計算機アーキテクチャ 73－8
（1988. 10. 31, 11. 1）

マルチベクトルプロセッサＶＰＰの処理方式

関戸 一紀　前田 明　真鍋 俊彦　岩佐 繁明　井上 淳　橋本 智
株式会社　東芝　総合研究所

画像処理を応用の１つとする並列処理システムＶＰＰ（Variable Processor Pipeline)を開発している。ＶＰＰは多数のベクトルプロセッサＰＵを高速で柔軟な結合部で結合したＭＩＭＤ方式のマルチプロセッサである。結合部はその基本構成要素であるＢＬを矩形状に配置し、Ｓループと Ｄループの２つのループで接続した簡単な構造をしている。結合部は１対１のデータ転送の他に１対多のデータ転送やステータスの返送機能を持っている。また、ＶＰＰのプログラミング言語として、ＦＯＲＴＲＡＮに並列性を記述する機能を付加した言語とその処理系を開発した。本言語ではデータ転送を陽にＳＥＮＤ，ＲＥＣＥＩＶＥで記述する。なお、その実現には結合部の機能を十分に活かせるようにした。

AN ARCHITECTURE OF MULTIPLE VECTOR PROCESSOR SYSTEM VPP

Kazunori SEKIDO　Akira MAEDA　Toshihiko MANABE
Shigeaki IWASA　Astushi INOUE　Satoru HASHIMOTO
R&D Center, Toshiba Corporation
1,Komukai, Toshiba-cho, Saiwai-ku, Kawasaki, 210, Japan

We are developing a parallel processing system VPP(Variable Processor Pipeline) which aims at many applications especially image processings. It consist of many element processors PU's(Processor Uint) and a high speed flexible network. Each PU is a kind of vector processor, and all PU's execute programs based on an MIMD manner. The network consists of many BL's(Bus Logic unit) which are arranged in a rectangular form and connected by S-loop and D-loop. It has not only a one-to-one transfer mode but also a one-to N, broad-casting transfer mode. We have also developed a parallel programming language for VPP which is an extension of FORTRAN. Data transfer programs are wirtten explicitly using SEND,RECEIVE primitives to make full use of the network capability.

1. はじめに

画像処理をはじめとする科学技術計算の種々の分野において高速処理の要求が高まっており、各種の並列処理計算機が提案されている。我々は、通産省工業技術院の大型プロジェクト「科学技術用高速計算システムの研究開発」の一環として、画像処理を応用の１つとしたＭＩＭＤ型の高機能並列処理プロセッサＶＰＰ（Variable Processor Pipeline）の研究開発を進めてきた。

一般に、画像処理では１画素を並列処理の単位とする細かな並列性と、ある領域または１つの処理を並列処理の単位とする粗い並列性が考えられる。前者の並列性を利用したシステムとしては、各画素対応に二次元状にプロセッサを配置したＭＰＰやＣＡＰ等がある。機能が単純なプロセッサを非常に多く用いることにより極めて他界性能を実現しようとしている。後者の並列性を利用したシステムとしては、ＴＩＰ－１等がある。パイプライン方式、マルチプロセッサ方式により高性能で柔軟性のあるシステムを追求している。

ＶＰＰは後者に属する並列プロセッサであり、複数のベクトルプロセッサＰＵ(Processor Unit)を柔軟で高速な結合部で結合した方式を採用している。現在、ＰＵの主要部分をゲートアレイ化することによりＰＵを１枚の基板にまとめ、ＰＵ８台からなるＶＰＰパイロットモデルの開発評価を行っている。以下では、前半でＶＰＰ全体について、後半では並列処理に焦点を当て、結合部や言語について説明する。

2. ＶＰＰの構成

ＶＰＰは図１に示すように、多数のベクトルプロセッサＰＵ，画像メモリ，制御プロセッサを高速な結合部で結合した構成である。制御プロセッサは、プログラムのコンパイル、アセンブル、リンクなどの言語処理系の実行と、プログラムのロード、ＰＵの管理、ＰＵから要求される入出力などのＰＵとのインターフェィス処理を行う。なお、一旦起動されたＰＵは制御プロセッサの介入なしに、ＰＵ間で同期を取りながら自主的にプログラムの実行を行う。処理対象となる画像データはあらかじめ画像メモリに格納して置き、その部分画像をＰＵが順次読み出して処理を進めていく。

