ジェットパイプラインを用いた次世代スーパーコンピュータ

A Next Generation Supercomputer Using Jet Pipeline

中村　維男 （東北大）　　○片平　昌幸（東北大）

Tadao NAKAMURA, Tohoku University, Aoba-ku, Sendai
Masayuki KATAHIRA, Tohoku University

Key Word: Super Computer, Computer Architecture, Jet Pipeline, VLIW, Superscalar

１．はじめに

パイプライン方式に基づくスーパーコンピュータの出現と、その発展によってもたらされた高速な計算能力は、大量な計算を必要とする各種の分野において重要な役割を果たしつつある。しかし、ベクトル化によるパイプライン方式の有効利用が不可能な処理は現在のスーパーコンピュータでは高速化が困難である。

本稿では、より高性能をめざす次世代のスーパーコンピュータのためのアーキテクチャを提案し、またそのシュミレータを構築して簡単な性能評価を行った結果について報告する。

２．パイプライン方式スーパーコンピュータ

現在、流体力学や熱力学など、科学技術計算の分野で広く用いられているスーパーコンピュータは、ほとんど全てがパイプライン方式によるスーパーコンピュータである[(1)(2)]。この方式では、パイプライン化された演算器に、各クロックサイクルごとにデータが送り込まれる。演算器は数段階のステージに分かれており、流れ作業的に処理が行われる。あるデータが演算器に入ってから結果が得られるまでにはステージ数だけ時間が必要であるが、見かけ上は各クロックサイクルごとに結果が得られることになる。

パイプライン化された演算器に対してデータを効率よく送り込むために、スーパーコンピュータにはベクトルレジスタと呼ばれる一時記憶装置が装備されている。演算はベクトルレジスタの内容に対して実行され、結果もベクトルレジスタに格納される。ベクトルレジスタと主記憶間には数個のベクトルロード・ストアユニットがあり、演算と並列してデータの転送が可能になっている。

以上のような、パイプライン方式スーパーコンピュータにおけるベクトル演算の概念を図１に示す[(2)]。このようなハードウエアを効率よく利用し、最大性能を引き出すためには、ベクトル化という処理を行わなければならない。例えば、図１中のループの形のスカラ処理プログラムは、その下のベクトル演算命令１つに置き換えることができる。

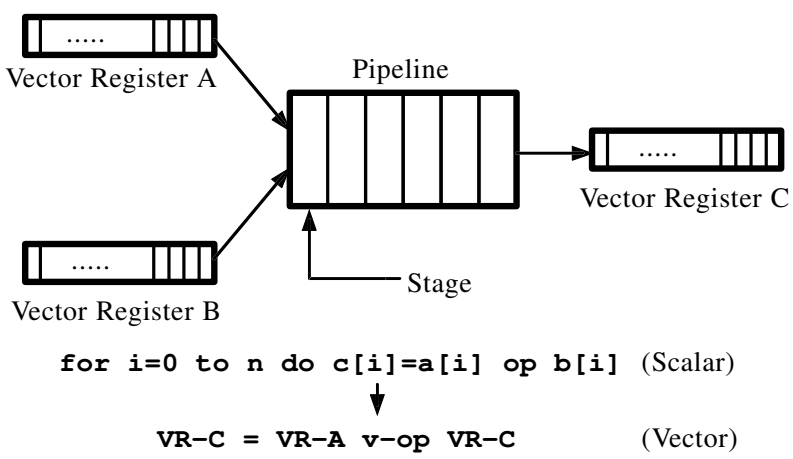

Fig.1 Vector Processing of the Super Computer

３．ジェットパイプラインアーキテクチャ

前節で述べたように、現在のスーパーコンピュータはプログラムをベクトル化し、パイプライン方式の演算器を効率よく動作させることによって高速な演算能力を得ているが、もしプログラム中でベクトル化できる部分の割合（ベクトル化率）が低ければ、本来持つ性能を発揮することが不可能となる。

そこで我々は、MIMD方式の並列処理機構である密結合マルチプロセッサシステムをより密接に結合させ１カプセル化し、さらにスーパーコンピュータのベクトル演算パイプライン方式をも組み合わせた新しいアーキテクチャであるジェットパイプライン方式を提案する。その概念図を図２に示す。このアーキテクチャは命令パイプラインを複数個持ち、それらが整数演算用のALUや浮動小数点演算用の演算パイプラインなどからなる演算部を制御する形式をとっている。命令パイプラインに投入される命令(scalarまたはvector)は、VLIW的に演算ユニットを最大限に効率よく利用できるようにコンパイラによるスケジューリングを施された後各命令パイプラインにマッピングされて実行が行われる。これは、ハードウエアによってスケジューリングが行われるSuperscalar方式とは異なる。Superscalar方式を採用しなかった理由として、以下の２点が挙げられる。

