C-15 VLDP3 アーキテクチャの構想 (2) ～ソフトウェア支援～

服部 直也[†] 山口 健輔[†] 谷地田 瞬[†‡] 田中 裕治[†]
入江 英嗣[†] 飯塚 大介[†] 坂井 修一[†] 田中 英彦[†]

1. はじめに

1.1 研究の背景

半導体技術の進歩により、マイクロプロセッサが利用可能なトランジスタ数は年々増加している。しかし現在の主流であるスーパースカラ方式では、半導体資源を利用して積極的な投機技術の導入や命令ウィンドウの拡大を行っても、制御オーバーヘッドが問題となるため効果的な性能向上は期待できない。これを受けて、ソフトウェア支援を取り入れることで制御を簡略化した VLDP アーキテクチャが提案されてきた [1]。

1.2 研究の目的

2001 年度までの VLDP (以下、便宜上 VLDP2 と記載) では、分散レジスタ構成を採用していたため、分散ユニット間のデータ供給レイテンシが問題となっていた。これを受けて我々は VLDP2 のデータ供給を改善する VLDP3 アーキテクチャを提案する。本稿では Instruction Block(以下 IB) を基本単位とする VLDP3 ISA の概略を述べ、データ供給や制御投機ミスによるオーバーヘッドを削減する観点から IB 仕様の初期検討を行う。

2. VLDP3 の命令仕様

VLDP3 では、"複雑性の低減" と "大規模な並列性抽出及び投機実行" の両立を目的としており、そのために VLIW のバンドルの概念を拡張した "IB" を採用している。IB は複数命令の塊で、この塊を Fetch, Decode や Retire といった処理の基本単位とすることで制御を単純化する。ただし、小さな IB を多数実行するのでは目標とする単純化が期待でないため、IB には十分な大きさが要求される。この理由から、IB には 制御・データ依存のある命令も含むことができるものとする。この点で IB はバンドルと異なる。

2.1 IB 内データ供給

依存を含む命令の塊を単純かつ高速に処理するために、VLDP3 では IB 内のデータ転送経路を静的に指定する。そして IB 実行機構 (ALU-Net) は、IB をそのままハードウェアに mapping して処理する (図 1)。この意味では IB を、ALU-Net に対する Configuration 情報と捉えることもできる。

IB 内で閉じたデータ供給 (以下 IB 内通信と呼ぶ) にはレジスタを使用せず、Local Wire (ALU-Net 内に限定的に用意された高速データ配線) を用いて転送経路を指定する。このために VLDP3 Compiler は、FPGA Compiler と同様の "配置配線" を行う。

2.2 IB 間データ供給

IB が生成したデータはレジスタやメモリを介して外部に提供されるが、IB はその内部に複数の Basic Block を内包することができるため、IB が出力すべき レジスタ／メモリ の状態は複数存在する。通常のプロセッサでは分岐命令を用いてこれらの状態を選択しているが、VLDP3

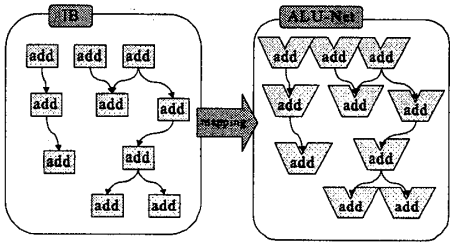

図 1: IB と ALU-Net
(データ転送経路は静的に指定されるので、
ALU-Net は IB をそのまま実行できる)

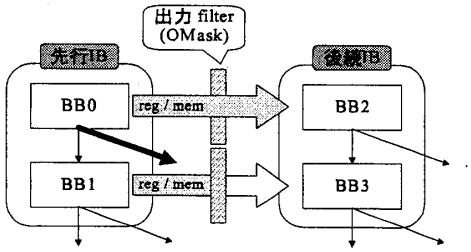

図 2: 複数 BB の封入と OMask による出力 filtering
(実行が無効化された BB の出力を適切に Mask する)

では制御を簡略化するために、出力を無効化するフィルタ (OMask) を用いて選択する (図 2)。Compiler は適切な OMask を用意し、実行時にその中の 1 つが採択される。

IB 内通信に対して、IB 間に跨るデータ供給を IB 間通信と呼ぶことにする。IB 間通信は IB 内通信と比べて、経路を動的に決める、フィルタリングが必要、などの理由からレイテンシが大きくなると予想される。そのため、ソフトウェア支援により IB 間通信を削減することが期待されている。

2.3 Mask 予測

今日の多くのプロセッサと同様に VLDP3 も制御投機を行うが、予測対象となるのは分岐方向ではく、採択される Mask の ID となる。Mask 予測に関しても分岐予測と同様、予測ミス時に後続命令の実行取り消しや再実行といったオーバーヘッドが加算されるため、ソフトウェア支援により Mask 予測ミスを削減することも期待されている。

3. IB 構成と制約条件の検討

前述のように、IB は ALU-Net を動かす為の mapping 情報であるため、VLDP3 の性能は Compiler が構築する IB の影響を強く受けることが予想される。具体的には以下に示す要素などが性能を左右すると考えられる。

(1) IB 内通信のオーバーヘッド (配置配線の品質)。
(2) IB 間通信のオーバーヘッド。
(3) Mask 予測ミス。
(4) メモリ依存投機。

[†]東京大学
[‡](株) 日立製作所

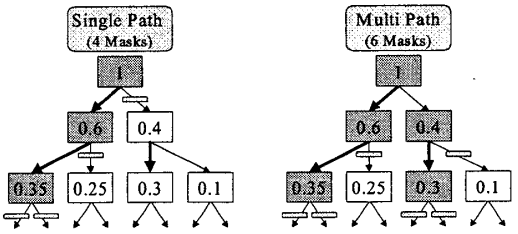

図 3: IB 内の制御依存モデルの違い
(□は BasicBlock とその採択確率、■は IB に封入したこと、
⬭ は OMask を示す)

