EC77-77

連想プロセッサのハードウェア構成と処理機能

THE STRUCTURE AND FUNCTIONS OF ASSOCIATIVE PROCESSOR

元岡 達　　田中英彦　　上森 明　　河村 悟

Tohru MOTO-OKA　Hidehiko TANAKA　Akira UEMORI　Satoru KAWAMURA

東京大学　工学部

Faculty of Engineering, University of Tokyo

[1] はじめに

新しい構造及び処理機能を取り入れた連想プロセッサを開発し、その第一段階の試作システム（DREAM-エ）を製作したので、ここに紹介する。これは、

- 二次元記憶
- ビット処理機能
  - バレルシフト
  - ビットグループ交換
  - バブルロジック
  - プライオリティ・エンコーディング

等の諸機能を、一つのモジュールにまとめ、複数のモジュールで並列処理する事を目標にしている。DREAM-エは、1台のモジュールに、試験用のコントローラを付けた実験システムであり、将来、マルチモジュールのフルシステムを製作する予定である。

従来のプロセッサは、汎用性を有する代わりに、問題に適した構造や機能を備えていなかった。特に、連想処理、パターン認識の前処理や照合、FFTやアダマール変換、データベース管理、等高度の実時間処理を必要とする分野では、それらの問題に適した構造や機能を有する計算機が必要とされている。

今回のDREAM-エの試作においては、1つのモジュール内に、どのような処理機能を置くべきかを調べる事に重点を置いた。マルチモジュールにして並列処理する時のモジュールの構成は、モジュール間結合及び通信方式によって、多少の変更を必要とするだろうが、処理機能は、さして変わらないと考えられる。

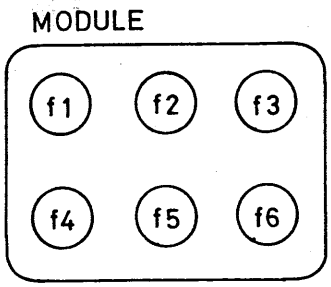
DREAM-エの特徴は、（1）マイクロプログラムの採用により、種々の問題向きの機械語体系、すなわち、問題向きの計算機の実現が可能な事、（2）データへのアクセス方法に、柔軟性を持たせた事（二次元アクセス、コンテントアドレッシング）、（3）パターン処理の効率を上げるため、ビット処理機能を各種用意した事、（4）ビットシリアルな演算を並列に処理可能な事、等である。最後の特徴は、データの記憶上でのスペースの浪費を除き、冗長性をなくして演算すれば、並列処理による効果が得られる事を意味する。

本報告では、最初に、連想モジュールの構成と処理機能にふれ、次に、試作システムDREAM-Iの記述、及び、応用例について述べる。

[2] 連想モジュールの構成

連想モジュールは、連想処理、パターン処理、等の効率向上に有用と考えられそうな機能の集合である。（図1参照）必要とされる処理能力に応じて、モジュールの数を増す事のできる、マルチモジュールの連想プロセッサである。今回のDREAM-エでは、1モジュールのみのシステムとし、モジュール内の処理機能の有効性の実証を目的とした。

DREAM-Iのモジュール内には、以下のような6個の主な処理機能が含まれる。



Functions of Module
- f1. Two-dimensional Access
- f2. Barrel Shift
- f3. Bit Group Exchange
- f4. Bubble Logic
- f5. Priority Encoding
- f6. Arithmetic Logic Operation

図1．連想モジュールの処理機能

ｆ１．二次元アクセス

直交アクセスとも言い、ＯＭＥＮ[4]、ＳＴＡＲＡＮ[5]等の計算機で用いられた技術である。これは、行方向、列方向いずれの方向のアクセス（Read／Write）も、同一時間で可能な二次元記憶である。データの直交変換や、連想処理のビットシリアル左比較や演算、等に用いられる。

ｆ２．バレルシフト

これは、１マイクロサイクルで、任意のビット数のサイクリックシフトを行なうもので、データのアラインメント（配置修正）や、データのフィールド抽出、等データの局所的処理に向いている。

ｆ３．ビットグループ交換

これは、２の巾乗個のビット数のグループ毎に、交換や（鏡像）反転操作を行なうもので、バイトやディジット単位の処理、ＦＦＴにおけるシャッフリング等で用いる事ができる。

ｆ４．バブルロジック

もともと、この機能は、ＥＬＬＩＡＣ－皿[6]のパターン処理装置内の回路として採用されているものである。左端又は右端へ、１ないし０のビットを掃き寄せるもので、パターン認識の前処理に用いられている。この機能を、１ビットずつシフトして寄せるような方法で実現すると、サイクル数を必要とするので、ＤＲＥＡＭ－Ｉでは１サイクルで済むような、専用回路を採用した。

