『順風』: MSF (*Multithreaded Streaming/FIFO*) 型ベクトル・プロセッサ・プロトタイプ

——MSFV アーキテクチャに関する評価——

橋本 隆[†] 弘中哲夫[†] 岡崎恵三[†*]
村上和彰[†] 權 五鳳[††] 富田眞治[†††]

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* アーキテクチャおよび『順風』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，ベクトル命令レベルでのマルチスレッド処理，ベクトル命令実行停止機能，といった特長を有する．本論文では，*MSFV* アーキテクチャおよびそのプロトタイプ・ベクトル・プロセッサ『順風』について述べている．さらに，*MSFV* アーキテクチャ自身の特長である FIFO ベクトル・レジスタ (*F*)，マルチスレッド処理 (*M*)，ストリーミング (*S*) について，その性能をシミュレーションにより評価している．マルチスレッド処理によりパイプラインの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数を増やすことが可能となる．その結果，従来型のベクトル・プロセッサと比べて，LFK (Livermore Fortran Kernels) 1～14 で最低 0.0%～ 相乗平均 43.6%～ 最高 333.8% の性能向上が得られた．また，FIFO ベクトル・レジスタによりストリップ・マイニング・オーバヘッドを排除でき，最低 0.0%～ 相乗平均 10.7%～ 最高 24.2% の性能向上が得られた．さらに，ストリーミングにより LFK 13，LFK 14 に対してそれぞれ 32.9%，29.4% の性能向上が得られた．総合的には，『順風』は従来型のベクトル・プロセッサと比べて，最低 16.3%～ 相乗平均 59.0%～ 最高 334.9% 性能が良いことが判明した．

A Vector-Processor Prototype Based on MSFV (*Multithreaded Streaming/FIFO Vector*) Architecture

—Evaluation of MSFV Architecture—

Takashi Hashimoto,[†] Tetsuo Hironaka,[†] Keizo Okazaki,[†*]
Kazuaki Murakami,[†] Oubong Gwun[††] and Shinji Tomita[†††]

We have been developing a prototype vector processor based on the *MSFV* (*Multithreaded Streaming/FIFO Vector*) architecture. The *MSFV* architecture has several unique features, such as *FIFO vector-register*, *streaming*, flexible chaining capability, *multithreading at vector instruction level*, and vector-operation termination. This paper tries to identify the strength and weakness of those architectural features. The results for Livermore Fortran Kernels are reported in terms of *normalized FLOPC* (*floating-point operations per clock cycle*). Multithreading at vector instruction level improve the utilization of a pipeline, and increase a number of vector instructions which can be executed concurrently. By the use of FIFO vector registers, stripmining is no longer neccessary. And with the streaming capability, scalar instructions can compute on vector operands in FIFO vector registers as well as vector instructions. Consequently, multithreading improve performance by 44% (geometric mean), and FIFO vector-registers do by 11% (geometric mean). Furthermore, streaming enhance performance by 30% on the LFK 13 and LFK 14. The paper concludes that the *MSFV* prototype is 1.59 times (geometric mean) faster than a hypothetical conventional vector processor.

† 九州大学大学院総合理工学研究科情報システム学専攻
Department of Information Systems, Interdisciplinary Graduate School of Engineering Sciences, Kyushu University
†† 九州大学工学部情報工学科
Department of Engineering, Faculty of Information Sciences, Kyushu University
††† 京都大学工学部情報工学科
Department of Engineering, Faculty of Information Sciences, Kyoto University
* 現在 日産自動車(株)
Presently with Nissan Motor Co., Ltd.

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-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[1),7)]．*MSFV* アーキテクチャは，その名前のとおり以下の特長を有する．

①**FIFO ベクトル・レジスタ** (*F*: *FIFO*): ベクトル・レジスタとしてリングFIFOバッファ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**ストリーミング** (*S*: *Streaming*): 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**マルチスレッド処理** (*M*: *Multithreaded*): 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[10]に関して，その方向および対象命令を柔軟なものとしている．上記③のマルチスレッド処理および柔軟なチェイニング機能により，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．

プロセッサのハードウェア設計と並行して，ソフトウェア・シミュレーションによる性能評価を行ってきた．これまでに，評価項目として，実パイプライン割当アルゴリズム[2),5)]，「IF 文を含む DO ループ」への対処法[3),4),7)]，メモリ・バンド幅[9)]，従来のスーパコンピュータとの比較[9)]，等について評価した結果を報告した．そこで本稿では，*MSFV* アーキテクチャ自身の特長である FIFO ベクトル・レジスタ(*F*)，マルチスレッド処理(*M*)，ストリーミング(*S*)について，シミュレーションにより評価する．

