計算機アーキテクチャ　147−17
ハイパフォーマンス　89−17
コンピューティング
（2002．3．7）

VLIWプロセッサのための デュアルパス投機実行手法の性能評価

島尻 寛之　　吉田 たけお
琉球大学 工学部 情報工学科

あらまし：動的スケジューリングを行わないVLIWプロセッサは，分岐命令の多い非数値計算プログラムでは十分な並列性を得ることができないという問題点がある．以前，我々はこの問題点を解決するために，ハードウェアコストを限りなく抑えた複数パス投機実行手法としてデュアルパス投機実行手法を提案した．本稿では，実行ユニットの構成やパイプライン段数が異なる様々なVLIWプロセッサに提案手法を適用し，その有効性について検討する．SPEC95ベンチマークに対するシミュレーションを行った結果，実行ユニットを多く実装したVLIWプロセッサでは約18%，実行ユニットが少ないVLIWプロセッサでも約10%ほどIPCを向上できることがわかった．
キーワード：VLIWプロセッサ，複数パス投機実行，分岐予測

Evaluation of Speculative Dual-path Execution for VLIW Pocessor

Hiroyuki SHIMAJIRI and Takeo YOSHIDA
Department of Information Engineering, Faculty of Engineering,
University of the Ryukyus

Abstract：　Speculative execution is an effective technique for VLIW processors to achieve a high performance in non-numerical programs. One of the problems for implementing multi-path speculative execution mechanism on VLIW processors is huge hardware costs. We have proposed a multi-path speculative execution method, called speculative dual-path execution, which greatly reduces hardware cost. This paper shows performance evaluation and analysis of speculative dual-path execution. The software simulation results show that a speed-up of 10% - 18% was achieved in the SPECint 95 benchmarks.
KEYWOARDS：VLIW Processor, Multi-path Speculative Execution, Branch Prediction

1　はじめに

VLIWプロセッサは，コンパイル時にのみスケジューリングを行うことによって，アーキテクチャを簡略化している．しかし，VLIWプロセッサは，ハードウェアスケジューラを省略したため，プログラムの実行状況に応じたスケジューリングを行うことができない．そのため，分岐命令が多く含まれる非数値計算プログラムでは，十分な並列性を引き出すことができないという問題点がある[1,2]．

非数値計算プログラムのように分岐命令の多いプログラムから，多くの並列性を抽出する手法として投機実行手法が知られている．なかでも，同時に複数のパスを投機的に実行する複数パス投機実行手法は，より多くの並列性を抽出できることが知られている[3,4]．前述のVLIWプロセッサの問題点を解決するために，我々は文献[5]において，VLIWプロセッサに複数パス投機実行手法を適用することについて検討し，ハードウェアコストを抑えたデュアルパス投機実行手法(以降，DP手法)を提案した．

DP手法は，分岐命令が成立する場合に実行される分岐先パスと成立しない場合に実行され

る後続パスの 2 つのパスに対して，投機実行を行う．DP 手法では，一方のパスの VLIW 命令に含まれる NOP 命令の代わりに，他方のパスの VLIW 命令内の有効な命令を埋め込み，1 つの VLIW 命令にまとめる．なお，この 2 つのパスの VLIW 命令を 1 つの VLIW 命令にまとめる処理を合成と呼ぶ．この合成された VLIW 命令を実行することによって，同時に分岐先パスと後続パスを投機的に実行する．このように DP 手法では，分岐命令後方の 2 つのパスを同時に投機実行するため，動的に分岐予測を行う必要がなく，分岐予測機構を省略することができる．

本稿では，実行ユニットの構成やパイプライン段数の異なる様々な VLIW プロセッサに対して DP 手法を適用し，その性能を比較することによって DP 手法の有効性を検証する． 以降 **2** では，VLIW プロセッサの投機実行手法に関する研究について述べる．**3** では，DP 手法の実行例と DP 手法を実現する機構を示す．続いて **4** では，DP 手法を VLIW プロセッサに適用し，その有効性について検討する．

2 関連研究

これまでに，VLIW プロセッサのための投機実行手法がいくつか提案されている．ここでは，VLIW プロセッサの投機実行手法に関する研究として，プレディケートと GIFT，ダイナミックブースティング手法 (以降，DB 手法) について述べる [6–9].

