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

2003－ARC－154 (21)
2003／8／6

BTBを利用したVLIWプロセッサ向けデュアルパス投機実行手法

島尻 寛之　　平良 健太郎　　吉田 たけお
琉球大学 工学部 情報工学科

あらまし： VLIWプロセッサは，非数値計算プログラムでは高い性能を実現できないという問題がある．この問題を解決するために，以前我々は，VLIWプロセッサ向けの複数パス投機実行手法として命令合成機構を用いたデュアルパス投機実行手法（合成型DP手法）を提案した．本稿では，合成型DP手法の問題点を解決するために，パイプラインを2重化することによって命令合成機構を省略することができる2パス型DP手法を提案する．さらに，命令フェッチの効率化を図るためにBTBを利用したBTB型DP手法も提案する．SPEC95ベンチマークに対して各DP手法の評価を行った結果，提案する2つの手法を併用した場合，合成型DP手法に比べてIPCを最大で約15%向上できることがわかった．
キーワード：VLIWプロセッサ，BTB，複数パス投機実行，in-oroder実行

Speculative Dual-path Execution Using a Branch Target Buffer for VLIW Processors

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

Abstract : We have proposed a method of speculative dual-path execution for VLIW processors. In this paper, we propose a new method speculation dual-path execution that uses BTB. The software simulation results on the SPECint 95 benchmarks show that our new method improves a performance by 15% in a VLIW processor.
KEYWORDS : VLIW Processor, Branch Target Buffer, Multi-path Speculative Execution, In-order Execution

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り，複雑なスケジューリング機構を省略し，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，非数値計算プログラムのように，振舞いが不規則な分岐命令が多く含まれているプログラムに対しては，高い性能を実現できないという問題点がある [1-3].

我々は以前，上記のVLIWプロセッサの問題点を解決するために，VLIWプロセッサ向けの複数パス投機実行手法であるデュアルパス投機実行手法 (以降，DP手法) を提案した [4]．DP手法は，プログラムの実行中に，分岐する2つのパスに対して同時に投機実行を行う．これにより，投機実行失敗時に発生するペナルティを大幅に削減している．文献 [4] に示した命令合成機構を用いたDP手法 (合成型DP手法) では，分岐する2つのパスのVLIW命令に含まれるNOP命令を利用して，2つのVLIW命令を1つのVLIW命令に合成することにより，2つのパスを同時に投機実行を行うことを実現し

ている．

一方，近年プロセッサは高クロック化が進み，パイプライン段数が増加する傾向にある．これはVLIWプロセッサについても同様であると考えられる．合成型DP手法では，パイプライン段数の少ないVLIWプロセッサを対象としていたため，高クロックで動作するVLIWプロセッサを考慮していなかった．特に，合成型DP手法は，分岐先アドレス計算器やVLIW命令の合成機構など，DP手法を実現するための多くの機構を命令フェッチステージに実装するため，VLIWプロセッサのクロック周期に悪影響を及ぼす可能性がある．

そこで本稿では，分岐命令の分岐先アドレスを計算することなく，分岐する2つのパスを効率良くフェッチするために，DP手法にBTB (branch target buffer) を利用することを検討する．また本稿では，VLIW命令の合成を行うことなくDP手法を実現することができる，新たなハードウェアの構成についても提案する．

以降，**2**では，DP手法の概要を述べ，高クロック化の際の問題点について検討する．続く**3**では，命令合成機構を必要としないDP手法のハードウェア構成を示し，**4**では，DP手法におけるBTBの利用方法について説明する．**5**章で性能評価を行い提案手法の有効性を示す．

2 デュアルパス投機実行手法

2.1 DP手法の概要

DP手法では，分岐命令を含むVLIW命令をフェッチした次のサイクルから，投機実行を開始する．投機実行中は，分岐命令が成立しなかった場合に実行される後続パスと，成立した場合に実行される分岐先パスのVLIW命令を同時にフェッチする．次に，フェッチした2つのパスを同時に投機実行を行い，分岐命令の分岐先が確定するまで投機実行を継続する．分岐命令の分岐先が確定した後は，分岐しなかった方のパスの実行結果を無効化し，プログラムに矛盾が生じるのを防ぐ．このようにDP手法では，分岐する2つのパスを同時に投機実行を行うため，投機実行失敗に伴うペナルティを大幅に削減することができる．