ここで、一連の画像処理、処理１（歪み補正）、処理２（フィルタリング），処理３（ＦＦＴ）を全画素に対して施す場合を例に、ＶＰＰの処理形態を考える。ＶＰＰでは、全ＰＵが同じ処理を分担するのではなく、図１に示すように処理１をＰＵ１が、処理２をＰＵ２，ＰＵ３が、処理３をＰＵ４，ＰＵ５，ＰＵ６が担当する。ＰＵ１は画像メモリから小部分画像を最初にＰＵ１が読みだし、処理１を行って結果を画像メモリに戻さずに処理２を担当するＰＵ２かＰＵ３に送る。これを受け取ったＰＵは処理２を行って次の処理３を担当するＰＵ４，ＰＵ５，ＰＵ６のいずれかにこの部分画像を送る。これを受け取ったＰＵは処理３を行った後で画像メモリに書き込む。これらの処理を全部分画像に対して施し、一連の処理を行う。この方法はＰＵが１つのパイプラインを構成し、その中を小画像データが流れていくように見なせるので、これをプロセッサパイプライン方式と呼ぶ。ＶＰＰでは与えられた処理に応じてプロセッサパイプラインを自由に設定（パイプラインの構成がアーキテクチャによって固定されない）できる。この意味で本システムをＶＰＰ…Variable Processor Pipeline と呼んでいる。

3. ＰＵの特長

ＰＵは図２に示すように、

a．プログラムや制御情報を格納するプログラムメモリ（ＰＭ）

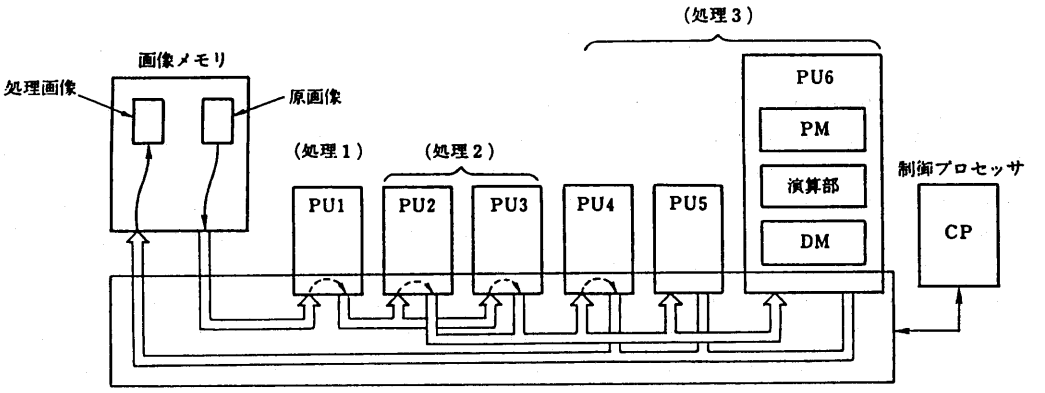

図１．ＶＰＰの構成

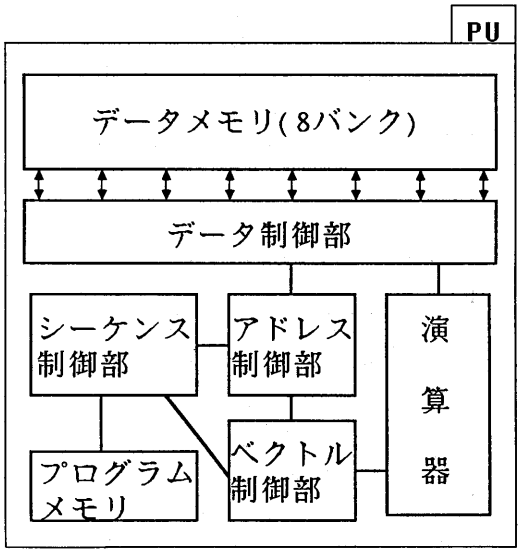

図２　　ＰＵの構成

ｂ．演算対象となるデータを格納するデータメモリ（ＤＭ）

ｃ．ＰＵ全体を制御するシーケンス制御部

ｄ．ベクトル演算などでのパイプラインを制御するベクトル制御部

ｅ．オペランドや演算結果のデータパスを制御するデータ制御部

ｆ．ベクトル演算のオペランドのアドレスを生成するアドレス制御部

ｇ．ベクトル／スカラー演算を実行する演算部

から構成される。以下にＰＵの特長を説明する。

①配列の添字の集合をインデックスセットとして定義し、これに対して陽に操作する機能を有する。この機能を用いて条件分岐や間接アドレス演算を含む処理も効率良くベクトル的に実行できる。

