多数演算器方式における演算器利用率の向上手法

吉瀬謙二, 田中英彦
東京大学大学院工学系研究科

1　はじめに

マイクロプロセッサの性能に対する要求は留まるところを知らない。しかしながら、パイプライン処理やスーパースカラ方式による性能向上に限界が見え始めており、新しい実行方式に関する研究が盛んにおこなわれている [?]。一方ユーザの立場からは、実行コードの互換性に対する強い要望があり、スーパースカラ方式とのコード互換性を維持しながら数十倍の性能向上を目指す新しい実行方式を議論することが今求められている。

我々が注目する多数演算器方式は、非常に多数の演算器をチップ上に集積し、それらの接続形態を動的に変更することでプログラムの最適な実行を目指す実行方式である。本報告では、多数演算器方式のフェッチ手法として確率を用いた枝刈りを採用した場合を考え、演算器利用率の向上を目指す一手法について検討する。

2　多数演算器方式

スーパースカラ •プロセッサの実行コードには、レジスタ使用の競合により発生するデータ依存関係やレジスタ数の不足により必要となるロード/ストア命令といった、ハードウェアの資源競合が原因で加えられた処理が多数存在する。また、レジスタ •ファイルを介したデータの移動という枠組が持つ無駄が存在する。多数演算器方式では、処理装置 (Execution Unit) と呼ばれる動的に接続形態を変える多数の演算器を用いて計算の本質を表すデータフローグラフの最適な実行を目指す。

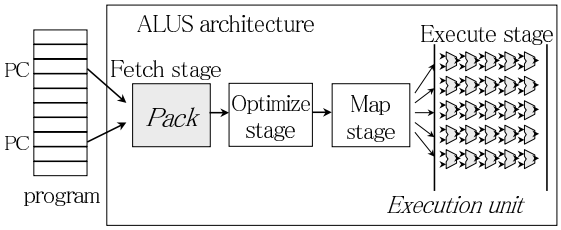

図 1: 多数演算器方式の実行ステージ

多数演算器方式の実行ステージの構成を図??に示す。フェッチ •ステージでは、単一制御流のフェッチではなく、分岐成立と不成立の両方の制御流における命令を投機的にフェッチする。フェッチ •ステージにおけるパラメータとして、フェッチ段数と呼ぶ値を定義する。これは、跨ぐことのできる分岐命令の数に 1 を足した値であり、フェッチ段数 n とは、それぞれの制御流に存在する基本ブロックの上限が n 個であることを意味する。フェッチ •ステージでフェッチした命令の集合をパックと呼ぶことにする。

最適化ステージではレジスタ •リネーミングにより不要なデータ依存関係を解消し、パック内の命令が持つ真のデータ依存関係を解析する。マッピング •ステージでは処理装置の構成と実行するデータフローグラフの対応をとり、実行ステージではパック内の命令を実行する。

本報告では、単一のプログラム •カウンタからの命令フェッチのみを検討の対象とするため、今回の評価における多数演算器方式の実行イメージはパックを単位としたパイプライン処理と捉えることができる。

3　Pruning Eager Execution フェッチモデル

フェッチ •ステージでは確率を用いて制御流の枝刈りをおこなう Pruning Eager Execution [?] を用いる。PEE ではそれぞれの制御流に到達する確率を計算し、その確率が閾値に満たない部分をフェッチしないことで、フェッチ命令数の増加を防ぐ。図??に Pruning Eager Execution の様子を示す。矢印は制御流を表しており、図??左には分岐成立 80%として各制御流に到達する確率を付けた。確率 p を用いた枝狩りを p-PEE と表すことにする。右図は確率 20%より小さい制御流の命令をフェッチしない場合で 20%-PEE に対応する。

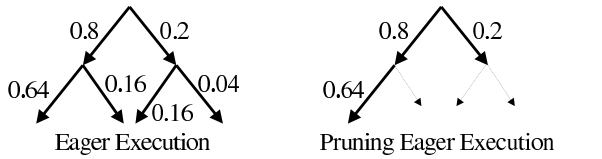

図 2: Eager Execution and Pruning Eager Execution

PEE では、それぞれの制御流に到達する確率を計算する必要がある。CALL, JMPL, Ticc 命令では必ず分岐が成立するので分岐成立の確率を 100%とすればよい。Bicc, FBfcc 命令では、分岐履歴を保存するテーブルにプログラム •カウンタをタグとして過去の分岐成立の数と分岐実行数を保存する。確率は履歴テーブルの情報を元に計算する。テーブルの内容は分岐命令が実行された時に変更する。

Improving the ratio of ALU usage on ALUS architecture
Kenji KISE, Hidehiko TANAKA
Graduate school of Engineering, The University of Tokyo

レジスタ間接アドレッシングで分岐先を決めるJMPL命令の飛び先アドレス予測方法として、命令のPCをタグとして分岐先アドレスを保存するテーブルを用い、前回と同じ分岐先を予測アドレスとする方法を用いる。