ｆ５．プライオリティ・エンコーディング

これは、最左端又は最右端の１ないし０のビットを検出し、そのビット位置を出力するものである。検索処理において、キーワードと一致のとれた語の位置を示すフラグビットに、この機能を働かせる事により、コンテントアドレッシングが可能である。

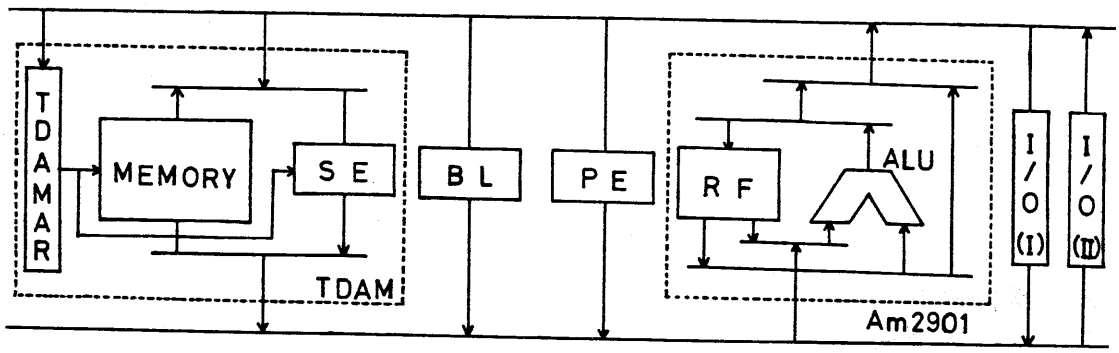
ｆ６．ＡＬＵ演算

通常の計算機で用いられている、加減算や、論理和、論理積、等の演算を行なう。

具体的な連想モジュールの構成は、図２のようになっている。上下二本のバスは、共に１６ビットの巾である。二次元アクセス記憶（ＴＤＡＭ）は、メモリーとアドレスレジスタと、データ並べ替え回路（Shuffle-Exchange）とから成る。ＡＬＵは、レジスタファイルと共に、モジュールの中核となり、４ビットスライスのマイクロプロセッサＡｍ２９０１－４個から成る。

Ｉ／Ｏポートレジスタは、マルチモジュールシステムに拡張した際の、モジュール間通信やデータ入出力の手段として組み込んである。向

図2. The Detail of Associative Module



TDAMAR: Two-dimensional Access Memory Address Register
TDAM: Two-dimensional Access Memory
SE: Shuffle Exchange Network
BL: Bubble Logic
PE: Priority Encoder
RF: Register File
ALU: Arithmetic Logic Unit

きの異なるI/Oポートが1つずつあるのは、モジュール内のレジスタファイルとTDAMのどちらにも、直接I/Oポートから読み書きできるようにするためである。

図2で注意する必要があるのは、全てのデータの流れが許されるのではない事である。すなわち、PEやBL、及び二次元アクセス以外の時のSEは、マイクロプロセッサ(Am2901)内のレジスタファイルから出たデータを入力として、再びそのレジスタファイルへ出力するようなパスしか認めていない。従って、I/Oポートレジスタや、TDAMから、BLやPEを使う事はできない。SEや、BL、PEは、ALUと同様、単なる組合せ回路であり、データのラッチを含んでいない。

図2のモジュール構成の中で、データがどのように流れるかは、マイクロ命令のフォーマットの所で詳細に説明する。

[3] 二次元記憶

二次元アクセス可能な記憶をTDAM(Two-dimensional Access Memory)と呼んでいるが、その簡単な実現方法として、(1) Skewed Array方式と、(2) EOR-Skewed Array方式がある事は、既に述べた。[2] DREAM-Iでは、EOR-Skewed Array方式を採用した。二次元アクセスを可能にするためには、各メモリーチップ毎に、アドレス変換の方法を変えればよく、後者は、EORゲートを用いて、アドレス変換する方法である。

アドレス変換回路の他に、データの並べ替えが、TDAMの入出力時に必要で、そのための回路がSE(Shuffle-Exchange Network)回路である。これの実現方法も前に報告しているので[2]、詳細は省くが、これはパーフェクトシャッフルと交換用セレクタとから成る。SE回路は、TDAMの二次元アクセスのためのデータの並べ替え回路として用いるだけでなく、バレルシフトとビットグループの反転操作等のために、単独でメモリーと切り離して使う事ができる。従って、TDAMは、ちょうど表1のように、マイクロプロセッサからは3つの機能の集合のように見える。

