計算機アーキテクチャ　92－5
（1992．　1．　23）

『順風』：MSF 型ベクトル・プロセッサ・プロトタイプ
— 演算パイプラインの構成 —

橋本 隆†, 岡崎 恵三†, 弘中 哲夫†, 村上 和彰†, 富田 眞治‡
†: 九州大学 大学院総合理工学研究科
‡: 京都大学 工学部

MSFV (*Multithreaded Streaming/FIFO Vector*) アーキテクチャを提案し，そのプロトタイプ・ベクトル・プロセッサ『順風』を開発している．*MSFV* アーキテクチャでは，FIFO ベクトル・レジスタ，ストリーミング，柔軟なチェイニング機能と並んで，ベクトル命令レベルでのマルチスレッド処理が大きな特長となっている．

『順風』では，実在する実パイプラインとしては，1本の実加減算パイプライン，1本の実乗除算パイプライン，および，2本の実ロード／ストア・パイプラインを備える．一方，ベクトル命令のディスパッチ対象となる仮想パイプラインとしては，8本の仮想演算パイプラインと8本の仮想ロード／ストア・パイプラインを設けている．仮想パイプラインの実パイプラインへディスパッチ方法としては，切替間隔は毎クロック・サイクルと一定かつ固定だが，ディスパッチ間隔は動的に決まる *round-robin* 方式を採用している．

本稿では，マルチスレッド処理を可能とした演算パイプラインについて，そのパイプライン処理過程およびハードウェア構成を述べている．

A Vector-Processor Prototype Based on *MSFV* (*Multithreaded Streaming/FIFO Vector*) Architecture
— Organization of the Arithmetic Pipelines —

Takashi HASHIMOTO†, Keizo OKAZAKI†, Tetsuo HIRONAKA†,
Kazuaki MURAKAMI†, and Shinji TOMITA‡

†: Department of Information Systems
Interdisciplinary Graduate School of Engineering Sciences
Kyushu University
6-1 Kasuga-koen, Kasuga-shi, Fukuoka 816 Japan
‡: Kyoto University

E-mail:{hashimot, keizo, hironaka, murakami}@is.kyushu-u.ac.jp

We have been developing a vector processor prototype based on *MSFV*(*Multithreaded Streaming/FIFO Vector*) architecture. The *MSFV* architecture has several architectural features, such as FIFO vector register, streaming, flexible chaining, and multithreading at vector-instruction level.

The *MSFV*prototype inplements a real ADD/SUB pipeline, a real MUL/DIV pipeline, and two real LOAD/STORE pipelines. It introduces eight virtual arithmetic pipelines and eight LOAD/STORE pipelines, each of which a vector instruction is issued to. Round-robin multithreading is employed for dispatching a virtual pipeline to a real pipeline.

This paper describes the pipeline structure and hardware organization of the the arithmetic pipelines that implement the multithreaded vector execution.

1 はじめに

従来のベクトル演算方式に比べて，より柔軟なベクトル処理およびベクトルースカラ協調処理を可能とする *MSFV* (*Multithreaded Streaming/FIFO Vector*) アーキテクチャを提案し，その試作プロセッサ『順風』の開発を行っている [4, 9]. *MSFV* アーキテクチャは，その名前の通り以下の特長を有する.

① *FIFO* (*F*)：ベクトル・レジスタとしてFIFOレジスタを用いることにより，1個のベクトル命令で一時に処理可能なベクトルの長さを制限しない．すなわちストリップ・マイニング処理が不要となり，ベクトル命令再発行に伴うオーバヘッドを排除できる.

② *Streaming* (*S*)：スカラ命令およびベクトル命令の双方からFIFOレジスタ内のベクトルデータにアクセスできる．これにより，スカラ命令のループでもFIFOレジスタ内のベクトルデータに対する演算が行える．すなわち，小さなオーバヘッドでベクトル—スカラ協調処理が可能となる.

③ *Multithreaded* (*M*)：ベクトル命令レベルでマルチスレッド処理を行う．すなわち，1本のパイプラインは，1個のベクトル命令の実行に占有されるのではなく，複数個のベクトル命令に時分割共有される．このとき，個々のベクトル命令をディスパッチする対象であるパイプラインを仮想パイプライン (*virtual pipeline*) と，また，実在するパイプラインを実パイプライン (*real pipeline*) とそれぞれ呼ぶ．これにより，パイプライン使用率を向上させると同時に，一時に実行可能なベクトル命令の数を増やせる.

