

キュー構造プログラムカウンタによる多重投機実行機構

芝 浩 二 郎[†] 有 田 五 次 郎^{††}

分岐の影響が ILP プロセッサでプログラムを実行する際の大きな障害となっている。分岐の複数方向実行は、条件分岐の結果が判明する前に分岐の両方向を実行する方式である。従って、複数方向実行では単一方向実行より分岐予想精度の影響ははるかに少ない。しかし、複数方向実行では多くのパスを同時に実行するので多くの資源が必要である。本論文では、キュー構造のプログラムカウンタによって分岐の複数方向を多重に投機実行するモデルについて提案する。提案するモデルの機能ユニットの資源量と性能の関係及びモデルを構成する際に導入した分岐を並列実行するグループ数について、シミュレーションで確認した結果について報告する。

Multi-path Speculative Execution Model with Program-counter Queues

KOJIRO SHIBA[†] and ITSUJIRO ARITA^{††}

The branch effects are the biggest obstacle to gaining significant speedups general-purpose code on ILP processor. Multi-path is the execution of code from both pathes of a branch before the condition of the branch has been evaluated. Therefore, the effects of branch prediction accuracy on multi-path is less than on the single-path. But, multi-path execution takes many resources for configuration, because it needs the execution of many pathes concurrently. In this paper, we propose multi-path speculative execution model with program counter queues. We describe the results that we simulate the model in the following case; the relationship between the number of function unit resources and performance, the number of groups which execute branch-path codes concurrently in this model.

1. はじめに

条件分岐によるペナルティへの対処として高精度の分岐予想に基づく単一方向の投機実行が主流であるが、分岐の両方向をフェッチして実行する方式の研究は従来はコスト高となるため、マイクロプロセッサへの適用に関してはあまり検討されなかった。近年、制御依存を多重実行によって大幅に減らし、処理性能を数十倍に向上できる可能性についての報告¹⁾があり、現実のものとして検討されつつある。また、シミュレーションにより文献¹⁾の可能性を検証した報告²⁾もある。我々は、この制御依存を回避するために複数の分岐先を多重実行する機構として、プログラムカウンタをキュー構造とした機構を検討している。この機構は FIFO キューを同期手段に用いたマルチプロセッサ方式³⁾をベースとしている。

FIFO キューを同期手段に用いる方式とはプログラムレベルにおいて制御依存による複数の分岐パスをスレッドと考えた場合、スレッド単位でスレッドの実行順序を指定し、実行順序を保持した形で各プロセッサ毎に備えられた FIFO キューにスレッドの開始番地を貯え、実行する方式である。実行中に同期の操作が不要になるこの機構をベースにすることによって、機能ユニットを多数準備して実行する方式において、ダイナミックに同期をとる必要がない機構を前提にハードウェアを検討することができる。

本論文では、この FIFO キューモデルの機構を複数の分岐先を多重実行するプログラムカウンタキュー(PCQ)プロセッサ機構としてモデル化したので、その機構について提案する。また、この提案するモデルで必要となる機能ユニットの資源量と性能の関係、及びモデルを構成する際に導入した分岐を並列実行する際に必要となるグループの数の数について、シミュレーションの結果に基づき予想した結果について報告する。分岐を全て並列的に多重実行する場合、実行すべき並列パスが指数関数的に増大すると予想されるが、PCQ プロセッサモデルでは、並列パスの実行を行うグループ数の増加をあまり招かずに行うことができると考えている。

[†] 鹿児島工業高等専門学校情報工学科
Information Engineering of Kagoshima National College
of Technology

^{††} 九州工業大学情報工学部
Faculty of Computer Science and Systems Engineering,
Kyushu Institute of Technology

