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

古関 聰[†] 小松秀昭[††] 深澤良彰[†]

近年，プロセッサのパイプライン速度の向上が目覚ましい．このようなアーキテクチャでは，条件分岐に際する分岐予測や命令供給が非常に重大な問題である．我々は，これまでに，非数値計算プログラムを含む汎用アプリケーションの高速実行を目指してVLIWの改良提案を行ってきた．しかしながら，非数値計算プログラムを効率良く実行するためには，これまでの分岐予測や命令供給方法では，解決できない問題点が存在する．なぜならば，効率の良い分岐処理をするためには，①分岐予測の正確さ，②分岐予測が成功したときの命令供給の効率，③分岐予測が失敗したときの命令供給の効率を同時に向上させなければならないが，近年の高いパイプラインピッチ，深いパイプライン段数を持ったアーキテクチャにおいて，これらすべてを同時に向上させることは難しいと考えられるからである．本論文では，分岐の動作特性を2つに分類し，これらに合わせた2つの分岐処理機構をVLIWに追加することで，この問題点を解決することを試みた．また，これらの機構を追加したことによる性能改善の評価を行い，本機構の有用性を確認した．

Branch-handling Mechanisms in the Extended VLIW Processor GIFT

Akira Koseki,[†] Hideaki Komatsu[††] and Yoshiaki Fukazawa[†]

The pipelining techniques of processors have been improved, and the issue rate of instructions has also been increasing. In these architectures, branch prediction and instruction issue become more and more important. We proposed an extension of VLIW processors to achieve a high performance in executing various applications including non-numerical programs. However, there exist some serious problems with branch prediction and instruction issue to execute non-numerical programs, because it is very difficult to improve following things simultaneously on an architecture with high issue rate of instructions and a deep instruction pipeline: ① accuracy of branch prediction, ② efficiency of instruction issue when branch prediction is correct, ③ efficiency of instruction issue when branch prediction is incorrect. In this paper, behaviors of branches are categorized into two classes and two extensions of VLIW are proposed to solve the problems respectively. At last, we evaluate the performance of our processor with the new extensions.

1. はじめに

VLIWは，これまで，主に科学技術計算を対象として研究されてきたアーキテクチャであり，それを背景としてトレーススケジューリングなどのコンパイラ技術が開発されてきた[1]．我々は，VLIWアーキテクチャの問題点を克服し，科学技術計算も含めたOSやコンパイラなどすべてのソフトウェアに対して，並列・高速化を図ることのできるアーキテクチャとコンパイラフレームワークを確立することを目的として研究を行ってきた[2),3)]．

本論文では，VLIWの更なる改良として，ジャンプ発生時のパイプラインの分断による実行効率低下を防ぐ機構について述べ，分岐処理の効率に関する評価を行う．

2. 本研究の背景

我々は，論文2)において，非数値計算プログラムをVLIW上で実行する場合に発生する問題点を明らかにし，これを解決するためのアーキテクチャGIFT（Guarded Instruction architecture for Fine-grain Technique）を提案した．具体的には，コンパイラによる並列化をサポートするハードウェアを導入することで，VLIWの並列処理能力を十分に活かすことを試

† 早稲田大学理工学部
School of Science & Engineering, Waseda University
†† 日本IBM株式会社東京基礎研究所
Tokyo Research Laboratory, IBM Japan, Ltd.

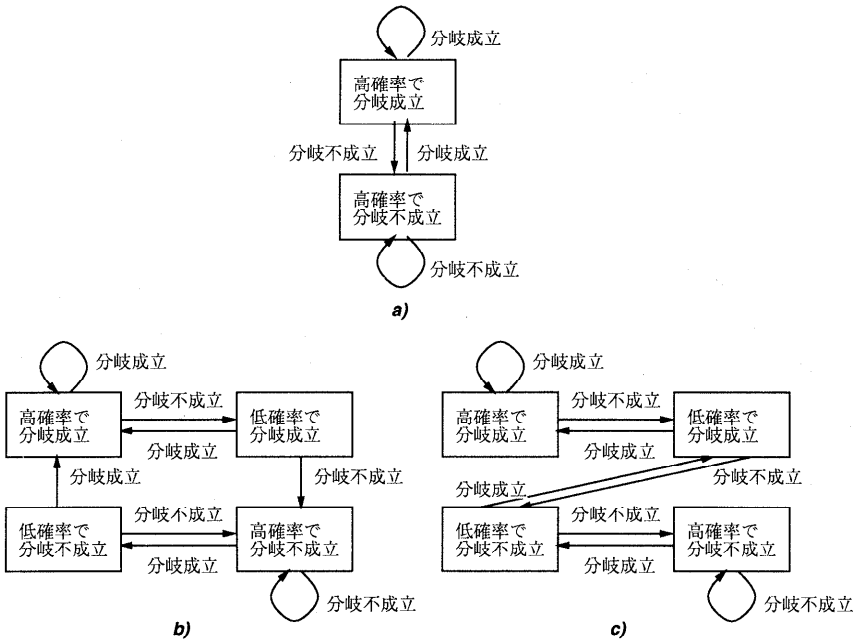


図1 分岐の履歴に基づいた分岐予測
Fig. 1 Branch predection based on branch history.

