

# 大規模データパスアーキテクチャの コード最適化に関する研究

岩崎 慎介<sup>†</sup> 服部 直也<sup>††</sup> 飯塚 大介<sup>†††</sup> 坂井 修一<sup>††</sup> 田中英彦<sup>††</sup>  
<sup>†</sup> 東京大学工学部 <sup>††</sup> 東京大学情報理工学系研究科 <sup>†††</sup> 東京大学工学系研究科

## 1 始めに

### 1.1 研究の背景

半導体技術の進歩によりトランジスタの集積度は年々増え続けている。この傾向は今後も続くと予想され、次世代マイクロプロセッサでは現在の数倍のトランジスタが利用可能となる。しかし現在マイクロプロセッサの主流であるスーパースカラは、その構造的複雑性より資源の投入による効果的な性能向上は期待できない。これを受けて、新しいアーキテクチャとして、大規模データパスアーキテクチャ(VLDP: Very Large Data Path)アーキテクチャが提案されている [1]。

### 1.2 研究の目的

VLDP では従来にない処理単位として命令ブロックという複数命令の集まりを導入している [2]。命令ブロックはコンパイラにより生成される。この生成手法を工夫することで VLDP の性能向上が期待できる。よって本研究ではコンパイラによる命令ブロックの最適な生成手法を比較、検討することを目的としている。

## 2 大規模データパスアーキテクチャ

### 2.1 大規模データパスアーキテクチャの概要

VLDP は将来利用可能であると考えられる豊富な資源を利用し、大規模な投機的実行を行うことにより性能向上を目指す、従来のアーキテクチャの延長上にはない新しいアーキテクチャである。平均実行 IPC (Instruction Per Clock cycle) 8 の性能を目標として設計が行われている。

VLDP では命令ブロック (IB: Instruction Block) と呼ばれる最大 32 の命令からなる命令列を処理単位として導入している。分散した複数の実行ユニット (EU: Execution Unit) を持ち、各々が一つずつ IB を処理する。

各 EU ごとにレジスタを持つ分散レジスタ構成を取っており、同一 EU 内へのデータアクセスは高速であるが、異なる EU へのデータアクセスは時間を要する。

*Code Optimization for Very Large Data Path Architecture*  
 Shinsuke IWASAKI<sup>†</sup>, Naoya HATTORI<sup>††</sup>, Daisuke IIZUKA<sup>†††</sup>, Shuichi SAKAI<sup>††</sup>, Hidehiko TANAKA<sup>††</sup>

{iwasaki, hato, iizuka, sakai, tanaka}@mtl.t.u-tokyo.ac.jp

<sup>†</sup>School of Engineering, The University of Tokyo

<sup>††</sup>Graduate School of Information Science and Technology,

The University of Tokyo

<sup>†††</sup>Graduate School of Engineering, The University of Tokyo

### 2.2 命令ブロック

IB の構成を図 1 に示す。IB は 4 つの Field (最大命令数 8 の BB) から構成され、1 つの Field は 8 つの命令スロットから構成される。各 Field 最後の 4 つのポイントを BP (Break Point) と呼ぶ。分岐命令は BP にのみ配置でき、分岐の飛び先は IB の先頭のみ指定できる。BP 以外の場所からの分岐すること、IB の途中に分岐してくることは許されない。この制限により命令を埋めることのできないスロットには NOP 命令が挿入される。

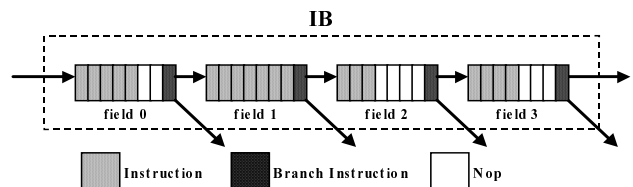


図 1: IB の構成

## 3 命令ブロックの生成

### 3.1 基本生成法

まず命令数が 8 を越える BB は 8 命令ごとに区切って、Field を生成する。この生成された Field を 4 つ結合し、空いた命令スロットに NOP 命令を詰めて 32 命令とすることで IB が生成される。ただし IB の途中へ分岐してくることは許されないので、Field が 4 つ未満でも次の Field への分岐があればそこで IB を区切る。また関数呼び出し、RET 命令などコードが途切れる命令の直後でも同様に IB を区切る。これにより 2.2 節で述べたような規則を満たす IB が生成される。

### 3.2 最適化の方針

異なる IB 間のデータ依存があると 2.1 節で述べた通り EU 間のデータ通信が起こり、多くの時間を要してしまう。よって IB 間データ依存をできるだけ少なくする必要はある。

しかし 2.2 節で述べた通り、分岐の飛び先やコードの途切れる命令により IB が区切られるため、IB 内命令数が少なくなり、IB 間のデータ依存が多くなってしまふ。これにより EU 間データ通信が増加してしまい、好ましくない。よってこのように IB が細かく区切られてしまう原因に対処し、できるだけ IB 内命令数を大きくして、EU 間通信を減らす最適化の実装について検討した。