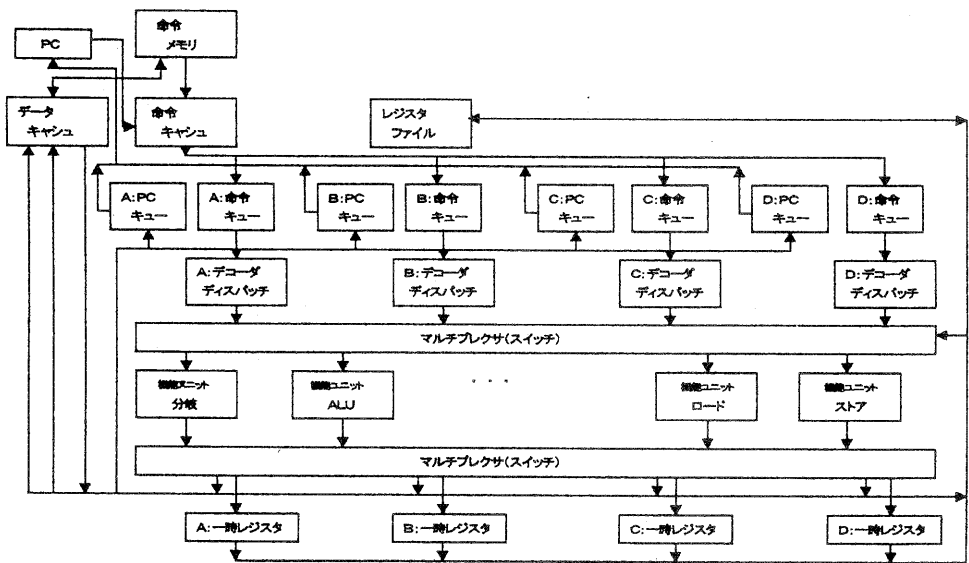


図1 プログラムカウンタキューを備えたプロセッサの構成

本論文では、2. でPCQ プロセッサの構成について述べ、3. でPCQ プロセッサを用いて分岐を多重実行する際の各部の動作について述べる。4. でPCQ プロセッサモデルをシミュレーションするための環境について説明し、5. でシミュレーションによる性能評価結果について報告する。

2. PC キュープロセッサの構成

本論文で提案するPCQ プロセッサは、基本的にはプログラム記述によってデータ依存を明示し、複数のプロセッサに並列に実行可能なプログラム部分を分散させ、データ依存に関係する同期に関してはその依存関係をFIFO キューにプログラムカウンタ（PC）値を実行順序に格納することによって制御する機構^{3),6)}に基づいている。本論文で提案するPCQ プロセッサにおいては、スーパースカラ型のプロセッサをプロセッサエレメントとして分岐の並列的な多重実行を行わせるものである。図1にその構成を示す。命令キャッシュ、データキャッシュ、レジスタファイル、起動時に使用するPC（プログラムカウンタ）は標準的なプロセッサ機構と同一である。複数の分岐を並列実行するための機構として、それぞれにPC キュー、命令キュー、デコーダ、レジスタ（分岐が解決するまで一時的にデータを保持する）を備えている。また、並列実行に際してのALU、ロード/ストア、分岐先のアドレス計算などの機能ユニットは分岐の並列実行に際し

てアプリケーションによって最適な個数が想定されるので可変構造としてある。図1で示したPC キューの役割は次の通りである。

- ・各グループのPC キューによって、あるパスに後続する分岐パスの並列実行が連続的に実行できる。すなわち、条件分岐による2つのパスの並列実行を集中制御せずに、各グループのPC キュー制御に任せることができる。

- ・標準的なスーパースカラ方式において機能ユニットを資源競合が生じないレベルまで準備できたとしても各リザベーションステーションに発行した命令は、投機実行がミスした際には再実行という大きなペナルティを払う必要がある。PC キュー方式では、分岐が解決した時点で確定した分岐パスを実行しているグループを選択することでこのペナルティを大部分回避できる。

- ・PC キューの個数に対する柔軟性（スケラビリティ）があることによってイーガークスキューション（可能性のあるパスを可能な限り並列に実行すること）のレベルを実装技術に合わせて選択できる。

