ストリームFIFO方式の構想
5T-10 －SIMP方式のベクトル処理への適用－

弘中哲夫　村上和彰　富田眞治
(九州大学)

1. はじめに

我々は，シングル・ユーザ向けデスクトップ・スーパーコンピュータ『新風』DTS－edition開発の一環として，そのメイン・プロセッサである『新風』プロセッサを開発中である[1]．本稿では，『新風』プロセッサに要求されるベクトル処理能力を得るために新たに考案した，“ストリームFIFO方式”と呼ぶベクトル処理方式について述べる．

2. ベクトル処理ユニットに対する要件

『新風』はSIMP（単一命令流／多重命令パイプライン）方式に基づくプロセッサであり，そのスカラ処理ユニットに付加する形でベクトル処理ユニット（ストリーム処理ユニットと呼ぶ）を設ける．すなわち，命令ブロックのフェッチ（IF）およびデコード（D）ステージは，スカラ処理ユニットで行ない，ベクトル命令（ストリーム命令と呼ぶ）のみをベクトル処理ユニットで実行する．

一般に，ベクトル処理ユニットに対する要件は次のようなものである；

①ベクトル化率の向上：できるだけ多くのループをベクトル化可能とし，ループ内各命令のデコード時間，および，ループを制御する分岐命令の実行時間などに起因するオーバーヘッドを除去する．

②大容量ベクトル・レジスタの実装：大容量のベクトル・レジスタを介して，ベクトル・ロード／ストアとベクトル演算間の並列実行およびチェイニングを行うことにより，ロード／ストアのオーバーヘッドを除去する．

③ベクトル演算パイプライン間のチェイニング：ベクトル演算パイプライン間をベクトル・レジスタを介してチェイニングすることで，データフローに従った高速ベクトル演算を行う．

④複数ベクトル命令の並列実行：上記②および③のように，複数のベクトル命令を並列に実行させて，パイプラインの使用効率向上を図る．

さらに，『新風』におけるベクトル処理ユニットに対しては，以下の要件がある；

⑤より多くのベクトル命令の並列実行：『新風』では，最大16もの命令（スカラあるいはストリーム命令）が演算（E）ステージに存在し得る．これらのベクトル命令を有機的にチェイニングすることで，さらに自然な形でデータフローに沿ったベクトル演算を行わせる．

⑥ベクトル命令のout－of－order実行：スカラ処理ユニット同様，ベクトル処理ユニットでも命令のout－of－order実行を行う．しかし，その実現には，データ依存関係の検出および保証（チェイニングも含む）が必要となり，これを一般のベクトル・レジスタ上で行おうとすると膨大なハードウェア量を要する．したがって，なるべく軽量のハードウェアでこれを実現する．

⑦セグメント化の回避：一般のベクトル・プロセッサでは，ベクトル長がベクトル・レジスタ長より大きい場合，ベクトル・データのセグメント化が必要となる．セグメント化は，ベクトル・ロード／ストアの回数を増すばかりでなく，データフロー実行の中断を伴う．このようなオーバーヘッドを招くセグメント化をできるだけ避けるようにする．

⑧スカラ命令によるベクトル処理：ベクトル処理ユニットがロードしたベクトル・データをスカラ命令で処理する．また，その反対に，スカラ命令が生成した演算結果をベクトル処理ユニットがベクトル・データとしてストアする．これにより，ベクトル化不可能なループでもスカラ命令のままでベクトルを対象として処理する[2]．

これらの要件を満たすために，我々は次節で述べる“ストリームFIFO方式”を考案した．

3.『新風』のベクトル処理方式

3.1 FIFOレジスタ

一般のベクトル・レジスタに代えて，『新風』のベクトル処理ユニットでは，“データFIFO（FIFO）”と呼ぶFIFO動作をするレジスタを16本備える．これらは，命令セット・アーキテクチャ上は通常の汎用レジスタなどと同様，演算命令（スカラおよびストリーム命令）のソース・レジスタ，および，ディスティネーション・レジスタとして指定できる．

3.2 ストリーム演算命令

『新風』のベクトル処理ユニットにおいて，ベクトル演算命令に相当するのが“ストリーム演算命令”である．ストリーム演算命令は，そのソース・オペランドに対して，指定された回数だけ同じ演算を繰り返す．ベクトル長に相当する繰り返し回数は，ストリーム・カウント・レジスタ（SCR）で指定する．ソース・レジスタにFIFOが指定されている場合は，その先頭データから順に演算を施していく．また，ディスティネーション・レジスタにFIFOが指定されている場合は，演算結果を順に書込んでいく．なお，ソースおよびディスティネーションとして，汎用レジスタ（GR）および浮動小数点レジスタ（FR）を用いることも許される．

3.3 ストリームLOAD／STORE命令

同じく，ベクトルLOAD／STORE命令に相当するのが“スト

Stream FIFO Architecture for Vector Processing on the 【Jimpu:】 Processor

Tetsuo HIRONAKA, Kazuaki MURAKAMI, and Shinji TOMITA
Kyushu University

リームLOAD／STORE命令”である．ストリームLOAD／STORE命令は，メモリ上の指定されたアドレスから，指定された増分に従って，指定された回数だけロード／ストアを繰り返す．繰り返し回数および増分はそれぞれ，ストリーム・カウント・レジスタ（SCR）およびストリーム・ストライド・レジスタ（SSR）で指定する．LOAD命令でディスティネーション・レジスタにFIFOが指定されている場合，ベクトル・データを順にロードしていく．また，STORE命令でソース・レジスタにFIFOが指定されている場合，ベクトル・データを順にストアしていく．なお，ソースおよびディスティネーションとして，汎用レジスタ（GR）および浮動小数点レジスタ（FR）を用いることも許される．また，ベース・レジスタにFIFOを指定すると，リスト・ベクトル・アクセスを行う．