まず，2章で，*MSFV* アーキテクチャの動作原理，マルチスレッド処理，および，ストリーミングについて述べる．3章では，『順風』のハードウェア構成およびパイプライン処理過程を概観する．そして，4章および5章で，シミュレーションによる評価を行い，その結果について考察する．

2. MSFV アーキテクチャ

2.1 動作原理

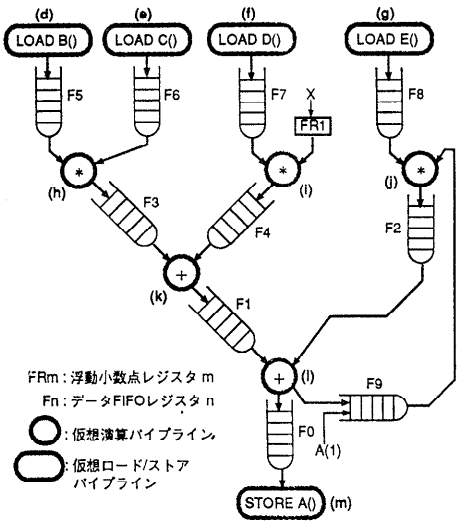
MSFV アーキテクチャの基本動作原理を**図1**の例を用いて説明する．これは，静的データ駆動 (*static dataflow*) 方式の動作原理に似ている．まず，図1(a)のループは回帰演算を含んでおり，従来のベクトル・プロセッサではマクロ演算命令と専用ハードウェアによってのみベクトル処理可能である．一方，*MSFV* アーキテクチャでは，図1(c)に示す一般の

```
      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
Fig. 1 The principle of *MSFV* architecture.
(a) Source code.
(b) Object code.
(c) Chaining graph.

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としている．

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．

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3参照) 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である (図3参照)．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[7]．

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（実パイプライン割当アルゴリズム）には，次の3方式がある．『順風』では，round-robin 方式を採用している．

①*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[11]．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決まる．

②*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アルゴリズムを用いている[2),5)]．

③*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．

2.3 ストリーミング

図2に，後でベンチマーク・プログラムとして用いる LFK (*Livermore Fortran Kernels*) 13 をストリーミングの適用例として示す．

LFK 13 の配列Hは，その添字が再び配列EおよびFで与えられる間接添字の形になっている．したがって，配列Hに関するデータ依存関係の有無がコンパイル時に断定できないので，LFK 13 の元の DO ループ 13 全体をベクトル化することはできない．そこで，図