さらに，チェイニング機能 [2] に関して，その方向および対象命令を以下のように柔軟なものとしている.

- 先行ベクトル／スカラ命令→後続ベクトル／スカラ命令
- 後続ベクトル／スカラ命令→先行ベクトル／スカラ命令
- ベクトル命令→同一ベクトル命令

この柔軟なチェイニング機能および上記③のマルチスレッド処理により，特殊なマクロ演算命令ないし専用ハードウェアを用いることなく，回帰型演算（総和，内積，最大値／最小値検索，1次回帰，等）のベクトル処理を可能としている．ちなみに，従来のベクトル・プロセッサにおいては，先行ベクトル命令→後続ベクトル命令のチェイニングしか許されない [2].

本稿では，マルチスレッド処理を可能とした『順風』の演算パイプラインの構成について述べる．まず，2章で，*MSFV* アーキテクチャの動作原理，および，マルチスレッド処理について述べる．3章では，『順風』の構成を述べた後，演算パイプラインで実行するベクトル演算命令のうち特徴的な命令を紹介する．そして，4章で，マルチスレッド処理を可能とした演算パイプラインの構成を述べる.

2 MSFV アーキテクチャ

2.1 動作原理

図1の例を用いて，*MSFV* アーキテクチャの動作原理を説明する．図1(a) のループは回帰演算を含んでおり，従来のベクトル・プロセッサではマクロ演算命令と専用ハードウェアによってのみベクトル処理可能である．これに対して，*MSFV* アーキテクチャでは，図1(c) に示す一般のベクトル／スカラ命令のチェイニングにより，次のようにベクトル処理可能である.

① スカラ LOAD 命令 a により，浮動小数点レジスタ FR1 にスカラ X をロードする．スカラ LOAD 命令 b は，FIFO レジスタ F9 をベクトル要素 A(1) で初期化する.

② スカラ MOVE 命令 c でベクトル長を 99 に設定する.

③ ベクトル VLOAD 命令 d, e, f および g により，ベクトル B, C, D および E を FIFO レジスタ F5, F6, F7 および F8 にそれぞれロードする.

④ ベクトル VMUL 命令 h, i および j は，ベクトル B と C，ベクトル D と X，および，ベクトル E と A の乗算をそれぞれ行い，各乗算結果を FIFO レジスタ F3, F4 および F2 にそれぞれ格納する.

⑤ ベクトル VADD 命令 k と l とで，3つの乗算結果 (F3, F4, F2) の加算を行い，ベクトル VADD 命令 l はその加算結果を2つの FIFO レジスタ F0 と F9 に同時に格納する．F0 と F9 の中身は同じ内容であり，F0 はメモリへのストア用に，また，F9 は回帰演算のためベクトル VMUL 命令 j のソース・オペランドとして用いられる.

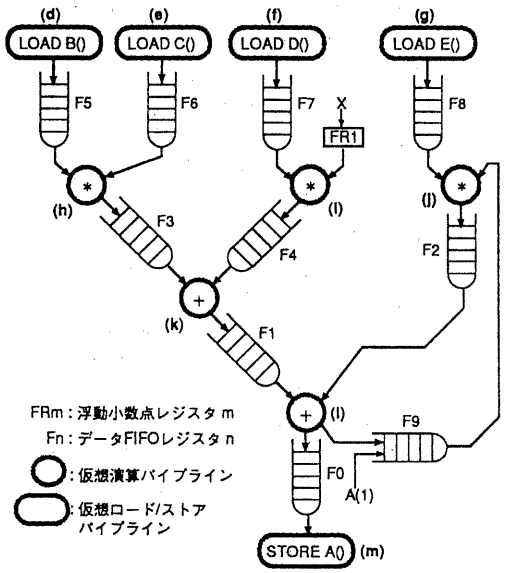
⑥ ベクトル VSTORE 命令 m により，F0 内の演算結果をメモリにストアする.

```
   DO 10 I = 1, 99
      A(I+1) = B(I)*C(I) + D(I)*X + E(I)*A(I)
10 CONTINUE
```

