スーパミニコン内蔵型ベクトルプロセッサの演算制御方式†

阿部重夫†† 高藤政雄†† 坂東忠秋††
平沢宏太郎†† 加藤猛†††

近年増加の一途にある大規模科学演算の高速処理の要求に答えるため，スーパコンピュータ，あるいはミニコンに外付するベクトルプロセッサ等，ベクトル演算を高速処理する専用マシンの開発が進められている．本論文では，制御用スーパミニコンに内蔵するベクトルプロセッサの高速・高精度化のアーキテクチャについて述べる．従来の専用マシンでは，単一データフォーマットしかサポートしていないが，このため単精度ベクトルプロセッサでは，内積演算，関数演算等で桁落ちの可能性が生じる．このため今回開発したベクトルプロセッサでは，倍精度および部分倍精度パイプライン制御方式を採用した．これにより内積演算では単精度と同一の演算ステップで，また関数演算でも単精度演算とほとんど変わらない処理ステップで倍精度演算が可能となった．電力系統の解析で生じるスパース行列演算は，規則的ベクトル演算の高速処理を主体にして設計されたスーパコンピュータが最も苦手とする分野である．このスパース行列演算の高速化を図るため，インデックスサーチの高速化等アドレス演算器の機能強化を図った．これによりスパース連立方程式の求解では，汎用大型機 HITAC M 200 H の 1.6 倍の高速化が実現できることが明らかとなった．

1. まえがき

大規模科学演算を高速に実行するマシンとして大型機では，CRAY-1[1] 等のスーパコンピュータ，あるいは汎用機に内蔵するベクトルプロセッサ[2] の開発が行われている．またミニコンピュータの分野でも科学演算，信号処理演算を高速に実行する外付型ベクトルプロセッサ[3] が開発されている．

スーパコンピュータでは，規則的な行列演算，とくに内積演算を高速化することを主体に設計されているため，実際のプログラムを走らせた性能すなわち実質性能は最大性能の 20～30% といわれている．極端な場合たとえば，CRAY-1 で電力系統解析等で生じるスパース連立方程式を解いた場合，最大性能 160 MFLOPS (Millions of Floating Operations per Second) に対してわずか 2.2 MFLOPS しか出なかったという報告例[4] もある．

また内積演算，関数演算等で演算の桁落ちを除いて入力データと同じ精度の出力を得るためには，部分的に倍精度演算が必要といわれている[5),6]．しかしながらこれまでに開発されているベクトルプロセッサでは，ハードウェアでは単一のデータフォーマットしかサポートしていないため，これらの演算で桁落ちが生じる可能性がある．スーパコンピュータでは，64 ビットを基本語長としているため，精度的問題は生じにくいが，32 ビット演算を基本としたミニコン用ベクトルプロセッサでは，精度的な問題が生じやすくなる．

本論文では，以上の問題点を解決すべく導入した関数演算の高速・高精度化方式，スパース行列演算の高速化方式等について述べ，それに基づいたスーパミニコン HIDIC V90/50 に内蔵するベクトルプロセッサのアーキテクチャおよび性能評価結果について述べる．

2. 内蔵型ベクトルプロセッサ高速化方式

汎用スーパミニコンにベクトルプロセッサを接続する方式としては，外付型[3] と内蔵型の二つが考えられる．外付型では，ベクトルプロセッサはホストコンピュータの一つの入出力機器として取り扱われるため，ベクトルプロセッサへの必要データの転送等起動のオーバヘッドが大きいという欠点があるが，ホストコンピュータとの並列処理が可能である．これに対し内蔵型では，ホストコンピュータから命令でベクトルプロセッサとリンクするため，起動のオーバヘッドは小さいが，その反面ホストコンピュータとの並列処理はできない．内蔵型にすべきか，外付型にすべきかは，応用分野によって決まりホストコンピュータとのインタラクションが多い電力系統制御等の分野は内蔵型が，また大量のデータ処理が必要でホストコンピュータとのインタラクションがあまりない画像処理等の分野は外付型が適しているといえる．ベクトルプロセッサの応用分野の一つとして電力系統の制御を考えたた

† Integrated Vector Processor for Super Minicomputer by SHIGEO ABE, MASAO TAKATOO, TADAAKI BANDOO, KOOTARO HIRASAWA (Hitachi Research Lab., Hitachi, Ltd.) and TAKESHI KATOO (Omika Works, Hitachi, Ltd.).

†† (株)日立製作所日立研究所
††† (株)日立製作所大みか工場

め内蔵型の方式を採用した.

汎用スーパミニコンにベクトルプロセッサを内蔵して高速演算を実現するためには，演算ユニットへのデータ供給能力を確保することおよび繰返し演算のステップ数を可能な限り短くするためにマイクロプログラム制御能力の強化が必要である.