```
      DO 13 IP= 1,N
         I1= P(1,IP)
         J1= P(2,IP)
         I1= 1 + MOD2N(I1,64)
         J1= 1 + MOD2N(J1,64)
         P(3,IP)= P(3,IP) + B(I1,J1)
         P(4,IP)= P(4,IP) + C(I1,J1)
         P(1,IP)= P(1,IP) + P(3,IP)
         P(2,IP)= P(2,IP) + P(4,IP)
         I2= P(1,IP)
         J2= P(2,IP)
         I2= MOD2N(I2,64)
         J2= MOD2N(J2,64)
         P(1,IP)= P(1,IP) + Y(I2+32)
         P(2,IP)= P(2,IP) + Z(J2+32)
         I2= I2 + E(I2+32)
         J2= J2 + F(J2+32)
         H(I2,J2)= H(I2,J2) + 1.0
13    CONTINUE
        ↓ ループ分割およびスカラ・エクスパンション
      DO 13 IP = 1,N
         TI11(IP)= P(1,IP)
         TJ11(IP)= P(2,IP)
         TI12(IP)= 1 + MOD2N(TI11(IP),64)
         TJ12(IP)= 1 + MOD2N(TJ11(IP),64)
         P(3,IP) = P(3,IP) + B(TI12(IP),TJ12(IP))
         P(4,IP) = P(4,IP) + C(TI12(IP),TJ12(IP))
         P(1,IP) = P(1,IP) + P(3,IP)
         P(2,IP) = P(2,IP) + P(4,IP)
         TI21(IP)= P(1,IP)
         TJ21(IP)= P(2,IP)
         TI22(IP)= MOD2N(TI21(IP),64)
         TJ22(IP)= MOD2N(TJ21(IP),64)
         P(1,IP) = P(1,IP) + Y(TI22(IP)+32)
         P(2,IP) = P(2,IP) + Z(TJ22(IP)+32)
         TI23(IP)= TI22(IP) + E(TI22(IP)+32)
         TJ23(IP)= TJ22(IP) + F(TJ22(IP)+32)
13    CONTINUE
      DO 131 IP = 1,N
         H(TI23(IP),TJ23(IP))
                 = H(TI23(IP),TJ23(IP)) + 1.0
131   CONTINUE
```

図 2 LFK 13 (2-D Particle Cell)
Fig. 2 LFK 13 (2-D Particle Cell).

2の下部に示すように，DO ループ 13 と DO ループ 131 の2つにループ分割する．さらに，スカラ・エクスパンションも施す．この結果，DO ループ 131 はベクトル化不可能なままだが，DO ループ 13 はベクトル化可能となった．

ここで，DO ループ 131 は DO ループ 13 に対して，配列 TI 23 および TJ 23 に関してフロー依存関係にある．ここに，ストリーミングを適用する．すなわち，従来のベクトル・プロセッサでは，配列 TI 23 および TJ 23 をベクトル・ストア命令でいったんメモリにストアした後，DO ループ 131 中のスカラ・ロード命令で配列要素 TI 23 (IP) および TJ 23 (IP) を再ロードしなければならなかった．これに対して，ストリーミングを用いれば，DO ループ 131 中のスカラ命令は，FIFO レジスタを介して配列 TI 23 および TJ 23 に直接アクセスできるようになる．これにより，ロード／ストア回数が削減できるだけでなく，ベクトル・ユニットとスカラ・ユニットとの間のオーバラップ処理が効率的に行えるようになる．

3. 『順風』の概要

3.1 ハードウェア構成

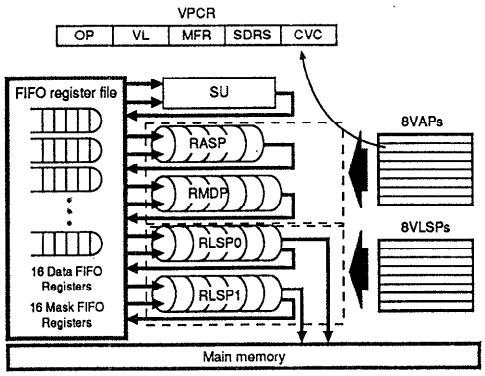
図 3 に『順風』の全体構成を示す．スカラ・ユニットが命令キャッシュから命令をフェッチする．フェッチした命令をデコードした結果，スカラ命令であればスカラ・ユニットに当該命令を発行し，ベクトル命令であればベクトル・ユニット内の仮想パイプライン (の実体である VPCR) に対して当該命令を発行する．ベクトル・ユニットは次に示す主要ユニットから構成される[5]．

(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-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，いったん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



