるため、複数のフェッチ候補の中からさらに 1 つに絞り込む必要がある。この選択アルゴリズムをフェッチ戦略と呼び、可能な限り正しいパス上の IB を先にフェッチするようにフェッチ戦略を調整する必要がある。

フェッチ機構の処理の流れは、以下のようになる。

(1) ある IB がフェッチされた場合、その次に実行される確率の高い IB をフェッチ候補としテーブルに保存する
(2) フェッチ戦略に従って、複数の候補の中から実際にフェッチする IB を選択する
(3) フェッチした IB に対して上記 1,2 を繰り返す

このようにフェッチ機構には、フェッチ候補を挙げる機構と、候補の中からフェッチする IB を決定する機構の 2 つが必要となる。どちらか一方でも精度が低い場合には EU に無駄な IB を供給する確率が高くなり高いスループットを確保できなくなる。そのため、2 つの機構共に高い精度を要求される。

3. フェッチ機構の提案

本章ではまず CS 全体の処理の流れを紹介した後、VLDP のフェッチ機構を提案する。

3.1 CS の概要

CS は 5 つのモジュールから構成されており、そのブロック図を図 3 に示す。各モジュールは以下の機能を果たす。

Prediction Unit(PU)
フェッチした IB の次に実行される確率の高い IB を予測する

Fetch Management Unit(FMU)
フェッチ候補を管理し、その中からフェッチする IB を選択する

IBinfo Management Unit(IBiMU)
フェッチした IB に IBinfo を付加する

Path Management Unit(PMU)
実行パス情報を管理し、コミット/無効化情報を作成する

RMS Buffer(RMSB)
論理レジスタと物理レジスタの関係を示す Reg-

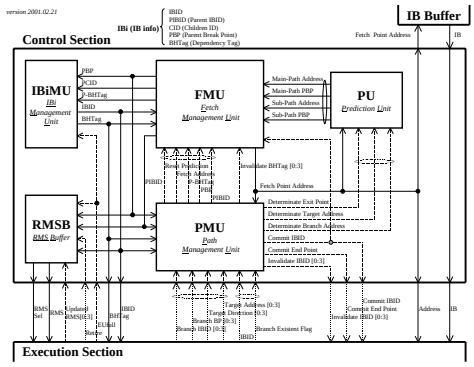

図 3 Control Section Block Diagram

ister Map Set(RMS) を管理する

まず FMU がフェッチする IB を決定し、その IB のアドレスを IBiMU、RMSB、PU、PMU に出力する。IBiMU は入力された IB の IBinfo を、RMSB は RMS を管理し、IB とともに ES に出力する。PU は次に実行される確率の高い IB のアドレスを予測し FMU に出力する。FMU は予測された IB のアドレスをテーブルに保持し、フェッチ候補として管理する。PMU はコントロールフローをすべて把握し、ES から出力される IB の分岐結果からパスのコミット/無効化情報を作成し、ES、FMU、PU に出力する。これをもとに ES では EU の解放を、FMU ではフェッチ候補の無効化、PU では予測テーブルの更新をそれぞれ行う。

3.2 フェッチ戦略

VLDP が複数パス実行をする上でのフェッチ戦略の基本方針は以下の 4 つである。ここで、複数のパスのうち、最も実行確率の高いパスをメインパスと呼び、その他のパスをサブパスと呼ぶ。また、ある IB から見た際に、一番実行確率の高い子 IB をメインパス候補、二番目に実行確率の高い子 IB をサブパス候補と呼ぶ。

- フェッチはメインパス、サブパスから交互に行う。ただし、サブパス上の候補がない場合はメインパス上の IB を続けてフェッチする。
- サブパス上の IB のサブパス候補はフェッチ対象としない。
- フェッチされた IB の最大 2 つの子 IB(メインパス候補、サブパス候補) をフェッチ候補とする。(PU によってその候補は選ばれる)
- サブパス上の IB は、フェッチ候補に挙がった順にフェッチされる。

このフェッチ戦略では、フェッチ候補中メインパス上の IB は常に 1 つで、残りはすべてサブパス上の IB となる。またフェッチ候補数については、各サイクル 1IB がフェッチにより出力され、2 つの IB がフェッチ候補として入力されるため、毎サイクル 1 つずつ増える。しかし、パスの確定により不要な候補は無効化されるため、フェッチ候補数の爆発的な増加を避けることが可能となる。そのため、フェッチ候補の管理の複雑化を回避できる。