②シーケンス制御部とその他の制御部は並列に動作し、ベクトル演算中に次の演算の準備や種々の割り込み処理を行うことができる。これにより演算部の稼働率を高めることができる。

③内部に画像処理専用の機能を持っている。Ｘ座標、Ｙ座標のペアをピクチャインデックスとして定義し、これを用いて２次元アドレスで配列をアクセスできる。また、ＦＦＴ演算で段数が進むにつれてベクトル長が短くなるのを防ぐ機構を設け、ＦＦＴを効率良く実行できる。

④バンク選択方式が工夫されており、プログラミングの際にユーザがメモリにおけるアクセス競合を考慮しなくても十分な性能が得られる。また、ＤＭへのアクセスにおいて生成されたアドレスを格納するアドレスレジスタを２重化することにより、アドレスの競合が発生しても性能低下が小さくなるようにした。

⑤パイプラインの空きを最小限に抑えるため、パイプラインレベルの同期に種々の機構を用意している。これらの機構により、演算間のデータ依存等による同期を効率良く制御できる。

４．　結合部

並列処理システムにおいて性能向上を計るには、並列化に伴うオーバヘッドをできる限り少なくする必要がある。そこで、ＶＰＰの結合部にはプロセッサ間でのデータ転送にかかる時間を極力押さえるため、次のような特徴がある。

①各ＰＵは演算結果を直接（ＤＭに一度バッファリングすることなしに）結合部へデータを転送できる。このため、ベクトル長が長い場合、転送時間はその演算時間とオーバラップされ、見掛上は陽に現れない。

②全ＰＵが同時にデータ転送を行っても（転送先がすべて相互に異なっていれば）転送路上では競合が生じない。

③ＰＵを幾つかのグループの分け、グループを対象としたデータ転送や同期がとれる。つまり、グループ内の全ＰＵに同一のデータをブロードキャスト転送したり、グループ内の全ＰＵが相互に同期を取ったりできる。

④転送はすべてブロック転送で行い、転送に伴うオーバヘッドを軽減している。

⑤データが送られている経路を用いてステータスを返すことができる。即ち、データの受信側ＰＵのステータスを別のデータ転送で送るのではなく、データが送られている経路を逆方向に伝わってステータスを返す。

４．１　結合部の構成

結合部の構成を図３に示す。結合部は各モジュール（ＰＵ，画像メモリ，制御プロセッサなど）に対応させてＢＬ(Bus Logic) を設け、これを矩形状に並べて各行方向、各列方向をループ状に接続した構成をしている。行方向のループをＤ(Destination) ループ、列方向のループをＳ(Source)ループと呼ぶ。このループ上をデータを転送するスロットが各ＢＬに対応して回っている。例えば左端のＳループではＢＬ0，ＢＬ3，ＢＬ6，ＢＬ9 に対応したスロットが、最下段のＤループではＢＬ9，ＢＬ10，ＢＬ11に対応したスロットが回っている。

また、ＢＬ内にはＤループ上の各ＢＬに対応したバッファレジスタが設けられている。例えばＢＬ6 には図４に示すようにＢＬ6，ＢＬ7，ＢＬ8 に対応したバッファレジスタが、ＢＬ10にはＢＬ9，ＢＬ10，ＢＬ11に対応したバッファレジスタが存在する。バッファレジスタはコントロール、