3.4 ストリームFIFO方式

ストリームFIFO方式では，FIFOを介してベクトル演算パイプライン間およびメモリ－ベクトル演算パイプライン間のチェイニングを行う．すなわち，2個のストリーム命令が同一FIFOをディスティネーションおよびソース・レジスタに指定すると，一方の命令がFIFOに書込んだベクトル・データを別の命令がソース・オペランドとして用いることが出来る．なお，1個の命令で1個のFIFOをソースおよびディスティネーションとして同時に指定することも可能である．

この結果図1に示すように，FIFOを枝とし，ロード／ストア・パイプラインおよびベクトル演算パイプラインをノードとする演算木が構成される．この演算木はデータフロー・グラフに相当しており，これに従ってデータフロー実行的に演算結果を次々とチェイニングしていく方式を“ストリームFIFO方式”と呼ぶ．

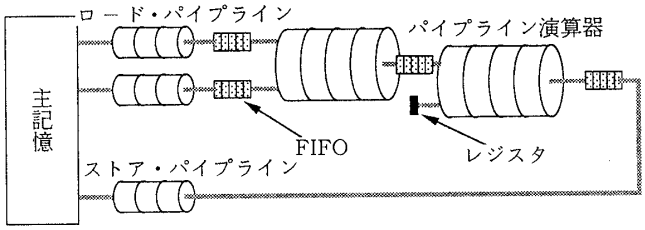

図1．ストリームFIFO方式の概念図

4. ストリームFIFO方式の特長

Cray－1をはじめとする現在のスーパーコンピュータにおいては，大容量のベクトル・レジスタを基礎としたレジスタ－レジスタ演算方式（以後，ベクトル・レジスタ方式と呼ぶ）が主流である．また，CDC CYBER205などでは，メモリ－メモリ演算方式（以後，メモリ方式と呼ぶ）が用いられていた．以下，これら従来のベクトル処理方式と我々の提案するストリームFIFO方式との比較を行う．

4.1 ベクトル長

①ベクトル・レジスタ方式：ベクトル・レジスタ方式では，ベクトル・データを主記憶からベクトル・レジスタにロードしておき，全ての演算をレジスタ－レジスタ間で行ない，そして最終的な結果のみを主記憶へストアする．よって，メモリへのアクセス頻度を最低限に抑え，高速なベクトル・レジスタを用いた高速なベクトル演算が可能である．ただし，「ベクトル長がベクトル・レジスタ長より短いこと」がその条件である．条件を満たさない場合，ベクトル・レジスタ長をセグメント長としてベクトル・データのセグメント化を行い，セグメント単位でベクトル処理を繰り返さなければならない．このときセグメント間で，本来なら必要のないベクトルLOAD／STORE命令を実行しなければならないことから，そのオーバーヘッドが問題となる．つまり，ベクトル・レジスタ方式では，レジスタ長を越えるベクトル長の演算はあまり効率がよくない．

②ストリームFIFO方式とメモリ方式：ストリームFIFO方式では，ベクトル長がいくら長くとも（実際には2^{32}が上限），セグメント化の必要がない．この特長は，メモリ方式と共通の特長である．

4.2 チェイニング

①メモリ方式：チェイニングは不可能ではないが，行われていない．

②ベクトル・レジスタ方式：先行ベクトル命令→後続ベクトル命令間でのごく単純なチェイニングしかできない．よって，ベクトル演算パイプライン間の特殊なチェイニングを要する演算（内積，総和，最大値検索，再帰演算，など）はマクロ演算命令としてしか提供されていない．

③ストリームFIFO方式：ストリームFIFO方式では，先行命令→後続命令間のチェイニングは当然として，さらに，後続命令→先行命令，ストリーム命令→スカラ命令，スカラ命令→ストリーム命令，および，同一命令内でのチェイニングが可能である．なぜなら，チェイニングを行う際の制限が，命令の順序，種類，数ではなく，FIFOに対する単一書込み＆単一読出し（1命令だけが書込み，1命令だけが読出す）という条件だけにあるからである．この条件さえ満足すれば，プログラム中の命令順序とは無関係に，かなり自由に演算木が作れる．よって，回帰演算などの異なるイタレーション間のチェイニングも可能であり，イタレーション間に依存関係がある場合においてもベクトル化可能となる．さらに，ベクトル・レジスタ方式ではマクロ演算命令としてしか実現できない演算も，一般の演算命令のチェイニングで実現できる．

5. おわりに

以上，『新風』プロセッサにおけるベクトル処理方式であるストリームFIFO方式について述べた．現在，ベクトル処理に必要な機能の詳細化を行っている．

参考文献

1）村上ほか：『新風』DTS－edition：SIMP（単一命令流／多重命令パイプライン）方式に基づくデスクトップ・スーパーコンピュータ，情処37全大論文集，4N－1（1988年9月）

2）Wm．A．Wulf：The WM Computer Architecture，ACM SIGARCH Computer Architecture News，vol. 16，no. 1，pp. 70－84，March 1988