C-11

分岐履歴を用いた条件付き双方向パスの考察

Consideration of Selective Dual Path Using Branch History

原 淳†　　宮内 新†　　荒井 秀一†
Hara Atsushi　Arata Miyauchi　Syuuichi Arai

1．　はじめに

プロセッサの性能を高めるためには,命令の流れを妨げる要因をできるだけ排除する必要がある.妨げる要因の一つに分岐命令がある.分岐命令移行アドレスを命令のフェッチ時点で予測し,その予測アドレスを移行アドレスと考え,実行することを分岐予測と呼ぶ.

双方向パスとは分岐予測の失敗におけるリスクを減らす手法である.分岐命令の Taken パスと Not-Taken パス両方を実行するが,莫大なハードウエアコストを支払わなければならない[1].

分岐予測に関して,さらなる予測精度の上昇は困難であることから,双方向パスにおける研究がなされている.片山ら[2]は将来実現可能と考えられるプロセッサに対し双方向パスを適用することで性能が向上するのを確認した.双方向パスの研究では十分な命令供給量を用意するためか,8 命令の発行を許可している.本研究ではプログラム中の並列度が最大 3~4 命令であることを考え,4 命令発行の双方向パスに関して考察を行う.

また,ハードウエアコストをできるだけ払わない機構を提案し,その評価を行う.

3．双方向パスの手法

3.1 双方向パスの問題点

双方向パスを単純に実現するには,設計するプロセッサの何倍も必要になる.それは分岐命令をデコードしてから,結果がでる間のサイクルに現れる分岐命令に対し双方向実行するために O(2^n)のハードウエアを費やす必要がある.

3.2　分岐履歴を用いた分岐予測

分岐履歴を用いた分岐予測では,分岐するであろう方向の偏りを利用することで予測方向を決定する.2bit 履歴分岐予測では taken と not taken をそれぞれ 2bit で表すことで移行する頻度を示し,その情報をもって予測をたてる方式である.

履歴を用いることで予測を立てることは,「前回分岐した方向に分岐命令は今回も同じ方向に分岐する」といった考えに基づく.それをもっともよく表すのはループである.ループは繰り返しで用いられるために,ループを続ける事を taken,つづけない事を not taken とするならば,not taken になるのはループ中 1 つだけであり,それ以外は taken であり続ける.しかもその taken は連続して現れる.

逆に履歴を用いても入力で変わるような物は,予測をたてづらくなる.入力データによって条件が変化する物は,プログラムを実行するたびに結果が違うことを示す

† 武蔵工業大学大学院電気工学専攻
Graduate School of Electrical Engineering,
Musashi Institute of Technology

このように履歴を用いた場合,予測を高確率で的中できる分岐とできない分岐に分類できる.

3.3　分岐フィルタ

双方向パスの問題点である無駄なパスの削減を行うには.常に分岐命令に対しての適用を避けなければならない.これを防ぐ手段として単純なフィルタを設ける.

例えば 2bit 履歴分岐予測での bit を”00”,”01” を taken,”10,”11”を not taken とする.このことを図 1 に示す.

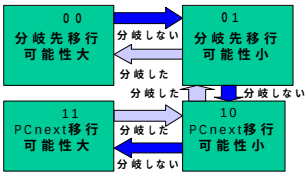

図 1:2bit 履歴分岐予測の状態遷移図

“00”,”11”にいる bit は予測を高確率で的中できる分岐命令と仮定し,”01”,”10”にいる bit は予測を的中しづらい分岐命令と仮定する.当然,的中しづらい分岐命令の方が予測を失敗するリスクを的中できる分岐命令より背負っている.そこで的中しづらい分岐命令だけに双方向パスを適用する.

3.4　必要となるハードウエア

双方向パスを適用するためには,十分な資源を必要とする.しかしスーパースケーラプロセッサ自体がすでに多数のハードウエアを必要としている為に,ハードウエアの過重な増加をさけねばならない.次に双方向パスに必要とされる構成要素を示す

- 命令キャッシュのキャッシュバンド幅
  双方向パスは分岐の別れ方によっては多くのアドレスを持つ必要がでてくる.1 クロック内にアクセスできる数を制限するか,ポートの多重化が必要となる

ハードウエアスケジューリングを行うには,命令キャッシュから読み出した命令をどこかに保存しておく必要があるために,いくら命令フェッチャが多くの命令を取得しても早期に保存しておく場所が詰まるだけとなる(その分,命令の選択性は増える).

- リオーダバッファ(リネームレジスタ)
  分岐予測は,予測の失敗に備える必要がある為に必ず回復を行う機構が必要になる.リオーダバッファは命令順序を維持し回復を行える機構[1]であるために標準で備えている必要がある.この機構を多重化する事で双方向パスでの実行結果の保存を行う必要がある.