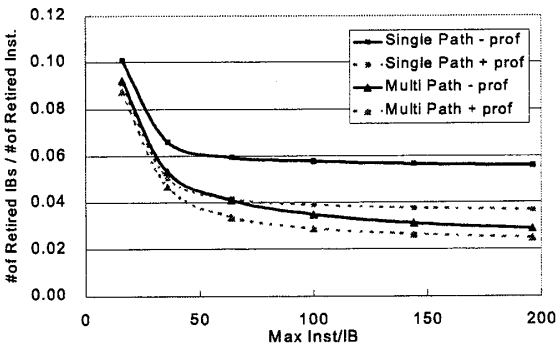

図 4: Scheduling 戦略、Profile の有無と Retire IB 数
(数値が小さいほど IB 間通信 overhead が少ないと考えられる)

(2) に関しては、レジスタやメモリを介したデータ供給は近い命令間に多いため、総 Retire IB 数は、IB 間通信のオーバーヘッドと比例に近い関係にあると予想される。(3) に関しては、Mask 予測が外れた際にパイプラインバブルが発生すると考えると、予測ミス回数は、予測ミスオーバーヘッドと比例に近い関係にあると予測される。そこで本稿では、初期評価として (2), (3) に着目し、Retire IB 数と Mask 予測ミスの回数を測定した。

3.1 IB 内の制御依存モデル

IB は制御依存のある命令も許容しているため、複数の Basic Block を含むことができるが、これらの選択に関していくつかのモデルが考えられる。図 3 にその違いを示す。Multi Path は IB 内に分岐を許すモデル、Single Path は IB 内に分岐を許さないモデルである。Retire IB 数の観点からは静的分岐予測への依存性が低い Multi Path が優れているが、Multi Path は IB 内の Mask 数が多くなり易いため Mask 予測の観点では不利と考えられる。本稿ではこの両モデルに対しての測定を行い、比較検討する。

3.2 評価環境

今回の調査では配置配線は理想化した。SPECint95 の compress, li, go, ijpeg, m88ksim の train に対して、理想的な Profile を用いた場合と用いなかった場合それぞれの Retire IB 数と Mask 予測ミス回数を測定した。ただし、5 つのアプリケーションに対する性質を平均化平均値を調べるために、両値とも 総 Retire 命令数 で正規化した値の算術平均を用いた。Mask 予測機構としては単純な Last Value Predictor を用い、エントリ数は無限とした。

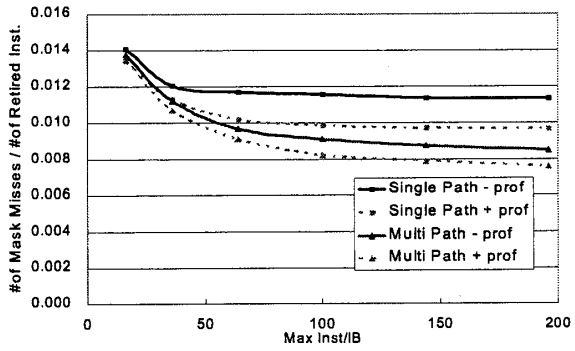

図 5: Scheduling 戦略、Profile の有無と Mask 予測ミス
(数値が小さいほど予測ミス overhead が少ないと考えられる)

3.3 結果と考察

Retire IB 数に関する結果を図 4 に示す。予想した通り Multi Path は Single Path よりも Retire IB 数を少なく抑えられている。また、Profile の有無による Retire IB 数の差を見ると、Multi Path は Single Path よりも変動率が小さい。Compile 時に常に Profile 情報が使用可能とは限らないので、変動率は小さい方が望ましく、この観点からも Multi Path モデルの優位性が示された。

次に Mask 予測ミス回数に関する結果を図 5 に示す。当初の予想に反して、Profile の有無に関わらず Multi Path は Single Path よりも Mask 予測ミスの回数が抑えられており、Profile の有無による変動率に関しても Multi Path の方が小さく抑えられている。この理由としては、Mask 数の増加によるミス率増大効果よりも、Retire IB 数削減効果の方が大きかったために、結果としてミス回数が減少したと考えている。

最後に、IB に封入する命令数に関しては、Retire IB 数、Mask 予測ミス数とも命令数の増加に伴って緩やかに減少しているため、この結果から最適な大きさを議論することは難しい。しかし、(1) 64 以降と以前では傾きが比較的大きく変化していること。(2) IB を大きくすると無効命令が増加し易く ALU-Net の有効稼働率が下がること。などを受け、64 命令程度が妥当ではないかと考えている。

4. おわりに

本稿では、IB を用いた VLDP3 の実行モデル概要を示し、IB 仕様の初期検討を行った。IB 間データ供給や Mask 予測ミスオーバーヘッドの観点から、Multi Path 型の制御依存モデルの優位性が明らかになり、最大命令数は 64 程度が妥当であろうという知見が得られた。

今回は実際の配置配線を行っていない等、部分的な評価しかできなかったが、今後は配置配線コンパイラやサイクルレベルシミュレータを開発し、詳細な評価を行っていく予定である。

参考文献

[1] 辻秀典, 安島雄一郎, 坂井修一, 田中英彦: 大規模データパス・プロセッサの提案, 情報処理学会研究報告 計算機アーキテクチャ研究会, 2000-ARC-139, Vol. 2000, No. 74. pp. 55-60

[2] 岩崎慎介, 服部直也, 飯塚大介, 坂井修一, 田中英彦: 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, 情報処理学会第 64 回全国大会, No. 5ZB-4