4　演算器利用率の向上手法

演算器の利用率を式(??)で定義する。演算器の利用率は、単一制御流から命令フェッチをおこなった時に100%となり、投機的に多数の制御流から命令をフェッチするに従い低下していく。

$$\text{演算器の利用率} = \frac{\text{有効命令の数}}{\text{処理装置にマップした命令数}} \quad (1)$$

本稿では、フェッチした命令流の合流に注目した演算器利用率の向上手法を考える。2つの制御流で同一の基本ブロックをフェッチしている場合には、それらの基本ブロックを共有できると考える。図??の例では、灰色で示した基本ブロックを同じ基本ブロックとして共有している。　ただし、図??に示す様に、基本ブロックを共有

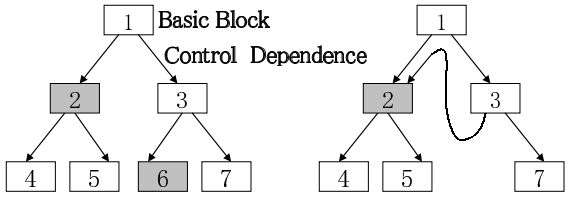

図 3: 演算器利用率の向上手法

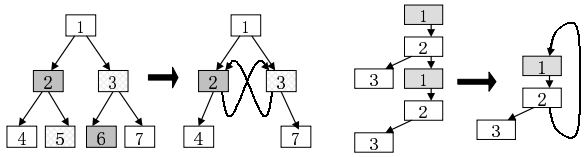

図 4: 基本ブロックを共有できない例

することによって制御依存グラフにループを作る場合にはグラフの変換をおこなわない。このような制御依存は処理装置における実行を複雑にするためである。多くの基本ブロックを共有することで処理装置にマップする命令数を削減し、演算器の利用率の向上が期待できる。一方、この共有により命令をスケジューリングする範囲が制限されるため、得られる並列性の低下が問題となる。

5　評価結果と考察

演算器利用率の向上手法を評価する。評価はSPEC92で用いられるcc1, compress, espresso, scに合成ベンチマークのdhrystoneを加えた5種類を対象に、各プログラム1千万命令のトレース・データを用いておこなう。アーキテクチャはSPARC architecture version 8である。

??章で示した手法を用いない場合の演算器利用率を表??に示す。フェッチ段数の増加と、閾値の低下により演算器の利用率は大きく低下する。閾値1%, フェッチ段数20で利用率が10%となり、閾値0.1%, フェッチ段数50で利用率が1%を下回ることがわかる。

表 1: 演算器の利用率 (%)

フェッチ段数	10	20	30	40	50
4%-PEE	32.2	20.8	16.5	14.6	13.7
1%-PEE	23.2	10.0	6.30	5.10	4.58
0.1%-PEE	17.6	4.20	1.80	1.12	.854

演算器利用率の向上手法を用いた場合の利用率を表??に示す。2つの表を比較することで、提案した手法により利用率は1.8〜4.7倍に向上していることがわかる。

表 2: 最適化を用いた場合の演算器利用率 (%)

フェッチ段数	10	20	30	40	50
4%-PEE	59.0	42.2	35.6	33.0	30.8
1%-PEE	45.3	23.9	17.8	15.3	14.1
0.1%-PEE	36.7	13.6	6.61	4.77	4.03

表??と表??の比較から、提案した手法により演算器利用率が改善できることがわかった。しかし、本手法によるパック内並列性に関する評価は現在調査中である。本手法を用いなかった場合の評価は、1%-PEE、フェッチ段数30という設定において平均7.2程度の命令レベル並列性を利用できることを文献[?]で確認している。提案した手法により、この並列性がどのように変化するか調査し、並列性の低下を抑えながら演算器の利用率の向上を目指す方法を検討する必要がある。

6　まとめ

多数の演算器の接続形態を動的に変更し、プログラムの最適な実行を目指す多数演算器方式の演算器利用率を検討した。制御の合流に注目した演算器利用率の向上手法を用いることで演算器利用率が1.8〜4.7倍に改善されることを示した。提案した手法は、利用率の改善を達成する一方で命令スケジューリングの範囲を狭めてしまうために利用できる並列性の低下を招く。今後、提案した手法を用いた場合の並列性と演算器利用率の関係を調査していく。

本研究の一部は文部省科学研究費 (一般研究 (B) 課題番号07458052　大規模データパスプロセッサの研究」) による。

参考文献

[1] 吉瀬謙二, 中村友洋, 辻秀典, 安島雄一郎, 田中英彦. 多数演算器方式における演算器利用率の検討. 情報処理学会研究報告 97-ARC-125-21, August 1997.

[2] 中村友洋, 吉瀬謙二, 辻秀典, 安島雄一郎, 田中英彦. 大規模データパスプロセッサの構想. 情報処理学会研究報告 97-ARC-124, pp. 13–18, June 1997.