# 1Q-10 ALU-NET: VLDP アーキテクチャにおける命令実行機構

**辻 秀典, 中村 友洋, 吉瀬 謙二, 安島雄一郎, 高峰 信, 坂井 修一, 田中 英彦**
東京大学大学院 工学系研究科

## 1 はじめに

命令レベル並列実行によって性能を得るマイクロプロセッサが一般的となった現在、さらなる性能向上をめざした次世代マイクロプロセッサに関するさまざまな研究が行われている。その一つとして我々は、大規模データパス (Very Large Data Path – VLDP)・アーキテクチャ [1] を提案している。これは、マイクロプロセッサにおける命令実行の本質が演算であることに注目し、命令列中のデータフローを ALU-NET と呼ぶ複数の ALU によって構成される機構上で実現することによって、命令実行のスループットを向上させようとするアーキテクチャである。本稿ではこの ALU-NET に焦点をあて、VLDP アーキテクチャにおける命令実行が、従来のアーキテクチャと比較してオーバヘッドが縮小されていることについて述べる。

## 2 マイクロプロセッサにおける命令実行

マイクロプロセッサにおける処理は、メモリ空間上のデータに何らかの演算処理を施し、得られた結果を再びメモリ空間上に書き戻すことが基本である。この演算処理は、単一であったり複数であったりさまざまであるが、必要なデータが用意されれば複数の演算処理であっても連続的に行うことが出来る。実際にはあるデータフローに注目した場合、その処理の流れは 図 1 のようになる。

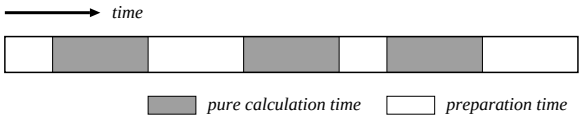


図 1: データフローにおける処理の流れ

*pure calculation time*とは演算器において演算が行われている時間に相当し、*preparetiontime* とは演算を行うためにデータを準備、格納する時間である。この時間は具体的には次のようなものである。

1. データ待ち合わせ時間
2. メモリアクセス時間
3. レジスタアクセス時間
4. パイプラインやクロックによるオーバーヘッド

ある処理 B の入力 $x, y$ のうち、$x$ が処理 A の結果である場合、$y$ が準備できていたとしても処理 B が終了しなければ処理 A は行えない。この場合 $y$ は一時的にメモリまたはレジスタに格納される。この格納されている時間がデータ待ち合わせ時間である。

実際の演算処理は、その入力や演算の種類によって回路の遅延時間は変動している。一般的にパイプラインピッチやクロックはその遅延時間の最大値をもって設定されるために、全ての演算の時間がその最大値に丸め込まれる事になる。この丸め込みがパイプラインやクロックによるオーバヘッドである。

## 3 VLDP アーキテクチャにおける命令実行

先に述べたデータ待ち合わせ時間そのものは、マイクロプロセッサにおける命令処理に対して直接影響を与えるものではないが、データ待ち合わせは必ず 1 回以上のレジスタ (メモリ) アクセスを伴う。このレジスタ (メモリ) アクセスはデータの流れの上では不必要なものであるが、アーキテクチャ上の都合によって行われる中間的なデータアクセスである。図 2 の上はレジスタアクセスを伴うデータ待ち合わせ、下はレジスタアクセスを伴わないデータ待ち合わせである。レジスタアクセス伴わない待ち合わせの場合、データが必要となったときにすぐに演算を開始できるため処理時間を削減できる。

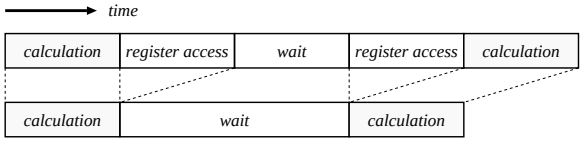


図 2: レジスタアクセスによるオーバヘッド

しかし、現在のマイクロプロセッサにおいては図 2 の下のような動作をサポートする機構はフォワーディング機構以外には存在しない。この、フォワーディング機構は短時間の待ち合わせを実現するのみである。

これまでのマイクロプロセッサの性能向上はパイプライン化および命令レベル並列性を利用したスーパスカラ機構によるもので、ここで述べた中間的なデータアクセスによる命令処理のオーバヘッドに関する改善を行っていない。命令レベル並列実行では複数の機能ユニットが

---

ALU-NET: The Execution Mechanism of VLDP Architecture
Hidenori TSUJI, Tomohiro NAKAMURA, Kenji KISE, Yuichiro AJIMA, Makoto TAKAMINE, Shuichi SAKAI, Hidehiko TANAKA
Graduate School of Engineering, The University of Tokyo