表1. Functions of TDAM

(i) Two-dimensional Access

(ii) Barrel Shift (cyclic)

(iii) Bit Group Exchange

- Swap

- Mirror

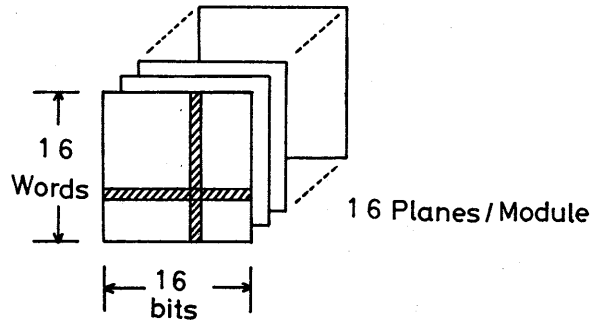


表1の中で、後2つは、次の節で説明する事にし、最初の二次元アクセスをここで説明すると、これには、図3のように、ワードスライスとビットスライスの2つのアクセス法がある。通常の入出力は、ワードスライスで行なり、連想処理をする時は、ビットスライスでアクセスして、ビットシリアルな演算を行なう。

① Word Slice R/W ② Bit Slice R/W

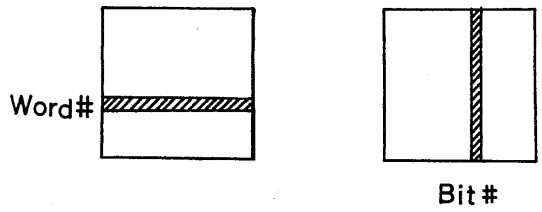


図3、二次元記憶のアクセス方法

DREAM-Iでは、TDAM内のメモリーのサイズは、16語×256ビットであり、一度に二次元アクセス可能な範囲は16語×16ビットに制限されている。従って、図4のように、16語×16ビットの二次元記憶が16枚あるように見える。ただし、ビットスライスアクセスだけは、0～255ビットの連続プレーンとして、アクセスする事が可能である。

TDAMからビットスライスアクセスで取り出されたデータは、16ビットのマイクロプロセッサに渡されるので、ビットシリアルな演算が16倍の並列性で実行できる。

図4、DREAM-Iの二次元記憶の構成

[4] 連想モジュールのビット処理機能

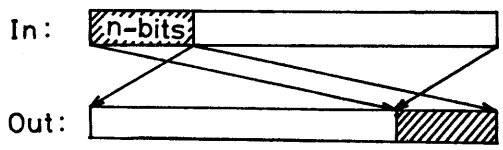
従来の汎用計算機で、パターン処理を行なう際の問題点として、データの局所的処理の難しさがある。これは、データを冗長に記憶している事、また、必要なビットを取り出すために、時間が余計にかかる事、すなわち、問題の構造に対する不適合性、等が原因となっている。例えば、パターン認識の前処理等で、特定の画素のみを取り出すために、ビットのアドレス計算や、1ワード中の特定のビットを取り出すためのシフト等に処理時間を要する。

このような、従来の計算機の欠点を改良するために、このシステムでは、以下のような種々のビット処理機能を用意した。

4-1. バレルシフト

バレルシフトとは、図5に示すように、任意のビット数のシフトを、1サイクルタイムで実行するものである。これは、ILLIAC-IVの演算装置に用いられているバレルスイッチと機能的には同じである。異なる点は、シャッフルエクスチェンジ回路で実現している事、エンドアラウンドのシフトだけをサポートしている

図5、バレルシフトの動作



※ shift any n-bits in one cycle

事である。

バレルシフトは、プライオリティ・エンコーダ（最左の1検出回路）と組合せる事によって、2値画像のパターン認識における、連結成分の抽出とか、浮動小数点演算における、数の正規化、等の操作に用いる事ができる。

DREAM-Ⅱのバレルシフトは、1から15ビットまでの右方向シフトである。16ビットデータのシフト量を指定するのには、4ビット必要であるが、シフト量の指定は、マイクロ命令から指定する場合の他に、レジスタからも指定可能である。これは、浮動小数点数の正規化のように、シフト量がデータに依存するような場合を考慮したためである。

4-2、ビットグループ交換

ビットグループ交換には、表1に示したように、スワップ（入れ替え）とミラー（鏡像反転）の2種類の操作がある。これも、シャッフルエクスチェンジ回路で実現されているが、これは、二次元アクセス記憶のデータの並べ替え操作と全く同じである。従って、16ビットのデータに対する制御ビットは4ビットであり、これをI_3、I_2、I_1、I_0（MSB-I_3、LSB-I_0）とする。