### 3.3 profile の利用

ある PC から始まる IB は、図 2 のように最大 8 通り考えられる。このうち静的分岐予測を用いてもっとも実行確率の高いものだけを選びその 1 つだけを生成する。これによってよく実行される IB 内の命令数が大きくなり、EU 間通信が削減される。例えば図 2 で太い矢印の方が実行確率が高い場合、IB は色の付いた部分から構成される。

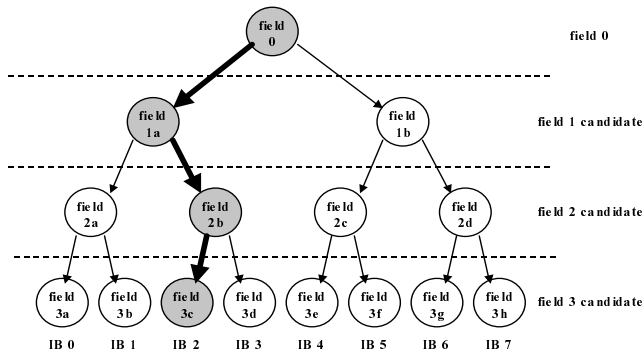


図 2: profile の利用

### 3.4 合流の除去

図 3a) のような 4 つの Field から IB を生成する場合を考える。IB の途中へ分岐してくることは許されないため、IB の入り口はその先頭のみである。よって図では入り口が複数ある Field D の手前で IB を区切らなければならない。しかし図 3b) のように Field D を複製することによって、IB が途切れるのを防ぎ、EU 間データ通信を削減することができる。

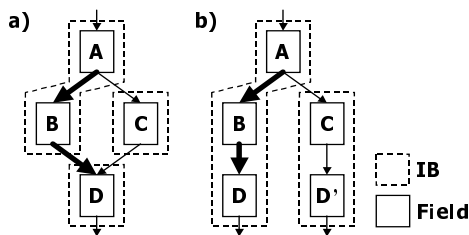


図 3: 合流の除去

### 3.5 ループの展開

図 4a) のような 1 つの Field が繰り返し実行されるループは図 4b) のようにループを展開する。これによって IB 内命令数を増やし、EU 間データ通信を削減することができる。

## 4 評価

最適化 C コンパイラ newcc[3] に 3.1 節で述べた IB の基本生成法、3.3 節、3.4 節、3.5 節で述べた最適化を実装し、最適化によって EU 間データ通信をどの程度削減できるか評価を行った。それぞれの最適化個々の効果と 3 つ全てを合わせた場合の効果について測定した。ベンチマークとして SPECint95 を用いた。

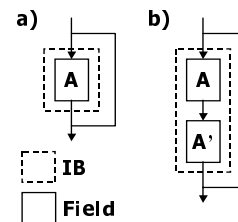


図 4: ループの展開

結果を図 5 に示す。全ての最適化を合わせた場合において平均 5% の削減に成功した。

なお評価においてユニット数、EU 間ネットワークバンド幅は無限大とした。またキャッシュは全てあたるものとし、分岐予測は 100% ヒットとした。

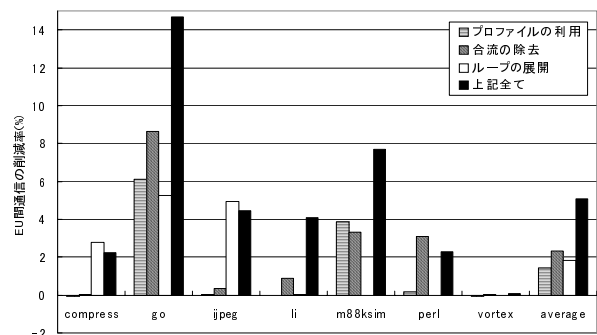


図 5: 最適化による EU 間通信の削減

## 5 おわりに

本稿では大規模データベースアーキテクチャにおける命令ブロックの最適な生成手法について検討した。今後は I キャッシュを考慮した場合の最適化について研究を行う。

## 参考文献

- [1] 辻秀典、安島雄一郎、坂井修一、田中英彦  
大規模データベース・アーキテクチャの提案  
情報処理学会研究報告 2000-ARC-139, pp.49-60, 2000.
- [2] 塚本泰通、安島雄一郎、辻秀典、坂井修一、田中英彦  
大規模データベース・アーキテクチャにおける命令ブロックの検討  
情報処理学会研究報告 2000-ARC-139, pp.61-66, 2000.
- [3] 飯塚大介、小沢年弘、坂井修一、田中英彦  
C コンパイラにおけるループ最適化の検討  
情報処理学会研究報告 1999-HPC-77, pp.65-70, 1999.