みた．しかしながら，非数値計算プログラムを効率良く実行するためには，依然として分岐処理に関する非効率が問題点として残っていると筆者らは考えている．

近年では，プロセッサのパイプライン速度の向上が目覚ましいが，このような高いパイプラインピッチ，深いパイプライン段数を前提とすると，非数値計算プログラムの分岐処理には以下に述べる問題点が存在する．

効率の良い分岐処理を行うためには少なくとも次の3点を考えることが重要である．

- 分岐予測の正確さ
- 分岐予測が成功したときの命令供給の効率
- 分岐予測が失敗したときの命令供給の効率

しかしながら，高いパイプラインピッチなどの上記の前提において，これらすべてを同時に向上させることは難しい．

分岐予測を正確にするためには様々な方法があるが，その中でも，分岐の履歴を数種類の状態で表し，その状態間を遷移しながらダイナミックに予測を行う方法[4]が代表的である．

図1 a) は一履歴に基づいた分岐予測，図1 b)，c) は二履歴に基づいた分岐予測を示す状態遷移図である．これらに基づいた予測は，数値計算を主体としたアプリケーションではかなり高い予測成功率を得ることができる．しかしながら，非数値計算プログラムではこの方法を用いても予測成功率はあまり高くならない．このため，分岐履歴を表す状態を増やす方法[4]や，ブランチコリレーションを利用した方法[5]，また，2レベル分岐予測と呼ばれる，過去数回の分岐のパターンについて図1の状態遷移を行う方法[6),7]が提案されている．しかし，我々は，不規則に変化する非数値計算プログラムにおける分岐のすべてを，このような方法で包括して処理するのは難しいと考える．また，処理を複雑にすることで，予測アドレスを得るまでの時間が延び，予測が成功したときの命令供給の時間を長くしてしまうことも問題である．

予測が成功したときの命令供給については，予測アドレスを得るまでの時間の短さがその効率を決定する重要なファクタである．筆者らは，文献8) において，VLIWの命令キャッシュに，予測を行うための情報と予測アドレスを格納し，ジャンプ時にも連続的な命令供給を行う試みを実行してその効果を確認している．この方法は非常に効果的であり，UltraSPARC等のプロセッサでも同様の方式が実装されている[9]．このような高速な命令供給は非常に重要であるが，分岐予測機構が複雑になってしまっては，非常に短いパイプラインピッチに合わせてアドレスを予測し，命令供給を行うことは難しい．

一方，最近では，予測が失敗したときのペナルティが大きくなっている．分岐予測による命令供給は投機

的であり，予測が成功したかどうかが実際に判明するのは数サイクル後のことである．このため，パイプラインの段数が深くなるにつれ，予測が外れたときのペナルティが増大している．非数値計算プログラムでは，予測の失敗は少なからぬ割合で発生すると考えられるので，プログラム全体での命令供給の効率が悪くなると考えられる．

3. 命令供給機構の設計方針

前章において，高いパイプラインピッチ，深いパイプライン段数，非数値計算プログラムの命令供給を考えた場合，難しい問題点が存在することを指摘した．これらの問題を，我々は以下のように考えている．

プログラムは様々な制御動作によって処理が行われ，それぞれの動作にはそれぞれの特性がある．これまでの分岐予測方法は，すべての制御動作を対象としており，その予測のアルゴリズムは，「分岐とはこのような動きをするものである」と仮定されて設計される．しかしながら，プログラムの制御の流れは入力データにより様々に変化し，どのように複雑なアルゴリズムを用意したとしても，意図した動きが得られないデータの存在は明らかである．一方，ループを構成する分岐や例外処理の分岐等は，入力されるデータにかかわらずその動作は規則的であり，その規則をアルゴリズムに反映することで，十分正確な分岐予測を行うことができる．

すなわち，すべての分岐を

- 規則的な分岐（規則的な挙動を示す分岐）
- 不規則な分岐（不規則な挙動を示す分岐）

に分類し，それぞれの特性に合わせた分岐処理を導入することが必要である．

これまでの方法は，この違いを考慮することなく分岐予測を行ってきたので，前章で述べた問題に直面せざるをえない．そこで，我々は，VLIWにおける命令供給の問題を以下のように解決する．

(1) 不規則な分岐に対しては，ジャンプの方向を予測することをやめ，ジャンプをせずに条件処理を行う機構を導入する．
(2) 規則的な分岐は，分岐予測が有効であるので，ジャンプ履歴を基にした予測機構を活用する．
(3) プログラム中の分岐の分類を，最適化コンパイラによって行う．

以上のようなアプローチにより，以下のような利点が得られる．

(1) 挙動が規則的な分岐だけを予測すればよいので，簡単な機構で正確な予測を行うことができる．
(2) 予測機構を簡単にすることができるので，予測アドレスを得るまでの時間を短縮することができる．
(3) 不規則な分岐に対しては，分岐の両方向の命令を供給するので，予測が失敗したときのペナルティがない．

VLIWは最適化コンパイラの助けを借りながら，シンプルなハードウェアで高速な処理を行うことが特徴である．我々の拡張は，この特徴を継承したものであり，非数値計算プログラムの効率の良い命令供給のためには最適なアプローチであると考える．

本論文では，以降，4章において一般的な分岐の規則性について述べ，また，ハードウェアの有限性から，特定のマシン上で取り扱われる分岐の規則性について述べる．次に，5章でGIFTについて簡単に述べた後，6章においてGIFTプロセッサ上で実装されるべき分岐の規則性を定め，その規則性に従った挙動を行う分岐を処理するための分岐予測機構と，分岐先アドレス供給機構を導入する．次に，7章において，不規則な分岐を処理するため，条件分岐をジャンプなしで実行する条件実行機構を導入する．8章では，両機構を十分に活用するための，最適化コンパイラによる条件分岐コード生成方法について述べる．最後に，9章において，本手法の評価を行う．

4. 分岐の規則性・不規則性

前章において，プログラム中の分岐の分類について述べた．本章では，分類の際の基準となる「規則性」について述べる．

ある分岐が「本質的に」規則的であるとは，その分岐における分岐成立の可能性の高低が，その分岐や他の分岐の挙動履歴で決定可能なことを意味するものとする．分岐が本質的に規則的な場合，適当な予測機構を組み込むことによって，入力データの変化にかかわらず，(その分岐の) 分岐予測の精度を非常に高めることができる．

ある分岐が「本質的に」不規則であるとは，その分岐における分岐成立の可能性の高低が，その分岐や他の分岐の挙動履歴で決定することができないことを意味するものとする．分岐が本質的に不規則な場合，どのような予測機構を組み込んでも，高い分岐予測の精度をつねに得ることはできない．