プレディケートは，コンパイル時に分岐命令を削減する手法であり，実行する全ての命令に条件を付加し，異なるパスを 1 つのパスにまとめることによって分岐命令を削減することができる [6, 7].

GIFT は，プレディケートと通常の投機実行手法を併用した VLIW プロセッサであり，分岐予測が難しい分岐命令にはプレディケートを適用し，分岐予測が簡単な命令に対しては通常の投機実行手法を適用している．これにより，単純な分岐予測機構でも高い予測成功率を得ることができる [8].

DB 手法は，広域スケジューリング手法の 1 種であるブースティング手法を動的に行う手法であり，通常の投機実行手法に比べて効率良くプログラムを実行することができる [9].

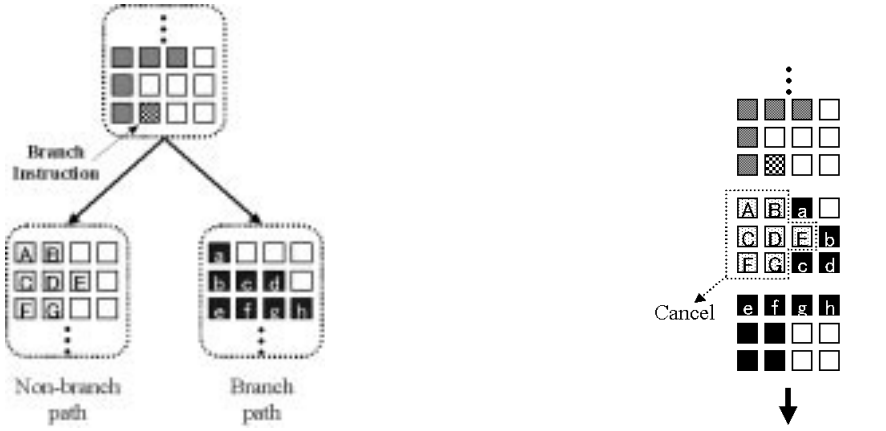
ハードウェアコストを抑えることを重視している DP 手法は，投機実行の処理を簡略化しているため，GIFT や DB 手法ほどの性能向上を望むことはできない．しかし DP 手法は，プログラムカウンタ (PC) ，命令キュー (IQ) の 2 重化および VLIW 命令を合成する機構を実装するだけで実現することができるため，複雑な分岐予測機構を実装する GIFT や DB 手法に比べてハードウェアコストの面において非常に有利であると考えられる．

一方，プレディケートと比較した場合，プレディケートはループの先頭方向に戻る分岐命令などには適用できないのに対して，DP 手法は分岐命令の分岐先に関係なく適用することができる．逆に DP 手法は，**3.1**でも述べるように，連続して現れる分岐命令には適用できないのに対して，プレディケートは，そのような分岐命令にも適用することが可能である．このことから両手法を併用することにより，さらなる性能向上を実現できると考えられる．またプレディケートは，DP 手法と同様に少ないハードウェアコストで実現できるため，両手法を併用した場合でもハードウェアコストを抑えることができる．なお，プレディケートとの併用に関する検討は，本稿では割愛する．

3 デュアルパス投機実行手法

3.1 デュアルパス投機実行手法の実行例

まず DP 手法の実行例を示す．DP 手法を適用した VLIW プロセッサ上で，図 1 (a) に示すサンプルコードを実行した様子を図 1 (b) に示す．なお実行例では，1 つの VLIW 命令で 4 つの命令を指定でき，分岐命令の分岐先が確定するまでに 3 サイクルを要するものとしている．また，1 つの VLIW 命令には，分岐命令は 1 つまでしか指定できないとしている．さらに，コンパイ