- ・条件分岐解決時点での確定パスの実行結果の選択が容易である。

3. PCQ プロセッサの動作

PCQ プロセッサで分岐の並列多重実行を行う際の動作について述べる。図2に図1のPCQ プロセッサ

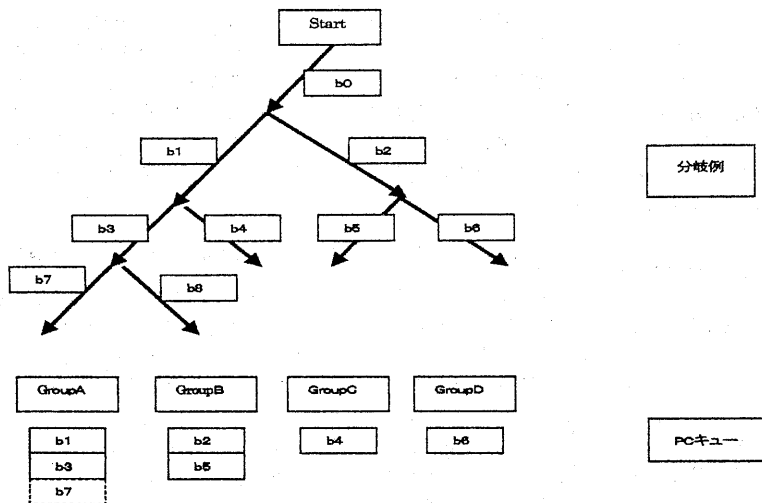


図 2 分岐パスと PC キューの状況

が分岐を並列実行する際の実行状況の例を示す。以下、図 1 と図 2 に基づいて並列実行機構について述べる。PCQ プロセッサの目的は、命令レベル並列化 (ILP) の内、実行対象となるアプリケーションのコードの基本ブロック (条件分岐から条件分岐の間のコード) を投機的に多重実行し、最終的に分岐が確定したブロックの結果を有するグループの実行結果を採用することによって制御依存による制限を除去するハードウェアモデルを提示することである。言い換えると単一方向の投機実行を拡張して複数方向の投機実行を実施することによって投機実行失敗時のペナルティを軽減するハードウェアモデルである。単一方向の投機実行に際して予測から外れた分岐に対しても並行して投機実行を実施し、投機実行失敗時のペナルティに備える。この条件分岐 (制御依存) が発生するごとに新たな投機実行を行うために複数のプログラムカウンタを備えた図 1 に示すハードウェア構成を使用する。

図 1 の PC キューは各グループ毎に複数存在する基本ブロックの先頭番地を指定する。複数の投機実行を行う際の基本ブロックの実行パスを保持するためのキューである。この PC キューによって命令の実行が制御される。以下にアプリケーションの実行順序に従ってその実行動作内容について述べる。図 1 で示す並列実行のための機構は可変構造であるがここでは 4 グループであるとして説明する。このグループ数の評価も本論文のシミュレーションの目的である。

3.1 動作の概要

まず、アプリケーション起動時に PC にセットされたシステム指定の既定値に基づいて命令キャッシュから命令キュー A に命令をフェッチする。3.2 で述べる制

御順序によって各グループの PC キューにセットされているエン트리値に基づいて各グループの命令キューに命令をフェッチする。すなわち、まず A グループによる分岐命令のデコードによって並列実行する分岐パスの開始番地を求め、適当なグループの PC キューにセットする。PC キューにセットされた分岐パスの後続の分岐の処理に関しては、当該 PC キューの属するグループの機構 (デコーダー等) によって後続する分岐命令が出現した時点で実行される。命令フェッチに当たっては各 PC キューは命令に空きがある場合には、命令キャッシュにラウンドロビン方式でアクセスする。切り替えの単位は分岐パスに含まれる命令数すなわち基本ブロック分の命令数とする。切り替えの方向は、命令キュー A, B, C, D の順とする。全ての命令フェッチが終了している状態であれば、条件分岐が解決し、新たに PC キューへの登録が可能になった PC キューを起点とする。

