計算機アーキテクチャ 60－6
（1985. 12. 6）

データフロー型イメージ・パイプラインド・プロセッサLSI，ImPPについて

松本 恵治
（日本電気）

1. はじめに

最近のディジタル画像処理技術の進展には、目覚ましいものがあり、リモートセンシング，医用画像処理，FA（ファクトリ・オートメーション）、OA（オフィス・オートメーション）などの広い分野で応用されている。しかしながら、ディジタル画像処理は、一般に繰り返し演算を多量のデータに対して施こす必要があり、膨大な計算量となる。例えば、ディジタル画像処理のひとつである3×3のコンボルーションを512×512の画素に行なうには、約230万回の乗算と約200万回の加算を必要とする。この処理を通常の16ビットマイクロプロセッサを用いて行なわせると約40秒もの長時間を要する。このため、処理の高速化を図るには、ビットスライス型プロセッサや専用ハードウェアを用いて実現することとなり、処理装置は、大型で高価なものとなる。

ここで紹介するイメージ・パイプラインド・プロセッサ（μPD7281，以降ImPPと称す）は、小型、高性能で低価格なディジタル画像処理装置を実現することを目的として開発したもので、この目的達成のために、VLSIテクノロジを用い、データフロー方式とパイプライン方式を基本アーキテクチャとして採り入れている。本稿では、ImPPのアーキテクチャ，内部ハードウェア構成，システム構成例ならびに処理性能について述べる。

2. アーキテクチャ

ImPPは、大量データに対する繰り返し演算を高速に行なわせるために、非ノイマン型アーキテクチャのひとつであるデータフロー方式と、処理の高速化手法のひとつであるパイプライン方式を基本アーキテクチャとして並用している。

2.1 データフロー方式

今日、一般的に用いられているノイマン型アーキテクチャにもとづくコンピュータでは、処理対象のデータだけでなく、命令コードやアドレスを生成するためのデータなども、ひとつのバスを介してプロセッサとメモリ間を移動するためにフォン・ノイマン・ボトルネックと呼ばれるバスネックが生じる。また、プログラムで決められたシーケンスに従って処理を進めるので、基本的に処理の並列化が困難である。これらの面を打開するために、1972年にMITのデニスらによって提唱された非ノイマン型アーキテクチャのひとつであるデータフロー方式をImPPに採用している。

データフロー方式は、処理に必要なデータがそろった時点で、処理を実行する方式で、実行指示情報をもったデータ（トークン）を入力することにより、トークンが、演算器を求めて移動し、処理がなされる。これは、ノイマン型アーキテクチャでの演算器が必要とするデータをメモリからフェッチする方法とは、逆の発想である。この方式の採用により、バスネックの解消ならびに並列処理を可能とし、高速処理を実現している。

2．2　パイプライン方式

パイプライン方式は、処理過程をいくつかのフェーズに分割し、流れ作業的に処理を実行し処理時間の短縮を図ろうとするもので、ディジタル画像処理のように大量のデータに対して繰り返し演算を高速に行なわせるものに適した方法のひとつである。しかしながら、通常のパイプライン方式は、各フェーズにおける処理が固定されていて、専用処理の高速化に適しているものの、汎用性をもってパイプライン処理を行なわせるのは、困難である。ImPPは、この面を打開すべくLSI内のプログラム領域をRAM（ランダム・アクセス・メモリ）で構成し、プログラム領域の書き換えを可能とし、各フェーズにおける処理をユーザプログラマブルとすることによって、汎用性のあるパイプライン処理を実現している。

また、パイプライン方式とデータフロー方式を並用することにより、ImPPのカスケード接続やリング接続を容易に実現できる。このことは、処理の並列度の向上や、プログラム領域の拡張が容易であることを示している。さらに、前述のように、パイプラインの各フェーズにおける処理を可変にできる構造を持っているImPPは、ディジタル画像処理のみでなく、一般に大量データに対する繰り返し演算の高速処理が必要なディジタル信号処理や、行列演算などの数値演算処理にも適している。