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 3 『順風』の全体構成
Fig. 3 Structure of 【dʒúmpu:】.

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：TI 社製の SN 74 ACT 8847 を使用）および実乗除算パイプライン（RMDP：TI 社製の SN 74 ACT 8847 を使用）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割当先の実演算パイプラインが定まっている[8]．パイプライン・レイテンシは，実加減算パイプラインが5クロック・サイクル，実乗除算パイプラインが6クロック・サイクルである．演算スループットは，いずれも1倍精度演算／クロック・サイクル（除算を除く）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4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 SN 74 ACT-8831 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 SN 74 ALS 2232 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 パイプライン処理過程

ベクトル・ユニットにおけるパイプライン処理過程は，以下のステージから構成されている[6),8)]．マルチスレッド処理の導入によりクロック・サイクル時間が大きくならないように，実パイプライン割当アルゴリズムを以下のステージ①と②とでパイプライン処理している．パイプライン・ピッチは採用した演算器（TI 社製 SN 74 ACT 8847）の動作周波数の関係で 60 ns としている．

①**FCC** (*FIFO Condition Check*) ステージ：個々の仮想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，各実パイプラインは，複数の割当要求の中から1本の仮想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，演算およびロード/ストアを行う．実加減算パイプライン（RASP）は3ステージ構成，実乗除算パイプライン（RMDP）は4ステージ構成，実ロード／ストア・パイプライン（RLSP）は4ステージ構成となっている．

⑤**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 レジスタおよびマスク FIFO レジスタに，演算結果ないしロード結果を書き込む．

上記のステージのうち，SCH と FR の2ステージがマルチスレッド処理を採用したことにより増加したものである．

4. 評価方法

MSFV アーキテクチャ自身の特長である FIFO ベクトル・レジスタ（*F*），マルチスレッド処理（*M*），ストリーミング（*S*）について，ソフトウェア・シミュレータを用いて評価を行った．

4.1 シミュレータ

本シミュレータは，3.2 節で述べた『順風』のパイプライン処理過程を忠実にモデル化している．これにより，毎クロック・サイクルのプロセッサ状態を忠実にシミュレーション可能である．さらに，本シミュレータは，設計上の各種選択肢が性能に与える影響を調べるために，ハードウェア資源数，パイプライン・レイテンシ，メモリ・アクセス・タイム，等をパラメータ化している．**表1**に，『順風』設計の際に行った選択肢決定の結果を示す．

4.2 ベンチマーク・プログラム

ベンチマーク・プログラムには，LFK (*Livermore Fortran Kernels*) 24 の中から LFK 1～LFK 14 を用いた．シミュレータへの入力は，『順風』の機械命令（ベクトル命令およびスカラ命令）である．『順風』のオブジェクト・コードは，ハンド・コンパイルにより得た．

表 1 『順風』の仕様
Table 1 The specifications of 【dʒũmpu:】.

RASP	本数	1
	開始間隔サイクル数	1
	所要サイクル数	5
	演算スループット	1倍精度演算／サイクル
RMDP	本数	1
	開始間隔サイクル数	1 (除算以外) 15 (除算)
	所要サイクル数	6 (除算以外) 15 (除算)
	演算スループット	1倍精度演算／サイクル (除算以外)
RLSP	本数	2
	開始間隔サイクル数	1
	所要サイクル数	6
	バンド幅／RLSP	8バイト／サイクル
ベクトル・レジスタ (データ FIFO)	個数	16
	語長	64 ビット
	レジスタ長	32語
	バンド幅／レジスタ	1語／サイクル
スカラ・レジスタ	汎用レジスタ	32 個
	浮動小数点レジスタ	32 個
制御レジスタ	仮想パイプライン制御 (VPCR)	8 個 (VAP) 8 個 (VLSP)
	ベクトル長 (VLR)	1 個
メモリ	ポート数	2
	バンク数	8
	ポート・サイズ	8バイト
	インタリーブ方式	8バイト・インタリービング
	バンク・ビジー時間	1サイクル
	バンド幅／ポート	8バイト／サイクル
クロック・サイクル時間		60 ns

4.3 評価指標