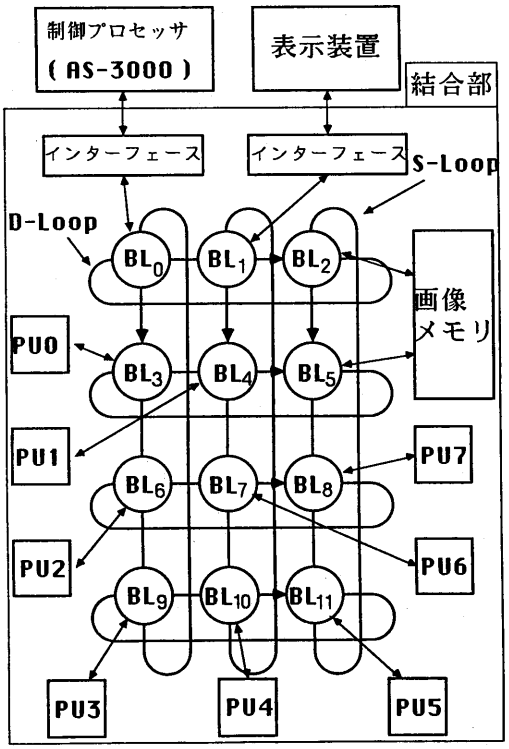

図４　　結合部の構成

グループ、データの３つのフィールドから構成され（図４）、コントロールはバッファレジスタの状態を、グループは対応するＢＬが属しているグループ番号を、データは転送データを保持するために用意されている。

４．２　１対１のデータ転送

上記スロットとバッファレジスタを用いて、下記の手順で１対１のデータ転送が行われる。

①データを送りたいモジュールはＢＬを介してＳループ上の自分に対応したスロットにデータをのせる。Ｓループ上にはデータの送り手に対応したスロットが回っており、送り手は必ずスロットにデータをのせることができる。

②Ｓループ上を順々に転送されたデータは、転送先のＢＬが含まれるＤループとの交点に来た時、その交点に位置するＢＬに一時保持される。このデータはそのＢＬ内にある転送先のＢＬに対応するバッファレジスタのデータフィールドに格納される。

③データフィールドに格納されたデータは、Ｄループ上を転送先に対応したスロットが回って来た時に、そのスロットに移される。Ｄループ上には受け手に対応したスロットが回っており、転送先が同じでない限りデータを移すスロットは空であり、必ずデータをのせることができる。

コントロール　グループ　データ

ＢＬ０		１	－
ＢＬ１		３	１００
ＢＬ２		２	５０

ＢＬ０内のバッファレジスタ

コントロール　グループ　データ

ＢＬ０		２	５０
ＢＬ１		１	－
ＢＬ２		２	５０

ＢＬ０内のバッファレジスタ

コントロール　グループ　データ

ＢＬ０		３	１００
ＢＬ１		２	５０
ＢＬ２		２	５０

ＢＬ０内のバッファレジスタ

図４　　ＢＬ内のバッファレジスタ

④Ｄループ上のデータは目的のＢＬに来た時に、スロットから取り込まれ対応するモジュールに送られる。

なお、Ｓループ、Ｄループ上のスロットはＰＵの１マシンサイクル以内の時間でループを一周するため、ベクトル演算の結果をそのまま転送しても演算を遅らせることにはならない。上記の転送手順から明らかなように、転送先が異なる限り任意の２つのデータ転送間でＳループ、Ｄループのスロットを奪い合う事は無く、このため競合は発生しない。また、モジュール間のデータ転送は全て同一の手順で行われるため、物理的な位置に関係なく各モジュールの接続は同等である。このため、任意の形態のプロセッサパイプラインを構成できる。

４．３　１対多のデータ転送

同様に、上記スロットとバッファレジスタを用いて、下記の手順で１対多のデータ転送が行われる。

①データを送りたいＰＵはＢＬを介してそのデータと転送先のグループ番号をＳループ上の自分に対応したスロットにのせる。

②このスロットはＳループ上を順々に転送されループを一周する。この間、このスロットが到着したＢＬではスロットにのっているグループ番号とＢＬ内の全バッファレジスタのグループフィールドとの比較を行い、値が一致するバッファレジスタにだけスロットにのっているデータを格納する。この処理をスロットの周回に従ってＳループ上の全

ＢＬに対して行うことにより、Ｓループ上の全ＢＬの全バッファレジスタの中で、グループフィールドが指定されたグループ番号と一致する全てのバッファレジスタにスロットのデータが格納される。（図４参照）

ここで、各モジュールが属するグループの番号は、そのモジュールが接続されているＢＬが含まれるＤループ上の全ＢＬにあらかじめ伝えられており、各ＢＬ内のそのモジュールに対応するバッファレジスタのグループフィールドにセットされているものとする。

③②の処理で各ＢＬに保持されたデータは、Ｄループ上を転送先グループに属するモジュールに対応したスロットが回って来た時に、ＢＬ内のそのスロットに対応したバッファレジスタを調べることにより検出され、そのスロットに移される。この処理は②でデータを保持した各ＢＬで独立に行なわれ、各Ｄループで並列にデータが転送される。