ImPPに採用しているデータフロー方式とパイプライン方式は、LSIの設計面においても優れた効果を発揮している。すなわち、パイプライン方式により、LSI内部は、パイプラインの各フェーズ毎のモジュール単位に分割できる。また、データフロー方式により、データ自身が処理内容を携えているので、各モジュール間の複雑な制御信号が不要である。したがって、各々のモジュールの回路やマスクレイアウトの設計／検証作業を複数の設計者で並行して進めることができ、かつランダムロジック部を極力減らした規則性のよい設計ができる。ImPPは、チップ全体の85％以上が規則正しい構造であり、チップの有効領域の集積密度は、3,000トランジスタ／mm^2である。

3．内部ハードウェア構成

図．1にImPPの内部ハードウェア構成ブロック図を示す。ImPPは、大きく10個のモジュールより構成されている。これらのモジュールの中で、LT，FT，DM，Q，PUが、LSI内部でパイプラインリングを構成し、ICより入力されたトークンは、この内部リングを周ることによって処理が施こされ、処理されたトークンが、OQ，OCを通ってImPPより出力される。

（1）IC（インプット・コントローラ）

入力トークンを制御するモジュールである。ImPPへの入出力トークンのフォーマットは、図．2に示すように32ビットより構成され、上位16ビットが、制御情報で、下位16ビットが、処理対象となるデータである。制御情報は、4ビットのMN（モジュール番号），7ビットのID（行き先アドレス）および4ビットのCTLF（制御フィールド）より成っている。ICは、16ビットの入力データバス（IDB0～IDB15）と、入力トークンを受け取るための2本の制

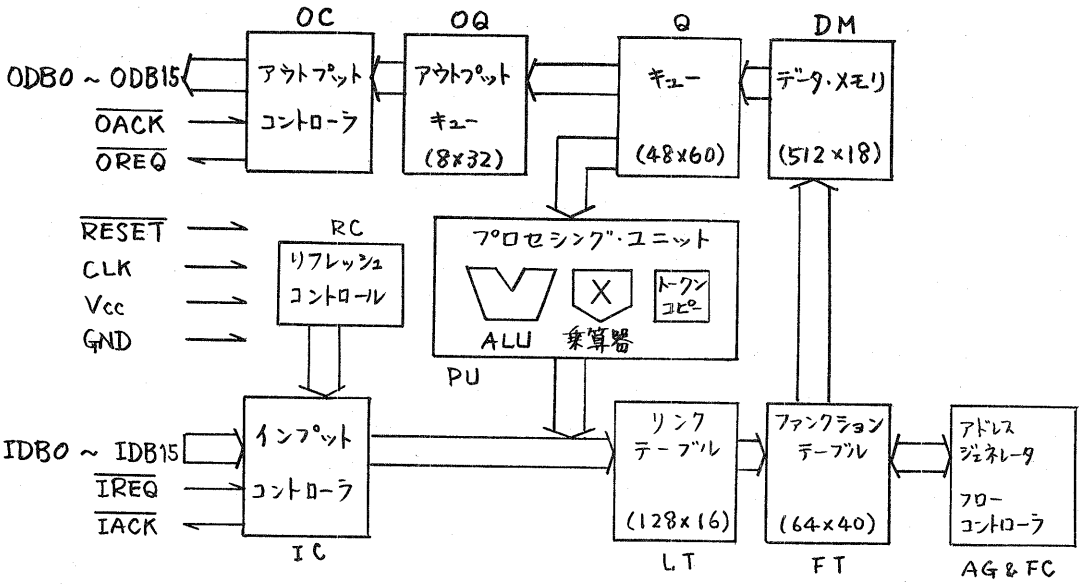


図. 1　ImPP内部ブロック図

御信号（$\overline{IREQ}$, $\overline{IACK}$）によって2度に分けて入力したトークンのMNを、ImPPのリセット時に設定したImPP自身のモジュール番号と比較し、一致した場合は、LT部へ、不一致の場合は、OC部へ送り出す働らきをする。MNは、チップレベルでの識別番号であり、IDおよびCTLFは、チップ内での識別情報ということができる。

