# 2H-07 VLDP アーキテクチャにおける ALU-NET の構成と実行制御の検討

**辻 秀典**, **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 [2] 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、これを構成する複数の ALU 間の接続モデルおよび各 ALU 上での命令実行の制御方法について検討する。

## 2 VLDP アーキテクチャにおける命令実行

命令実行の本質部分といえる演算を効率良く実行することを目的とする VLDP アーキテクチャでは、そのために ALU-NET という機構を用意している [1]。

現在のマイクロプロセッサにおいては、短時間のデータの再利用をフォワーディング機構によって実現しているのみで、基本的に演算間のデータのやりとりはレジスタを介す構造になっている。これに対し、複数の ALU とそれを接続するパスによって構成された ALU-NET は、レジスタを介した中間的なデータのやりとりを複数の ALU 間の局所的な接続として実現することができ、これによって不要なレジスタアクセスを削減することが可能である。また、現在のマイクロプロセッサにおいてフォワーディングパスは、複数のステージを越えた接続となるためクリティカルパスとなりうるが、ALU-NET ではこれを短くすることも可能である。このような理由により、ALU-NET はデータの授受に伴うオーバヘッドを削減できる高速な演算処理を可能としている。

この ALU-NET を持つ VLDP アーキテクチャの基本構成は図 1に示すようなものである。実際に演算処理を行う機構は ALU-NET であるが、データフロー解析に基

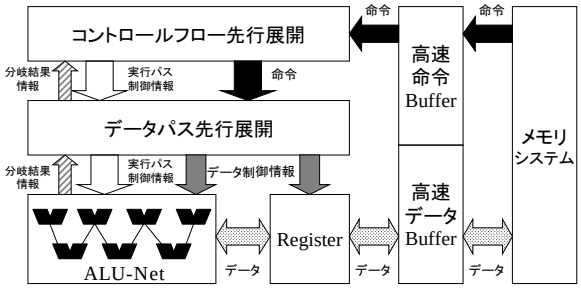


図 1: VLDP アーキテクチャブロック図

づく ALU-NET 内部の各 ALU への命令の割り当ておよびデータの供給指示は、データパス先行展開が行う。コントロールフロー先行展開は、分岐命令がもたらす制御依存関係を解消する機構であり、複数パス同時実行を可能とする命令フェッチを行う。まとめると、VLDP アーキテクチャによる性能向上は次の 2 点により得られるものである。

1. ALU-NET によるデータの授受の高速化
2. 大規模な投機処理による分岐制御ペナルティの削減 (複数パス実行)

さらに、VLDP アーキテクチャは、高性能かつ大規模化を可能とするアーキテクチャであり、大規模化に伴うハードウェア的なクリティカルパスを短くする必要がある。そのため、コントロールフロー先行展開、データパス先行展開、ALU-NET をはじめとする主要な機構を独立させ、各機構間のやりとりのスループットを向上させることによって、全体の処理を高速に動作させる。また、現在のマイクロプロセッサでは集中しているアーキテクチャ自体の制御も、各機構間によって独立させ、その情報交換をパイプライン化することによってスループットを向上させる。

## 3 ALU-NET における実行制御

ALU-NET 内部の ALU への命令の割り当て、ALU 間の接続、ALU へのデータ供給、結果の書き戻しの指示はデータパス先行展開から行われる。この指示 (ALU 制御情報) に基づく ALU の制御を直接データパス先行展開が行う場合、制御信号線が伸び制御遅延が大きくなるだけでなく信号線の数が増える。そのため、ALU の制御はデータパス先行展開とは分離し、制御は ALU-NET

---

*Study of the ALU-NET structure and the execution control on VLDP architecture.*
Hidenori TSUJI, Shuichi SAKAI, Hidehiko TANAKA
Graduate School of Engineering, The University of Tokyo