(a) ソース・コード

```
LOAD    FR1     ← X          … a
LOAD    F9      ← A(1)       … b
MOVE    VLR     ← #99        … c
VLOAD   F5      ← B(1)       … d
VLOAD   F6      ← C(1)       … e
VLOAD   F7      ← D(1)       … f
VLOAD   F8      ← E(1)       … g
VMUL    F3      ← F5,F6      … h
VMUL    F4      ← F7,FR1     … i
VMUL    F2      ← F8,F9      … j
VADD    F1      ← F3,F4      … k
VADD    F0,F9   ← F1,F2      … l
VSTORE  A(2)    ← F0         … m
```

(b) オブジェクト・コード



(c) チェイニング・グラフ

図 1: *MSFV* アーキテクチャの動作原理

2.2 マルチスレッド処理

図1の例では，10個のベクトル命令 (d～m) がチェイニングされて同時に実行可能である．しかし，これを実現しようとすると，10本ものパイプライン (5本のロード／ストア・パイプライン，3本の乗算パイプライン，2本の加算パイプライン) が必要となる．また，ループが回帰演算を含んでいるので，データ依存による遅延でパイプラインにバブルが生じ，パイプラインの使用効率が低下する．

MSFV アーキテクチャでは，これらの問題をマルチスレッド処理により解決している．たとえば，『順風』(図2参照) では3.1節で述べるように，1本の加減算パイプライン (RASP: *Real ADD/SUB Pipeline*)，1本の乗除算パイプライン (RMDP: *Real MUL/DIV Pipeline*)，および，2本のロード／ストア・パイプライン (RLSP: *Real LOAD/STORE Pipeline*) しか備えていない．そのかわり，8本の仮想演算パイプライン (VAP: *Virtual Arithmetic Pipeline*) と8本の仮想ロード／ストア・パイプライン (VLSP: *Virtual LOAD/STORE Pipeline*) を設けている．そして，ベクトル命令を実パイプラインではなく仮想パイプラインにディスパッチするようにした．

個々の仮想パイプラインの実体は，仮想パイプライン・コンテキスト・レジスタ (VPCR: *Virtual Pipeline Context Register*) と呼ぶ制御レジスタ[1]である (図2参照)．VPCR は，対応する仮想パイプラインにディスパッチされたベクトル命令のコンテキスト (オペレーション指示子 (OP)，指定ベクトル長 (VL)，マスク FIFO レジスタ指示子 (MFR)，ソース／デスティネーション・レジスタ指示子 (SDRS)，ベクトル要素カウント (CVC)) を保持する [9]．

仮想パイプラインを実パイプラインにディスパッチする方法には，次の3つの方法がある．

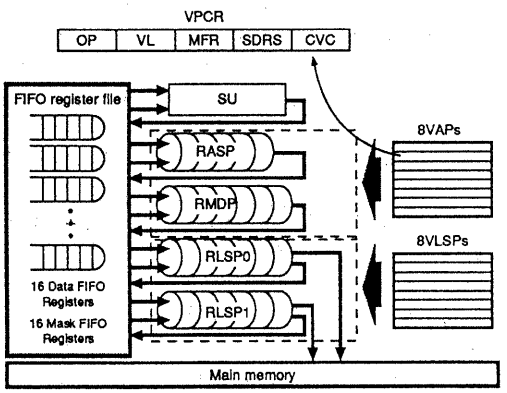
- *Periodic*：切替え間隔およびディスパッチ間隔ともに一定かつ固定 [8]．たとえば，毎クロック・サイクル，仮想パイプラインを切替え (*cycle-by-cycle interleaving*)，同一仮想パイプラインが実パイプラインにディスパッチされる間隔もコンパイラあるいはアーキテクチャにより静的に定まる．
- *Round-robin*：切替え間隔は一定かつ固定だが，ディスパッチ間隔は可変で動的に決まる．『順風』はこの方式を採用しており，トークン・リング／タイム・スライス方式と呼ぶ実パイプライン割当てアルゴリズムを用いている [5, 6]．
- *Forced*：切替え間隔およびディスパッチ間隔ともに可変で動的に決まる．すなわち，仮想パイプラインの処理を阻止する要因が発生するまで，当該仮想パイプラインの処理を継続する．