各グループ毎に命令キューから逐次的に命令を取り出し、デコードしディスパッチ (各グループの機能ユニットへの命令種類ごとの命令の発行) する。この部分は、標準的なスーパースカラ機構と等価な機能である。

3.2 分岐の多重実行制御

(1) イニシャル時の動作

PC キューへの基本ブロックのエントリは最初はグループ A のみであるので、そこから動作を開始する。以後、PC キューのエントリに基づいて各グループが並列動作する。図 1 のグループ A, B, C, D の各 PC キュー、命令キュー、一時レジスタ及びレジスタファイルはこの段階では空である。PC は、システム指定の

既定値をとる（イニシャルプログラムの先頭番地）。機能ユニットはアプリケーション毎にマルチプレクサーによりA, B, C, Dの各グループに再構成される。マルチタスク環境の基では、複数のタスクが時分割で動作するので、アプリケーション毎の再構成のオーバーヘッド（タイムスライス時間に対する再構成時間の割合）の評価は重要であるが、今回のシミュレーションでは実施していない。

(2) 最初の分岐の並列実行直前まで

図1のPCに基づいて命令キャッシュからある単位（キャッシュのライン単位）で命令をプリフェッチし、命令キューAに格納する。デコードの結果、条件分岐命令であることが判定されるまで上記を続行する。それまでの間、グループAのみが処理を行う。この時点で処理は標準的なスーパースカラ機構と等価である。

(3) 最初の条件分岐命令以降（b1-b2分岐命令以降）でb1-b2分岐命令解決直前までの処理

b1-b2分岐命令による分岐パスb1とb2の先頭番地をそれぞれPCキューA, Bにセットする。b1, b2の各パス上に次の条件分岐命令が出現するまでグループAとグループBは独立に処理を続行する。b3-b4分岐命令, b5-b6分岐命令の内, b3-b4分岐命令の出現が早ければ, b3の先頭番地をPCキューAにセットし, b4の先頭番地をPCキューCにセットする。その後, b5-b6分岐命令が出現したら, b5の先頭番地をPCキューBにセットし, b6の先頭番地をPCキューDにセットする。さらに, その後, b7-b8分岐命令が出現したら, その時点でPCキューへのセット処理をストールする。（本モデルでは, PCキューが4つとした場合であるのでb7をPCキューAにセットできるが, b8はセットすべきPCキューが無いので）、この時点で各グループがそれぞれのPCキューにセットされたパスを実行完了したら, b1-b2分岐命令解決まで各グループとも処理がストールする。b1-b2分岐命令においてPCキューAにb1, PCキューBにb2をセットしているがこの選択は任意でよい。以下, 各条件分岐命令とも同様の扱いである。b1-b2分岐命令が解決されるまで図1で示す各PCキューにセットされた分岐パスの状況は, 保持される。

(4) b1-b2分岐命令解決時点での処理

b1-b2分岐命令がb1パス確定として解決する場合, PCキューB, Dの内容がクリアされると同時に, PCキューの不足によりストールしていた前項で述べたb7-b8分岐命令の出現によるPCキューAへのb7の先頭番地のセットとPCキューB（b1-b2分岐命令によって開放されたPCキューB, Cの一つ）へのb8の先頭番地のセットが実施される。一時レジスタB, Dも同じタイミングでクリアされる。一時レジスタAのb1パス処理に該当する部分がレジスタファイルに書き戻される。

(5) b3-b4分岐命令解決時点での処理

b3-b4分岐命令がパスb4確定として解決する場合, PCキューA, Bの内容がクリアされる。一時レジスタA, Bも同じタイミングでクリアされる。一時レジスタCのb4パス処理に該当する部分（b3-b4分岐命令が解決した時点でかなりの処理が進んでいる）がレジスタファイルに書き戻される。