(a) Sample Code　　(b) Proposed Method

図 1: デュアルパス投機実行の実行例

ル時に静的分岐予測を行い，後続パスに分岐命令の予測先が偏るようにスケジューリングされているものとする．

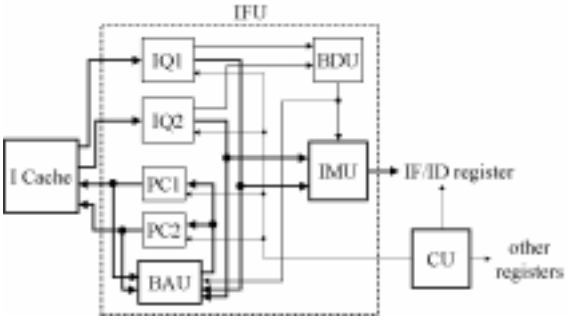


図 2: DP 手法を適用した IFU の構成

図 1 (b) に示すように DP 手法では，後続パスと分岐先パスの VLIW 命令を合成し，後続パスと分岐先パスを同時に実行する．VLIW 命令の合成の際には，コンパイル時の静的分岐予測の結果を利用するために，後続パスの VLIW 命令が優先的に実行されるように合成する．そのため，分岐先パスの VLIW 命令が実行されるのは，後続パスの VLIW 命令内に NOP 命令が含まれている場合に限られる．なお DP 手法においてミスペナルティが発生するのは，分岐命令の分岐先が分岐先パスに確定し，かつ，分岐先パスの実行が遅れた場合のみに限られる (図 1 の例では，1 サイクルのペナルティ)．分岐命令の分岐先が確定した後は，プログラムの実行結果に矛盾が生じるのを防ぐために，分岐しなかったパス (図 1の例では，後続パス) の実行結果を無効化する．また，投機実行中に新たな分岐命令が現れた場合には，現在実行中の分岐命令の分岐先が確定するまで NOP 命令を挿入し，分岐先が確定した後に投機実行を行う．

3.2　デュアルパス投機実行機構

ここでは，DP 手法を実現するための機構について述べる．図 2 に示すように，デュアルパス投機実行機構は命令フェッチユニット (IFU) 内に実装する．デュアルパス投機実行機構の各ユニットは以下に示す **1.**∼**5.** の処理を行うことによって，**3.1** で示した DP 手法を実現する．

1. 分岐命令の検出

分岐命令検出ユニット (BAU) は，フェッチしてきた VLIW 命令を常に監視し，分岐命令の検出を行う．

2. 後続パスと分岐先パスのフェッチ

投機実行中に後続パスと分岐先パスの VLIW 命令をフェッチするために，プログラムカウンタ (PC) と命令キュー (IQ) をそれぞれ 2 重化する．なお投機実行を行っていない時は，1 組の PC と IQ のみが使用される．

3．分岐先アドレスの計算

分岐先アドレス計算ユニット (BAU) は，分岐命令の有無に関わらず，PC と分岐命令フィールドの即値から分岐先アドレスの計算を行う．また BAU は，分岐命令が検出された場合にのみ，その時点で使用されていない方の PC に分岐先アドレスを格納する．これにより，分岐命令を検出した次のサイクルから分岐先パスの VLIW 命令をフェッチすることができる．

4．VLIW 命令の合成

命令合成ユニット (IMU) は，投機実行中に後続パスと分岐先パスの VLIW 命令の合成を行う．なお合成の際に，分岐先パスの VLIW 命令を合成しきれなかった場合，IMU は分岐先パスの VLIW 命令を保持している IQ に対して，残りの命令を保持するように制御信号を出力する．

5．分岐先確定後の無効化処理

無効化ユニット (CU) は，分岐命令の分岐先が確定した後に，分岐しなかった方のパスを保持している IQ と PC，実行結果を保持しているパイプラインレジスタの初期化を行う．