ハードウェアの制限を考慮しない場合，様々な分岐の挙動の規則に合わせた機構をマシン上にインプリメントすることで，多くの規則的な分岐の挙動を予測することができる．しかしながら，事実上は，ハードウェ

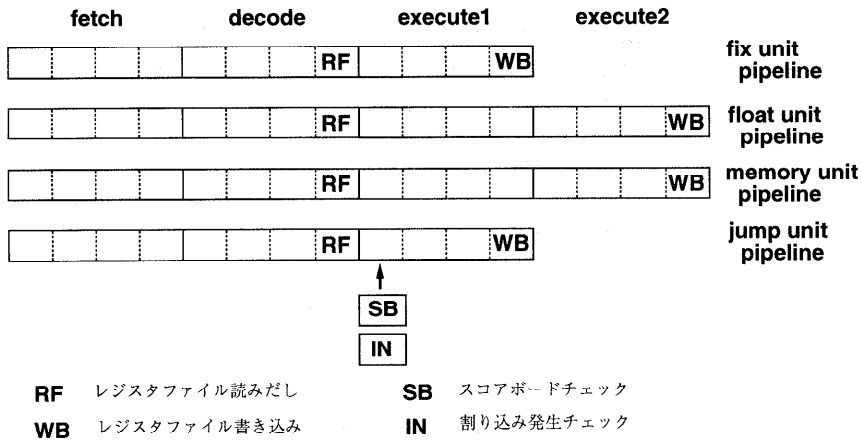


図2 パイプライン構成
Fig. 2 Pipeline architecture.

アの有限性からマシン上にインプリメントする規則を限定しなければならない．本論文では，分岐が「特定のマシン上で」規則的である，または，不規則であるとは以下のことを意味するものとする．

- 分岐成立の可能性の高低を決定する規則を1つ定めた場合，その規則に従うものを規則的，それ以外のものを不規則とする．

5. GIFTのパイプラインアーキテクチャ

GIFTは機能非均質型VLIWで，固定小数点演算ユニット，浮動小数点演算ユニット，ジャンプ処理ユニット，メモリ処理ユニットそれぞれを複数個持っている．

GIFTの基本的パイプライン構成は，**図2**のようになっている．固定小数点パイプラインとジャンプパイプラインは，フェッチステージ，デコードステージ，実行ステージから構成され，浮動小数点パイプラインとメモリパイプラインは，フェッチステージ，デコードステージ，第一実行ステージ，第二実行ステージから構成されている．各命令は，この各ステージをメジャーサイクルとして投入される．また，各ステージは4つのマイナサイクルに分割されている．

6. GIFTの分岐予測機構および分岐先アドレス供給機構

6.1 GIFTが採用する規則性

我々は，分岐予測の方法として，図1a)に示す状態遷移を表す規則を採用し，これに従う分岐を「GIFT上で」規則的と定める．また，それらの分岐のための予測機構を導入する．この規則は，1度分岐方向が変われば，以降，その方向に分岐が続けて行われることを想定したものである．GIFTでは，分岐予測を以上のような挙動を示す分岐に限定している．このことによって，単純なハードウェアで十分正確な分岐予測を行うことが可能である．また，単純なハードウェアを用いることで，動作を非常に高速にすることが可能である．

6.2 本機構の動作

6.2.1 次候補アドレスによる分岐先アドレス供給

GIFTでは，命令キャッシュメモリに格納されている命令語に，次に供給すべきアドレス（次候補アドレス）を付加することで，方向予測が成功したときには，分岐時にもパイプラインを分断しない高速な分岐先アドレス供給機構を導入している．

次候補アドレスとは，ある命令語が実行された後に実行されると予想される命令語のアドレスである．命令語の供給はすべて次候補アドレスを参照して行われ，通常のようなプログラムカウンタを用いた命令供給は行っていない．命令キャッシュメモリから供給された命令語と次候補アドレスは，デコードユニットに送られる．また，同時に，読み出した次候補アドレスが示すアドレスを用いて次の命令語が供給される．命令のデコードが終わると，実際に実行されるべきアドレスが判明するので，このアドレスと次候補アドレスが比較される．これらが一致した場合はそのまま処理が続けられるが，もし，一致しなかった場合は，実際に実行されるべきアドレスから新たに命令供給を行うよう命令供給機構に通知するとともに，パイプラインを無効化する．

命令語がメインメモリからキャッシュメモリに格納される際，次候補アドレスには，その命令語の次の命令語のアドレスが格納される．したがって，次候補ア

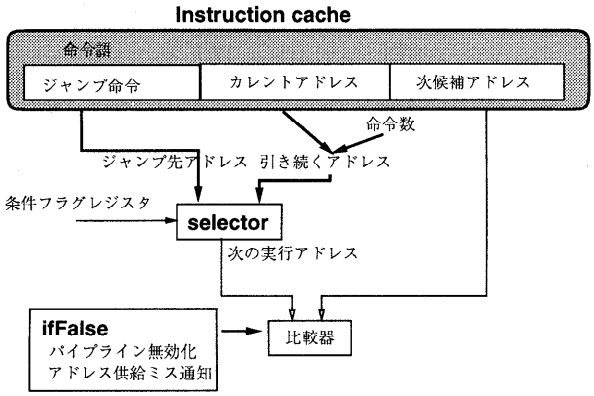


図3 分岐予測および分岐先アドレス供給機構
Fig. 3 Branch prediction and branch address providing mechanisms.