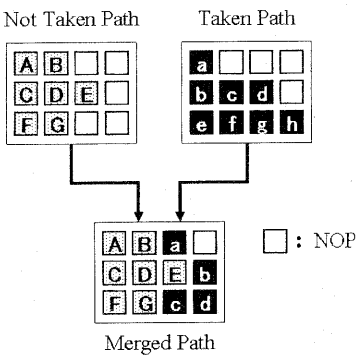

図 1: VLIW命令の合成

合成型DP手法では上記のDP手法を実現するために，VLIW命令の合成を行っている．図1に示すように，後続パスのVLIW命令に含まれるNOP命令の代わりに，分岐先パスのVLIW命令の有効な命令を埋め込み1つのVLIW命令に合成する．この合成したVLIW命令を投機的に実行することによって，DP手法を実現している．

なおDP手法と同様に，分岐する2つのパスを同時に投機実行を行う手法としてプレディケートが知られている [3,5–7]．プレディケートは，コンパイル時に異なるパスを1つのパスにまとめるため，適用できる分岐命令が限定されている．一方，DP手法はプログラムの実行時に，2つのパスに対して同時に投機実行を行うため，実行中に現れる全ての分岐命令に対して適用することができる．DP手法とプレディケートとの違いの詳細については，文献 [4] を参照されたい．

2.2 高クロック化の影響

1でも述べたように，合成型DP手法では，DP手法を実現するためのハードウェア機構の多くを命令フェッチステージに実装する．

中でも，VLIW命令の合成を行う命令合成機構は，VLIW命令を入れ替えるスイッチで構成されており，命令フェッチステージのクリティカルパスに大きな影響を及ぼす．そのため，命令合成機構は，高クロック化の際に大きな障害になると考えられる．これに対しては，VLIW

命令の合成を行うためのステージを設けることによって，命令フェッチステージから命令合成機構を分離することが考えられる．しかし，無駄にパイプライン段数が増えてしまうため，プロセッサの性能が低下してしまう恐れがある．

またDP手法では，投機実行中，後続パスと分岐先パスのVLIW命令を同時にフェッチする必要がある．しかし，分岐先パスのVLIW命令をフェッチするためには，分岐命令の分岐先アドレスを計算しなければならない．合成型DP手法では，分岐先アドレスの計算時間を隠蔽するために，プリフェッチを行っている．しかし，プリフェッチ機構を導入するため，VLIWプロセッサの命令フェッチステージが複雑化してしまい，VLIWプロセッサのクロック周期とパイプライン段数に影響を及ぼす可能性がある．

以上のことから，VLIW命令の合成とプリフェッチを行うことなくDP手法を実現することができれば，命令フェッチステージを簡略化することができ，VLIWプロセッサの高クロック化に貢献できると考えられる．また，このことは，パイプライン段数の多いVLIWプロセッサに対しても，DP手法を容易に適用できることを意味している．

一般に，プロセッサの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が長くなり，投機実行中に分岐命令が現れる可能性が高くなる．合成型DP手法では，パイプライン段数が少ないVLIWプロセッサを対象としており，ハードウェアを簡略化するために，投機実行中に分岐命令が現れた場合には，先行する分岐命令の分岐先が確定するまでNOP命令を挿入して対処していた．しかし，投機実行中に分岐命令が頻繁に現れる場合には，多くのNOP命令が挿入され，DP手法の効果がほとんど得られないという問題が生じる．

3 多重化したパイプラインを利用したデュアルパス投機実行手法

ここでは，VLIWプロセッサから命令合成機構を省くために，後続パスと分岐先パスのVLIW命令を合成することなくDP手法を実現する方法について検討する．これを実現するため，VLIWプロセッサ上では，in-orderでプログラムが実行されていることに着目する．