4. シミュレーション環境

PCQプロセッサにおいて, 分岐を並列実行するグループ数をどの程度準備すれば効果的か, また機能ユニットをアプリケーション毎に再構成して各グループに割り付ける際の各グループ毎の効果的な機能ユニット数についてシミュレーションを実施した。

(1) シミュレーションの条件

PCQプロセッサのシミュレーションに際して次の条件を設定している。分岐の多重実行に際しての図1で示すグループの必要数と機能ユニットの効率的な各グループ毎の個数の評価が目的であるので, 命令フェッチは指定された命令数分だけ理想的に行えるとする。すなわち, フェッチできる命令数を指定するのみで, 命令キャッシュのミスなどは存在しないものとする。各グループはこの条件のもとで命令フェッチをおこなう。各グループに割り付ける際の各グループ毎の効果的な機能ユニット数については各グループごとに幾つかの固定値を与え, その条件でシミュレーションし効率的な個数を実験的に求める。ベンチマークプログラムには標準的なベンチマークプログラムSPECINT95を使用した。また, シミュレータとしては既存のスーパースカラ型のマイクロアーキテクチャをPCQプロセッサモデルの各グループのコアとして使用でき, 内部の制御動作を詳細に検討できる実行駆動型のシミュレータを使用した。

(2) シミュレータについて

PCキュープロセッサをシミュレートするためにILPプロセッサを用いたメモリ共有型マルチプロセッサシステムのシミュレーションを目的に開発された実行駆動型シミュレータRSIM⁴⁾を使用した。RSIMで使用しているILPプロセッサのマイクロアーキテクチャはMIPSR10000⁵⁾をベースとしたアーキテクチャである。特に, R10000の現在実行中の命令を保持しているアクティブリスト（命令ウィンドウあるいはリオーダーバッファともいう）、論理レジスタから物理レジスタへのマッピング情報を保持しているレジスタマッピングテーブル（リネーミングレジスタともいう）、分岐予想に基づく投機実行に際して投機実行失敗時の回復を行う際のレジスタマッピングテーブル情報を保持しているシャドウマッピングなどの機構が実行駆動型のシミュレータとして実装されており, 本論文のPCQプロセッサをシミュレーションするのに適している。RSIMとそ

表 1 投機実行分岐命令数

units	base-length	branches-until-resolved
4222(go)	4.72	3.76
8422(go)	4.95	3.48
4222(compress)	10.35	3.15
8422(compress)	7.19	3.27

のベースとなっている MIPS 10000 の代表的なマイクロアーキテクチャについて述べておく。

MIPSR10000 は、1 サイクル当たり 4 命令をフェッチデコードする。4 つまでの分岐を投機実行できる。ダイナミックなアウトオブオーダー実行が行える。マッピングでレジスタリネーミングを行う。正確な例外処理を実現するため終了（コミット）操作はインオーダーに実施する。5 つの独立の実行ユニット；ノンブロッキングロード/ストアユニット、整数型 ALU、浮動小数点ユニット、加算ユニット、乗算ユニット、を備えている。それぞれ 32 K バイトの一次命令キャッシュとデータキャッシュをもつ。

RSIM で実装している ILP プロセッサは MIPSR10000 とほぼ等価なアーキテクチャであるが異なる部分を示す。RSIM では命令キャッシュは無く、フェッチできる命令数を任意に指定できる。従って命令デコードと命令フェッチが 1 ステージで実行する形でパイプラインを実装している。すなわち、MIPS10000 では、命令フェッチ、デコード、発行、実行、完了、終了の 6 段のパイプラインであるが、RSIM の実装では命令フェッチとデコードを 1 ステージとした 5 段のパイプラインとなっている。命令フェッチ、デコードおよび終了がインオーダーで発行、実行、完了がアウトオブオーダーである。