ドレスの書換えが行われていない状態では，通常のプロセッサにおいてプログラムカウンタを1つずつ増やしていった場合と同様の命令供給が行われる．次候補アドレスにその命令語の次の命令語のアドレスが格納されている状態でジャンプが発生した場合は，予測的にパイプラインに投入した命令語を無効にして，新たにジャンプ先アドレスの命令語から供給を行わなければならない．このように，次候補アドレスと実際に実行されるべきアドレスが異なった場合は，パイプラインの分断が発生する．この損失を抑えるため，命令語の次候補アドレスを，以下に述べるアルゴリズムで書き換える．この書換えにより，次回に次候補アドレスが正しいアドレスを示した場合には，再び連続的な命令供給が行われる．

6.2.2 次候補アドレスの書換え

次候補アドレスの書換えは，次候補アドレスが正しいがどうかの信号（アドレス供給ミス通知）によって，以下のように行われる．

アドレス供給ミス通知が真である場合，すなわち，実際に供給したアドレスと次候補アドレスが異なっている場合，次候補アドレスを，実際に供給したアドレスに書き換える．アドレス供給ミス通知が偽である場合は，正しいアドレスを予測しているので，なにも行わない．

このアルゴリズムによって次候補アドレスを書き換えることにより，図1a) に相当した分岐予測を行うことができる．

6.2.3 本機構の実装

図3 に分岐予測および分岐先アドレス供給機構の実装，**図4** に計算タイミングを示す．

図3に示すように，命令語のジャンプフィールドに置かれた命令から，ジャンプ先アドレスが計算される．

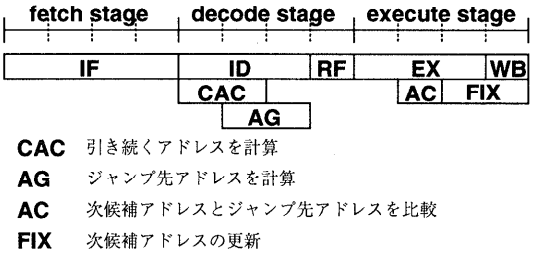


図4 タイミング
Fig. 4 Timing.

GIFTでは，直接ジャンプと，相対ジャンプをサポートしており，ジャンプ先アドレスは実行サイクルの直前までに決定される．また，現在の命令語の次の命令のアドレス（引き続くアドレス）も，実行サイクルの直前までに計算される．実行サイクルに入ると，条件フラグレジスタを参照して，次に実行すべき命令のアドレスが決定される．次に実行されるべき命令のアドレスと次候補アドレスの比較は，実行ステージの2番目のマイナサイクルで行われる．

6.3 本機構の導入のコスト

本機構の導入に関して議論しなければならない点は，アドレス供給機構，次候補アドレスの書換え機構の導入が，高ピッチパイプラインプロセッサの設計に関して妨げにならないかどうかである．

まず，アドレス供給機構においては，次に実行すべきアドレスは命令キャッシュ上に置かれているため，非常に高速な動作が可能である．また，次候補アドレスの書換えは他の演算と並行して行うことができるので，全体のパイプラインピッチを圧迫するようなことはない．したがって，本章で提案した機構は，高ピッチパイプラインプロセッサに非常に適したものであると考えられる．

7. GIFTの条件実行機構

本章では，前章で定めた規則性に従わない分岐を処理するための条件実行機構について述べる．

7.1 不規則な分岐と条件実行

図5は，ある配列の要素について，奇数のものと偶数のものの個数をカウントするプログラムである．このプログラムでは，入力データに偏りがなければ，奇数かどうかを判定する分岐の動作に規則性はないと考えられる．

これを，分岐が失敗したときのペナルティがcサイクルであるプロセッサで実行することを考える．分岐予測の正確さをp (< 1)，if文の実行回数をNとすると，全体で，$c \times (1-p) \times N$サイクルが無駄になる計算になる．図5のようなプログラムにおいて，高い分岐予測の正確さを得ることは難しく，また，最近のプロセッサでは分岐が失敗したときのペナルティは小さくはないことを考えると，この$c \times (1-p) \times N$サイクルの無駄は深刻である．

以上のような分岐は，ジャンプを行わずに処理することで，無駄を軽減することが可能である．これは，プログラムのthenブロックの命令とelseブロックの命令を同時に供給し，分岐の条件に合わない命令を無効化するハードウェアを導入することで実現できる．このような処理を，本論文では条件実行と呼び，そのためのハードウェアサポートを条件実行機構と呼ぶ．同様な処理は，文献10)~13)でも提案されている．

条件実行を利用して命令供給を行うことで，分岐の動作の規則性にかかわらず，パイプラインの分断が発生することなく命令を処理することが可能である．

7.2 条件実行機構の実装

GIFTの条件実行機構は，VLIWの1命令語（a very long instrucion word）の各命令に付加された，実行時に満足していなければならない条件部（ガード）と，条件フラグレジスタ（cf register）と，条件に合わない命令を無効化する仕組みから成っている．条件部は，各条件フラグレジスタと1対1に対応したフィールドで構成されており，この条件部を使って，各命令の実行条件を記述することができる．

```
for (i= 1; i < N; i++){
   tmp = ARRAY[i];
   if (ODDP(tmp)){
      odd++;
   }else{
      even++;
   }
}
```

図5 サンプルプログラム
Fig. 5 Sample program.

図6において，条件部のcc_jと条件フラグレジスタcf_jが対応している．cf_jはプログラム上の比較命令等によって真偽がセットされる．各命令の条件部には，条件フラグレジスタのそれぞれをどのように参照するかを決定するため，cfに対応したフィールドに次の3つのうちの1つを記述することができる．

T 対応するcfが真のときに真
F 対応するcfが偽のときに真
* 対応するcfの真偽にかかわらず真

GIFTにおいては，分岐の両方向の命令が同時に供給される．条件部が記述された各命令は，実行時にすべてのcc_jとcf_jが比較され，cf_jの状態が完全にcc_jの記述にマッチしたもののみが実行される．その他の命令は，演算結果をレジスタに書き込むことを禁止するなど，命令の副作用が生じる書き込みを取り消すことによって無効化する．