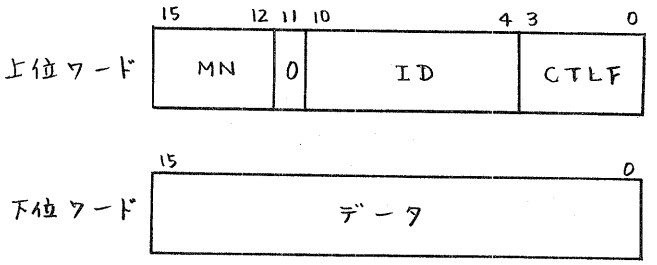


図. 2　入出力トークンのフォーマット

(2) LT（リンク・テーブル）

LTは、128ワード×16ビットのDRAM（ダイナミックRAM）で構成するプログラム領域である。LTには、次のパイプラインサイクルで参照するFTのアドレス情報や、パイプラインリングを一周して、再度LTを参照するために必要なID情報を格納する。LTに格納しているこれらの情報は、LTに到着したトークンが携えているID部によってアドレスされ、読み出されたLTの内容をトークンに新らたに付加して、次のFTへ送られる。

(3) FT (ファンクション・テーブル)

FTは、60ワード×40ビットのDRAMで構成するプログラム領域である。FTには、AG&FCで実行する処理内容や、OQ, PUで実行する処理内容が格納される。FTは、LT部で付加したFTアドレス情報によってアドレスされ、読み出された内容をトークンに付加し、AG&FC, DM, PUなどその後に続くパイプラインサイクルでの処理情報となる。

LTおよびFTのプログラム領域をRAMで構成し、ユーザプログラマブルな構造としているので、パイプライン方式を採用しているにもかかわらず、各パイプラインモジュールでの処理内容を可変とできる。

(4) AG&FC (アドレス・ジェネレータとフロー・コントローラ)

AG&FCは、DMアドレスの生成(AG)とトークンのフロー制御(FC)を行ない、FT部と同じパイプラインサイクルで動作する。AGの機能としては、DMのアドレス生成, DMへの書き込み, 読み出し制御, 二項演算のための待ち合わせ制御などがあり、FCの機能としては、このモジュールに到着したトークンのID情報の修飾によるフロー制御がある。

二項演算のための待ち合わせ制御は、このモジュールに到着したトークンの演算のパートナーとなるトークンが、まだ到着していない場合、到着したトークンをDMに書き込み、待ち合わせを行なう。パートナーが既に到着している場合は、パートナーをDMより読み出し、二項そろってQ, PUまたはOQに送って、二項演算や、二項出力を行なう。この制御は、データがそろった時点で演算を開始するというデータフローの象徴的な処理のひとつである。

(5) DM (データ・メモリ)

DMは、512ワード×18ビットのDRAMで構成する内部メモリで、AG&FCで生成したDMアドレスによって参照し、定数やテンポラリデータの格納, 二項演算データの待ち合わせなどに用いる。

(6) Q (キュー)

Qは、60ビット×48レベルのDRAMで構成するFIFO(ファーストイン・ファーストアウト)レジスタで、48レベルの中で、32レベルをDQ(データキュー)として、16レベルをGQ(ジェネレータキュー)として用いる。DQは、PUやOQに送るトークンを、GQは、ジェネレート命令(PUにおいて、ひとつのトークンより複数のトークンを生成する命令)を実行するトークンを、一時的に退避するために用いる。これは、パイプラインリング内でのトークン数が、パイプラインステージ数を超える場合があるためで、Qは、これら超過したトークンを一時的に蓄えておく。

(7) PU (プロセシング・ユニット)

PUは、FT部で付加した演算情報をもとに演算を実行するユニットである。論理演算や加減乗算, バレルシフト, 大小比較, ビット操作, ビットチェック, 倍精度調整など32種類の演算を備えている。これら以外に、数値の発生やデータのコピーなどのジェネレート命令もPU部で実行する。

(8) OQ (アウトプット・キュー)

OQは、32ビット×8レベルのSRAM (スタティックRAM) で構成するFIFOレジスタで、OCよりImPP外部へ出力するトークンを、出力ビジー状態のときに、一時的に退避するために用いる。