汎用スーパミニコンでは，データ供給能力を高めるためにキャシュメモリが用いられている．通常のプログラムを実行する場合は，一度アクセスされたデータは近い将来再度アクセスされる可能性が高いというプログラムの局所性により，キャシュメモリのデータ供給能力にほぼ等しい供給能力を実現できる．しかしながらメモリ上で連続して記憶されたデータを読み出して演算を行い別の連続したメモリ上の領域に記憶するベクトル演算では，上記のプログラムの局所性は保証されないためキャシュのヒット率があがらず大幅な性能低下につながる.

このためベクトルプロセッサ内に大容量の2組のローカルメモリおよびレジスタファイルを設け，1マシンサイクルに2ワード以上のデータ供給能力を実現した.

またベクトル演算の高速化のためには4.2節で述べる二つの判定の同時処理等，マイクロプログラム制御機能の大幅な強化が必要である．V 90/50 のマイクロプログラムコントローラの改造なしでこれを実現するために，ベクトルプロセッサ内にマイクロプログラムコントローラを置き，V 90/50 のマイクロプログラムコントローラからの制御信号に従ってベクトルプロセッサが動作する一体モードと，V 90/50 とは独立にベクトルプロセッサ内のコントローラに従って動作する単独モードとを設けた．一体モードでの処理では高速化が得られないような演算に対しては，一体モードで必要なデータをローカルメモリに転送後，単独モードで演算することにより高速演算が可能となる．ローカルメモリに転送後演算を実行するという点からは，外付型ベクトルプロセッサと同じであるが，内蔵型であるため命令レベルでホストコンピュータからリンクできること，またデータ転送のバンド幅を外付型より大きくとれるためデータ転送に伴うオーバヘッドは，外付型ほど問題とならない.

3. 関数演算の高速・高精度化方式

3.1 区間縮小演算

関数演算は，科学計算における基本演算であるから高速化は必須であるが，演算過程に桁落ちが生じる特殊な演算が含まれるため，精度面からも十分な考慮が必要である．その代表的なものが区間縮小演算[5]である．たとえば sin X を求める場合，X を

$$X=(F+I)\cdot\pi/2,\quad -1<F<1,\ I:\text{整数} \qquad (1)$$

と分離すると

$$\sin X=\begin{cases}(-1)^{I/2}\sin(\pi/2)F & \text{for } I:\text{偶数}\\ (-1)^{(I-1)/2}\cos(\pi/2)F & \text{for } I:\text{奇数}\end{cases} \qquad (2)$$

となり，$\sin(\pi/2)F$ あるいは，$\cos(\pi/2)F$ を多項式近似することにより求めることができる．ここで(1)式においてFを求める場合，Xに $2/\pi$ を乗じて，その結果を小数部分と整数部分とに分離するが，単精度でこの計算を実行すれば，$|X|>\pi/2$ のとき，桁落ちにより，Fは単精度の精度が得られなくなる．このためFの計算には，倍精度演算が必要となる.

単精度演算を基本にしたプロセッサで倍精度演算を実行すると性能が大幅に低下する．このため性能と精度の両方を満足させるものとして部分倍精度演算方式を導入する．すなわち乗算器の機能として

(1) (単精度)×(単精度)→(単精度)
(2) (単精度)×(単精度)→(倍精度)
(3) (倍精度)×(単精度)→(倍精度)
(4) (倍精度)×(倍精度)→(倍精度)

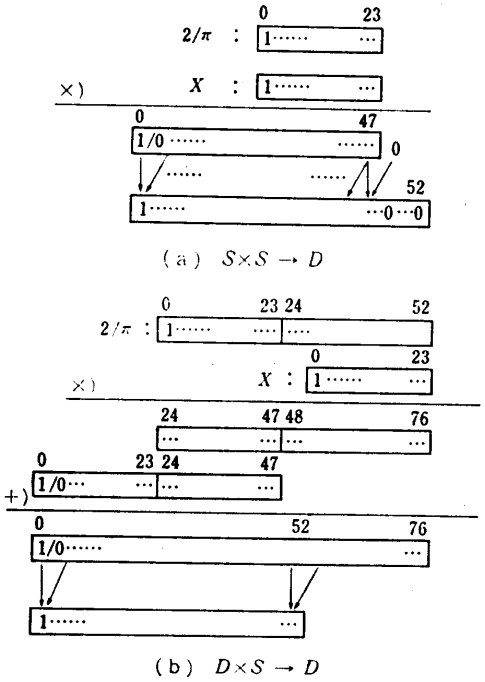
を設け，また加算器の機能として，

(1) (単精度) op. (単精度)→(単精度)
(2) (倍精度) op. (倍精度)→(単精度)
(3) (倍精度) op. (倍精度)→(倍精度)

を設ける．ただし op. は加算等の演算を示すとし，単，倍精度は，以下 S, D と略す.