4　性能評価

4.1　プロセッサモデルと評価方法

ここではまず，評価に用いたプロセッサモデルと評価方法について説明する．今回，評価のために 3 つのプロセッサモデルを用意した．各プロセッサは，MIPS 社の R3000^{TM}をベースにしており，パイプライン段数を 5 段とした．また，1 つの VLIW 命令に最大 4 つの R3000 の命令を指定できるものとした．以下に，各プロセッサの実行ユニットの構成を示す．

プロセッサ A　実行ユニットとして R3000 の CPU コアを 4 つ実装し，VLIW 命令内に指定できる命令の組み合わせに制限がないモデル．ただし，DP 手法を適用するために，1 つの VLIW 命令には分岐命令は 1 つまでしか指定できない．

プロセッサ B　実行ユニットとして ALU を 2 つ，ロード・ストアユニットを 1 つ，分岐命令処理ユニットを 1 つ，合計 4 つの実行ユニットを実装するモデル．1 つの VLIW 命令で，ALU 系の命令を 2 つ，ロード・ストア命令，分岐命令をそれぞれ 1 つずつ指定することができる．

プロセッサ C　プロセッサ B に，ALU とロード・ストアユニットをそれぞれ 1 つずつ追加したモデル．VLIW 命令で指定できる命令はプロセッサ B と同じ．

評価はソフトウェアシミュレーションによって行い，各プロセッサに DP 手法を適用した場合と適用しない場合の IPC (Instruction Per Cycle) を比較した．評価には，SPEC95 ベンチマークを用いた．各ベンチマークプログラムは GNU の GCC 2.6.3 を用いてコンパイルし，その結果をリストスケジューリングによって，各プロセッサの実行ユニットの構成に合わせてスケジューリングしたものを使用した．

4.2　評価結果

図 3 に各プロセッサの IPC を示す．なお，DP 手法を適用していない場合は，分岐命令を遅延分岐によって処理しており，各プロセッサとも 1 サイクルの遅延スロットを挿入した．また，DP 手法の投機実行サイクル数も 1 サイクルとした．

図 3 より，遅延分岐を適用した場合の IPC に比べて DP 手法を適用した場合の IPC が約 11 ～ 30% ほど高いことがわかる．DP 手法を適用した場合，プロセッサ A と C は平均 18%，プロセッサ B は平均 10%ほど IPC が向上している．これは，プロセッサ B は実行ユニットの数が少なく，投機実行の際に VLIW 命令が合成しきれず，多くのペナルティが発生したためだと考えられる．

また **3.1**でも述べたように，DP 手法では投機実行中に分岐命令が現れた場合には，分岐が確定するまで NOP を挿入しなければならず，投機実行を行うことができない．また，ジャンプ命令のように DP 手法を適用できない命令も存在する．図 4 に，各プログラムにおける DP 手法を適用可能な分岐命令の割合を示す．図 3,4 より，DP 手法の効果が大きい **gcc, m88ksim,**

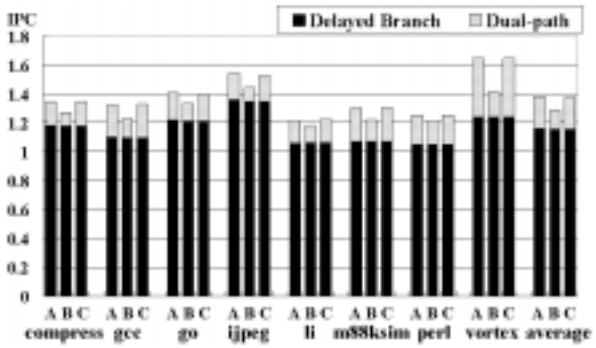


図 3: 各プロセッサの IPC
(パイプライン段数 5 段)

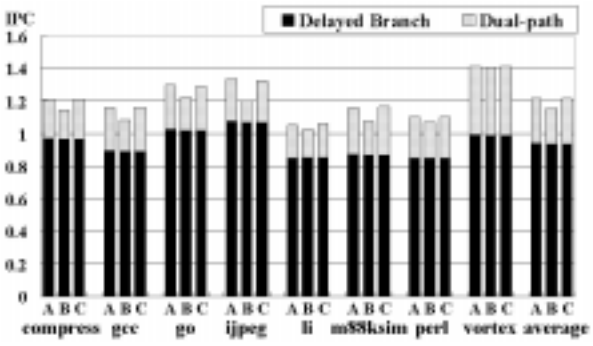


図 5: 各プロセッサの IPC
(パイプライン段数 6 段)