3 『順風』の概要

3.1 全体構成

図2に『順風』の全体構成を示す．スカラ・ユニットが命令キャッシュから命令をフェッチする．フェッチした命令をデコードした結果，スカラ命

[1] これまで，SS (Streaming Station)[7] と呼んでいた．



CVC : ベクトル要素カウント
MFR : マスク FIFO レジスタ指示子
OP : オペレーション指示子
RASP : 実加減算パイプライン
RLSP : 実ロード／ストア・パイプライン
RMDP: 実乗除算パイプライン
SDRS : ソース／デスティネーション・レジスタ指示子
SU : スカラ・ユニット
VAP : 仮想演算パイプライン
VL : 指定ベクトル長
VLSP : 仮想ロード／ストア・パイプライン
VPCR: 仮想パイプライン制御レジスタ

図 2: 『順風』の全体構成

令であればスカラ・ユニットに当該命令を発行し，ベクトル命令であればベクトル・ユニット内の仮想パイプライン（の実体である VPCR）に対して当該命令を発行する．ベクトル・ユニットは次に示す主要ユニットから構成される [6].

(1) 仮想パイプライン・コンテキスト・レジスタ

8 本の仮想演算パイプライン (VAP) と 8 本の仮想ロード／ストア・パイプライン (VLSP) に対応して，16 個の仮想パイプライン・コンテキスト・レジスタ (VPCR) を備える．

(2) 実ロード／ストア・パイプライン

同一構成／機能の実ロード／ストア・パイプライン (RLSP) を 2 本備え，メモリー FIFO レジスタ間でベクトル・データのロード／ストアを行う．仮想ロード／ストア・パイプラインと実ロード／ストア・パイプラインとの対応関係は動的に決まるが，一旦対応関係が決まると当該ロード／ストア命令の実行終了までそれは変わらない．これにより，ベクトル・ロードの際，ベクトル要素の FIFO レジスタへの格納順序が矛盾しないことを保証する．メモリ・アクセスのレイテンシは 6 クロック・サイクルである．

(3) 実演算パイプライン

実加減算パイプライン (RASP) および実乗除算パイプライン (RMDP) をそれぞれ 1 本ずつ備える．仮想パイプラインと実演算パイプラインとの対応関係は動的に決まるが，ベクトル命令の種類により表 1 のように割り当てられる．パイプライン・レイテンシは，実加減算パイプラインが 5 サイクル，実乗除算パイプラインが 6 サイクルである．

(4)FIFO レジスタ・ファイル

以下のデータ FIFO レジスタおよびマスク FIFO レジスタをそれぞれ 16 本ずつ備える．

- **データ FIFO レジスタ**：浮動小数点データ（単精度／倍精度）および整数データを格納する．レジスタ幅はデータ 64 ビットおよびコンディション・コード 5 ビットの計 69 ビットで，レジスタ長は 32 である．読出しポートおよび書込みポートは，スカラ・ユニットおよび 5 本の実パイプラインが各々占有できるよう，それぞれ 10 ポートと 5 ポート構成となっている．3 ポート（読出しポート 1，書込みポート 1，読出し書込みポート 1）レジスタ・ファイルの TI 社製 SN74ACT8831 を用いて実現している．
- **マスク FIFO レジスタ**：レジスタ幅はマスク 1 ビットおよびコンディション・コード 1 ビットの計 2 ビットで，レジスタ長は 32 である．読出しポートは，16 個の仮想パイプライン・コンテキスト・レジスタが各々占有できるよう 16 ポート構成となっている．また，書込みポートは，マスクを生成する実加減算パイプラインおよび 2 本の実ロード／ストア・パイプラインが各々占有できるよう 3 ポート構成となっている．FIFO メモリの TI 社製 SN74ALS2232 を用いて実現している．

なお，コンディション・コードとして，ベクトルの最終要素であることを示す EOS (End Of Stream) を設けている．EOS には，対応するデータの有効／無効に応じて，EOS_V と EOS_I の 2 種類がある．

3.2 ベクトル演算命令

表 1 に，『順風』におけるベクトル演算命令と実加減算パイプラインおよび実乗除算パイプラインとの対応関係を示す．以下，特徴的なベクトル演算命令について簡単に説明する．

- コピー命令 (VCOPY)：ベクトル・レジスタが破壊読出しの FIFO レジスタであるので，データの再利用性がない．よって，データを再利用する場合は，ディスティネーション・レジスタの 2 重指定（ダブル・デスティ

