オンチップハードウエアによるループ並列化機構

5N－5

玉造潤史　松本尚　平木敬

東京大学大学院 理学系研究科 情報科学専攻

1　はじめに

ハードウエア集積技術の発展により得られる大きなハードウエア量を有効に活用する様々なアーキテクチャが提案されている。なかでも命令レベルの並列性とスレッドレベルの投機並列性を組み合わせオンチップMIMDによって実現するアーキテクチャ[6][7]が注目されている。現在の命令レベルの並列性抽出の問題点はハードウエアの構造は複雑になってしまい大きな資源を用いた設計が困難であることである。一方、MIMDを用いた場合、プロセッサアーキテクチャの重要な性質であるバイナリコンパチビリティを持たせることができない。これらの問題を解決するものとして、我々はオンチップ MIMD 上で逐次バイナリプログラムを並列化してバイナリコンパチビリティを実現するアーキテクチャである実行時再構成法[3][4]を提案した。本稿では、この実行時再構成方式による並列化ハードウエアの実現法について述べる。

2　実行時再構成方式

実行時再構成方式では逐次プログラムにおけるコントロールフローとデータフローを解析しそれぞれをリネーミングすることで並列化を実現している。リネーミングだけで並列化を行うため、現在用いられている逐次プログラムからさらなる性能を得ることができ、MIMD の要素プロセッサは現在用いている命令レベル並列性を活用したものを用いることができる。実行時再構成方式の投機並列実行には次のような特徴がある。

- 逐次形式のプログラムからループを抽出し、並列化のためのレジスタ依存をバイナリから解析する。(Loop Analyzer)
- レジスタ依存を解決するヘッダを生成し、ヘッダを用いて投機実行する。管理はバリア機構[5]を用いて行う。(Speculative Execution Manager)
- メモリアクセスは投機的に行い、ハードウエアが RAW ハザードを検出する。(Speculative Memory Buffer)

これらの機能を備えたオンチップ MIMD テストベットが OCHA-PRO[1][2](図1) である。OCHA-PRO は命令レベルのスーパースカラプロセッサを要素プロセッサとして持ち上記の機能を各要素プロセッサに分散して付加するアーキテクチャとなっている。

3　ループ解析手法

実行時再構成方式は逐次プログラム中からループ構造を検出し、少ないハードウエア資源だけで並列化する手法である。一般に並列化で解析を行うのは、ループ内のメモリデータ依存とレジスタデータである。このうち前者は前述のようにハードウエアの援助により投機実行を行い解析を行わず、実行時にアーキテクチャで依存を検出する。解析を必要とするのは、プロセッサが実行時に用いるレジスタ上のデータを投機実行によって正しく生成する条件 (ヘッダ) を抽出することである。

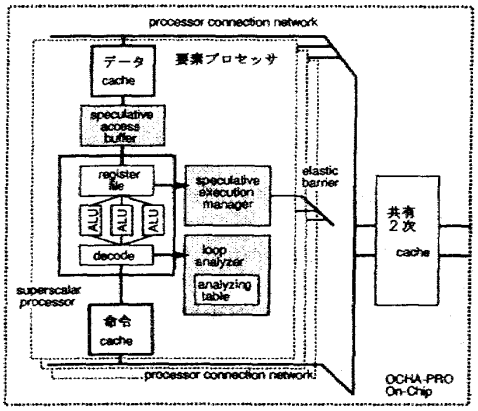

Figure 1: OCHA-PRO のアーキテクチャ

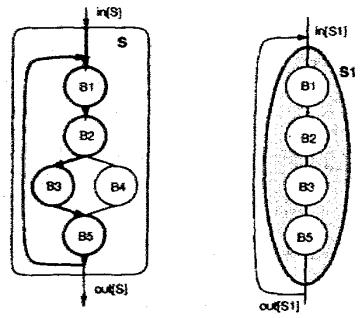

Figure 2: 投機実行ループのデータフロー

図2 は、制御構造を持ったループの例である。節点は、基本ブロックを意味している。このループを分岐予測バッファを用いて一つの経路だけを抽出し、一つのブロックとする。

さらに、一般にループのデータフロー方程式は、

$$out[S] = gen[S] \cup (in[S] - kill[S])$$

$$out[S_1] = gen[S_1], in[S_1] = gen[S_1]$$

であるが、実行時再構成法ではループを一回実行してから並列化を行うことで $in[S] - kill[S]$ の解析を不要とする。したがって、図3 のように実行を行う。すると、ループのヘッダ $gen[S]$ は次式を満せばよい。