リオーダバッファは FIFO のリングバッファを取る為に単純なエントリの増加ではなく,ユニットの多重化が必要と

なる.単純なエントリの増加では,書き込みと読み出し地点の判定が複雑になる為である

- 実行ユニット

実行ユニットは ALU,アドレス演算機などオペランドの計算を行うユニットである.

実行ユニットは双方向パスを用いることで使用頻度が増加する.それは双方向パスを行うたびに命令が分割され命令並列度が一時的に上昇する為である.もし実行ユニットを現在の量では足りないのであれば,多く持つ必要がある.

２． 実験・結果

3.1 構成

スーパースケーラプロセッサを以下のように想定した.基本的な物として,４命令フェッチャ,2biti 履歴分岐予測機構,16 エントリの命令ウインドウ,16 エントリのリオーダバッファ,4 エントリのストアバッファを所持している

実行ユニットとして論理演算機*2,バレルシフタ,ロード/ストア用アドレス演算機,分岐用アドレス演算機を所持する.

この構成は現在のマイクロプロセッサよりも機能を多く持っていないこれは.少ない所持で双方向パスの効果を確認する為である.

3.2 双方向パスの手法

用いた双方向パスの手法は

- 双方向パス実行許可を１分岐のみ
- 2bit 履歴分岐予測を用いた分岐フィルタ

また,向いていた分岐履歴の予測 bit の方向を正方向とし,反対側を負方向とすると,正方向側を最優先に実行し分岐予測を行う方式採用した.負方向側は分岐命令をデコードする直前の命令か,保存するリオーダバッファの量をオーバーした時に限りデコードを終了し,正方向側での命令フェッチ量を回復させる(通常,双方向パスを行った時点で命令フェッチ量は半分となる).

3.3 ハードウエアの増加割合

命令キャッシュのデュアルポート化,リオーダバッファ,ストアバッファ,命令ウインドウのエントリ増加を行った.

3.4 命令レベルでの性能評価

設計したプロセッサに対し簡単なアルゴリズムを用いたプログラムを走らせることでベンチマークを行った.表 1 は各種プログラムを走らせた際の,CLK 数と性能上昇率である.

表 1:ベンチマーク結果

	基礎(CLK)	提案手法(CLK)	性能向上率(%)
2分探索10個	582	521	10.4
線形ソート10個	657	626	4.7
バブルソート10個	494	510	-3.2

また,提案手法においての分岐予測的中率,負方向に移行した確率(負方向移行率),双方向パスを実行した分岐命令の予測を失敗し,提案手法が得た利益(双方向パス解放数)を表 2 に示す.

表 2:予測的中率,負方向移行率,提案手法の利益

	分岐予測的中率(%)	負方向移行率(%)	双方向パス解放数(個)
二分探索	57.0	59.0	1.0
線形ソート	82.4	35.0	3.8
バブルソート	86.0	24.0	1.8

線形ソートでは性能が向上し,バブルソートで性能が減少したことから,ある一定の分岐予測精度(80％～)を持つ場合,分割元の命令後に 3~4 個以上の命令をデコードし,完了しておく必要があると考えられる.

また,分岐精度が低い(50%~60%)のなら,双方向パスの実行は分割元の命令後に多くデコードし完了しなくとも性能を上昇させることができる.

本研究で設計したプロセッサは分岐命令の処理方法としてステータスレジスタ方式を用いていた.そのために負方向側はデコードを止める分岐命令の前に存在するであろうコンペア命令でデコードを止めざるを得なかった.

そこで,分岐命令の対処の仕方に汎用レジスタ方式を用い,分岐命令直前で止めるのではなく,分岐命令を実行し,移行アドレスを格納することで提案手法の利益をさらに増すことができる.

４．おわりに

分岐履歴 bit を用い,命令のバンド幅やデコード量を増加させることなしに行える,分岐命令における条件付きでの双方向パスの実行に関しての考察を行った.

双方向パスの実行は分岐予測を行う際の失敗におけるリスクを減少させることができる.しかし予測の負方向側の命令が多く実行できない場合には命令フェッチ能力の減少が,双方向パスの利益を上回ってしまい性能減少を招く問題がある.この問題は本研究では実行できなかった命令を実行することで解決すると考えられる.

また,間接分岐の問題が存在する.双方向パスがどちらも答えを持ち得ないときに問題が発生すると考えられる.

ハードウエア量に関して,基本となるリオーダバッファエントリは[1]で最適とされる 16 エントリに 8 エントリを双方向パス用に追加した.しかしこの量が最適とは必ずしもいえない.命令ウインドウやストアバッファに関しても同じ事がいえる.命令レベルからの考察より最低 4 命令必要とされる.その最大値を求める必要がある.

参考文献

[1] Mike Johnson
「スーパースカラプロセッサ」
日経 BP 出版センター,1994

[2] 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」