④Ｄループ上のデータは１対１のデータ転送の場合と同様にして、目的のＢＬに到着した時に取り込まれ対応するモジュールに送られる。

以上のように、結合部ではＢＬ内のバッファレジスタのグループ番号と転送先のグループ番号が一致する全モジュールにブロードキャスト転送することができる。ブロードキャスト転送は同一のプログラムを複数のＰＵにロードする場合や、画像メモリ内のデータを複数のＰＵに転送する場合にデータ転送回数を減らすために用いる。

４．４　ステータスの返送

結合部を用いて行なわれるデータ転送はベクトル演算との整合性やオーバヘッドを削減するため、全てブロック転送である。ブロックデータの最初のデータが転送される時、その経路の途中にあるＢＬ内のバッファレジスタやＤループ上のスロットにロックを掛け、転送先が同じ転送が同時に発生しても２つのデータが混ざらないようにしている。よって、ブロック転送の途中で、実際にデータが送られていなくても、通信経路は確保されたままになっている。この状態を利用して、下記の手順でステータスの返送を行う。

①受け手のモジュールはＢＬを介してステータスをＤループ上の自分に対応したスロットにのせる（Ｄループでは先に述べたようにデータの受け手に対応したスロットが回っている）。

②Ｄループ上を順々に転送されたステータスは、そのスロットをロックしたＢＬに到着した時そのＢＬに一時保持される。保持されたステータスはＢＬ内の受け手に対応したバッファレジスタのコントロールフィールドに格納される。

③このステータスはＳループ上の送り手に対応したスロットが回って来た時、そのスロットに移される。

④Ｓループ上のステータスは送り手のＢＬに到着した時取り込まれ、対応するモジュールに送られる。

以上のようにして、受け手から送り手へステータスを返すことができる。なお、ステータスはスロットにデータがのっていない場合にのみ返せる仕様になっており、送り手と受け手の間でいつステータスを返すかあらかじめ決めておく必要がある。この機能の使い方は次の章で説明する。

５．　並列処理記述言語

ＶＰＰにおける並列動作を記述する言語として、ＦＯＲＴＲＡＮにＳＥＮＤ，ＲＥＣＥＩＶＥなどのデータ転送やＰＡＲＡ－ＰＲＯＧＲＡＭなどのプロセス定義を記述する文を加えた言語を開発した。並列処理システムのプログラミング言語には種々のレベルが考えられる。

①対象となっているシステムのアーキテクチャをベースとして新しい言語を作る方法

②既存の言語をそのまま用いて並列に実行できる部分をその中から取り出す方法

③既存の言語に並列処理記述の機能を追加する方法

などがある。我々は以下の理由により③のアプローチを取った。ＶＰＰはＭＩＭＤ方式を取っており比較的汎用なタイプであるため、それに適した新しい言語を定め切れなかった。②の方法では並列性を取り出せる部分が限られてしまい、ＶＰＰの性能を十分に引き出せない恐れがある。また、並列アルゴリズムの研究には直接並列処理を記述できた方が良いことや、将来②のアプローチを取る場合にも③の言語はその中間言語と見なせ、②への拡張が容易であると考えられる。

５．１　並列処理プログラムの構成

並列実行の記述をシーケンシャルな言語に付加する場合、ｆｏｒｋ文や並列セパレータ等を導入してステートメントレベルで行う方法と、並列に実行される単位（プロセスなど）を予め定義しておきそれに起動を掛ける方法が考えられる。本言語ではＶＰＰには画像メモリを除くとＰＵ間で共有できるメモリがなく、前者の方法では大量の実行環境のコピーが必要となることから後者の方法を採用した。また、現段階のＶＰＰではマルチプロセスの管理や動的な資源管理の機能をサポートしておらず、リンク処理のロードモジュール生成時に各並列実行単位がどのＰＵで実行されるか決定しなければならない。これらの点を考慮して、概略以下のような言語仕様を決定した。

①並列実行単位を記述するためにＰＡＲＡ－ＰＲＯＧＲＡＭ文を用いる。この文とＥＮＤ文で囲まれたステートメントとその中で呼ばれる関数、サブルーチンが並列実行の単位である（これをパラプログラムと呼ぶ）。この中に通常の演算のほかに他のパラプログラムとのデータ転送を記述する。