3.3 Prediction Unit

Prediction Unit(PU) では、フェッチされた IB の次にフェッチされる可能性の高い IB を予測する。図 1 に示すように IB の出口候補は最大 5 つ (BP_0〜BP_4) 存在する。VLDP は複数パス実行を行うため、PU は 5 つの出口候補の中から 1 番確率の高い候補 (メインパス候補) と 2 番目に確率の高い候補 (サブパス候補) を出力する。ただし、出口が一つしかない場合や出口履歴に大きな偏りがある (予測の確信度が高い) 場合はメインパス候補のみ出力する。

3.3.1 PU の構造

PU の構造は図 4 のようになっており、IB のアドレスをインデックスとするそれぞれ 5 つの出口に対応した Prediction Table(PT_0〜PT_4) を持つ。各 PT は図 5 のようになっており、テーブルは 3 種類存在する。1 つ目はメインパス候補の出口を予測するためのカウンタ (Main Path Counter:MC_0〜MC_4)、2 つ目はサブパス候補の出口を予測するためのカウンタ (Sub Path Counter:SC_0〜SC_4)、3 つ目はそれぞれの出口に対応する IB のアドレスを予測するためのテーブル (IB Target Buffer:ITB_0〜ITB_4) である。MC は飽和カウンタ、SC はリセッティングカウンタの動作をする。

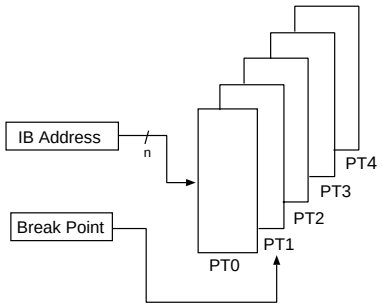

図 4 Prediction Unit の構造

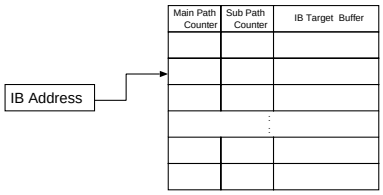

図 5 予測テーブルの構造

3.3.2 PU の動作

PT の更新

コミットした IB が BP_N から抜けた場合、PT_N にアクセスする。IB のアドレスをインデックスとして MC,SC,ITB それぞれを更新する。具体的には MC_N をインクリメント、SC_N を最大値にし、それ以外の SC と MC をデクリメントする。また,IB の飛先アドレスを ITB_N に格納する。

予測

フェッチした IB のアドレスをインデックスとして PT_0〜PT_4 にアクセスする。各 PT の MC のうち一番大きいものが MC_n の場合、BP_n をメインパス候補、SC_n 以外の SC_0〜SC_4 のうち一番大きいものが SC_m の場合、BP_m をサブパス候補とする。ただし、サブパス候補に関して SC_n 以外が全て 0 の場合はサブパス候補はなしとする。また各予測 BP に対応する IB のアドレスは ITB_n から得られる。

3.4 Fetch Management Unit

Fetch Management Unit(FMU) は、PU によって予測されたフェッチ候補 IB を管理し、その中からフェッチ戦略に従ってフェッチする IB を選択する。ここでは、どのように管理、選択を行うかを述べる。

3.4.1 FMU の構造

図 6 に示すように 5 つの要素をもつ Fetch Management Table(FMT) を用意する。各要素は IB のアドレス (Address), フェッチ優先度 (Priority), サブパスのサブパス上を示すフラグ (SubSub Path Flag), エントリが有効な値かを示すフラグ (Valid Bit),IB の親子情報 (IBinfo) である。

また、メインパス上のエントリはどこか (Main Path Index), 次にメインパス、サブパスどちらをフェッチするべきか (Path Flag), 前回メインパス、サブパスどちらをフェッチしたか (Path Bit), 現在最下位の優先度 (Priority Counter)、という 4 つの制御情報も持っている。