評価指標には，浮動小数点演算数および動作周波数の双方を正規化した FLOPC (*floating-point operations per clock cycle*) を用いる．

4.4 評価項目および評価モデル

評価項目は次のとおりである．

①FIFO ベクトル・レジスタ (F)：ベクトル・レジスタをリング FIFO バッファとして動作させた場合，および，従来のベクトル・プロセッサ同様のレジスタとして動作させた場合とを比較する．なお，後者の場合，ストリップ・マイニング操作が必要となるが，そのセクション・サイズは FIFO レジスタの物理レジスタ長 32 とした．

②マルチスレッド処理 (M)：ベクトル命令レベルのマルチスレッド処理を行う場合，および，従来のベクトル・プロセッサ同様にこれを行わない場合とを比較する．

③ストリーミング (S)：LFK 13 と LFK 14 に対して，ストリーミングを適用した場合とそうでない場合とを比較する．

上記の評価項目①と②に対しては，以下の 3 つの評価モデルを設定した．

- *SV*: 従来のベクトル・プロセッサのモデル．すなわち，マルチスレッド処理は行わず，ベクトル・レジスタもリング FIFO バッファとして動作しない．
- *MSV*: 従来のベクトル・プロセッサにマルチスレッド処理を採用したモデル．なお，ベクトル・レジスタはリング FIFO バッファとして動作しない．
- *MSFV*:『順風』のモデル．すなわち，マルチスレッド処理を行い，ベクトル・レジスタはリング FIFO バッファとして動作する．

上記いずれのモデルにおいても，LFK 13 と LFK 14 に対してはストリーミング (S) を適用している．

各評価モデルにおけるハードウェア資源数，パイプライン・レイテンシ，メモリ・アクセス・レイテンシ，等のパラメータは『順風』の仕様（表 1）に基づいている．ただし，メモリ・アクセス・レイテンシに関しては，これを変えて性能に与える影響を調査した．これは，『順風』の現仕様ではクロック・サイク

ル時間が 60 ns と大きいため，バンク・ビジー時間が1クロック・サイクル，ロード／ストア・パイプラインのメモリ・アクセス・レイテンシが6クロック・サイクルと，相対的に小さくなっているからである．これらの値は，現在の一般的なスーパコンピュータに比べると非常に小さい値であり，実情に即さない．そこで，メモリ・アクセス・レイテンシに関して，次の2つの性能モデルを設定した．

- *fm*(*fast memory*)：『順風』のメモリ仕様そのもの．バンク・ビジー時間1クロック・サイクルで，バンク数は8バンク．ロード／ストア・パイプラインのメモリ・アクセス・レイテンシは6クロック・サイクルである．
- *sm* (*slow memory*)：『順風』のメモリ仕様を現在のスーパコンピュータのそれに合わせたモデル．バンク・ビジー時間は8クロック・サイクルで，バンク数は 32 バンク．ロード／ストア・パイプラインのメモリ・アクセス・レイテンシは 13 クロック・サイクルと，*fm* の約2倍になる．

結局，評価モデルは SV_{fm}, SV_{sm}, MSV_{fm}, MSV_{sm}, $MSFV_{fm}$, および，$MSFV_{sm}$ の6モデルとなった．

さらに，メモリ・アクセス・レイテンシが変化すると，仮想パイプラインへの実パイプライン割当アルゴリズムが性能に与える影響が異なってくることが予想される．そこで，マルチスレッド処理を採用している MSV_{fm}, MSV_{sm}, $MSFV_{fm}$, および，$MSFV_{sm}$ の4モデルについては，2.2 節で述べた3方式のうち round-robin 方式 (rr) と forced 方式 (fo) の2方式が性能に与える影響も調査した．

5. 評価結果および考察

図4 に，評価モデル SV_{fm}, MSV_{fm} および $MSFV_{fm}$ それぞれに対するシミュレーション結果を示す．

『順風』は演算スループット1倍精度演算／クロック・サイクルの実加減算パイプラインと実乗除算パイプライン1本をそれぞれ1本ずつ備えているので，いずれの評価モデルでもピーク性能は 2FLOPC となる．

図4の評価モデル SV_{fm}（従来型のベクトル・プロセッサのモデル）と評価モデル $MSFV_{fm}$（『順風』のモデル）の性能に着目すると，『順風』は従来型のベクトル・プロセッサに比べて最低 16.3%～ 相乗平均 59.0%～最高 334.9% 性能が良いことがわかる．それでは，その内訳を見てみよう．

5.1 FIFO ベクトル・レジスタの評価

図4の評価モデル MSV_{fm}（ベクトル・レジスタが FIFO 動作しない）と評価モデル $MSFV_{fm}$（ベクトル・レジスタが FIFO 動作する）の性能に注目する．$MSFV_{fm}$ の MSV_{fm} に対する性能向上率は，最低 0.0%～相乗平均 10.7%～ 最高 24.2% である．

上記の性能向上は，ベクトル・レジスタをリング FIFO バッファとして動作させて，ストリップ・マイニング・オーバヘッドを除去したことによって得られたものである．処理すべきベクトルが非常に長い場合には，FIFO ベクトル・レジスタの効果はかなり期待できる．例えば，LFK 12 のベクトル長は 1001 と長く，FIFO ベクトル・レジスタによる性能向上は最高の 24.2% に達している．一方，ベクトル長が短い場合にはあまり効果が期待できない．例えば，LFK 2 のベクトル長は最長でも 32 と短く，FIFO ベクトル・レジスタによる性能向上は全くない (0.0%)．