条件実行機構における各処理の計算タイミングを**図7**に示す．実行条件は，デコードステージで決定され，条件フラグレジスタの内容はデコードステージの終わりまでに読み出される．これらの比較は，演算と並行して実行ステージの第1マイナサイクルで行わ

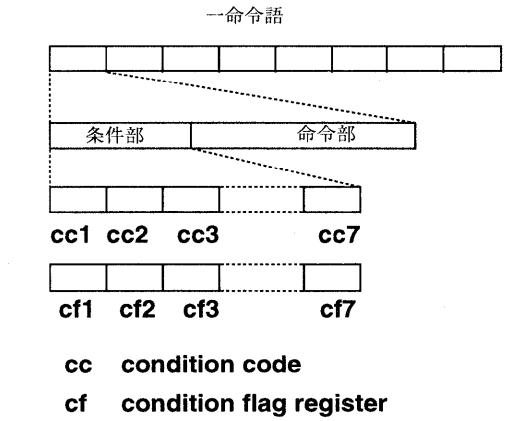


図6 ガード付き命令と条件フラグレジスタ
Fig. 6 Guarded instructions and condition flag registers.

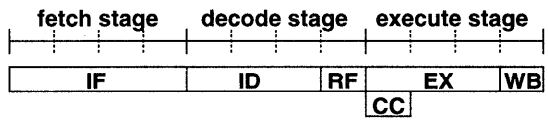


図7 条件実行機構のタイミング
Fig. 7 Timing of conditional execution mechanism.

れる．

7.3 条件実行機構導入のコスト

6.3節と同様に，条件実行機構の実現のためにハードウェアが複雑になってしまい，パイプラインピッチを圧迫してしまうことがないかどうかを議論することが必要である．

条件実行機構の導入に際して追加されるハードウェアは，条件フラグレジスタ，実行条件（ガード）を保持するラッチ，比較器などである．命令無効化までの動作は前節で示したとおりであるが，図7に示すとおり，レジスタの読み込み，比較など，すべて他の動作と並行して行われている．また，それぞれの動作は単純であり，条件実行機構の導入は，パイプライン速度の向上を妨げるものではないと考えられる．

8. 分岐の分類

本手法では，分岐の挙動の規則を6.1節に示すように決定した．4章の議論より，これに従うものを規則的，それ以外のものを不規則として扱う．GIFTでは，規則的な分岐の処理のために，6章で述べた分岐予測および分岐先アドレス供給機構，不規則な分岐の処理のために，7章で述べた条件実行機構を用意している．

GIFT用の最適化コンパイラでは，プログラム中の分岐を分類し，規則的なものの多くが予測機構で処理され，不規則なものの多くが条件実行機構で処理されるようにする．本論文では，分類の1手法として以下のものを実装した．

不規則：分岐の方向が前向きのもの

規則的：分岐の方向が後向きのもの

ここで，分岐の方向が前向きとは，分岐を構成するジャンプが両方とも前向きであることを示し，後向きとは分岐を構成するジャンプのどちらかが後向きであることを示す．また，ここで，ジャンプが前向きとは，ジャンプ先のアドレスが分岐命令のアドレスより小さい場合を意味するものとする（GIFTでは，命令が小さいアドレスから大きいアドレスに向けて供給されている）．同様に，後向きとはジャンプ先のアドレスが分岐命令のアドレスより大きい場合を意味するものとする．プログラムのすべての分岐は，以上の規則に従って分類され，前向きの分岐のうち，有限の条件フラグレジスタで記述可能なもの，および，事前に分岐の偏りが判明しているものを除いた分岐に対し，ガード付きの命令が生成される．また，その他の分岐に対しては，通常の条件ジャンプ命令が生成される．前向きの分岐の偏りは，例外処理や前判定のループにおけるif文に現れることがある．本コンパイラでは，プロファイルやプログラムの構造を基に分岐の偏りを判断し，条件実行で処理しない前向きの分岐を決定する．

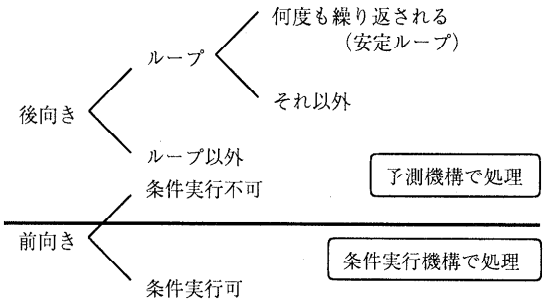


図8 分岐の分類
Fig. 8 Classification of branches.

本手法での分岐の分類を図8に示す．本手法では，ループはすべて予測機構により処理される．特に，何度も繰り返されるループ（安定ループと呼ぶ）における分岐は，規則的な挙動を示すので，このようなループを構成する分岐を高い精度で予測することができる．しかしながら，繰返し数が非常に少ないループと，ループを構成しない後向きのジャンプ，および，ガード付き命令で実行できない前向きの分岐を構成するジャンプは，予測の精度を落とす原因となる可能性がある．

9. 評　　価

いくつかのプログラムに対し，分岐予測の成功率を計測した．また，その中のチェッカーボードパズル（ある決まった形状を持つブロックを，すき間なく長方形に並べる問題）と8クイーンパズルの2つのプログラムに対して，プログラム中の各ジャンプの分岐状態と分岐予測成功率を計測した．プログラムの選択に関しては，非数値計算プログラムについての評価を示すため，非数値計算プログラムというものを不規則な挙動を示す分岐がある程度含まれているプログラムだととらえ，不規則な挙動を示す分岐の分岐回数が全体の3割以上であるようなものを選択した．

ターゲットアーキテクチャは，固定小数点演算，ロードストア，浮動小数点演算，分岐をそれぞれ最大4個実行可能で，条件フラグレジスタを4個持つGIFTを用いるものとした．並列ユニットの個数は，本評価のデータに直接影響を及ぼすものではないが，文献2), 3)のデータに基づき，非数値計算プログラムが持つ並列度に不足がないように決定した．

本評価では，コンパイラにより分岐の分類と条件実行コードの生成を行い，そのコードの実行をシミュレーションした．分岐の分類は，8章で述べたとおり，

表1 チェッカーボードパズルの分岐状態
Table 1 Behaviors of branches in checker board puzzle.