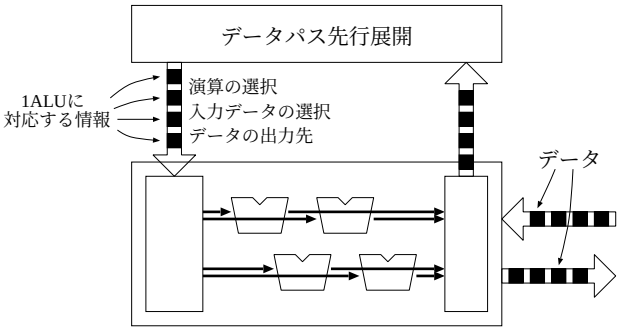


図 2: ALU-NET の実行制御

内部で行う。このとき ALU 制御情報はパイプライン化され ALU-NET に送られるため、制御の遅延および不要な信号線の増加を招かない。

具体的には 図 2 に示すような制御情報のやりとりが行われる。この制御情報にはどの ALU に対する指示かの情報も含まれているため、これに基づいて指定された ALU の制御を ALU-NET 内部で行う。各 ALU は、すべての出力先に対してデータを供給できてはじめて開放することができるため、データの出力先という情報が必要である。ALU 開放の情報はデータパス先行展開に送られ、再割り付け可能である事を知らせる。このような ALU の開放情報を管理すれば、データパス先行展開が演算の状況を直接把握する必要がなくなる。このような実行制御によって、制御の集中の回避と高速化を実現する。

## 4 ALU-NET の構成

### 4.1 ALU 間の接続モデル

ALU-NET は複数の ALU により構成されるが、その接続方法によっていくつかのモデルが定義される。

最も基本的な構成は 図 3 の (a) に示したモデルである。これは各 ALU が自由に接続可能となっており、柔軟なデータフローの実現が可能である。ALU 単位での接続が自由であるため ALU の利用率が高くできる。しかし、ALU の数が増加した場合に信号線と制御が複雑化するだけでなく、それに伴う動作遅延の増加を招くため、常に現実的なモデルとはいえない。

次に示すモデルは 図 3 の (b) に示すもので、適当な規模の (a) に示したモデルの ALU-NET を複数接続す形である。このように階層構造を持たせることによって、制御と接続を適当な規模で分割できるため、モデル (a) と比較して大規模化が可能であると考えられる。しかし、ALU の利用率およびデータ転送の遅延を最適化するためには、マイクロプロセッサにおける計算モデルの解析に基づく適切な階層構造と接続を設定する必要がある。またこれは、データパス先行展開で行うデータ依存解析

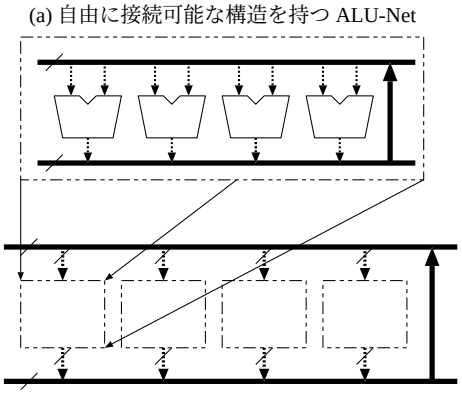


図 3: ALU-NET の接続構造

および ALU 割り当てアルゴリズムに適したものになっていなければならない。

### 4.2 ALU の構成

ALU は単位となる演算器とラッチにより構成される。単位となる演算器をすべて汎用的に同じとすると、ALU 割り付けは簡単化されるが ALU 単体の回路規模の増大を招く。機能を限定した複数のタイプの ALU を設定すると回路規模の最適化は行えるものの、ALU 利用の制約が与えらるために ALU 割り付けが複雑化する。

ラッチ数も ALU の構成を考える際に重要となる。ラッチの増加はデータの再利用と ALU 間のデータの流れの制御を容易にするが、その場合ラッチによる遅延が増大するために、局所的なデータアクセスの高速化をめざす ALU-NET の目的に反する。逆にラッチ数を減らすと高速化は望めるものの、データの流れの制御が限定され ALU の利用率が低下する。また実行頻度が高い複数の演算の組合せに対しては、ラッチを介さないデータの授受を行うことによって、演算遅延を削減することも可能である。

## 5 おわりに

ALU-NET を構成する ALU の接続とその構成単位となる ALU は、その規模および制御の複雑さによってさまざまなモデルが定義できる。これらについて今後、実際の実行に基づく評価が必要である。

## 参考文献

[1] 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: ALU-NET: VLDP アーキテクチャにおける命令実行機構, 情報処理学会 第 57 回全国大会, Vol. 1, No. 1Q-10 (1998).

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