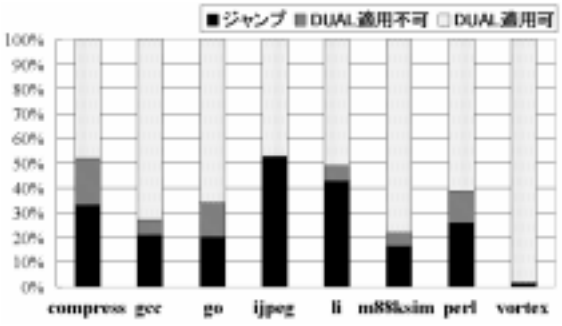


図 4: DP 手法を適用可能な分岐命令の割合
(パイプライン段数 5 段)

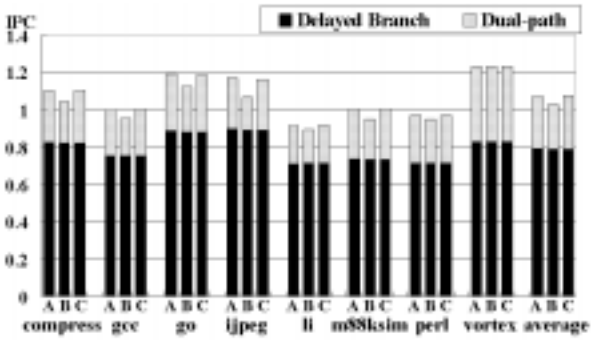


図 6: 各プロセッサの IPC
(パイプライン段数 7 段)

vortex では，DP 手法を適用できる分岐命令の割合が多く，DP 手法の効果が小さい **compress** や **ijpeg, li** では，DP 手法を適用できる分岐命令の割合が少ないことがわかる．

4.3 パイプライン段数の影響

ここでは，パイプライン段数が DP 手法に及ぼす影響について検討する．通常，パイプライン段数が多い場合，投機実行に要するサイクル数も多いと考えられる．そこで，前に示した各プロセッサのパイプライン段数を増加した場合の IPC を比較した．図 5,6 はそれぞれ，各プロセッサのパイプライン段数を 6 段と 7 段に増やした場合の IPC を示している．なお，投機実行のサイクル数と挿入する遅延スロットの数は，パイプライン段数の増加に伴い，段数 6 段では 2 サイクル，段数 7 段では 3 サイクルとした．この他，パイプライン段数を 7 段とした場合の DP 手法を適用可能な分岐命令の割合を図 7 に示す．

図 5,6 からもわかるように，パイプライン段数が増えるに従って，IPC が低下していることがわかる．段数が 5 段の場合と比較して，段数 6 段では平均して 15 %，段数 7 段では平均して 25%， IPC が低下している．また図 7より，DP 手法を適用できる分岐命令の割合も大幅に低下していることがわかる．この理由は，投機実行のサイクル数が増えたことによって，ペナルティの発生する割合やペナルティサイクル数が増えたためだとと考えられる．

続いて，DP 手法を適用した場合のペナルティ削減率を求めた．各プロセッサのペナルティ削減率を図 8 に示す．なお，ペナルティ削減率と

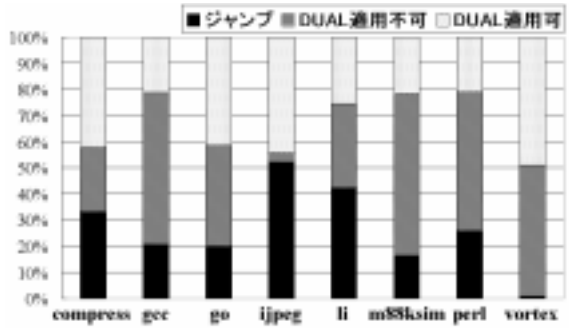


図 7: DP 手法を適用可能な分岐命令の割合
(パイプライン段数 7 段)