図1に $(2/\pi)\times X$ の乗算を二つの部分倍精度演算で行う場合を示す．ここでフローティングデータフォーマットは，IEEE 標準フォーマット[7]としているため，単精度および倍精度の仮数部は，おのおの24および53ビットで入力データの先頭ビットは1に正規化してあるとしている．図の(a),(b)を比較すれば，(a)では，25ビット目以降正しく求まらないため，(b)の $D\times S\rightarrow D$ の演算方式を用いる必要がある．この結果を用いて加算器で，D op. $D\rightarrow S$ の演算を実行すれば，小数部分Fが桁落ちなく求まることになる.

ここで演算器の構成を**図2**に示すように32ビットのパイプライン乗算器，および64ビットのパイプライン加算器とし，乗算器における $D\times S\rightarrow D$ の演算



(a) $S\times S \rightarrow D$

(b) $D\times S \rightarrow D$

図 1 部分倍精度演算
Fig. 1 Partial double precision calculation.

を，倍精度データを上32ビット，下32ビットの順で順次入力することにより実行する方式とすれば，区間縮小演算を，単精度で実行するのに比較してわずか1マシンサイクルの増加のみで桁落ちなく実行できることになる．

部分倍精度演算は，単精度入力データの内積演算の桁落ち防止にも有効で，$S\times S \rightarrow D$, $D+D \rightarrow S$ の機能により単精度演算と同一処理時間で部分倍精度内積演算が可能となる．

3.2 条件付加算

関数演算においては，二つのデータの演算内容を一方のデータの符号に従って変えることが，しばしば必要となる．たとえば e^X を計算するアルゴリズム[8]は，以下のようになる．

(1) $X/\ln 2+1=I+F+1/2$ (ただし $-1/2\leq F<1/2$, I: 整数) に分離する．

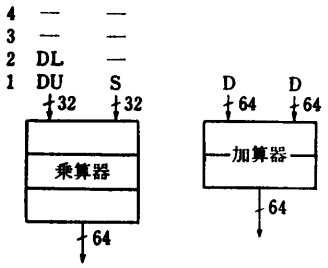
(2) $V(F)=\sum_{i=0}^{r} a_i F^i$ を求める．

(3) $e^X=2^I\cdot V(F)$ により e^X を求める．

ここで $X/\ln 2$ を区間縮小演算により，

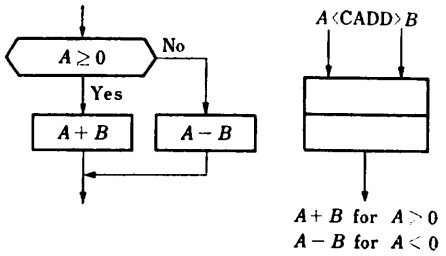
$$X/\ln 2=I'+F' \quad (\text{ただし } -1<F'<1,\ I': \text{整数}) \tag{3}$$

に分離したとすると，



$D\times S \rightarrow D$ $\quad D+D \rightarrow S$

図 2 パイプライン演算器による区間縮小演算
Fig. 2 Range reduction by pipeline calculation units.



(a) 条件分岐演算 (b) 加算器による条件付演算

図 3 条件付加算
Fig. 3 Conditional addition.

$$F=\begin{cases} F'-0.5 & F'\geq 0 \\ F'+0.5 & F'<0 \end{cases} \tag{4}$$

$$I=\begin{cases} I'+1 & I'\geq 0 \\ I' & I'<0 \end{cases} \tag{5}$$

により F, I が求まる．**図3**(a)にこのときのフローを示すが，条件に従って分岐が生じるため，パイプライン演算器へのデータの流れが中断し，高速処理を阻害する要因となる．

関数演算をベクトル化する場合も，条件分岐はベクトル化による高速化を阻むことになる．パイプラインプロセッサにおけるベクトル演算では，ある要素に対する演算が完了する前に次の要素に対する演算を開始するというオーバラップ演算により，高速化を図っている．したがってオーバラップの度合が大きいほど，一つの要素に対する演算が完了する前に次々と次の要素に対する演算を開始できるため，繰返し計算のループ部分のステップ数が短くなり，1要素あたりの計算をスカラ計算に対して高速化することができる．

ここで**図4**に示すように分岐処理があると，分岐処理の部分は，異なる添字間のオーバラップ演算が行えないため，ベクトル演算の高速化が図れない．このため図3(b)に示すように，パイプライン加算器に一方