これらのテーブルと制御情報を用いてフェッチ候補の管理を行う。

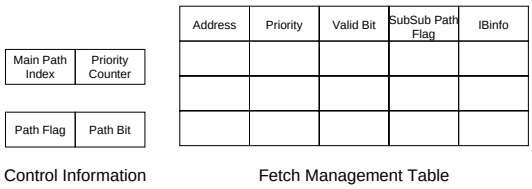

図 6 Fetch Management Unit の構造

3.4.2 FMU の動作

フェッチ候補は PU から各サイクル最大 2 つ FMU に渡され、FMT に記憶される。サブパスのサブパス上の IB の場合は、SubSub Path Flag がセットされる。

メインパス、サブパスから交互に IB をフェッチするため、Path Flag が 1 の場合はメインパス、0 の場合はサブパスからフェッチを行う。Path Flag は毎サイクル反転される。フェッチする IB を選択する際、メインパス候補は常に 1 つなので問題ないが、サブパス候補は多数存在する。そのため、サブパスを FMT に記憶する際に Priority Counter+1 の値を Priority に設定し、Priority Counter をインクリメントする。これにより、Priority の小さいものからフェッチを行えば、フェッチ候補に上がった順にフェッチできる。ただしサブパスのサブパス上の IB のフェッチは行わず、サブパス候補がない場合はメインパス候補の IB をフェッチする。フェッチするエントリが決まったら、該当する IB のアドレスを出力し、そのエントリの Valid Bit がリセットされる。

また、PMU から渡されるコミット/無効化情報と FMT 内の IBinfo を用いてどのエントリがメインパスを示すかなどの制御情報を更新し、誤ったパス上のエントリの Valid Bit はリセットされる。すべての制御情報は、予測ミスによりすべてのエントリが無効化された場合にすべてリセットされ、正しい IB からリスタートする。

4. 評　　価

VLDP におけるフェッチ機構を、IB を処理単位とし複数パス実行を行うシミュレータ上で評価する。性能は、IB の予測精度、FIPC(Fetched Instructions Per Cycle)、実効 IPC(Instructions Per Cycle) について、動的分岐予測を用いない場合と単一パス実行を行う場合と複数パス実効を行う場合で比較評価する。

4.1 評 価 環 境

プロセッサモデルとして、3 章で提案した PU と FMU を機構として持ち、IB をフェッチ、処理単位とする複数パス実行プロッセッサを考える。パス展開は 3.2 節で述べたフェッチ戦略に従って行う。

フェッチ機構の設定は以下のようになっている。

FMU　フェッチ候補エントリ数は 32 とする

PU　予測テーブルの各エントリは衝突ミスが起こらない十分な数を用意し、各 SC は 3bit、MC は 4bit のカウンタになっている。ITB はフルアドレスが格納できる。

フェッチと実行はそれぞれ以下のような条件でシミュレーションを行った。

フェッチ

通常のプロセッサでは、基本的にプログラムカウンタに 1 加えた命令が次に実行する命令となる。一方 VLDP では、次にフェッチする IB を毎サイクル動的に予測するが、初期参照などにより予測値が得られない場合が存在する。その場合はフェッ

チが止まり性能低下を引き起こすため、あらかじめ静的分岐予測で各 IB の次にフェッチする IB のデータを持っておく。これにより基本的には動的分岐予測結果を用いてフェッチする IB を決定するが、動的分岐予測ができない場合には静的分岐予測結果を利用しフェッチを止めないようにする。評価対象の 1 つである「動的分岐予測を用いない場合」はすべて静的分岐予測結果を用いてフェッチを行う。

ここで、全実行プロファイルから各 IB について一番多い飛先アドレスを静的分岐予測結果とする。

実行

実行機構については、フェッチ機構に注目する意味で理想化を行った。IB の分岐結果はフェッチしてから固定 16 サイクル後に CS に渡されると仮定する。分岐結果が出た後は、無効なパス上の IB が割当てられている EU を解放し、フェッチ候補も無効化する。また、IB 間のデータ依存関係は考慮しない。

評価対象プログラムは SPECint95 の go、m88ksim、compress、li、ijpeg、perl、vortex を用い、パラメータは表 4.1 の通りである。分岐予測精度は各結果の算術平均、FIPC と IPC は調和平均となっている。