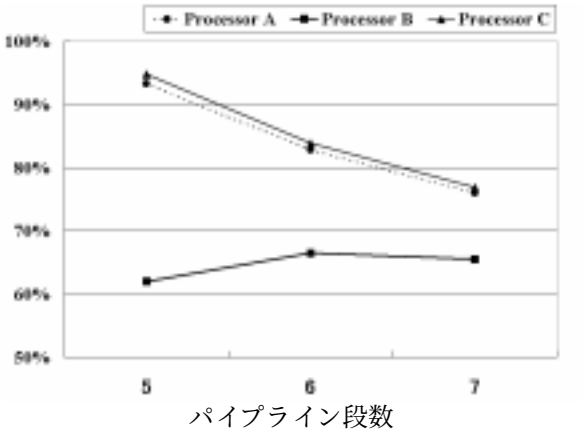


図 8: ペナルティ削減率

は，DP 手法を適用した場合，遅延分岐のみを適用した場合に発生するペナルティをどれだけ削減できたかを示している．図 8 に示すように，プロセッサ B のペナルティ削減率が最も低く，プロセッサ A と C はほぼ同じ値となっていることがわかる．またプロセッサ A と C は，パイプライン段数の増加に伴ってペナルティ削減率が低下していることもわかる．

5 むすび

今回，デュアルパス投機実行手法を，実行ユニットの構成やパイプライン段数が異なる 3 つの VLIW プロセッサに適用し，その性能比較を行うことによって DP 手法の有効性を検証した．ソフトウェアシミュレーションによる評価では，DP 手法を適用することによってプロセッサの IPC を最大で約 18% ，実行ユニットの数が少ない VLIW プロセッサでも約 11% 向上できることを示した．

今後の課題として，DP 手法を適用できない分岐命令が多いプログラムでも高い性能向上を実現するために，プレディケートなど他の投機実行手法と DP 手法の併用を検討する予定である．また DP 手法の効果を向上させるために，DP 手法に適したスケジューリング方法についても検討する予定である．

参考文献

[1] 中澤喜三郎，“計算機アーキテクチャと構成方式，” 朝倉書店, 1995.

[2] 富田眞治，“第 2 版 コンピュータアーキテクチャ，” 丸善, 2000.

[3] 片山清和，安藤秀樹，島田俊夫，“両パス実行の性能評価と実行判定精度の改善，” 情報処理，Vol. 42, No. 8, pp. 106–118, 2001.

[4] Pritpal S. Ahuja, Kevin Skadron, Margaret Martonosi and Douglas W. Clark,“Multipath Execution : Opportunities and Limits,”Proc. the 1998 International Conference on Supercomputing, pp. 101–108, 1998.

[5] 島尻寛之，吉田たけお，“VLIW プロセッサのための複数パス投機実行機構，” 信学技報，CPSY2000-40.

[6] 安藤秀樹，中西知嘉子，原哲也，中屋雅夫，“プレディケーティング：VLIW マシンにおける投機的実行のためのアーキテクチャ上の支援，” 情報処理，Vol. 37, No. 11, pp. 2039–2055, 1996.

[7] 小沢年弘，新井正樹，細井聡，木村康則，“分岐予測と条件付実行” 計算機アーキテクチャ研究会報告，ARC-134-19, pp. 109–114, 1999.

[8] 古関聰，小松秀昭，深澤良彰，“拡張 VLIW プロセッサ GIFT におけるブランチハンドリング機構．” 情報処理，Vol. 38, No. 12, pp. 2576–2587, 1997.

[9] 阿部孝之，仲池卓也，小林広明，中村維男，“VLIW アーキテクチャのためのダイナミックブースティング機構，” 信学論 (D-I), Vol. J83-D-I, No. 1, pp. 171–183, 2000.