表 1: ベクトル演算命令と実演算パイプラインとの対応関係

実加減算パイプライン (RASP)	実乗除算パイプライン (RMDP)
加算 (VADD)	乗算 (VMUL)
減算 (VSUB)	除算 (VDIV)†
論理演算 (VAND 等)	平方根 (VSQRT)†
シフト (VSHIFT 等)	分配 (VSPLIT)
型変換 (VFLOAT 等)	等差数列生成 (VGENAS)
コピー (VCOPY)	コピー (VCOPY)
比較 (VCOMP)	
最大値/最小値 (VMAX/VMIN)	
ベクトル実行停止 (VWHILE)	

†: 非パイプライン化命令

ネーション指定) [1] あるいは本 VCOPY 命令を用いる．本命令は，1 個のソース・レジスタ内のデータを 2 個のデスティネーション・レジスタにコピーする．

- 最大値／最小値命令 (VMAX/VMIN)：2 個のソース・オペランドを比較して，値が大きい／小さい方のデータをデスティネーション・レジスタに格納する．
- ベクトル分配命令 (VSPLIT)：マスクベクトルの真偽に従って，1 個のベクトルを 2 個のベクトルに分配する．本命令は，「IF 文を含む DO ループ」の 1 ベクトル化技法であるベクトル分配ー併合 (*vector split-merge*: ベクトル収集ー拡散の変形) において用いる [3]．
- ベクトル実行停止命令 (VWHILE)：基本動作は VCOMP 命令と同様だが，比較結果が一度でも偽となると EOS_I を出力して実行を終了する．本命令は，「IF 文を含む DO ループ」の 1 ベクトル化技法であるベクトル命令実行停止 (*vector-operation termination*) において用いる [3]．

4 演算パイプライン

図 3 に，『順風』の演算パイプラインの構成を示す．演算パイプラインは，次の 6 ステージから成る [7]．パイプライン・ピッチは 60ns である．

① **FCC** (*FIFO Condition Check*) ステージ：個々の仮想演算パイプラインに対して，対応する仮想パイプライン・コンテキスト・レジスタ (VPCR) に従って，ソースおよびデスティネーション FIFO レジスタの状態，マスク FIFO レジスタの状態，および，ベクトル要素カウントを調べる．

② **SCH** (*SCHedule*) ステージ：個々の仮想演算パイプラインに対して，実演算パイプラインへディスパッチ可能か否か判断する．もし，ディスパッチ可能なら，オペレーションに応じた実演算パイプラインの割当要求を行う．一方，各実演算パイプラインは，複数の割当要求の中から 1 本の仮想演算パイプラインを選択しディスパッチを許可する．

③ **FR** (*FIFO Read*) ステージ：VPCR のソース・レジスタ指示子 (SDRS) で指定される FIFO レジスタからベクトル要素を読み出す．

④ **EX** (*EXecute*) ステージ：VPCR のオペレーション指示子 (OP) に従って演算を行う．実加減算パイプライン (RASP) は 2 ステージ構成，実乗除算パイプライン (RMDP) は 3 ステージ構成となっている．

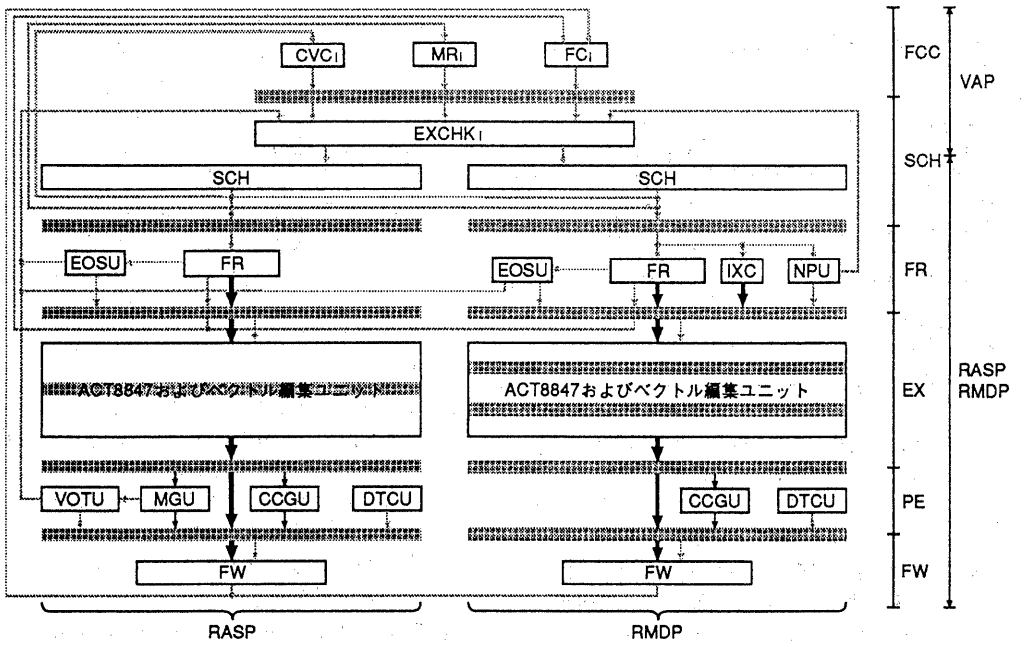
⑤ **PE** (*Post Execute*) ステージ：EX ステージの後処理を行う．