しかしながら，上記の性能向上率は，意外と低い値である．これには次のような理由が考えられる．『順風』のメモリ・アクセス・レイテンシは6クロック・サイクルと極めて小さく，セクションごとのスタートアップ時間（つまり，ストリップ・マイニング操作に伴うオーバヘッド）が低く抑えられている．ちなみに，メモリ・アクセス・レイテンシを 13 クロック・

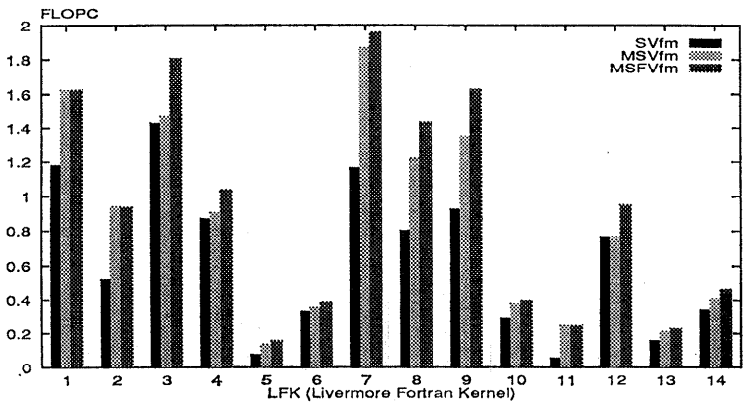


図 4 SV_{fm}, MSV_{fm} および $MSFV_{fm}$ の性能
Fig. 4 Performance of SV_{fm}, MSV_{fm}, and $MSFV_{fm}$.

サイクルと遅くした場合には，$MSFV_{sm}$ の MSV_{sm} に対する性能向上率は，最低2.1%～相乗平均28.4%～最高109.4%とかなり大きくなる（**図5**参照）．すなわち，FIFO ベクトル・レジスタの効果が出ている．

しかしながら，ベクトル・レジスタをリングFIFOバッファとして動作させることには，以下の問題点がある．すなわち，論理上は無限長のFIFOとして動作させるため，レジスタ読出しが破壊読出しとなってしまう．つまり，データの再利用性がない．この問題を解決するため，演算結果やロード結果を2個のデスティネーション・レジスタに格納したり（ダブル・デスティネーション指定），ベクトル・コピー（VCOPY）命令（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・レジスタに単にコピーする命令）を用いている．しかし，余分なベクトル命令を追加するため実行時間が長くなるし，また，2個のレジスタへ同時に書き込むためバス構成が複雑になるという問題が生じる．

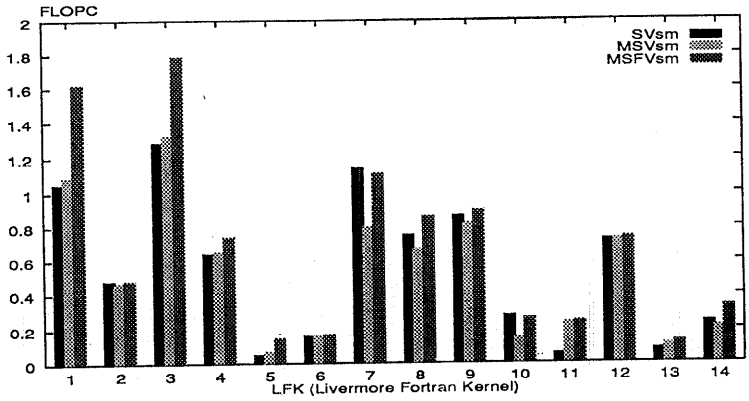


図5 SV_{sm}, MSV_{sm} および $MSFV_{sm}$ の性能
Fig. 5 Performance of SV_{sm}, MSV_{sm}, and $MSFV_{sm}$.

このように，ベクトル・レジスタをリング FIFO バッファとして動作させるか否かに関しては，性能とハードウェア・コストとのトレードオフに十分留意して決定する必要がある．

5.2 マルチスレッド処理の評価

図4の評価モデル SV_{fm}（マルチスレッド処理しない）と評価モデル MSV_{fm}（マルチスレッド処理する）の性能に注目する．MSV_{fm} の SV_{fm} に対する性能向上率，すなわち，ベクトル命令レベルでマルチスレッド処理することによる性能向上率は，最低 0.0%～相乗平均 43.6%～ 最高 333.8% である．

演算数が多いループに対しては，パイプラインを仮想化しマルチスレッド処理する効果は大きい．とりわけ，演算数が多く，かつ，回帰型演算（一次回帰演算，総和，内積，等）を含むような場合，その効果は顕著である．回帰型演算は，本来ベクトル化に不適なデータ依存関係にある．したがって，SV ではパイプライン内にバブルが生じ，パイプラインの使用率が低下するため高速なベクトル処理ができない．それに対して MSV では，マルチスレッド処理によりパイプラインの使用率を向上させることが可能であるため，SV と比べて性能が向上する．たとえば，LFK 11 (first summation) では，マルチスレッド処理により4倍以上もの性能が得られている．また，複数のベクトル命令が実行可能であるため，MSV では SV と比べてスタートアップ・タイムが短縮される．このため，スタートアップ・タイムによるオーバヘッドが小さくなり性能が向上する．

一方，性能向上率が低いのは LFK 3 (2.9%) と LFK 12 (0.0%) だが，これらのループは演算数自体が少なく (LFK 3 は乗算1と加算1，LFK 12 は減算1)，パイプラインを仮想化しても上述の効果が得られないためである．

しかしながら，メモリ・アクセス・レイテンシを 13 クロック・サイクルと遅くした場合，MSV_{sm} の SV_{sm} に対する性能向上率は最低 −45.1%～相乗平均 6.2%～ 最高 321.1% と小さくなっている（図5参照）．14 カーネル中6カーネルでは，逆に性能が低下している．

これは，もともとメモリへの連続アクセスだったものが，マルチスレッド処理により不連続かつランダム・アクセスに変わってしまい，その結果，バンク・コンフリクトが多発しているためと推察される．そこで，仮想パイプラインへの実パイプライン割当アルゴリズムを『順風』で採用している round-robin 方式 (rr) から forced 方式(fo)へと変更してみた．

評価モデル MSV_{fm}(rr), MSV_{fm}(fo), MSV_{sm}(rr) および MSV_{sm}(fo) それぞれに対するシミュレーション結果を**図6**に示す．これより，メモリ・アクセス・レイテンシが大きい場合，forced 方式 (MSV_{sm}(fo))

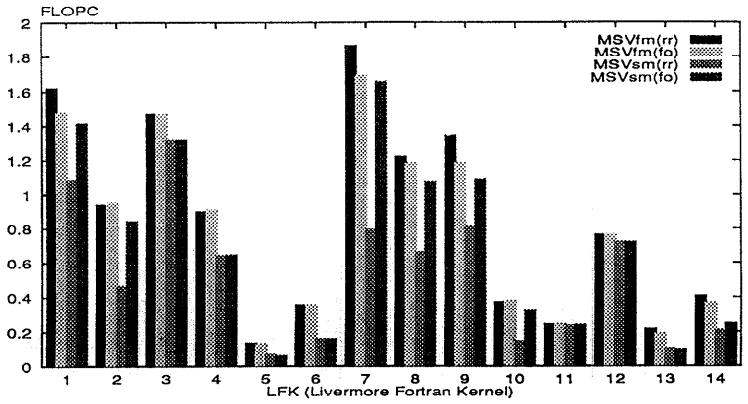