分岐番号	方向	分岐回数	ジャンプ回数	非ジャンプ回数	条件実行	安定ループ
0	backward	128	112	16	-	○
1	backward	16	14	2	-	○
2	backward	10	9	1	-	○
3	backward	64	56	8	-	○
4	backward	8	7	1	-	○
5	backward	13	12	1	-	○
6	forward	1878245	1688505	189740	○	-
7	forward	5764752	3793060	1971692	○	-
8	forward	1971692	973154	998538	○	-
9	forward	998538	547360	451178	○	-
10	forward	451178	190932	260246	○	-
11	forward	260246	91963	168283	○	-
12	forward	168283	34218	134065	○	-
13	forward	5764752	134065	5630687	×	-
14	forward	134065	2	134063	×	-
15	forward	407024	43952	363072	○	-
16	backward	407024	272961	134063	-	×
17	backward	5764752	5630688	134064	-	○
total	-	23970790	13401070	10569720	-	-

表2 8クイーンパズルの分岐状態
Table 2 Behaviors of branches in 8 queen puzzle.

分岐番号	方向	分岐回数	ジャンプ回数	非ジャンプ回数	条件実行	安定ループ
0	forward	5888	736	5152	○	-
1	backward	5888	5152	736	-	○
2	backward	736	644	92	-	○
3	forward	46752	7196	39556	○	-
4	forward	39556	6468	33088	○	-
5	backward	33088	31040	2048	-	○
6	forward	15720	2056	13664	○	-
7	forward	2056	92	1964	×	-
8	backward	15720	13755	1965	-	○
total	-	165404	67139	98265	-	-

ジャンプ先アドレスの方向に基づいて行った．また，以下の図表におけるデータは，コンパイラが生成したコードに分岐の挙動を記録するカウンタと予測コードを埋め込み，採集したものである．

まず，**表1**，**表2** に，チェッカーボードパズルと8クイーンパズルの分岐と分岐方向，ジャンプ回数，非ジャンプ回数，方向が前向きであった場合は，条件実行によって処理を行うかどうか，および，方向が後向きであった場合は，その分岐が安定ループを構成しているかどうかを示した．また，**表3**，**表4** に，それぞれの分岐に対する分岐方向の予測成功率を示した．ここで，takenはジャンプすることを予測，not takenはジャンプしないことを予測するものとしている．また，一履歴，二履歴b，cは，それぞれ図1に示した各方式を表すものとした．

次に，**図9** に，以上の2つのプログラムに加えて，クイックソート，ヒープソート，シェルソート，蜂の巣パズル（bee，7つの正六角形の隣接した辺の色を合わせる問題），ファイル解凍・圧縮プログラム（gzip，gunzip），ワードカウンター（wc），高速フーリエ変換の各プログラムに対し，それぞれの予測方法における分岐予測の平均成功率を示した．「□ + 条件実行」となっている欄は条件実行が可能である分岐（たとえばチェッカーボードパズルでは，表1において条件実行の欄が○になっている分岐）を条件実行機構で処理し，その他の分岐を「□」の方法で分岐予測することを示すものとした．

次に，**表5** に，いくつかのプログラムの全後向き分岐に対する，安定ループを構成しない分岐の個数と実行回数の割合，および，安定ループを構成する分岐と，構成しない分岐の，それぞれの本方式による分岐予測成功率を示した．また，**表6** に，プログラムの全前向き分岐に対する，条件実行を行わない分岐の個数と実行回数の割合，および，条件実行を行わない分岐

表3 チェッカーボードパズルにおける予測成功率
Table 3 Prediction accuracy in checker board puzzle.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
taken	88%	88%	90%	88%	88%	92%	90%	66%	49%	55%	42%	35%	20%	2%	0%	11%	67%	98%
not taken	12%	12%	10%	12%	12%	8%	10%	44%	51%	45%	58%	65%	80%	98%	100%	91%	33%	2%
一履歴	75%	75%	80%	75%	75%	85%	84%	84%	61%	58%	55%	56%	69%	95%	100%	79%	55%	95%
二履歴 c	86%	75%	70%	86%	75%	77%	10%	58%	59%	59%	56%	60%	74%	98%	100%	89%	67%	98%
二履歴 b	86%	75%	70%	86%	75%	85%	83%	56%	56%	50%	49%	42%	30%	87%	100%	89%	67%	98%

表4 8クイーンパズルにおける予測成功率
Table 4 Prediction accuracy in 8 queen puzzle.

	0	1	2	3	4	5	6	7	8
taken	13%	88%	88%	15%	16%	94%	13%	4%	88%
not taken	87%	12%	13%	85%	84%	6%	87%	96%	12%
一履歴	75%	75%	71%	73%	70%	88%	77%	91%	78%
二履歴 c	87%	87%	75%	70%	81%	94%	87%	91%	83%
二履歴 b	87%	87%	75%	85%	84%	95%	87%	91%	86%

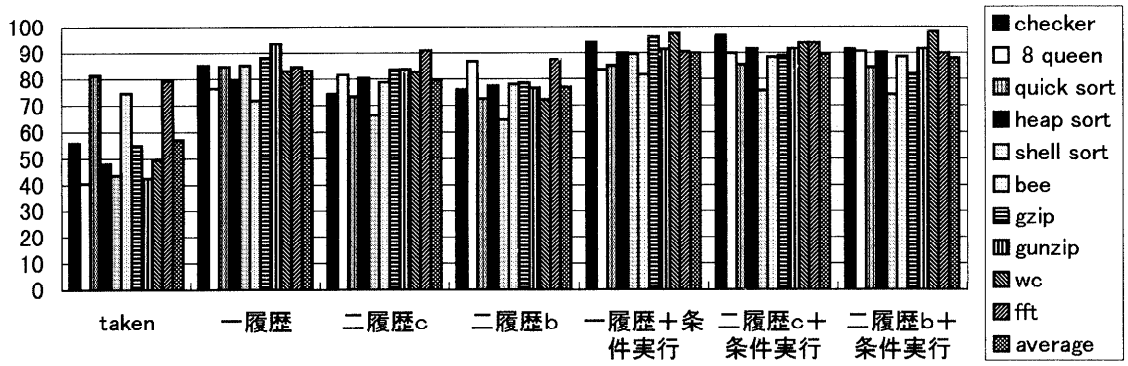


図9 分岐予測の成功率
Fig. 9 Prediction accuracy.