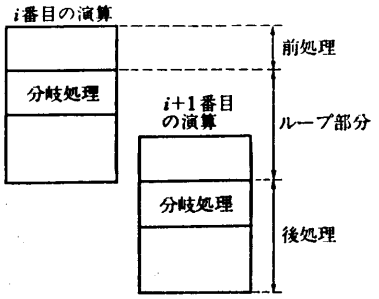
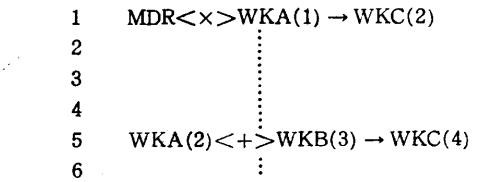
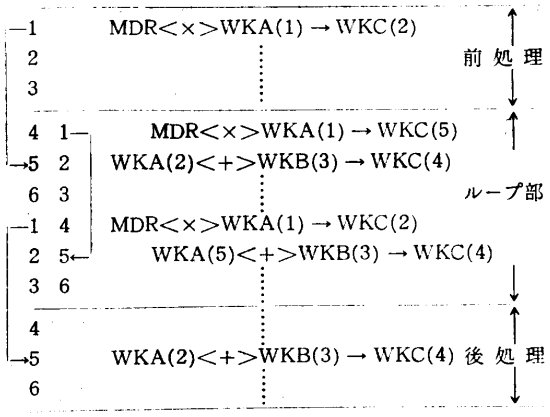


図 4 条件分岐があるときのオーバラップ演算
Fig. 4 Overlap calculation with conditional branch.



（a） スカラ演算



（b） ベクトル化

図 5 レジスタ方式によるベクトル演算
Fig. 5 Vector calculation with working register.

の入力データの符号に従って演算モードを変える条件付加算命令を追加した．これにより e^x の計算では条件分岐が解消し，スカラ演算25マシンサイクルに対して，ベクトル演算では1要素あたり10マシンサイクルに短縮が可能となった．

3.3 FIFO バッファ

乗算，加算は，パイプライン演算により高速化が可能であるが，除算はパイプライン処理化がむずかしい．このため関数演算をパイプライン方式のベクトルプロセッサで高速化するためには，多項式近似等除算の少ないアルゴリズムを適用する必要がある．多項式演算では，乗算あるいは加算結果をその後の計算で何回か使用する場合が生じるため，演算結果を一度作業用レジスタに待避することが行われる．このような関

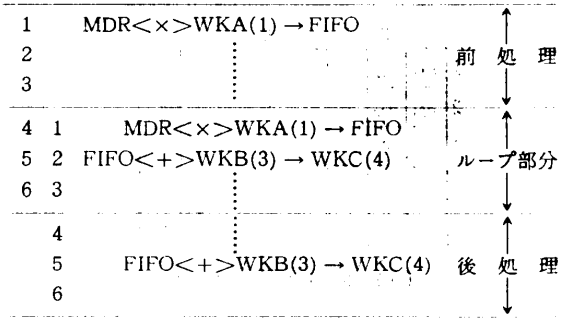


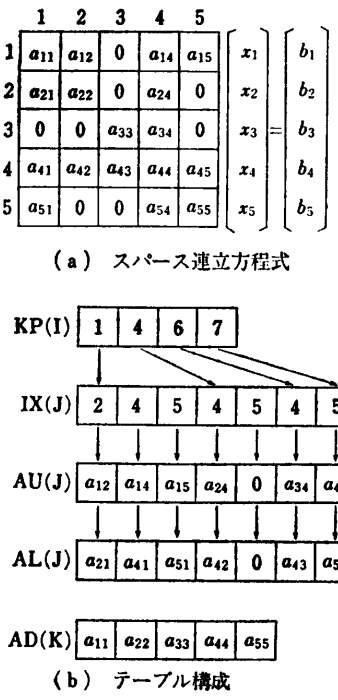
図 6 FIFO バッファによるマイクロプログラムステップの削減
Fig. 6 Reduction of microprogram steps by FIFO buffer.