(3) PCQ プロセッサモデルの実装

現在、RSIM シミュレータをベースとして、PCQ プロセッサモデルを実プロセッサに近い形のシミュレータとして実装中なので概要を次に示しておく。

- ・PC キューのグループは out-of-order 実行エンジンで n グループ実装する形式で実現する。

- ・命令キャッシュからの命令の読み込みは各グループ毎の実行エンジンが各 PC キューに基づいて並行して行う。

- ・各 PC キューへの登録と削除は、デコードした命令を分岐予想ユニットが条件分岐と判断した時点で当該 PC キューに登録し、未解決の分岐命令が解決した時点で各 PC キューのエントリの削除（フラッシュするグループを決定）を行う。

- ・各グループで処理された命令の実行結果は完了（complete）した時点で各グループの一時レジスタ（RSIM の active list を利用して実装）に格納する。

- ・未解決の分岐命令が解決したら上記に示す各グループのアクティブリストの該当するものを終了（graduate）対象アクティブリストとして採用する。

表 2 投機実行ミスペナルティ

units	penalty-cycle	penalty-inst
4222(go)	6.74	17.73
8422(go)	4.25	17.24
4222(compress)	11.28	32.56
8422(compress)	7.19	32.65

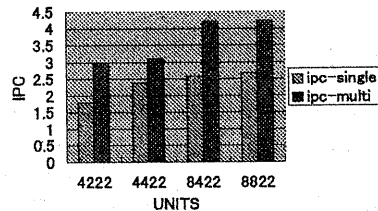


図 3 go の IPC 値

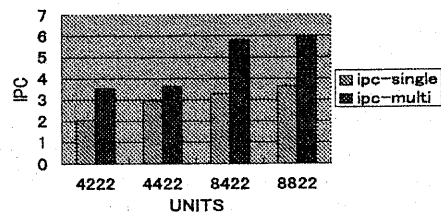


図 4 compress の IPC 値

5. 性能評価

PC キュープロセッサにおいて分岐を多重実行する際にまず、対象とするアプリケーションがどの程度の分岐を持ち、分岐を並列実行した際にどの程度の性能向上が図れるかシミュレーションによって求めた。次に PCQ プロセッサのグループ数の値をシミュレーション結果から予想する。その際、各アプリケーション毎に図 1 で示した分岐を並列実行するためのグループがどの程度あればよいかを評価する。以下に、使用したアプリケーションと評価条件、評価結果について示す。

(1) アプリケーションシミュレーション

使用したアプリケーションを次に述べる。SPECINT95 はプロセッサ自体の性能評価に標準的に用いられるベンチマークプログラムである。SPECINT95 から分岐命令が多く、かつ分岐予想の難しい go と compress の 2 つのプログラムを評価に使用した。PCQ プロセッサで分岐を多重実行する際に最もその効果を評価するのに適しており、かつ標準的に使用されているベンチマークプログラムである。これらのベンチマークプロ

グラムは,SPARC コンパイラでコンパイルした。評価においてコンパイラによる性能向上が最大の実行コードを得るために、コンパイラオプションは SPARC コンパイラの最適化オプション -x04 を指定してコンパイルしている。

また、評価には go については、実行コードの命令数で約 13,000,000 命令、compress については、約 60,000,000 命令をシミュレータで実行した。

(2) 評価結果

表 1 にユニット数 (units) を変化させたときの go プログラムと compress プログラムの分岐命令から分岐命令までのベースブロック長 (base-length) と分岐命令が解決するまでに新たに出現する分岐命令数 (branches-until-resolved) の平均値を示す。同様に表 2 に分岐命令ミス時の損失サイクル数 (penalty-cycle) と損失命令数 (penalty-inst) の平均値を示す。表 1 と表 2 のユニット数は、第 1 番目がデコーダ数、2 番目が ALU 数、3 番目が FPU 数、4 番目がアドレスジェネレータ数である。ユニット数が 4222 で示される構成は、MIPSR10000 と等価な構成である。

図 3、図 4 にベンチマークプログラム go と compress においてユニット数を変化させた時の性能を IPC 値で示す。ipc-single が標準的なスーパースカラプロセッサの単一方向投機実行の場合であり、ipc-multi が今回提案する複数方向を投機実行する PCQ プロセッサモデルのグループ数が eager-execution するのに充分あるとした場合 (理想値) の値である。PCQ プロセッサモデルでは、各グループのユニット構成を単一方向投機実行の場合と同一構成として評価する。