表5 後向きの分岐の分類
Table 5 Classification of backward branches.

		8 queen	checker	quick sort	heap sort	shell sort	bee	gzip
安定ループ以外 後向きの分岐	回数	0%	7%	23%	0%	0%	0%	3%
	個数	0%	13%	33%	0%	0%	6%	15%
安定ループの分岐の予測成功率		83%	95%	81%	85%	85%	100%	99%
それ以外の分岐の予測成功率		-	55%	66%	-	-	60%	49%

の分岐予測成功率を示した.

最後に, 図10 に, 各プログラムに対する既存の予測機構の有効性を示した.

なお, 本図表における, 後向きの分岐が安定ループを構成する分岐であるかどうかの判定は, 全体の分岐回数が数万回になるような入力データをいくつか用意し, そのトレース結果から, ループが持続するような方向に 75%以上分岐が行われたかどうかで行った.

表1, 表2 と表3, 表4 より, 分岐方向が後向きのものに関しては, 高い成功率で分岐予測ができることが分かる. これは, 後向きのジャンプは, ループを継続する等の処理であることが多く, 分岐方向が規則的で安定していることを示している. したがって, このような分岐だけを対象とすると, これまでの予測方法で十分なパフォーマンスを得ることができるといえる. 履歴を利用した予測では, 若干の違いはあるものの, 履歴の取り方にかかわらず高い精度が得られており, 後向きの分岐では, 履歴に基づいた分岐予測が有効であることが分かる.

これに比べて, 前向きの分岐では, どの方法でも成

表6 前向きの分岐の分類
Table 6 Classification of forward branches.

		8 queen	checker	quick sort	heap sort	shell sort	bee	gzip
非条件実行	回数	15%	42%	83%	28%	25%	62%	77%
前向きの分岐	個数	20%	20%	60%	71%	75%	42%	78%
分岐予測成功率		95%	91%	92%	94%	100%	98%	95%

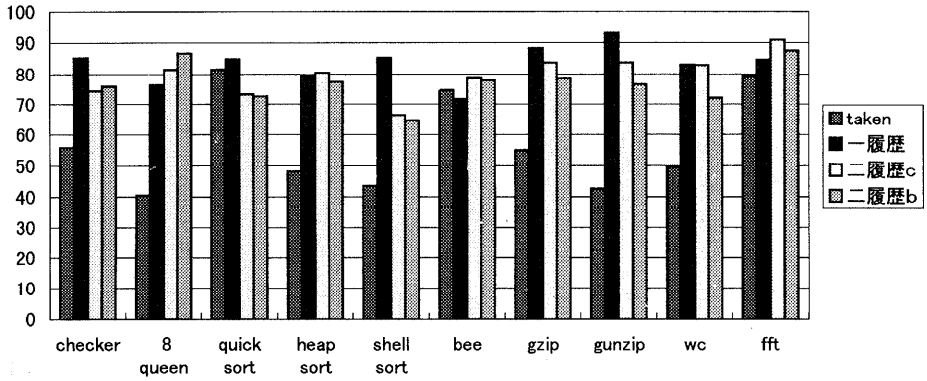


図10 予測機構の有効性
Fig. 10 Effectiveness of predection mechanisms.

功率が高くないものが存在する．これは，前向きのジャンプは，分岐の方向が不規則であることを示している．これらの分岐は，入力データにより様々な挙動をすることが予想され，固定したアルゴリズムで予測を成功させるのが非常に難しいことを示している．

すべての分岐に対しては，図10より，二履歴の方式は一履歴の場合よりもパフォーマンスが悪い場合があり，シェルソートでは20%近くの精度低下が発生している．これは，プログラムやデータの性質によって分岐の特性はまちまちであり，必ずしも複雑な予測機構が良いパフォーマンスを示すとは限らないことを示している．

我々のアプローチでは，不規則な分岐は，条件実行機構によりジャンプを行わずに処理を行う．図9は，GIFTの条件実行機構を考慮して分岐予測を行ったものを加えた，各プログラムの分岐予測成功率である．条件実行を取り入れたものでは，分岐予測の難しい前方向への分岐をジャンプせずに命令供給することで，安定して高い予測成功率を得ることに成功していることが分かる．図9より，条件実行を取り入れたものでは，どの方式も平均で85%以上の成功率を示しており，全体で10%近くのパフォーマンス向上が得られている．また，条件実行を行わない分岐に対しては，分岐方向の予測の正確さが重要である．GIFTではこの方式として，一履歴に基づく単純な分岐予測を採用し，全体として非常に高い予測成功率を得ることに成功している．これは，規則的な分岐に対しては，一履歴のもので十分なパフォーマンスが得られることによるものである．

次に，本コンパイラによる分岐の分類の妥当性を示す．表5より，各プログラムにおいて，後向きの分岐で安定ループを構成していない分岐の個数は少なく，本手法の分類が妥当であることが分かる．また，表5より，安定ループを構成している分岐の予測成功率は高いが，それ以外の分岐の予測成功率は高くないことが分かる．しかしながら，すべてのプログラムにおいて，安定ループを構成していない分岐の分岐回数の割合は低く，全体に対する影響は低い．したがって，後向きの分岐を規則的な分岐であるとする本手法の分類法は妥当であるといえる．