数演算を，ベクトル演算化する場合，i 番目の要素の演算において，作業用レジスタの内容を参照する前に，$i+1$ 番目の要素の演算を開始し，レジスタへの書込みが必要となる場合がある．たとえば**図5**（a）において，WKA, WKB, WKC を2ポートレジスタのおのおの二つの読出しおよび書込みポート，MDRは，メモリから読み出したデータをセットするレジスタとし，MDR と，WKA（1）を乗算して，WKC（2）に書き込み，その後で WKA（2）と WKB（3）を加算して，WKC（4）に書き込むとする．この演算をベクトル化するとき，図5（b）に示すように WKC（2）に書き込まれた内容が，参照される前に，次の要素の演算を開始して，MDR と WKA（1）の乗算結果をレジスタに記憶するものとする．このとき WKC（2）に書き込めば，アクセスする前にデータが書き替えられることになるため WKC（2）とは別のエリア WKC（5）に書き込む必要がある．このとき図に示すように一要素に対する演算ステップ数の2倍のステップ数のマイクロプログラムの記述が必要となり，プログラミングがきわめて複雑となる．この問題は，ワークのバッファとしてアドレスでアクセスするレジスタでなく，最初に書き込んだデータを最初に読み出すFIFO (First-in, First-out) バッファを用いれば，**図6**に示すように解決する．このためワークのレジスタとして，2ポートレジスタのほか，FIFO バッファを設けることとした．この2種類のワークレジスタの組合せにより，マイクロプログラムの記述性の向上，ステップ数の削減に寄与することとなった．

4. スパース演算高速化方式

4.1 スパース行列演算

電力系統解析においては，系統網の電力潮流を計算



(a) スパース連立方程式

(b) テーブル構成

図 7 スパース LU 分解のテーブル構成
Fig. 7 Table structure of sparse LU decomposition.

するために数百次元の非線形連立方程式を高速に解くことが要求される[4]. 高速化の方式としては，係数行列の零要素が 95～98% とスパース性がきわめて高いことから，Newton-Raphson 法とスパース LU 分解（三角行列分解）法とを組み合わせるのが一般的である.

LU 分解法[9]では，n 次元連立一次方程式を

$$AX=B \qquad (6)$$

としたとき，係数行列Aを，次式のように単位下三角行列Lと上三角行列Uとの積に分解する.

$$AX=L\cdot UX=B \qquad (7)$$

そこで $UX=Z$ とおいて，$LZ=B$ より Z を求め，ついで $UX=Z$ よりXを求める. なお，L, U は三角行列のため，Z, X は代入操作で求まりおのおの前進代入および後退代入と呼ぶ.

係数行列のスパース性を考慮する場合は，係数行列中の非零要素のみをインデックス付きで記憶する方式を採る[10]. ただしこのとき LU 分解によって零から非零に変わる要素は，非零要素としてエリアを確保しておく必要がある. **図 7** に 5 次元連立方程式を例にとってテーブル構成を示す. 係数行列Aは，一次元配列とした上三角行列 $AU(J)$, 下三角行列 $AL(J)$, 対角行列 $AD(J)$ の形で記憶されそれに対するアドレスを，$KP(I)$, $IX(J)$ の組合せで指定する. すなわち $KP(I)$ は，係数行列の I 行，あるいは I 列目の最初の非零要素 a_{ij}, $a_{ji}(j>I)$ の $AU(J)$, $AL(J)$ 内の相対アドレスを示し，$IX(J)$ は，$AU(J)$, および $AL(J)$ の J 番目の要素の列番号あるいは行番号を示している. $AD(J)$ には，対角要素 a_{JJ} が記憶されている.

$i-1$ 行 $i-1$ 列まで LU 分解が完了したとすると i 行 i 列での分解は，次の演算が中心となる.

$$\begin{aligned} a_{kj} &\leftarrow a_{kj}-a_{ij}\times a_{ki} \\ a_{jk} &\leftarrow a_{jk}-a_{ik}\times a_{ji} \end{aligned} \qquad j>k>i$$

ここで a_{ij}, a_{ik} および a_{ki}, a_{ji} はおのおの $AU(J)$, $AL(J)$ に連続的に記憶されているため，データの取出しのためのアドレス計算は，容易に行いうる. これに対して，a_{kj}, a_{jk} を取り出すためには，$IX(J)$ 内の相対アドレスが $KP(k)$ から $KP(k+1)-1$ までのデータの値で j となる J の値を決めるインデックスサーチが必要となる. この J により $AU(J)$, $AL(J)$ を取り出すことにより a_{kj}, a_{jk} が得られる. LU 分解では，このインデックスサーチが最内側の Do Loop となるためインデックスサーチの高速化がスパース連立方程式の求解の高速化に必須の条件となる. またスパース演算では，インデックス値によりメモリアドレスを修飾することが必要であるからインデックス付アドレス計算の高速化も必須である.

CRAY-1 等の従来のスーパコンピュータが電力スパース行列演算で低い処理性能しか実現できなかった理由は，インデックス付演算が中心であるため，規則的行列演算が中心のベクトル命令で置き換えることができなかったことによる. CRAY-1 以降のスーパコンピュータでは，インデックス付ベクトル命令もサポートされているが，スパース演算ではベクトル長がせいぜい 10 程度であるためそれほどの高速化は望めないと思われる.