②ＰＡＲＡＬＬＥＬ文は実際に並列実行されるパラプログラムを指定する文である。パラプログラムはその名前で指定するが、ＰＲＯＧ（ＮＰ）のような配列の形式で記述することによりパラプログラムの配列を取り扱える。
③ＥＸＥＣ文はＰＡＲＡＬＬＥＬ文で指定した全てのパラプログラムに起動を掛け、それらが全て終了するのを待つ実行文である。これによりシーケンシャルの実行する部分と並列に実行する部分の橋渡しを行う。
④ＬＩＮＫ文はロードモジュール生成時に値を決定する変数（リンク変数と呼ぶ）を指定する文である。リンク変数は配列やパラプログラム配列の大きさを指定するのに用いる。現段階のＶＰＰではプログラムの並列度が実際のＰＵ台数を越えられないので、リンク変数により実際のＰＵ台数に適合したプログラムを再リンクだけで生成できるようにした。
⑤プログラムの見通しを良くするため、並列処理に関連する文（ＰＡＲＡＬＬＥＬ，ＥＸＥＣ）を書ける部分を制限し、これをサブルーチン化した。これをパラサブルーチンと呼び、ＰＡＲＡ－ＳＵＢＲＯＵＴＩＮＥ文からＥＮＤ文で指定する。パラプログラムはＰＡＲＡ－ＣＡＬＬ文で呼び出す。パラプログラムの中でパラサブルーチンを呼ぶことにより、並列処理のネストが行える。

５．２　データ転送の機能とその実現

ＶＰＰにおけるパレプログラム間データ転送の記述にはＳＥＮＤ文、ＲＥＣＥＩＶＥ文を用いる。転送する値としてベクトルの式を書くことにより、演算結果をそのまま相手のパラプログラムに送れる。

（１）　データ転送の機能

パラプログラム間の転送はデータの複写などのオーバヘッドは極力避けることが望ましい。ｓｅｎｄ，ｒｅｃｅｉｖｅに基づくデータ転送には同期の取り方により種々の方法が考えられるが、我々は上記条件を考慮して、同期式で非封鎖型のｒｅｃｅｉｖｅでデータ転送することにした。

データ転送には同期式（ｓｅｎｄは対応するｒｅｃｅｉｖｅが実行されるまで待つ）と非同期式（ｓｅｎｄは対応するｒｅｃｅｉｖｅが実行されていなくてもシステムバッファにデータを格納してｒｅｃｅｉｖｅが実行されるまで待たない）があるが、非同期式では大きなバッファ領域とデータの複写が必要でありオーバヘッドが避けられない。なお、転送データのバッファリングにより性能向上が見込まれる処理に対しては、受信用配列をリングバッファ（配列の先頭アドレスを順次変更するするだけでデータの複写は行わない）で構成するように言語レベルでサポートしている。

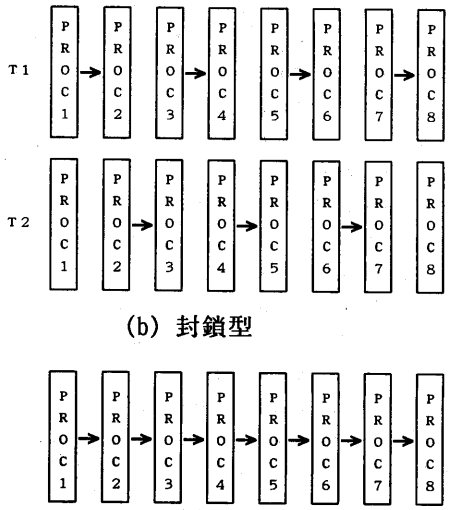

図５　　データ転送の例

また、ｒｅｃｅｉｖｅには最も一般的な封鎖型（ｒｅｃｅｉｖｅするデータが送られて来るまで待つ）と非封鎖型（ｒｅｃｅｉｖｅするデータが送られて来ない場合、ｒｅｃｅｉｖｅを実行した事を記憶して先に進む）があるが、封鎖型では全パラメータが同時にデータ転送する場合効率良くこれを実行できない。例えば図５（ａ）のように全パラプログラムが隣のパラプログラムに転送を行う場合、封鎖型では図５（ｂ）のように最初に偶数番目のパラプログラムがｓｅｎｄ，奇数番目のパラプログラムがｒｅｃｅｉｖｅを実行し、次に奇数番目のパラプログラムがｓｅｎｄ，偶数番目のパラプログラムがｒｅｃｅｉｖｅを実行するようにプログラムしなければならない。また、非封鎖型ではｒｅｃｅｉｖｅとｒｅｃｅｉｖｅしたデータの参照の間にある程度時間がありその間にデータ転送が終了する場合、受け手のパラプログラムではデータ転送の時間をまったく意識しなくて済むというメリットもある。