図 6 MSV_{fm}(rr), MSV_{fm}(fo), MSV_{sm}(rr) および MSV_{sm}(fo) の性能
Fig. 6 Performance of MSV_{fm}(rr), MSV_{fm}(fo), MSV_{sm}(rr), and MSV_{sm}(fo).

の方が round-robin 方式 (MSV_{sm}(rr)) よりも最低 −0.05%〜相乗平均 25.8%〜最高 114.5% 性能が良いことがわかる．ちなみに，メモリ・アクセス・レイテンシが小さい場合は，round-robin 方式 (MSV_{fm}(rr)) の forced 方式 (MSV_{fm}(fo)) に対する性能向上率は最低 −0.01%〜相乗平均 0.41%〜最高 13.7% とほとんど差がない．

このように，マルチスレッド処理は効果が大きいものの，その導入に当たっては，実パイプライン割当アルゴリズムの選択に慎重を期する必要がある．

5.3 ストリーミングの評価

表 2 に，LFK 13 と LFK 14 に対してストリーミングを適用した場合とそうでない場合における性能を示す．評価モデルは，$MSFV_{fm}$（メモリ・アクセス・レイテンシが『順風』の仕様通り小さい）と $MSFV_{sm}$（メモリ・アクセス・レイテンシが大きい）を用いた．

$MSFV_{fm}$ においては，ストリーミングを適用した場合 29.4%〜32.9% 性能が向上する．また，$MSFV_{sm}$ においても，23.4%〜33.3% とほぼ同程度の性能向上率である．

このように，ストリーミングは，メモリ・アクセス・レイテンシの大小に関係なく有効な機能である．

表 2 ストリーミングの効果 (FLOPC)
Table 2 The effects of streaming.

評価モデル	$MSFV_{fm}$		$MSFV_{sm}$	
ストリーミング	適用	未適用	適用	未適用
LFK 13	0.230	0.173	0.128	0.096
	(1.329 : 1)		(1.333 : 1)	
LFK 14	0.466	0.360	0.327	0.265
	(1.294 : 1)		(1.234 : 1)	

6. おわりに

以上，*MSFV* アーキテクチャ自身の特長である FIFO ベクトル・レジスタ (F)，マルチスレッド処理 (M)，ストリーミング (S) について評価を行った．その結果，次のことが判明した．

- FIFO ベクトル・レジスタ (F) の評価：ベクトル・レジスタを FIFO 動作させることにより，ストリップ・マイニング操作を排除することの効果はある．特に，メモリ・アクセス・レイテンシが大きくなると，その効果は顕著である．ただし，ハードウェア・コストの増加を招く可能性があるので，性能とのトレードオフに注意する必要がある．
- マルチスレッド処理 (M) の評価：ベクトル命令レベルでのマルチスレッド処理の効果は大きい．特に，演算数の多いループや回帰型演算を含む場合において，極めて効果的である．しかし，メモリ・バンク・ビジー時間が大きくなると，マルチスレッド処理の影響でメモリ・アクセスがランダムとなり，バンク・コンフリクトが多発するという弊害が生じる．この問題の解決には，仮想パイプラインへの実パイプライン割当アルゴリズムの最適化が必要で，今回の評価では『順風』で採用している round-robin 方式より forced 方式の方が良好な結果を出した．
- ストリーミング (S) の評価：ストリーミングの適用可能な LFK が 2 つだけと評価数は十分でないが，極めて効果的であることが確認された．しかも，その効果の程度がメモリ・アクセス・レイテンシの大小に左右されない点も評価できる．なお，『順風』のスカラ・ユニットの性能は，単純化したためそれほどよくない．スカラ・ユニットの性能が向上すれば，ストリーミングの効果はさらに大きくなるはずである．

本論文で示した評価結果から，*MSFV* アーキテクチャが有する特長により総合的には，『順風』は従来型のベクトル・プロセッサに比べて，LFK 1〜14 で最低 16.3%〜相乗平均 59.0%〜最高 334.9% 性能が良いことが判明した．

マルチスレッド処理に関しては，今後検討すべき項目が多くある．たとえば，今回の評価でも取り扱った実パイプライン割当アルゴリズム，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，等がそうである．これらについては今後の研究問題である．

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します．

参考文献

1) 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』，電子情報通信学会技術研究報告，CPSY 89-39 (1989)．
2) 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)．
3) 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—，情報処理学会研究会報告，ARC-85-3 (1990)．
4) 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-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)．