(1) 命令の互換性を考える必要がなかったこと。

(2) 実行に多くのクロック数を要するベクトル命令をハードウエアで無駄なくスケジューリングすることは困難であること。

また、VLIW方式とも異なる点として、各命令パイプラインは独立であり、全体として１つの長命令語により制御するのではなく、独立な複数の命令語の組によって制御が行われる。

このアーキテクチャを用いることにより、従来のスーパーコンピュータで高速化できるベクトル化可能な部分に加えて、ベクトル化不可能な部分についてもVLIW的な並列化を行うことによって高速化を図ることができる。さらに、プログラムが持つ並列性が非常に少なく、並列化が困難な場合についても、アドレスの計算とその他の演算処理などの低レベルの並列性を有効利用することにより高速化することが可能となる。なお、ジェットパイプラ

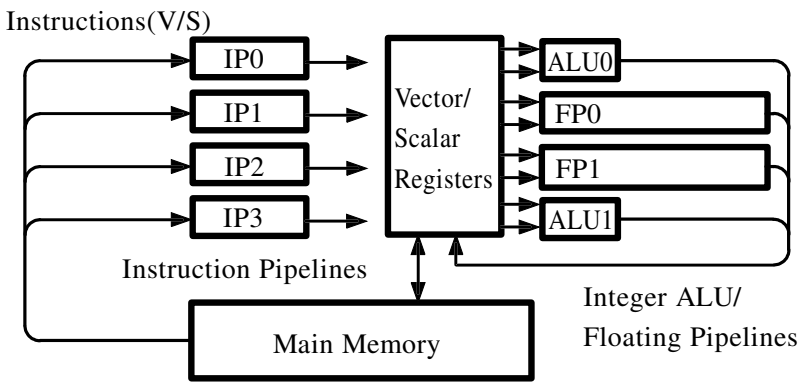

Fig.2 The Conceptual Design of the Jet Pipeline

インという名称は、図２の概念図において、命令を空気に、入力データを燃料に、演算結果を燃焼ガスに見立てたときシステム全体がジェットエンジンのように見えることから名付けられた。

４．シミュレータの構築と性能評価

前節で提案したジェットパイプラインアーキテクチャについて、我々はシミュレータを構築しベンチマークプログラムを実際に実行することによって性能評価を行うことにした。

シミュレータの構築を始める前に、まず命令セットを定義した。スカラ命令セットはRISCの考え方[3]を取り入れ、ロード・ストア命令とレジスタ間演算命令、そして分岐命令から構成されている。ベクトル命令セットも同様に、ベクトルロード・ストア命令、ベクトルレジスタ間の演算命令（整数演算及び浮動小数点演算）、その他のベクトル命令（マスクレジスタ操作など）から構成される。また、チェイニング演算も命令セットレベルで制御される。

命令セットの定義を行った後、シミュレータの構築を行った。まずシミュレーションを行う機能レベルとして、ここでは機能ブロックレベルを想定した。機能ブロックレベルで実際にシミュレータを構築する場合、図２の概念図をそのまま用いることはできないので、図３に示すような機能ブロックを想定し、命令パイプラインを４本持つシステムについてシミュレータを構築した。

命令パイプライン(Inst. pipe,IP)は６段構成で、２段づつオーバーラップして実行される。整数演算器(ALU)はスカラ命令の実行において 繁に利用されるので各命令パイプラインにひとつづつ対応させてある。浮動小数点演算パイプライン(FP calc. pipe)はベクトル命令、スカラ命令の両方で共用される。したがって、ベクトル浮動小数点演算命令実行中はスカラ浮動小数点演算命令は実行できない。また、ベクトル演算命令は各演算パイプラインに併設されたベクトル演算コントローラによって実行の制御が行われるので、浮動小数点演算以外のスカラ命令とベクトル演算命令の並列実行が可能である。加減算用と乗除算用の２つの浮動小数点演算パイプラインとベクトル整数演算用のALU、及びそれらに併設されるベクトル演算コントローラから構成されるパイプライン演算ブロックは２つの命令パイプライン間で共用される。各演算パイプライン間にはチェイニング演算のための相互結合ネットワークも存在する。また、レジスタはスカラレジスタとベクトルレジスタから構成される。レジスタ−主記憶間のデータ転送のためには、スカラ命令用のロードストアユニット及びベクトルロードストアユニットの両方が存在し、ベクトル演算等と並行してデータ転送が行えるようにしている。

シミュレータで実行するプログラムは、現在アセンブリ言語で記述されたものを用いており、専用のアセンブラを用いて機械語に変換し、それをシミュレータにロードして実行する。プログラムを実行後、シミュレータは実行に要したクロック数を出力する。