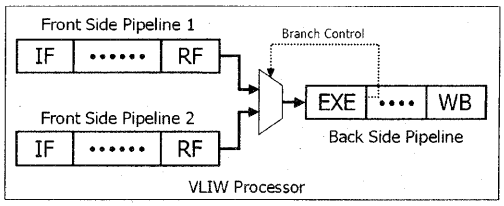

図2: パイプラインを2重化したVLIWプロセッサ

in-order実行の場合，後続の命令は先行する命令より先に実行されることはない．また，一般に分岐命令の分岐先は，パイプラインの実行ステージか実行ステージよりも前のステージで確定する．すなわち，投機的に実行された命令が実行ステージに到達する前に，分岐命令の分岐先が確定することになる．このことはDP手法において，分岐しなかったパスが実行される前に無効化できることを意味する．

そこで本稿では，図2に示すように実行ステージの手前までのパイプラインを2重化したVLIWプロセッサの構成を提案する．ここで，2重化した実行ステージ手前までのパイプラインステージをFSP(front side pipeline)と呼び，実行ステージ以降のパイプラインステージをBSP(back side pipeline)と呼ぶことにする．また，図2に示すVLIWプロセッサを利用したDP手法を2バス型DP手法と呼ぶことにする．

2バス型DP手法は，通常実行時には，2つあるFSPのうちのどちらか一方のみを使用してVLIW命令を実行する．投機実行時には，それまでに使用していたFSPに後続パスのVLIW命令を投入し，もう一方のFSPに分岐先パスのVLIW命令を投入する．先行する分岐命令の分岐先が確定したとき，分岐したパスが投入されているFSPをBSPに接続する．同時に，分岐しなかったパスが投入されているFSPを無効化する．これにより，VLIW命令の合成を行

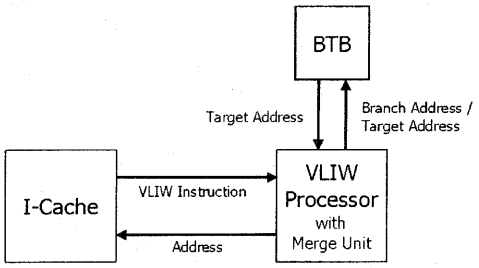

図 3: BTB を利用した DP 手法の構成例

うことなく DP 手法を実現することができる．

4 BTB を利用したデュアルパス投機実行手法

2 バス型 DP 手法によって，命令合成機構を実装することなく DP 手法を実現することができる．しかし，2 バス型 DP 手法でも，分岐命令の分岐先アドレスを計算するために，プリフェッチを行う必要がある．

ここではさらに，プリフェッチを行うことなく分岐先アドレスを得るために，BTB を DP 手法に応用することについて検討する．

4.1 BTB を利用した DP 手法の概要

DP 手法で利用する BTB は，実行中の分岐命令が成立したときに，その分岐命令のアドレスに基づき，所定のエントリに分岐先アドレスを格納する．これにより，分岐先アドレスが BTB に格納されている分岐命令を再びフェッチした時点で，その分岐先アドレスを得ることができる．なお，一度も成立していない分岐命令に関しては，その分岐先アドレスは BTB に格納されることはないため，BTB の容量を有効に活用することができる．図 3 に BTB を利用した DP 手法の構成例を示す．

4.2 BTB を利用した DP 手法の実行例

ここで，図 4 に示すサンプルコードを用いて，BTB を利用した DP 手法（以降，BTB 型 DP 手法）について説明する．なお，図 4 中の各ノードは基本ブロックを表しており，各基本ブロック内には 1 つ以上の VLIW 命令が含まれているものとする．今回の例では，基本ブロック A,B,E,F,G の順に，これが繰り返し実行されるものとする．