(9) OC (アウトプット・コントローラ)

OCは、IC部とOQ部より送られてくるトークンを、プライオリティ制御して、16ビット出力データバス (ODB0~ODB15) を介して、ImPPより出力する。トークンの出力方法は、トークンの入力の場合と同様で、2本の制御信号 ($\overline{\text{OREQ}}$, $\overline{\text{OACK}}$) によって、32ビットのトークンを、上位, 下位16ビットの順に出力する。

入力および出力のトークンの制御信号は、2線の非同期によるデータのハンドシェイクできるものであるので、複数個のImPPを構成する場合、互いの入出力データバスと入出力制御信号を結線するだけで簡単に実現できる。

(10) RC (リフレッシュ・コントロール)

RCは、内部のDRAMをリフレッシュするためのコントロール部である。内部のDRAMは、LT, FT, DM, Qに分散しているが、ひとつのリフレッシュ用トークンが、パイプラインリングを回ることにより順次これらのDRAMをリフレッシュできる。

以上説明したモジュールのうち、LT, FT, DMは1パイプラインサイクル, Q, PUは2パイプラインサイクルで処理実行する。したがって、内部のパイプラインリングは、合計7段のパイプラインで構成し、リング上に同時に7個のトークンが並行して存在できる。8個以上のトークンがリング内に存在するときに、あふれたトークンが、Qで待たされる。1パイプラインサイクルは、200ns (10MHzシステムクロック時) である。PU部には、16×16ビットの乗算器を内蔵しているが、この乗算器も200nsで実行できる。このようなハードウェア構成をもつImPPのチップ写真を図.3に示す。ImPPは、1.75ミクロン・ルールのNMOS技術を用い、6.93×6.99 mm^2 内に、115,000トランジスタを集積し、40ピンDIPのパッケージを用いている。

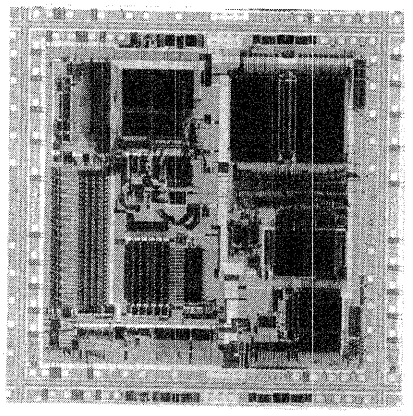
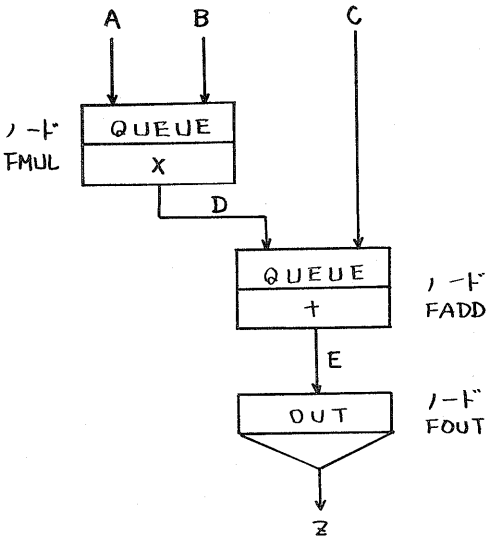


図.3 ImPPのチップ写真

次に、ハードウェア構成とプログラムの関係について述べる。データフロー型プロセッサの場合、フローグラフを用いると、プログラムを考える上で都合がよい。このフローグラフは、ノイマン型プロセッサで実行するプログラムのフローチャートに相当するものである。

フローグラフは、アークとノードの集合からなり、アークはデータの流れを、ノードはデータに対する処理内容を示している。図．4および図．5に、Z＝A×B＋Cの演算を例にとって、そのフローグラフとソースリストをそれぞれ示す。図．4で、A～E，Zがアークで、FMUL，FADD，FOUTがノードである。