$$out[S_1] = in[S_1]$$

ループの実行時再構成では $gen[S]$ を命令のレジスタフィールドでこの条件を満すレジスタを検出し、得ら

Loop parallerizing mechanism on On-Chip MIMD Hardware

Junji TAMATSUKURI , Takashi MATSUMOTO , Kei HIRAKI

Department of Information Science,Faculty of Science, the Univers ity of Tokyo

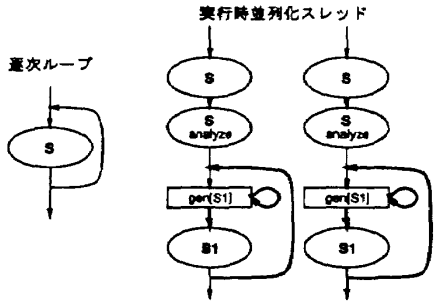

Figure 3: 投機実行スレッドの実行モデル

れた命令を繰り返すヘッダを作成することで並列化スレッドを実現している。

このヘッダが静的な並列化とのオーバーヘッドとなるが、命令レベルの並列性を用いた要素プロセッサと、逐次のコンパイラでも誘導変数の最適化によって $gen[S]$ が小さくなる最適化が行われるためヘッダのサイズは小さくなることから十分に効率的な並列化ができる。

4 再構成バッファを用いた実現

再構成するヘッダは命令流から抽出されるが、ループボディーに対してヘッダのサイズは小さいため通常の命令フェッチを行い、発行時にヘッダ実行中には不要な命令を破棄する方法を用いると命令フェッチ効率が低下してしまう。 そのため、抽出した命令の再構成の

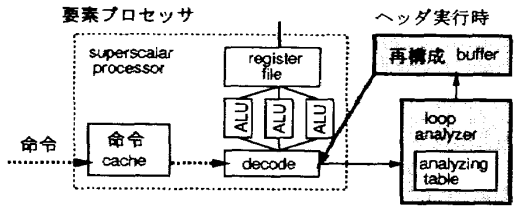

Figure 4: 再構成バッファ

ために命令キャッシュの他に命令再構成バッファ 4を装着する。並列実行中のヘッダは全て再構成バッファに格納されそこから命令フェッチにフィードされる。問題は再構成バッファのサイズが大きくなると並列実行出来ないものが出来てしまうことである。

Application	tomcatv	compress	mpeg
最大必要バッファサイズ(ワード)	13	4	13
命令フェッチ効率(%)	10.0	19.3	22.3

Table 1: 再構成バッファのエントリ数

再構成バッファを用いた場合と用いない場合についてベンチマークを行った。まず、再構成バッファの必要サイズの測定(表 1)を行った。計測を行ったのはSPEC ベンチマーク最大のループサイズのループを持つ tomcatv と整数アプリケーションの compress,mpeg である。この結果から再構成バッファのサイズは小さくしかも並列実行時のフェッチの効率が大きく低下してしまうことが分かる。その再構成バッファを使った

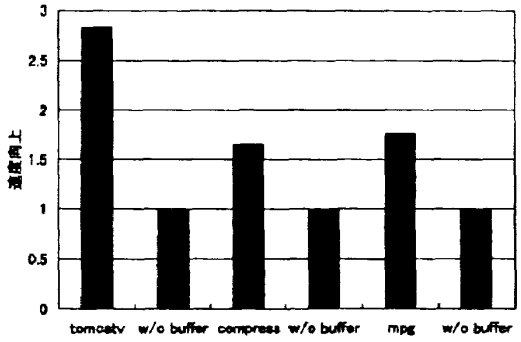

Figure 5: 再構成バッファの効果

場合と使わない場合の性能改善はグラフ (図 5) の結果となる。(要素プロセッサ 4 台の場合)

5 まとめ

実行時再構成法では、ループの並列化をレジスタ解析を行いループイテレーション間依存を解消するヘッダを逐次プログラムを構成する。実行時の再構成による命令フェッチの性能低下を少数のエントリを持つ再構成バッファを用いて実現できることを示した。

References

[1] 玉造 潤史 , 松本 尚 , 平木 敬., "オンチップ MIMD プロセッサにおける実行時並列化機構の性能評価" , 情報処理学会研究会報告 ARC 126-13,pp.73–78,1997

[2] 玉造 潤史 , 松本 尚 , 平木 敬., "On Chip MIMD における大規模投機実行機構" , 情報処理学会研究会報告 ARC 125-24,pp.139–144,1997

[3] Junji Tamatsukuri , Takashi Matsumoto , Kei Hiraki., "On-Chip Parallel Architecture for Runtime Loop Restructuring" ,Technical Report TR-04,University of Tokyo,1997

[4] 平木 敬 , 松本 尚., "プロセッサによる実行時ループ再構成方式" , In proceedings of the HPCS'97.

[5] 松本 尚., "Elastic Barrier: 一般化されたバリア型同期機構", 情処学会論文誌 Vol32,No.7,pp.886-896,July 1991

[6] J.Oplinger , D.Heine , S.W.Liao , B.A.Nayfeh , M.S.Lam , K.Olukotun., "Software and Hardware for Exploiting Speculative Parallelism with a Multiprocessor" ,CSL-TR-97-715,Stanford University,1997

[7] G.Sohi , S.Breach , T.Vijaykumar., "Multiscalar Processors",In proceedings of the 22nd Annual International Symposium o n Computer Architecture,1995