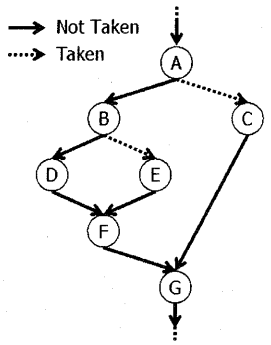

図 4: サンプルコード

1 回目の実行では，BTB にはサンプルコード中の分岐命令の分岐先アドレスは格納されていない．そのため，各ブロックの後続パス側のブロックが投機的に実行される．すなわち，ブロック A ではブロック B，ブロック B ではブロック D がそれぞれ投機実行の対象となる．しかし，ブロック B の分岐命令が成立しブロック E に分岐するため，投機実行が失敗しペナルティが発生する．

2 回目の実行では，BTB にブロック E の先頭のアドレスが分岐先アドレスとして格納されている．ブロック A では，まだ分岐命令が成立していないため，再びブロック B だけ投機的に実行される．続いてブロック B では，BTB に分岐先アドレスが格納されているので，DP 手法を適用することができる．ここではブロック D と E がそれぞれ対応する FSP に投入される．今回は，DP 手法が行われているため，ペナルティが発生することなくブロック E の実行を進めることができる．

4.3 投機実行中に現れる分岐命令

2.2 でも述べたように以前に示した合成型 DP 手法では，投機実行中に分岐命令が現れた場合には，先行する分岐命令の分岐先が確定するまで NOP 命令を挿入して対処していた．しかし，投機実行中に分岐命令が頻繁に現れる場合には，DP 手法の効果がほとんど得られないという問

題がある．この問題を解決するために，今回提案した2つのDP手法では投機実行中に分岐命令が現れた場合でも，DP手法を継続して行うことにする．以下に2パス型DP手法とBTB型DP手法を併用した場合の分岐命令の処理について述べる．

投機実行中に分岐命令が現れた場合，その後続パスのVLIW命令をそのままFSPに投入する．これと同時に，BTBにアクセスして，分岐先アドレスを取得しておく．先行する分岐命令の分岐先が，分岐命令が現れたパスに分岐した場合，取得しておいた分岐先アドレスから分岐先パスのVLIW命令をフェッチし，無効化したFSPに投入する．なお，投機実行中に現れた分岐命令の分岐先パスに関しては，発行が遅れることになる．そのため，投機実行中に現れた分岐命令が分岐先パスに分岐した場合，発行が遅れた分のペナルティが発生することになる．

図5の例に上記の処理を説明する．なお，図5に示すグラフの各ノードは1つのVLIW命令を表しており，それぞれのVLIW命令の分岐先アドレスは既にBTBに格納されているものとする．

図5の例では始めに，VLIW命令Aの後続パス（図5の1のパス）と分岐先パス（図5の2のパス）がそれぞれ対応するFSPに投入され，DP手法が適用される．VLIW命令Aの分岐先が後続パス（VLIW命令B）に確定した場合，VLIW命令Bの分岐先パス（図5の3のパス）が，無効化された分岐先パスのFSPに新たに投入される．その後，図5の1のパスの後続のVLIW命令と図5の3のパスに対してDP手法が適用される．ここで，VLIW命令BがVLIW命令Eに分岐した場合，ペナルティが発生することになる．

このように，投機実行中に分岐命令が現れる限り，DP手法を適用し続けることになる．

5 性能評価

SPEC95intベンチマークのプログラムに対して，今回提案した手法の評価を行った．評価に用いたVLIWプロセッサは，MIPS社のR3000^{TM}をベースにしており，128ビットのVLIW命令

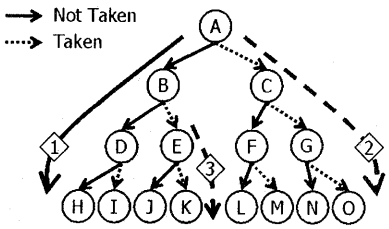

図5: 投機実行中に分岐命令が現れる例

に最大4つの命令を指定することができる．ただし，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ものとした．

評価は，遅延分岐方式を適用した場合，合成型DP手法を適用した場合，2パス型DP手法を適用した場合，BTB型DP手法と2パス型DP手法の両方を併用した併用型DP手法を適用した場合，のそれぞれ場合についてIPC(Instructions Per Cycle)を評価した．なお，BTB型DP手法に用いたBTBの構成は，1024エントリを保持できる4Wayセットアソシアティブ方式を採用した．