一方，前向きの分岐の分類では，表6より，条件実行で処理できなかった分岐の割合は，プログラムによってまちまちであることが分かる．これは，各プログラムによって，分岐のネストの深さが異なることによる．しかしながら，条件実行で処理できない分岐が多いとしても，それは必ずしも分岐処理の効率を下げる原因とはならない．なぜならば，分岐のネストが深く，すべての前向きの分岐を条件実行で処理できないとしても，その中に規則的な分岐が含まれていれば，それらを通常のジャンプ命令で処理し，それら以外を条件実行で処理することにより，全体の分岐を効率良く処理することが可能であるからである．表6では，

プロファイルを有効に活用することにより，条件実行で処理できない分岐を例外処理などに限定することができたので，それらの分岐の予測成功率を90%以上に保持することに成功している．

結論として，GIFTでは，パズルやソート，符合化処理のような非数値計算プログラムに対して，高い分岐予測成功率を得ることができ，非常に効率の良い分岐処理が行われていることが実証された．

10. おわりに

本研究では，VLIWの分岐処理に関する問題を分析し，これを解決すべき機構を提案した．また，いくつかのプログラム中のジャンプについて，本機構の評価を行った．

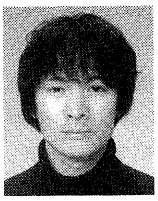
この提案は，GIFTのみに限定したものではなく，条件実行機構と単純な予測機構を兼ね備えたVLIWまたはスーパースカラプロセッサでも同様に有効である．すなわち，分岐予測機構の複雑化を避け，予測対象を規則的な分岐のみに限定することによって，高いパイプラインピッチを維持しながら，同時に正確な分岐予測を提供するプロセッサの設計に役立つものと思われる．

これからの研究としては，条件実行を十分に活かすためのコンパイラ技法を確立することが必要である．また，次候補アドレスを積極的に利用し，プログラムの実行中にこのアドレスを自由に書き換えることにより，更なる命令パイプライン分断の緩和手法を構成することを考えている．

参考文献

1) Ellis, J.R.: *Bulldog: A Compiler for VLIW Architectures*, MIT Press (1985).
2) 小松，古関，鈴木，深澤：拡張VLIWプロセッサGIFTにおける命令レベル並列処理機構，情報処理学会論文誌，Vol.34, No.12, pp.2599–2610 (1993).
3) 小松，古関，深澤：命令レベル並列アーキテクチャのための大域的コードスケジューリング技法，情報処理学会論文誌，Vol.37, No.6, pp.1149–1161 (1996).
4) Lee, J.K. and Smith, A.J.: Branch Prediction Strategies and Branch Target Buffer Design, *IEEE Computer*, Vol.17, No.1, pp.6–22 (1984).
5) Pan, S., So, K. and Rahmeh, J.: Improving the Accuracy of Dynamic Branch Prediction Using Branch Correlation, *Proc. 5th Annual Intl. Conf. on Architectural Support for Prog. Lang. and Operating Systems* (1992).
6) Yeh, T-Y. and Patt, Y.: Two-Level Adaptive Branch Prediction, *Proc. 24th Annual International Symposium on Micro Architecture*, pp.51–61 (1991).
7) Yeh, T-Y. and Patt, Y.: Alternative Implementation of Two-Level Adaptive Branch Prediction, *Proc. 19th Annual International Symposium on Computer Architecture*, pp.124–134 (1992).
8) 鈴木，小松，深澤，門倉：条件実行アーキテクチャGIFTのメモリインターフェース，信学技報，CPSY90-53, pp.91–96 (1990).
9) UltraSPARC-I Data Sheet, Sun Microsystems Inc. (1994).
10) Rau, B.R., Yen, D.W.L. and Towle, R.A.: The Cydra 5 Ddepartmental Supmercomputer, *IEEE Computer*, Vol.22, No.1, pp.12–35 (1989).
11) 安藤，中西，原，中屋：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，情報処理学会論文誌，Vol.37, No.11, pp.2039–2055 (1996).
12) Ebcioğlu, K.: Some Design Ideas for a VLIW Architecture for Sequential Natured Software, *Parallel Processing (Proc. IFIP WG 10.3 Working Conference on Parallel Processing)*, Cosnard, M., et al. (Eds.), North Holland (1988).
13) Mahlke, S.A., Lin, D.C., Chen, W.Y., Hank, R.E. and Bringmann, R.A.: Effective Compiler Support for Predicated Execution Using the Hyperblock, *Proc. 25th Annual International Symposium on Micro Architecture*, pp.45–54 (1992).

(平成8年9月17日受付)
(平成9年10月1日採録)

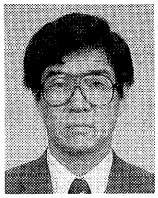


古関　聰（学生会員）

1969年生．1992年早稲田大学理工学部電気工学科卒業．1994年同大大学院理工学研究科修士課程修了．現在，同大大学院博士後期課程在学中．命令レベル並列アーキテクチャ，コンパイラの研究に従事．

小松 秀昭（正会員）

1960年生．1985年早稲田大学大学院理工学研究科修士課程修了．同年，日本IBM（株）入社．以来，東京基礎研究所において，HPFコンパイラ等の研究に従事．早稲田大学において，命令レベル並列アーキテクチャ，コンパイラの研究に従事．ACM会員．



深澤 良彰（正会員）

1976年早稲田大学理工学部電気工学科卒業．1983年同大大学院博士課程中退．同年相模工業大学工学部情報工学科専任講師．1987年早稲田大学理工学部助教授．1992年同教授．工学博士．ソフトウェア工学，コンピュータアーキテクチャなどの研究に従事．電子情報通信学会，ソフトウェア科学会，IEEE，ACM各会員．