図 3 から命令フェッチ数が同じであれば、ALU 数に関係なく ipc-multi の値はほぼ同じであることがわかる。図 4 においても同様のことが言える。したがって、単一方向投機実行において最大性能に近く、かつ ALU 数が 4 であるユニット数が 8422 の場合をベースとして PCQ モデルの評価を実施した。この場合の理想的な PCQ モデルにおける性能向上の割合は、ベンチマーク go で 63 % 増、ベンチマーク compress で 77 % 増である。

次に、理想的な PCQ プロセッサモデルに対して、現実的な PCQ プロセッサモデルの構成として図 2 で示すグループ数が 4 の場合についての性能向上の予測値を評価する。表 1 よりユニット数が 8422 の場合、分岐が解決するまでに新たに出現する命令数は、go の場合が 3.48、compress の場合が 3.27 である。この値を図 2 に当てはめると、ほぼ分岐パス b7 の実行が終了する時点で、b1-b2 分岐命令が解決されることになる。b1-b2 分岐命令で b1 方向のパスが選択される場合は、単一方向投機実行の場合と等価であるので、この場合を基準に b2 方向パスが実行される場合について評価すればよい。

b1-b2 分岐命令が解決された時点で、b1 パス方向では

3 段の投機実行を実施しているが、b2 パス方向では 2 段の投機実行までしか行なえない。したがって、b1-b2 の分岐命令解決時点で b2 パスが選択された場合、b1 パスが選択された場合に比べて 3 分の 2 の投機実行の効果となる。したがって、前述の理想的な性能向上値の 3 分の 2 の値、すなわち、go が 42 %、compress が 51 % の性能向上となる。

6. おわりに

PC キュープロセッサの構成とシミュレータによる評価結果、及び評価データに基づくモデルの性能向上の予測値について述べた。PCQ キューモデルの分岐の並列実行機構であれば、並列実行を行うグループ数が 4 グループの実装で単一方向投機実行に対して、ベンチマークプログラム go で 42 %、compress で 51 % の性能向上が期待できる。

今後、MIPSR10000 とほぼ等価なシミュレータである実行駆動型の RSIM シミュレータをベースとして実プロセッサに近い形のシミュレーションを実施し性能を確認していきたい。また、実行するアプリケーション毎に機能ユニットを再構成して機能ユニットの効率的な利用を行える構成をモデル化し、シミュレーションで性能を確認することも今後の課題である。

参 考 文 献

- 1) M.S.Lam, R.P.Wilson.: "Limits of Control Flow on Parallelism", Proc.19th Annual Int'l Symp. Computer Architecture, ACM Press New York, pp.46-57, (1992).
- 2) A.UHT, V.Sindagi, "Disjoint Eager Execution: An Optimal Form of Speculative Execution", Proc.28th Int'l Symp. Microarchitecture, IEEE CS Press, pp.313-325, (1995).
- 3) 大島, 都志見, 有田: "80286 をプロセッサエレメントとするメモリ共有型並列処理システムの開発 (HYPHEN B-16 のアーキテクチャ)", マイクロコンピュータ 47-1, (1987).
- 4) Vijay S. Pai, Parthasasathy Ranganathan and Sarita V. Adve, RSIM: "An execution-Driven Simulator for ILP-Based Shared-Memory Multiprocessors and Uniprocessors", IEEE TCCA Newsletter, Oct. (1997).
- 5) Kenneth C. Yeager, "The Mips R10000 Superscalar Microprocessor", IEEE Micro, pp.28-40, April (1996).
- 6) 芝, 有田: "実行ユニットを再構成し分岐を並列実行することによる性能向上に関する検討", 第 51 回電気関係学会九州支部連合大会講演論文集, p.17, (1998).