評価はソフトウェアシミュレーションによって行い，各ベンチマークプログラムはGNUのGCC 2.7.2.3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手法で再スケジューリングしたものを使用した．提案手法の効果を確認するため，命令キャッシュにはキャッシュミスは発生しないものと仮定した．

図6に，それぞれ投機実行サイクル数を1～4に変化させた場合の各手法を適用したときのIPCを示す．なお図6のIPCは，SPEC95intの各プログラムを実行したときのIPCの平均値を示している．

図6から，2パス型DP手法のIPCが最も高く，その他，IPCが高い順に併用型DP手法，合成型DP手法，遅延分岐方式となっていることがわかる．

また，合成型DP手法に比べて，2パス型DP手法では3%～18%，併用型DP手法では3%～15%ほどIPCが向上している．IPCの差は，遅延サイクル数が増加するにしたがって高くなっ

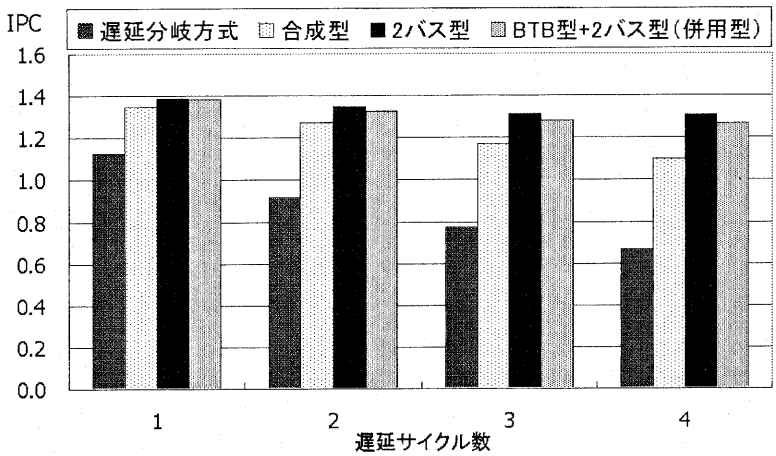

図 6: 各手法を適用した VLIW プロセッサ上で SPEC95int を実行した場合の IPC

ており，今回提案した 2 つの DP 手法が，パイプライン段数が多い VLIW プロセッサに対しても有効であることがわかる．

2 バス型 DP 手法と併用型 DP 手法を比較してみると，投機実行サイクル数 4 の場合，併用型 DP 手法の IPC が約 3% ほど低くなっている．これは BTB 型では，初めて現れる分岐命令に対しては DP 手法を適用できないためだと考えられる．しかし BTB 型 DP 手法を適用した場合，VLIW プロセッサの構成が最も単純になるため，この程度の性能低下は，クロック周波数を向上することで挽回できると考えられる．

6 おわりに

今回，以前に提案した合成型 DP 手法の問題点を解決するために，パイプラインを 2 重化した 2 バス型 DP 手法と BTB を利用した BTB 型 DP 手法を提案した．また性能評価を行った結果，今回提案した 2 つの手法の有効性を示した．

なお，合成型 DP 手法と 2 バス型 DP 手法を併用した場合，より多くのパスに対して同時に投機実行を適用することができると考えられる．今後の課題として，上記の考えをもとに，DP 手法を拡張した VLIW プロセッサ向けの複数パス投機実行手法について検討していく予定である．

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] 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.

[4] 島尻寛之，吉田たけお，“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，” 情報処理学会研究報告，2002-ARC-147, pp.97 – 102, 2002.

[5] 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.

[6] Gary Scott Tyson,“The Effects of Predicated Execution on Branch Prediction,”Proc. 27th Annual International Symposium on Microarchitecture, pp. 196–206, 1994.

[7] Scott A. Mahlke, Richard E. Hank, James E. McCormick, David I. August and Wen-mei W. Hwu,“A Comparison of Full and Partial Predicated Execution Support for ILP Processors,”Proc. 22nd Annual International Symposium on Computer Architecture, pp. 138–149, 1995.