4.2 高速化方式

スパース連立方程式の求解を，マイクロプログラムレベルで解析したところ，アドレス演算の占める割合が，演算ステップ数の 80% を占め，フローティング演算は，全体の 35% 程度となっている[9]. このためスパース行列演算を高速化するため**図 8** のようにアドレス演算器の機能強化を図った. すなわちローカルメモリに格納されたインデックスデータを直接 ALU (Arithmetic and Logical Unit) に入力することにより，インデックス付アドレス計算が 1 マシンサイクルで実行可能とした. またループカウンタを設け，アド

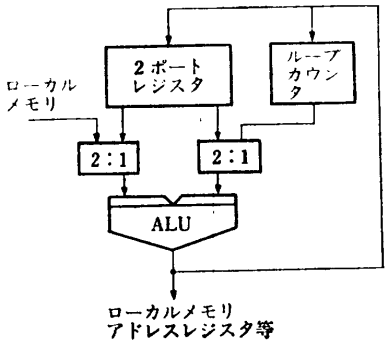


図 8 アドレス演算器の構成
Fig. 8 Block diagram of address calculation unit.

レス演算器による条件判定と，ループカウンタのオーバフロー判定が同時に行えるようにした．これによりスパース LU 分解での最内側ループであるインデックスサーチにおける1要素あたりの判定が1マシンサイクルで実行可能となった．

5. ベクトルプロセッサのアーキテクチャ

5.1 ハードウェア構成

図9に今回開発した内蔵型ベクトルプロセッサの構成を示す．ベクトルプロセッサは，インタフェース部を介して制御用スーパミニコン HIDIC V 90/50 に接続されており，V 90/50 と同じマシンサイクル 167 ns で動作する．

演算ユニットは，32 ビットパイプライン乗算器，64 ビットパイプライン加算器とよりなり，これらのユニットへのデータの供給は，V 90/50 からインタフェース部を介して，あるいはレジスタファイルおよび2組のローカルメモリとより行われる．レジスタファイルは，32 ワード2ポートレジスタ，および2ワードの FIFO バッファとよりなり，ローカルメモリは，16 k ワードおよび 64 k ワードから 512 k ワードまで拡張可能なアクセス時間 167 ns の CMOS メモリからなる．32 ビットアドレス演算器は，整数演算およびローカルメモリアドレス計算を行う．上記の各ユニットは，96 ビット幅 8 k ワードの水平型マイクロプログラムコントローラにより，同時に制御され並列動作が可能となる．

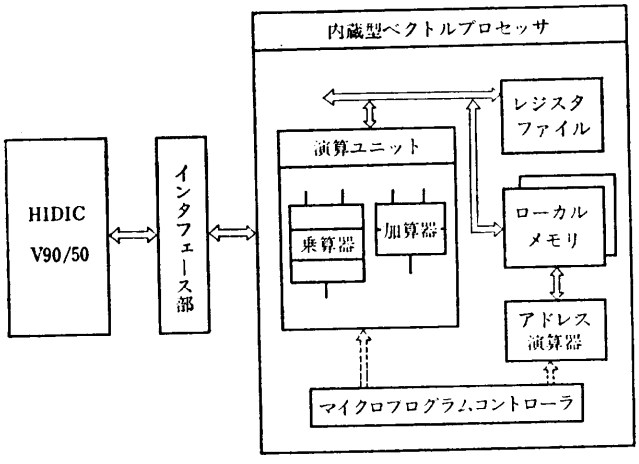


図 9 内蔵型ベクトルプロセッサの構成
Fig. 9 Block diagram of integrated vector processor.

表 1 パイプライン乗算器のコマンド
Table 1 Commands for pipeline multiplier.

No.	コマンド	処理内容
1	FMULT	$S\times S \rightarrow S$
2	FMS_D	$S\times S \rightarrow D$
3	FMSD_D	$S\times D \rightarrow D$
4	FMD_D	$D\times D \rightarrow D$
5	FMI	$I\times I \rightarrow I$
6	THRU	右入力データをそのまま出力

I: 整数データ，S: 単精度，D: 倍精度

表 2 パイプライン加算器のコマンド
Table 2 Commands for pipeline adder.