⑥ **FW** (*FIFO Write*) ステージ：VPCR のデスティネーション・レジスタ指示子 (SDRS) で指定される FIFO レジスタに演算結果を書き込む．

4.1 FCC ステージ

FCC (*FIFO Condition Check*) ステージは，以下のユニットから構成される．これらは，仮想演算パイプライン対応に設けられる．

- CVC (*Current Vector Count*)：仮想パイプライン・コンテキスト・レジスタ (VPCR) の 1 フィールドで，ベクトル演算命令の実行回数を計数・保持する．仮想パイプラインが実パイプラインにディスパッチされる毎に+1 更新する．VPCR の指定ベクトル長 (VL) フィールドと常に比較を行い，所定のベクトル長に達したらベクトル演算命令の実行完了を EXCHK に通知する．
- MR (*Mask Read unit*)：マスク FIFO レジスタからマスク・データを先読みする．このとき，EOS コンディション・コードを検出したらベクトル演算命令の実行完了を EXCHK に通知する．
- FC (*FIFO Condition unit*)：ソースおよびデスティネーション FIFO レジスタ，および，マスク FIFO レジスタの状態 (アンダーフロー／オーバーフロー) を先見する．



ラッチ　制御の流れ　データの流れ

CCGU	: コンディション・コード生成	IXC	: インデックス生成
CVC_i	: ベクトル要素カウント	MGU	: マスク生成
DTCU	: デスティネーション・タグ制御	MR_i	: マスク FIFO レジスタ読出し
EOSU	: EOS 後処理	NPU	: 非パイプライン化命令制御
EX	: EX ステージ	PE	: PE ステージ
$EXCHK_i$	: 実行条件判定	RASP	: 実加減算パイプライン
FC_i	: FIFO レジスタ状態チェック	RMDP	: 実乗除算パイプライン
FCC	: FCC ステージ	SCH	: 実パイプライン割当
FR	: ソース FIFO レジスタ読出し	VAP_i	: 仮想演算パイプライン
FW	: デスティネーション FIFO レジスタ書込み	VOTU	: ベクトル命令実行停止検出
i	: 仮想演算パイプライン識別子 ($0 \leq i \leq 15$)		:

図 3: 演算パイプラインの構成

4.2 SCH ステージ

SCH (*SCHedule*) ステージは，以下のユニットから構成される．

- EXCHK (*EXecution CHecK unit*)：仮想演算パイプライン対応に設けられる．対応する CVC，MR，FC，および，実演算パイプラインの FR からの制御情報に基づいて，当該仮想演算パイプラインがディスパッチ可能か否か判定する．ディスパッチ可能条件は次の通りである．
  - CVC 関係：ベクトル演算命令の実行回数が所定のベクトル長に達していない．
  - MR 関係：マスク付きベクトル演算命令の場合，マスク FIFO レジスタが空でなく，かつ，対応するマスク値がオフ（実行可）である．
  - FC 関係：ソース FIFO レジスタが空でなく，かつ，デスティネーション FIFO レジスタが満杯でない．
  - MR および FR 関係：EOS コンディション・コードが検出されていない．

  もしディスパッチ実行可能と判定したら，SCH に対して実演算パイプラインの割当要求を行う．なお，マスク値がオンの場合は，対応するベクトル要素に対する演算は行わず，単に CVC を更新するだけである．また，EOS コンディション・コードあるいは CVC=VL を検出したら，ベクトル演算命令の実行を完了とする．
- SCH (*SCHedule unit*)：実演算パイプライン対応に設けられる．各 EXCHK から出され