存在するものの、それらの機能ユニット間に密な結合はなく 1 命令単位で命令を独立に実行することを前提としているためである。

VLDP アーキテクチャはこの点に注目し、中間的なデータアクセスを出来る限り排することによって、命令単位の実行によるオーバーヘッドを削減し効率的な演算処理を行う。データパスと複数の ALU を用意することによって、データフローを演算機構の内部で実現しデータの待ち合わせもそこで行う。このデータパスと複数の ALU によって構成される機構が ALU-NET である。

なお VLDP アーキテクチャは、オーバヘッドを削減した実行だけでなく、大規模な投機的処理による分岐制御ペナルティ削減も行うことによって、これまでとは異なるアプローチでの性能向上を目指している。

## 4 ALU-NET による処理の利点

ALU-NET はデータフローを処理する機構であり、次のような特徴を持つ。

- データの待ち合わせが可能
- フォワーディングパスが短い
- データ駆動による処理

図 3 の上が従来のパイプラインの演算ステージの処理、下が ALU-NET における演算処理である。

従来の演算ステージでは処理されるデータはすべてサイクル毎に独立して供給、書き戻されていたのに対し、ALU-NET においては待ち合わせが必要なデータは ALU 間を接続するデータパス上のラッチによって保持される。これによって、中間的なデータアクセスなしにデータの待ち合わせを実現できる。

ALU-NET におけるフォワーディングパスは、ALU 間を接続するデータパスにより実現できるためその距離が短い。短時間のデータ待ち合わせによる中間的データアクセスは、スーパスカラプロセッサにおいてもフォワーディング機構によって高速化できるが、フォワーディング機構は複数のパイプラインステージを越えたパスであるため、距離的に短いパスではない。さらに、性能向上を目指してスーパスカラ機構を拡張すれば、フォワーディングパスがハードウェア的なクリティカルパスとなる。それに対し ALU-NET のフォワーディングパスは短いため、クリティカルパスとはなり得ない。

データ駆動による処理が可能であるために、ALU 間に存在するラッチが、直接接続している ALU へのパスを制御することにより待ち合わせを実現できるため、ALU のデータの入出力に関する制御を分散化できる。なおこのラッチは一時的なデータ保持を行うものであるためその動作は軽い。また、これによりデータパス毎で独立してデータが流れるために、従来のパイプラインステージ幅による演算時間の丸め込みが行われないだけでなく、フォワーディングパスが短くなることによって動作遅延も小さくなる。さらに、データ駆動に適した非同期式回路を用いてクロック制御をなくすことにより、データ駆動動作を効率良く実装できるだけでなく、クロック同期によるオーバヘッドを削減する事も可能となる。

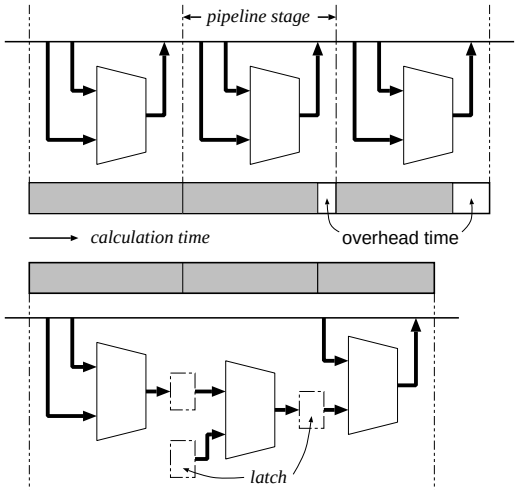

図 3 : ALU 間の接続と演算時間

スーパースカラプロセッサのような複数の機能ユニットを拡張する形で演算資源を大規模化した場合には、その演算資源に対するデータ供給能力の問題が生じる。これは 図 3 の上に示したような、各サイクル毎に独立したデータ供給を行っているためである。VLDP においても、ALU-NET 内部の各 ALU に対するデータ供給の総数は多くなるが、中間的データアクセスおよびフォワーディングパスが ALU-NET 内部の局所的な接続で実現される。投機処理および ALU-NET の規模にもよるが、データアクセスの半数以上が ALU-NET 内で吸収できる可能性 [1] があり、単純に ALU-NET に対するデータ供給が爆発的なものとはならない。

## 5 おわりに

VLDP アーキテクチャは命令列からデータフローを抽出し、複数の ALU とそれを接続するデータパスによって構成される ALU-NET によって処理を行う。ALU-NET は、中間的データアクセスを必要としないデータ待ち合わせが可能なだけでなく、データ駆動による処理が可能であり、これまでのマイクロプロセッサにおける命令単位の処理と比較してオーバーヘッドを削減した実行が可能である。

## 参考文献

[1] 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, 情報処理学会研究会 ARCH, Vol. 124, No. 3, pp. 13-18 (1997).