図．4　フローグラフ

入力データA，B，Cが、IC部を介して入力する入力トークンに、演算結果の出力データZが、OC部を介して出力する出力トークンに相当する。これらは、図．5に示したソースリストで、INPUT文，OUTPUT文に対応する。LINK文は、LT部に格納すべきプログラムに相当し、トークンの行き先情報を示し、FUNCTION文が、FT部に格納すべきプログラムに相当し、AG＆FC部での二項待ち合わせ制御やPU部での演算処理内容を示している。MEMORY文は、DMにおける二項待ち合わせに必要なメモリエリアの確保や定数などの格納を示している。この例では、二項待ち合わせのために、AG＆FC部で実行する命令のひとつであるQUEUE命令（このQUEUEは、データを一時的退避するQモジュールでの処理とは異なる）によって、16個分の待ち合わせ領域を確保している。A×Bの乗算の場合、Aデータが到着したときに、演算対象となるBデータがまだ到着していないならば、AデータをDMに一時的に退避させ、その後Bデータが到着したときに、A，Bの二項そろってPU部へ送られて乗算がなされる。乗算結果のトークンは、アークDによって入力データCとの二項待ち合わせ制御を別のQUEUE命令によって設定したDMにおける待ち合わせ領域で行ない、ふたつのデータがそろった時点でPU部へ送られて加算を実行する。この結果のトークンが、Z＝A×B＋Cの演算結果を示し、アークE，ノードFOUTにおけるOUT1命令，アークZを通ってImPP外部へ送り出されて処理が終了する。

```
INPUT     A,B,C;
OUTPUT    Z;
LINK      D=FMUL(A,B);
LINK      E=FADD(C,D);
LINK      Z=FOUT(E);
FUNCTION  FMUL=MUL, QUEUE(BUF1,16);
FUNCTION  FADD=ADD, QUEUE(BUF2,16);
FUNCTION  FOUT=OUT1(0,0);
MEMORY    BUF1=16 DUP(?);
MEMORY    BUF2=16 DUP(?);
```

図．5　ソースリスト

LT部を、128ワード×16ビット，FT部を、64ワード×40ビットのRAMで構成しているので、ひとつのImPPで、フローグラフにおけるアーク数が128個，ノード数が64個のプログラムまで実行可能である。