た複数の実演算パイプライン割当要求を調停し，1つの仮想演算パイプラインに対して実演算パイプラインへのディスパッチを許可する．そして，当該仮想演算パイプラインに対して，CVCの更新，および，MRによる次マスク・データの読出しを指示する．

4.3 FRステージ

FR (*FIFO Read*) ステージは，以下のユニットから構成される．

- FR (*FIFO Read unit*)：ディスパッチされた仮想演算パイプライン（コンテキスト・レジスタ:VPCR）のソース・レジスタ指示子(SDRS)で指定されるFIFOレジスタからベクトル要素を読み出す．このとき，EOSコンディション・コードを検出したらベクトル演算命令の実行完了をEXCHKおよびEOSUに通知する．
- EOSU (*End Of Stream Unit*)：EOSコンディション・コードを検出した際に必要となる以下の処理を行う．
  - 2種類のEOSコンディション・コードEOS_IおよびEOS_Vに応じた処置[1]
  - 命令実行完了通知の遅れにより，完了したはずの仮想演算パイプラインが誤って実演算パイプラインにディスパッチされる場合がある．この問題に対処するため，EOSを検出した仮想演算パイプラインを登録しておき，当該仮想演算パイプラインが誤ってディスパッチされた際にはその実行を無効化するようにしている．
- IXC (*IndeX Counter*)：実乗除算パイプラインにのみ備える．等差数列生成命令(VGENAS)を実行する．
- NPU (*Non-Pipelined instruction Unit*)：実乗除算パイプラインにのみ備える．非パイプライン化命令である除算命令(VDIV)および平方根命令(VSQRT)の実行は，複数サイクルにわたって実乗除算パイプラインを占有する．したがって，その間他の仮想演算パイプラインが実乗除算パイプラインにディスパッチされないよう制御を行う．

4.4 EXステージ

ディスパッチされた仮想演算パイプライン（コンテキスト・レジスタ:VPCR）のオペレーション指示子(OP)に従って演算を行う．EX (*EXecute*) ステージは，次の2つのユニットから構成される．いずれのユニットも，実加減算パイプラインは2ステージ構成，実乗除算パイプラインは3ステージ構成となっている．

- 演算器：実加減算パイプラインおよび実乗除算パイプラインともに，TI社のSN74ACT8847を用いる．大部分の演算機能は，この演算器で実現する．
- ベクトル編集ユニット：上記SN74ACT8847では実現できない命令（コピー命令(VCOPY)，最大値／最小値(VMAX/VMIN)，ベクトル分配命令(VSPLIT)）を実行する．

4.5 PEステージ

PE (*Post Execute*) ステージは，以下のユニットから構成される．

- CCGU (*Condition-Code Generation Unit*)：算術演算命令および論理演算命令を実行した際，5ビットのコンディション・コードを生成する．
- MGU (*Mask Generation Unit*)：実加減算パイプラインに設ける．比較命令(VCOMP)を実行して，マスクを生成する．SN74ACT8847から得られた比較結果と比較条件とから，マスク値を決定する．
- VOTU (*Vector-Operation Termination Unit*)：ベクトル実行停止命令(VWHILE)の実行において，条件不成立（偽）検出時に必要な処理を行う．これは，FRステージのEOSUと同様の処理である．
- DTCU (*Destination Tag Control Unit*)：ディスパッチされた仮想演算パイプライン（コンテキスト・レジスタ:VPCR）のデスティネーション・レジスタ指示子(SDRS)に従って，演算結果を格納するデスティネーションFIFOレジスタ番号を生成する．ダブル・デスティネーション指定時は，デスティネーションFIFOレジスタ番号の最下位ビットだけが異なる2個のレジスタ番号を生成する．また，実乗除算パイプラインにおいては，ベクトル分配命令(VSPLIT)実行の際，マスク値に従ってデスティネーションFIFOレジスタ番号を次のように切り替えながら生成する．
  - マスクが0の場合：デスティネーション・レジスタ指示子(SDRS)通りのレジスタ番号

– マスクが1の場合：デスティネーション・レジスタ指示子(SDRS)で指定されているレジスタ番号の最下位ビットを反転させたもの

4.6 FW ステージ

FW (*FIFO Write*) ステージでは，実演算パイプラインからの出力結果をデスティネーションFIFO レジスタに書き込む．

5 おわりに

以上，*MSFV* アーキテクチャに基づくプロトタイプ・ベクトル・プロセッサ『順風』の演算パイプラインの構成について述べた．*MSFV* アーキテクチャでは，FIFO ベクトル・レジスタ，ストリーミング，柔軟なチェイニング機能とならんで，ベクトル命令レベルでのマルチスレッド処理が大きな特長となっている．