（２）　データ転送の実現

本データ転送を実現するフローチャートを図６に示す。

まず、送信側が数語からなる制御情報を送信し、結合部を通して受け手からのステータスが返って来るのを待つ。制御情報にはこれから転送を行う受信側の変数（配列）に関する情報が含まれる。受信側ではこの制御情報によりシーケンス制御部に割り込みがかかり、制御情報に従ってその変数に対してｒｅｃｅｉｖｅが実行されたかチェックする。ｒｅｃｅｉｖｅの対象となる変数は図７に示すように

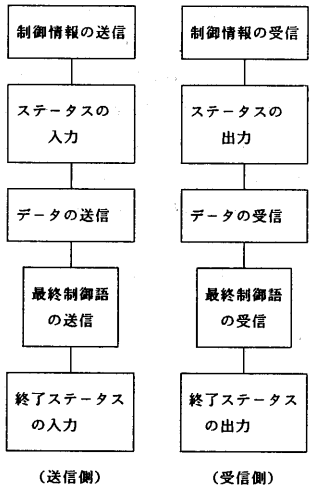

図６　　データ転送のフローチャート

ｆｌａｇとデータ領域へのポインタからなる管理情報を介して構成されている。ｆｌａｇはｒｅｃｅｉｖｅが実行されてデータ領域が空である時０で、データ領域に転送データが入っている時１になっている。よって、制御情報で指定された変数のｆｌａｇが０であるかチェックし、０であれば転送許可ステータス（ａｃｋステータス）を，１であれば転送不許可ステータス（ｎａｃｋステータス）を結合部に出力する。

このステータスは４．４節で述べたように、転送経路を逆方向に伝わって送信側に返送される。受信側はａｃｋステータスを出力した場合、受信用レジスタにデータ領域のアドレスを設定して割り込み処理を終了する。なお、３章で述べたように、シーケンス制御部とその他の制御部は並列に動作するので、割り込み処理中も受信側は演算を行っており、割り込みによる性能低下はほとんどない。

次に、送信側では返送されたステータスがａｃｋである

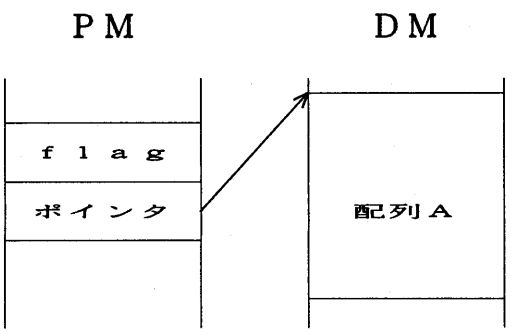

図７　　RECEIVE 変数の管理情報

ことを確認し、転送データを生成するベクトル演算を実行する。この演算結果は直接（一度内部メモリに格納されることなく）結合部を通して転送される。受信側では送られて来たデータを先に設定した受信用レジスタに従い自動的にメモリに格納される。

ベクトル演算によりデータの転送が終了した後で、送信側ではさらに終了制御語を送信する。終了制御語には送信側でデータ転送中にメモリパリティエラーや演算エラーが発生したか否かの情報がのせられている。受け手側ではこの終了制御語により再度シーケンス制御部に割り込みがかかり、終了制御語の内容およびデータ受信中のパリティエラーを調べ、両方にエラーがなければデータを受信した変数のｆｌａｇを１に設定する。さらに、エラーの有無に拘らず受信中のパリティエラーの情報を終了ステータスとして結合部へ出力し、割り込み処理を終了する。送信側ではこの終了ステータスにより受信側のパリティエラーの有無を知る。

制御情報の転送後にｎａｃｋステータスが返って来た場合やデータ転送中に送信側か受信側にエラーが発生した場合は、もう１度図６の処理を繰り返す。なお、図６のフローチャートはＳＥＮＤ文に関連するものであるが、ＲＥＣＥＩＶＥ文は単に受信する変数のｆｌａｇを０に設定するだけでよい。転送データが格納される変数を参照する時、そのｆｌａｇが１であることを常に確認し、転送データが格納されていない変数を参照しないようにしている。