No.	コマンド	処理内容
1	加減算 3種	$R+L, R-L, L-R$
2	ABSUB	$\|R\|-\|L\|$
3	DIV	除算
4	SCALE	$2^L \cdot R$
5	EOUT	指数部取出し
6	条件付加算 6種	$R-L$, for $R\geq 0$, $R+L$ for $R<0$ 等
7	FAND	小数部分取出し
8	FANDD_S	Dで小数部分取出しSへ変換
9	コンバート(F) 3種	$S+S\rightarrow D, D+D\rightarrow S, D-D\rightarrow S$
10	コンバート 4種	$I\rightarrow S, I\rightarrow D, S\rightarrow I, D\rightarrow I$
11	DIVI	整数除算

R: 右入力，L: 左入力，S: 単精度，D: 倍精度，I: 整数データ

パイプライン乗算器および加算器のコマンドを**表 1, 2** に示す．表1の THRU コマンドは，FFT (Fast Fourier Transform) 演算等で用いる．表2の1〜7までの 14 種のコマンドは，単精度，倍精度をサポートしており，加算器のコマンドは合計 37 種となっている．除算は，1マシンサイクルに2ビットの演算を行う non-restoring 方式を採用しており，16 マシンサイクルでフローティングデータの割り算を行う．

最大性能は，12 MFLOPS で，2組を入力データをおのおの2組のローカルメモリに別々に記憶して

内積演算を行うときに達成できる．このとき部分倍精度演算により，単精度入力データの倍精度内積演算が実行可能である．

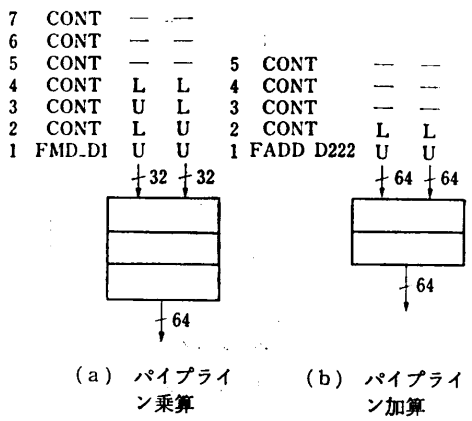
5.2 倍精度パイプライン制御方式

単精度のパイプライン演算器で倍精度演算を行う場合，倍長データの上位32ビット，下位32ビットを，パイプライン演算器に別々に入力する必要があるが，このときの制御信号をそのステップごとに個別にマイクロプログラムで与えていると，マイクロプログラムの記述がきわめて複雑となり，倍精度演算の使用が事実上不可能となる．このためマイクロプログラムの記述を容易にするため，演算開始時にその命令の実行に必要な制御データをすべて与え，それ以後は必要回数分命令実行継続の指令と，必要データを入力する方式とした．**図10**にパイプライン演算の例を示す．(a)は，倍精度乗算の例を示している．FMD-D1 は，倍精度入力データを乗算し，倍精度乗算結果を64ビットバスに一度に出力することを示しており，このコマンドを発行する時点で，64ビットの入力データの上位32ビットのデータを演算器に入力する．2マシンサイクル目からは，命令の継続を示す CONT 指令と必要なデータを入力し，7ステップ目の CONT 指令で演算結果が出力される．図10(a)に示すように倍精度乗算を上位32ビットから実行できるようにしているのは，メモリからのデータは，上位32ビット，下位32ビットの順で記憶され，かつまた読出しも上位32ビットから行われるため，演算開始のためのオーバヘッドが少なくなるからである．

図10(b)は，倍精度加算の例を示している．FADD は，フローティング加算を，また D222 は，倍精度演算で，左右の入力データとも上位32ビット，下位32ビットの順で2度に分けて入力し，加算した結果も上位32ビット，下位32ビットの順で2度に分けて出力することを示している．このコマンドを発行する時点で，左右の32ビットデータを入力し，次の CONT 指令で，下位32ビットを入力し，4および5ステップ目で上位32ビットおよび下位32ビットが出力される．加算器の入出力のタイプは，各入出力ごとに，64ビット1度，32ビットずつ2度の組合せがあり，合計8通りの組合せがある．

5.3 ユーザインタフェース

ベクトルプロセッサには，内積演算等の28種の基本ベクトル命令[2]と，18種の関数演算命令とが用意されており，ユーザは，これらのベクトル命令を意識し



(a) パイプライン乗算　(b) パイプライン加算

図10 倍精度パイプライン演算の例

Fig. 10 Examples of double precision pipeline calculation.