表 1 SPECint95 のパラメータ

go	9x9(1.4 億命令程度)
m88ksim	ctl.in(1.3 億命令程度)
compress	30000 q 2131(1.4 億命令程度)
li	train.lsp(2.1 億命令程度)
ijpeg	-image_file specmun.ppm(1.7 億命令程度)
perl	scrabbl.pl scrabbl.in(1.5 億命令程度)
vortex	vortex.in(1.9 億命令程度)

4.2 結　　果

IB 全体の分岐予測精度、条件分岐命令のみに注目した時の分岐予測精度、レジスタ間接分岐命令のみに注目した時の分岐予測精度を図 7 に示す。それぞれに対し、左から「複数パス実行」「単一パス実行」「動的分岐予測なし」という 3 つのモデルのデータを載せてある。「動的分岐予測なし」の場合、IB の予測成功率は 30%に留まる。しかし、動的分岐予測により「単一パス実行」で 80%、「複数パス実行」で 90%を越える予測成功率となっている。

1 サイクルに平均何命令をフェッチできたかを示す FIPC を図 8 に示す。「動的分岐予測なし」の場合 6 を切るが、動的分岐予測を用いる他の 2 つの FIPC は 18 を越えている。

1 サイクルに平均何命令をリタイアできたかを示す実効 IPC を図 9 に示す。「動的分岐予測なし」のみの場合 1.5 を切るが、動的分岐予測を用いると IPC は約 4 となる。

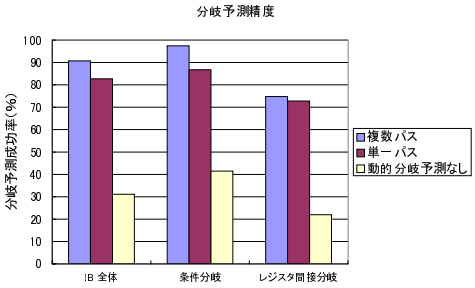

図 7 分岐命令種類別の分岐予測精度

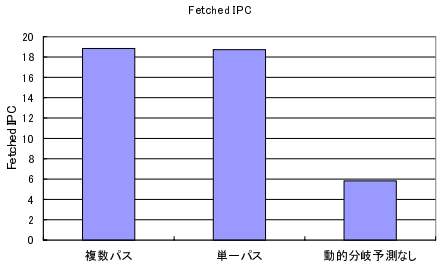

図 8 Fetched IPC

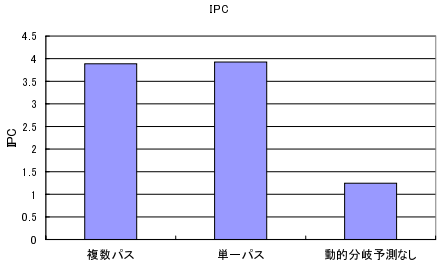

図 9 実効 IPC

4.3 考　　察

本論文で提案した分岐予測機構を用いた場合、静的分岐予測のみの場合に比べ約 3 倍の性能が出ている。以下、分岐予測、FIPC、IPC それぞれについて見ていく。

まず分岐予測であるが、条件分岐命令は予測機構を用いると「複数パス実行」は約 97%、「単一パス実行」は約 86%の予測成功率となっている。これは、4 つの分岐命令を含む IB 単位で分岐予測を行うことにより、分岐の傾向が命令単位より把握し易いためと考えられる。一方、レジスタ間接分岐命令は予測が条件分岐命

令より困難なため、ともに約 75%の予測成功率となっている。レジスタ間接分岐命令に関しては同じ IB の同じ出口でも飛先候補アドレスが複数あるため、複数パス実行の効果が表われない。条件分岐命令とレジスタ間接分岐命令を合わせた IB 全体の予測成功率は「複数パス実行」で約 91%、「単一パス実行」で約 83%となり、条件分岐予測に比べ精度が下がる。これは IB をレジスタ間接分岐命令で抜ける場合が 25%を占めるため、レジスタ間接分岐予測の精度の低さが全体性能低下に大きく影響していると考えられる。よって、さらなる分岐予測精度の向上を図るためには、レジスタ間接分岐命令の予測精度が向上する予測機構を考える必要がある。

次に FIPC であるが、予測機構は用いない場合約 6、用いた場合約 18 となっている。文献 [2] で、IB 内の平均命令数は約 21 となっている。よって、予測機構がない場合は 3.5 サイクルに 1 回しかフェッチできず、実行機構に効率良く IB を供給できない。一方、予測機構があれば、ほぼ毎サイクル実行機構に IB を供給できる。