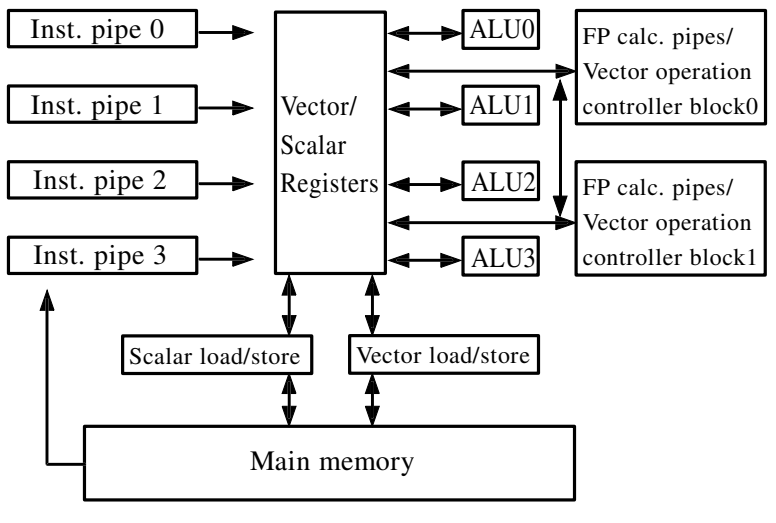

Fig.3 Functional Blocks of the Simulator

シミュレータを用いて性能評価を行うためのベンチマークプログラムとして、今回はスーパーコンピュータの性能評価にしばしば用いられるリバモアループを採用した[1]。リバモアループは高度にベクトル化可能なものからベクトル化が困難なものまで各種のループから構成されているが、今回はベクトル化及び並列化が容易なloop1、部分的にベクトル化ができ並列化も可能なloop3、ベクトル化が不可能で並列化も困難なloop5の３つのループについてシミュレータ上で実行に要するサイクル数を計測した結果を図３に示す。なおFORTRANで記述されているリバモアループのソースプログラムはハンドコンパイルによりアセンブリ言語に変換した。また、ベクトル化や並列化も人間が行った。図３では各ループについてIP数が１かつスカラ命令のみによる実行時間を１としたときのスピードアップを示している。

loop1はプログラム中に並列性を多く含み、かつベクトル化も可能なので、大きなスピードアップが得られている。特に、IP数４でベクトル化した場合には30倍近いスピードアップである。また、ベクトル命令を用いない4IPの場合でも4倍のスピードアップとほぼ理想的な値を得られている。

loop3はプログラム中に並列性を多く含んでいるが、全てをベクトル化できないプログラムである。したがってloop1よりもスピードアップは少ないが、それでも最大12倍程度の値を得られた。

loop5はプログラムに並列性がほとんどなく、またベクトル化も不可能である。したがって、従来はまったくスピードアップが不可能であったが、このアーキテクチャでは、わずかに存在する低レベルの並列性を利用することにより1.7倍のスピードアップを得ることができた。

５．まとめ

本稿では、従来のパイプライン方式を越える性能をめざす次世代スーパーコンピュータのためのアーキテクチャとして、ジェットパイプライン方式を提案し、シミュレーションによりその性能評価を行った。今後の課題としては、自動ベクトル化・並列化の可能なコンパイラの開発が第一に挙げられる。

[参考文献]

(1) 唐木：″スーパーコンピュータの使い勝手″, bit, Vol.16, No.5, pp.20-26, 1984

(2) 安村：″スーパーコンピュータとそのコンパイラ″, bit, Vol. 17, No.7, pp.19-30, 1985

(3) D.Tabak(大森 訳)：″RISCシステム″, 海文堂, 1991

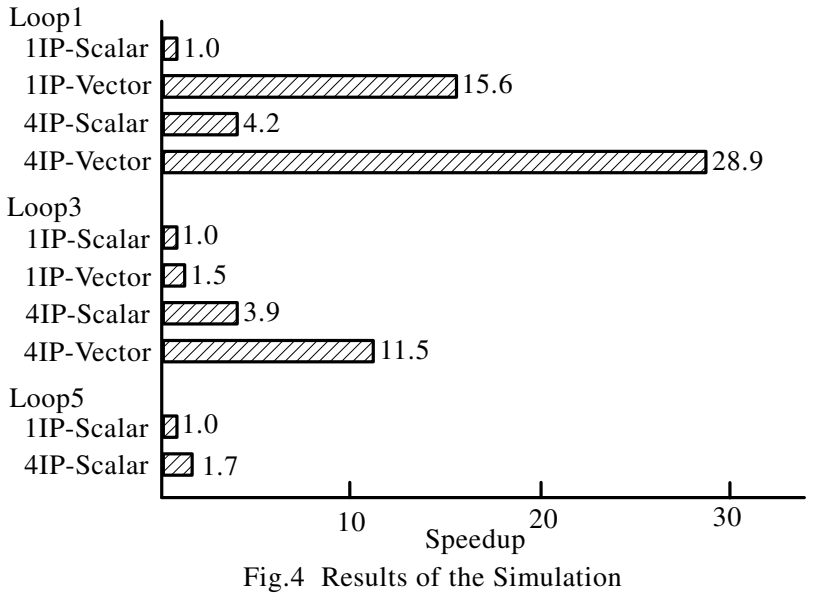

Fig.4 Results of the Simulation