てフォートランでプログラミングを行う．スーパコンピュータ，あるいは汎用機の内蔵型ベクトルプロセッサでは，ユーザがベクトル命令を意識することなくプログラミングできるようにフォートランコンパイラによりベクトル命令に展開する方式を採っているが，十分な性能を得るためにはプログラミングの工夫が必要といわれている．本ベクトルプロセッサでは性能重視の観点から，ユーザがベクトル命令を用いてプログラミングする方式を採った．また基本ベクトル命令でプログラミングして得られる以上の高速化を必要とする場合は，ユーザが新たな命令の定義により対処する．すなわち単独モードで動作するマイクロプログラムを記述することにより大幅な性能向上が得られる．

6. 性能評価

内蔵型ベクトルプロセッサの性能を評価するために4章で述べたスパース連立方程式を解くための LU 分解および前進，後退代入をマイクロプログラミングしてユーザ定義命令とした．高速化を図るため，ベクトルプロセッサ内のローカルメモリにデータが転送された後起動されるものとし，V 90/50 とは独立に動作する単独モードで記述した．

表3に370次元スパース連立方程式の求解時間を V 90/50 および汎用大型計算機 HITAC M 200 H と対比して示す．スパース連立方程式の求解ルーチンには，基本ベクトル命令で置き換わる部分がなく，また基本ベクトル命令で置き換わるように，プログラムを修正しても高速化の可能性が少ないため基本ベクトル命令を用いた演算は行っていない．

ベクトルプロセッサを V 90/50 に接続することに

表 3 370 次元スパース連立方程式の求解時間
Table 3 Calculation time for the 370-th order sparse matrix solution.

(単位: ms)

処理ルーチン	ベクトルプロセッサ	V 90/50	M 200 H
LU 分解	9.5	97.0	15.8
前進および後退代入	3.1	47.4	4.6
計	12.6 (1)	144.4 (11.5)	20.4 (1.6)

( ): ベクトルプロセッサを1としたときの処理時間

より，スパース連立方程式の求解処理全体で，V 90/50 単体時の 11.5倍，M 200 H の 1.6 倍の高速化が実現できることが明らかとなった．表中のベクトルプロセッサの求解時間には，V 90/50 のメインメモリとベクトルプロセッサ内のローカルメモリ間でのデータ転送時間は含まれていないが，データ転送時間は，1 ms 程度であり，データ転送時間を含めても同等の高速化が達成できる．

4.2節で述べたインデックス付アドレス計算およびインデックスサーチによる高速化の効果は，文献9)において高速化を行わないときの 1.6 倍と評価されているが，表3より，高速化を行わなければ，M 200 H と同等の性能であることがわかる．

7. むすび

スーパミニコン HIDIC V 90/50 に内蔵するベクトルプロセッサの高速・高精度化のアーキテクチャについて述べた．倍精度および部分倍精度パイプライン演算方式を実現することにより，内積演算，関数演算が桁落ちなく単精度演算とほぼ同一の演算ステップで可能となった．またアドレス演算ユニットの強化により，スパース行列演算の高速化が可能となり，スパース連立一次方程式の求解では，HITAC M 200 H の 1.6 倍の高速化が実現できることが明らかとなった．

謝辞 最後に本研究の遂行にあたりご助言いただいた大みか工場計算機設計部井手部長，日立研究所第8部西原部長，また直接開発を担当いただいた同所82研究室稲田氏，日立エンジニアリング(株)ソフトウェアシステム部小林，中沢両氏に感謝の意を表する．

参考文献

1) Russel, R. M.: The Cray-1 Computer System, *Comm. ACM*, Vol. 21, No. 1, pp. 63-72 (1978).
2) 小高，河辺：超高速演算の動向，情報処理，Vol. 21, No. 9, pp. 927-937 (1980).
3) Charlesworth, A. E.: An Approach to Scientific Array Processing: The Architectural Design of the AP 120 B/FPS-164 Family, *IEEE Comput.* Vol. 14, No. 9, pp. 18-27 (1981).
4) Pottle, C.: Solution of Sparse Linear Equations Arising from Power System Simulation on Vector and Parallel Processors, *ISA Trans.*, Vol. 18, No. 3, pp. 81-88 (1979).
5) 二宮：数学ソフトウェアの現状と問題点，情報処理，Vol. 23, No. 2, pp. 109-117 (1982).
6) Wilkinson, J. H.(一松訳)：基本演算における丸め誤差解析，培風館，東京 (1974).
7) IEEE Task P 754: A Proposed Standard for Binary Floating-Point Arithmetic, *IEEE Comput.*, Vol. 14, No. 3, pp. 51-62 (1981).
8) 山内他：電子計算機のための数値計算法 I，培風館，東京 (1965).
9) 高藤他：スパース行列演算の高速処理方式，情報処理第26回全国大会，IP-5 (1983).
10) Berry, R. D.: An Optimal Ordering of Electric Circuit Equations for a Sparse Matrix Solution, *IEEE Trans. Circuit Theory*, Vol. CT-18, No. 1, pp. 40-50 (1981).

(昭和58年 8 月 3 日受付)
(昭和59年 1 月17日採録)