『順風』では，実在する実パイプラインとして，1本の実加減算パイプライン (RASP)，1本の実乗除算パイプライン (RMDP)，および，2本の実ロード／ストア・パイプライン (RLSP) を備える．一方，ベクトル命令のディスパッチ対象となる仮想パイプラインとしては，8本の仮想演算パイプライン (VAP) と8本の仮想ロード／ストア・パイプライン (VLSP) を設けている．仮想パイプラインの実体は，仮想パイプライン・コンテキスト・レジスタ (VPCR) と呼ぶ制御レジスタである．仮想パイプラインの実パイプラインへのディスパッチ方法としては，切替間隔は毎クロック・サイクルと一定かつ固定だが，ディスパッチ間隔は動的に決まる *round-robin* 方式を採用している．マルチスレッド処理を可能とした演算パイプラインのパイプライン・ステージ数は，7（加減算パイプライン）および8（乗除算パイプライン）となった．このうち，2ステージがマルチスレッド処理を実装するために増えたステージである．

マルチスレッド処理を実装するに当っては，まだ検討すべき項目が多くある．たとえば，仮想パイプライン数と実パイプライン数の適切な比，仮想パイプラインと実パイプラインとの対応関係，仮想パイプラインの実パイプラインへのディスパッチ方法，等がある．今後は，『順風』の開発と並行して，これらの項目に関して評価を行っていく予定である．

謝辞

日頃ご討論頂く九州大学大学院総合理工学研究科 安浦寛人教授ならびに安浦研究室の諸氏に感謝致します．

参考文献

[1] 岡崎, 弘中, 村上, 富田, “『順風』:ストリーム FIFO 方式に基づくシングルチップ・ベクトルプロセッサ・プロトタイプ —「IF 文を含む DO ループ」への対処法 —,” 情処研報, ARC-85-3, 1990 年 11 月.

[2] 長島, 稲上, 阿部, 河辺, “動的チェイニングによるベクトルプロセッサの実効性能の向上,” 信学論, vol.J74-D-I, no.12, pp.836–845, 1991 年 12 月.

[3] 橋本, 岡崎, 弘中, 村上, 富田, “『順風』:ストリーム FIFO 方式に基づくシングルチップ・ベクトルプロセッサ・プロトタイプ — マクロ演算，ベクトルースカラ協調処理およびベクトル命令実行停止機能の評価 —,” 並列処理シンポジウム JSPP'91 論文集, pp.101–108, 1991 年 5 月.

[4] 弘中, 久我, 村上, 富田, “ストリーム FIFO 方式に基づくベクトル・プロセッサ『順風』,” 信学技報, CPSY89-39, 1989 年 8 月.

[5] 弘中, 岡崎, 村上, 富田, “ストリーム FIFO 方式に基づくベクトル・プロセッサ『順風』の構成と性能評価,” 並列処理シンポジウム JSPP'90 論文集, pp.201–208, 1990 年 5 月.

[6] 弘中, 岡崎, 村上, 富田, “ストリーム FIFO 方式に基づくベクトル・プロセッサ『順風』— ストリーム FIFO 方式および仮想パイプライン方式の実現法に関する評価 —,” 情処論, vol.32, no.7, pp.828–837, 1991 年 7 月.

[7] 弘中, 橋本, 岡崎, 村上, 富田, “『順風』:ストリーム FIFO 方式に基づくシングルチップ・ベクトルプロセッサ・プロトタイプ — 仮想パイプラインの実現法 —,” 情処研報, ARC-89-6, 1991 年 7 月.

[8] Chiueh, T.-C., “Multi-Threaded Vectorization,” *Proc. 18th Int'l. Symp. Computer Architecture*, pp.352–361, May 1991.

[9] Hironaka, T., Hashimoto, T, Okazaki, K., Murakami, K., and Tomita, S., “A Single-Chip Vector-Processor Prototype Based on Multithreading Streaming/FIFO Vector (MSFV) Architecture,” *Proc. 1991 Int'l. Symp. Supercomputing*, pp.77–86, Nov. 1991.