4. システム構成例と処理性能

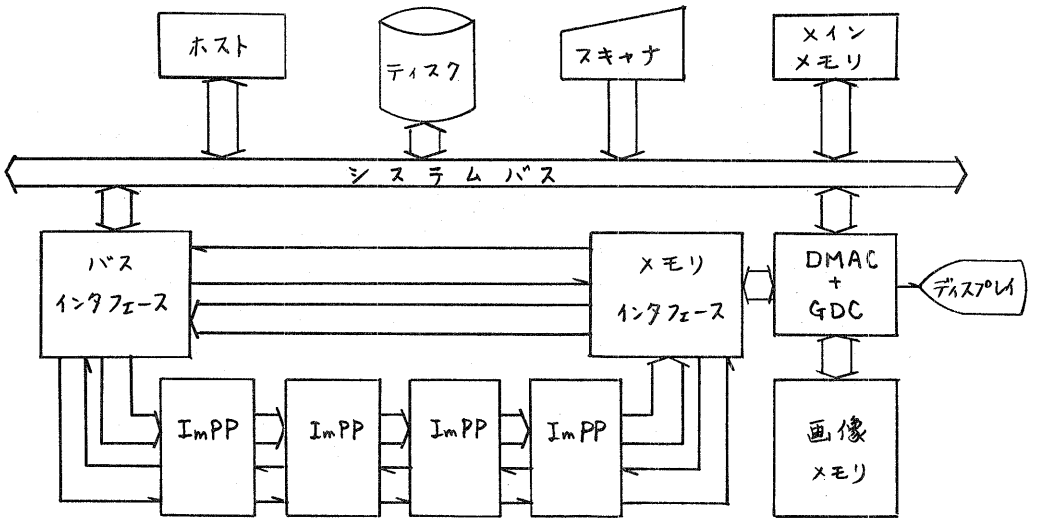


図. 6　システム構成例

図. 6 にImPPを4個用いたディジタル画像処理に適しているシステム構成例を示す。ImPPを複数個用いるシステム構成として、ImPPを縦続接続するカスケードの構成と、トークンが循環できるリング構成とが考えられる。このシステム構成例では、ImPP 4個を、バスインタフェース回路とメモリインタフェース回路を介して、リング状に接続している。画像処理を施こす画像メモリを表示メモリと兼用し、表示のためにGDC（グラフィック・ディスプレイ・コントローラ, μPD7220）を用いている。処理対象となる画像情報は、ホスト側より、バスインタフェース, ImPPおよびメモリインタフェースを介して画像メモリへ書き込む方法と、DMAC（ダイレクト・メモリ・アクセス・コントローラ, μPD8237）, GDCを介して画像メモリへ書き込む方法とが可能である。画像メモリとしては、256K×18ビットを実装している。

ImPPは、データフロー方式のプロセッサであるので、アドレスバスやデータバスの考え方がなく、アドレス情報やデータ情報を、トークンより生成する必要がある。バスインタフェース部は、ホスト側からのLTやFTへロードするプログラムや、プログラムを起動させる起動データを、上位16ビット, 下位16ビット合計32ビットのトークンの形に整形し、ImPPへ流し込み、逆に、ImPPからの処理結果データや処理終了を知らせるトークンを、ホスト側へ渡す役割をする。メモリインタフェース部は、ImPPと画像メモリとの間のデータ受渡の役割をする。ImPPからのメモリアクセス要求もトークンにより行なわれるので、トークンに、メモリの読みだし、書き込み, メモリアドレスなどの識別情報をあらかじめ決めておき、メモリインタフェース部で識別し、メモリアクセスを実行

する。小型，低価格のシステムを構成できるように，バスインタフェース回路とメモリインタフェース回路とをシングルチップ（MAGIC，メモリ・アクセス&ジェネラル・バス・インタフェース・チップ，μPD9305）化している。

アプリケーション		1 ImPP	2 ImPP	備考
二値画像	ブロック転送	30 ms	26 ms	512×512 ビットバウンダリ
	任意角回転	1.4 s	0.8 s	512×512
	縮少	0.28 s	0.16 s	1024×1024 → 256×256
	拡大	21 ms	15 ms	128×128 → 512×512
	平滑化	1.1 s	0.6 s	512×512
	エッジ検出	0.9 s	0.5 s	512×512
多値画像	階調変調	0.14 s	0.1 s	256×256
	二値化	0.13 s	0.08 s	256×256
	ディザ変換	0.13 s	0.08 s	256×256
	3×3 マスキング	2.8 s	1.5 s	512×512
	アフィン変換	0.7 s	0.5 s	256×256

表．二値画像，多値画像の処理性能

図．6に示したシステムを用いて実行した二値画像処理や多値画像処理のいくつかのアプリケーションについての処理性能を，表にまとめた。ImPPを1個用いた場合の処理速度は，μPD8086（8MHz）と比較して約10～20倍といえる。

ImPPを2個用いて実行した場合のほうが，1個使いの場合より処理速度は向上している。これは，ImPPの複数個使用により，処理の並列度が向上しているからである。表におけるブロック転送のように，ImPPを2個用いても処理速度が，それほど向上していないものがあるが，これは，ImPP内での処理が単純なものであり，処理速度が，ほぼメモリのアクセス時間で決ってしまうからである。

5. おわりに

以上述べたように，ImPPは大量データに対する繰り返し演算を高速に行なう能力に優れており，パイプライン方式とデータフロー方式の採用により，ユーザプログラマブルなマルチプロセッサ構成を容易に組み立てることができる。このため，ディジタル画像処理のみならず数値演算や信号処理の各種アプリケーションに適している。また，パイプライン方式とデータフロー方式を用いることにより，データ自身が処理内容を携え，各モジュールの制御がモジュール内で閉じているので，LSIの設計面，検証面にも優れた効果を発揮している。

このように，パイプライン方式とデータフロー方式は，装置への応用面，LSIの設計面において優れた方式ということができ，今後この種のVLSIのファミリィ展開を積極的に図っていきたい。