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』—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—，情報処理学会論文誌，Vol. 32, No. 7, pp. 828-837 (1991)．
6) 弘中，橋本，岡崎，村上，富田：『順風』：MSF 型ベクトル・プロセッサ・プロトタイプ—仮想パイプラインの実現法—，情報処理学会研究会報告，ARC-89-6 (1991)．
7) Hironaka, T., Hashimoto, T., Okazaki, K., Murakami, K. and Tomita, S.: A Single-Chip Vector-Processor Prototype Based on Multi-threaded Streaming/FIFO Vector (MSFV) Architecture, *Proc. Int'l. Symp. Supercomputing '91*, pp. 77-86 (1991)．
8) 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—演算パイプラインの構成—，情報処理学会研究会報告，ARC-92-5 (1992)．
9) Hironaka, T., Hashimoto, T., Okazaki, K., Murakami, K. and Tomita, S.: Benchmarking a Vector-Processor Prototype Based on Multi-threaded Streaming/FIFO Vector(MSFV) Architecure, *Proc. 1992 Int'l. Conf. Supercomputing*, pp. 272-281 (1992)．
10) 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，電子情報通信学会論文誌，Vol. J 74-D-I, No. 12, pp. 836-845 (1991)．
11) Chiueh, T.-C.: Multi-Threaded Vectorization, *Proc. 18th Int'l. Symp. Computer Architecture*, pp. 352-361 (1991)．

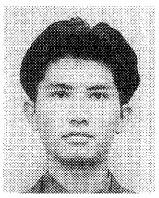
(平成4年9月16日受付)
(平成5年1月18日採録)

橋本 隆（正会員）

1967 年生．1991 年九州大学工学部情報工学科卒業．現在，同大学大学院総合理工学研究科情報システム学専攻修士課程に在学中．計算機アーキテクチャ，並列処理の研究に従事．

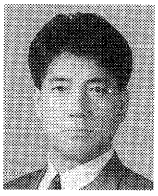
弘中 哲夫（正会員）

1965 年生．1988 年山口大学工学部電気工学科卒業．1990 年九州大学大学院総合理工学研究科情報システム学専攻修士課程修了．現在同大学院博士課程に在学中．計算機アーキテクチャ，並列処理システムの研究に従事．



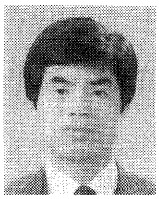
岡崎 恵三

1967 年生．1990 年九州大学工学部情報工学科卒業．1992 年九州大学大学院総合理工学研究科情報システム学専攻修士課程修了．現在日産自動車(株)勤務．並列処理，ベクトル処理，計算機アーキテクチャに興味をもつ．



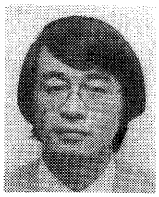
村上 和彰（正会員）

1960 年生．1982 年京都大学工学部情報工学科卒業．1984 年同大学院修士課程修了．同年富士通(株)本体事業部に入社，汎用計算機 M シリーズのアーキテクチャ開発に従事．1987 年九州大学工学部助手，1992 年同大学院総合理工学研究科講師，現在に至る．計算機アーキテクチャ，並列処理，性能評価などの研究に従事．著書「計算機システム工学（共著）」，「ヘネシー＆パターソン：コンピュータ・アーキテクチャ（共訳）」．本学会研究賞（平成3年度），本学会論文賞（平成3年度）受賞．電子情報通信学会，日本応用数理学会，ACM, IEEE, IEEE-CS 各会員．



權　五鳳（正会員）

1954 年生．1980 年韓国高麗大学電気工学科卒業．1983 年同大学院制御工学専攻修士課程修了．同年韓国大丘工業専門大学専任講師．1992 年九州大学大学院総合理工学研究科博士課程退学．同年九州大学工学部情報工学科助手，現在に至る．計算機アーキテクチャ，並列処理システム，コンピュータ・グラフィックスの研究に従事．ACM, IEEE 各会員．



富田　眞治（正会員）

1945 年生．1968 年京都大学工学部電子工学科卒業．1973 年同大学院博士課程修了．工学博士．同年京都大学工学部情報工学教室助手．1978 年同助教授．1986 年九州大学大学院総合理工学研究科教授，1991 年京都大学工学部情報工学科教授，現在に至る．計算機アーキテクチャ，並列処理システムなどに興味を持つ．著書「並列計算機構成論」「計算機システム工学」「並列処理マシン」など．電子情報通信学会，IEEE, ACM 各会員．