以上のようにデータ転送を構成することにより、（１）で述べた機能を実現できるとともに、受け手側の演算をほとんど妨げることなくデータ転送できる。また、データ転送のパスを逆方向に伝わってステータスを返送でき、受信側がデータ転送中であっても図６のフローは支障をきたさない。

（３）　プロセッサパイプラインの記述

プロセッサパイプラインでは複数台のＰＵの中から１台を選んでそのＰＵにデータを転送する必要がある。図１の場合、ＰＵ１はＰＵ２かＰＵ３の一方にデータを送らなければならない。この処理を記述するのにＳＥＮＤ－ＳＯＭＥＯＮＥ文がある。図８に図１のプロセッサパイプラインのプログラム例を示す。ＰＵ１は画像メモリからのデータを変数Ａに受取り処理１を行う。その結果をＳＥＮＤ－ＳＯＭＥＯＮＥ文でパラプログラム配列Ｐ２の１つに転送する。ＰＵ２，ＰＵ３ではＰＵ１からのデータを変数Ｂに受取り処理２を行い、その結果をＳＥＮＤ－ＳＯＭＥＯＮＥ文でプログラム配列Ｐ３の１つに転送する。ＳＥＮＤ－ＳＯＭＥＯＮＥ文では転送先のパラプログラム配列の名前を指定することにより、パラプログラム配列の中の１つにデータ転送を行う。

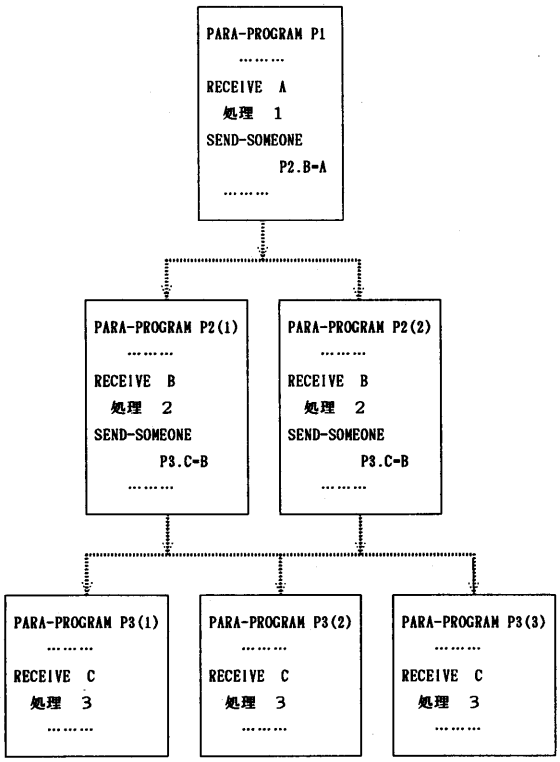

図８　プロセッサパイプラインのプログラム

なお、ＳＥＮＤ－ＳＯＭＥＯＮＥ文の実現は図６のデータ転送の部分はＳＥＮＤ文と同じであるが、受信側からｎａｃｋステータスが返って来た場合パラプログラム配列の他のパラプログラムに転送先を切り換えて（ポーリング）データ転送を行う。これにより遊んでいるＰＵに先にデータが行くようにしている。

６．　おわりに

ＶＰＰの全体構成、その単位プロセッサを結合する結合部、ＶＰＰ用並列記述言語について説明した。ＶＰＰは、非常に高速な結合部に複数のベクトルプロセッサを結合した並列システムで、多くの特長がある。本稿ではその中で、並列処理に焦点を当てて説明した。

現在、８台のＰＵから構成されるパイロットモデルを開発・評価を進めている。今後、実際のアプリケーションによりシステム全体の、特にプロセッサパイプライン方式の性能評価を進めていく予定である。また、プロセッサパイプラインにおける動的負荷分散についても検討してみたい。

参考文献

［１］関戸他：並列処理記述様言語の概要，第30回情報処理全国大会４Ｂ－１，１９８５

［２］前田他：並列処理装置の結合方式，第31回情報処理全国大会２Ｄ－２，１９８５

［３］関戸他：ＶＰＰにおけるプロセッサ結合方式，第36回情報処理全国大会６Ｃ－１，１９８８

［４］真鍋他：並列処理システムＶＰＰのアーキテクチャ，情処アーキテクチャ研究会，８８－ＡＲＣ－７１，１９８８