最後に IPC であるが、予測機構がない場合は約 1.2 と非常に低い値となっている。これは、3.5 サイクルに一回しかフェッチできない上、フェッチした IB のうち 5 回に 4 回は誤った IB ということになり、フェッチ、実行とも効率が悪い。これは、IB の予測精度が約 30%と低いためである。一方、予測機構がある場合、IPC は約 4 となる。FIPC は 18 であるため、4.5 サイクルに 1 回 IB の実行が完了することになる。以降で、この原因について考える。

「単一パス実行」において、メインパスの予測成功率は 80%に留まっており、5 回に 1 回予測を誤った IB のあとにフェッチした IB はすべて無効化される。よって、4 回連続で正しい IB をフェッチしてもそのあとフェッチした 12 個の IB は誤ったパス上の IB になる。よって、4.5 サイクルに 1 回の IB の実行完了は妥当なデータと言える。

「複数パス実行」では、単一パス実行の予測ミス 20%のうちの半分を救えている。しかし、サブパス候補の IB がフェッチされるタイミングはメインパス候補のフェッチよりも遅れる。よって、サブパス候補で正しい IB をフェッチしても、その IB が完了するまでには数サイクルを要する。また、サブパス候補をフェッチすることで、誤ったパス上の IB をフェッチする可能性も高くなり、メインパス候補のフェッチタイミングが遅れることに繋がる。このことから、単一パス実行で予測成功率が高いプログラムでは複数パス実行により性能が落ち、単一パス実行で予測成功率の低いプログラムでは、複数パス実行により性能が向上する。これを示したのが図 10 である。単一パス実行で予測成功率が 69%である go では複数パス実行することで IPC が向上しているが、予測成功率が 90%を越える

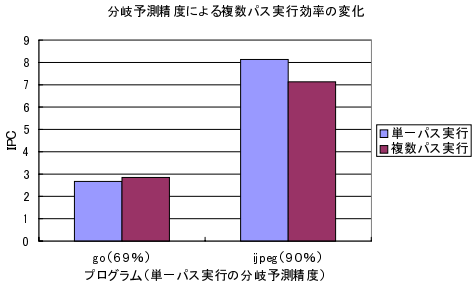

図 10 go と ijpeg の IPC と分岐予測成功率の関係

ijpeg では複数パス実行をすることで IPC が 1 低下する。

このような理由から、IB の予測成功率が高いほど複数パス実行をしても性能が伸び悩んでいる。これを改善するには、フェッチ戦略を改善しサブパスの選択をより効率的に行う必要がある。

5. おわりに

本論文では、VLDP におけるフェッチ機構を提案し、各ユニットの詳細と動作について説明した。また、その評価を行った。その結果、分岐予測は 90%の成功率でほぼ 1 サイクルに 1IB をフェッチできることが分かった。また、現在のフェッチ戦略では実効 IPC4 を達成するに留まり、さらなる改善が必要なことが分かった。

今後は、レジスタ間接分岐命令の予測精度向上とフェッチ戦略の改善を行っていく。

謝辞 本研究の一部は、文部省科学研究費補助金 (基盤研究 (B) 課題番号 11480066) および (株) 半導体理工学研究センターとの共同研究によるものである。

参 考 文 献

1) 辻秀典, 安島雄一郎, 坂井修一, 田中英彦. 大規模データパス・アーキテクチャの提案. 情報処理学会研究報告 2000-ARC-139, pp. 49–54, 2000.
2) 塚本泰通, 安島雄一郎, 辻秀典, 坂井修一, 田中英彦. 大規模データパス・アーキテクチャにおける命令ブロック構成の検討. 情報処理学会研究報告 2000-ARC-139, pp. 60–65, 2000.
3) 安島雄一郎, 辻秀典, 坂井修一, 田中英彦. 大規模データパス・アーキテクチャの実行機構. 情報処理学会研究報告 2000-ARC-139, pp. 55–60, 2000.
4) 入江英嗣, 安島雄一郎, 辻秀典, 坂井修一, 田中英彦. 大規模データパス・アーキテクチャに適したロードストアユニット構成. 情報処理学会研究報告 2000-ARC-140, pp. 43–48, 2000.