$I_0=0$、すなわち、偶数の時、スワップ操作となり、$I_3=1$なら、8ビット毎の入れ替え、$I_2=1$なら、4ビット毎の入れ替え、$I_1=1$なら、2ビット毎の入れ替えとなる。I_3～I_1のビットのうち、複数個1が立てば、それらを組合せた入れ替えになる。

$I_0=1$、すなわち、奇数の時、ミラー操作となり、$I_1=I_0=1$なら、4ビット単位で反転、$I_2=I_1=I_0=1$なら、8ビット単位で反転、$I_3=I_2=I_1=I_0=1$なら、16ビット全部反転となる。

スワップ操作は、バイトスワップやディジット（4ビット）スワップへ応用でき、バイト単位の入出力動作や、ディジット単位の10進演算等に有効と考えられる。

ミラー操作は、4-1、で述べたバレルシフトや、後で述べるバブルロジックやプライオリティエンコーディング等の処理で、左右の区別をなくすために用いる事ができる。又、スワップもミラーも、FFTのシャッフリングの操作と同じであり、FFTへの応用にも有用である。

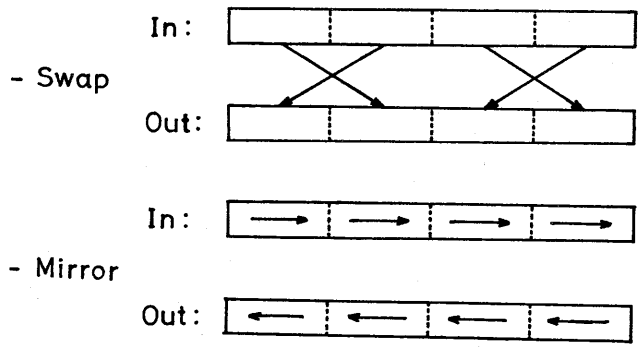
SE（シャッフルエクスチェンジ）回路は、二次元アクセスのための並べ替え、バレルシフト、ビットグループ交換、の3つの機能を、一つの制御回路で実現している。SE回路は、

- パーフェクトシャッフル
- 交換モジュール

とから成る。N入力N出力（$N=2^m$）のSE回路であれば、この組合せが、m段必要である。交換モジュールは、2入力2出力なので、一段当り、N/2個必要で、SE回路全体では、$\frac{1}{2}N\log_2 N$個用いる。N=16の時は、32個である。

交換モジュールは、1本の制御端子を持ち、その値によって、2つの入力をそのまま出力するか、又は、入れ替えて出力する。従って、N=16の場合、32個の制御端子を持つ事になるが、これを15個のグループに分けて制御する事により、EOR Skewの並べ替え（ビットグループの交換）とバレルシフトが容易に実現できる。制御データは、5ビットあり、PROMによって、15個のグループへの制御信号に変換される，従って、SE回路の制御に要するハードウェアの量は、比較的少量で済むし、SE回路自身の遅延時間も、$O(\log_2 N)$に比例し、問題はない。特に、TDAMへ入出力する時は、必ず、このSE回路を通るので、SE回路の遅延を小さくする必要がある。

図6、ビットグループの交換操作



4－3、バブルロジック

バブルロジックは、0又は1のビットを右又は左へ掃き寄せる回路である。（図7参照）泡（0又は1）が片側へ寄せられるのに似ているので、こう呼ばれる。これは、語長が1ビットのBitonic Sorter[7]でもって実現される。すなわち、バブルロジックをN入力N出力とすると、あたかもN個の1ビットデータを並列にバイトニックソートすればよい。これは、シフトと論理演算を用いて、マイクロプログラムで処理可能であるが、時間を要するので、このような専用のハードウェアを付け、1マイクロサイクルで実行可能とした。

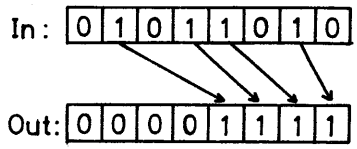


図7. Bubble Logic

このバブルロジックは、1ビットデータのバイトニックソートであるから、NANDとNORおよびパーフェクトシャッフルの組合せだけで構成できる。すなわち、比較エレメントを、2入力2出力で、出力の一方は、2つの入力のうちの小さい方を、他方は、入力のうちの大きい方を出力するような回路と定義する。比較エレメントは、NANDとNOR1つずつを組合せて作れ、これをパーフェクトシャッフルと交互に並べる。[2] 詳細は省くが、段数は、データの語長をNビットとすると、

$$\log_2 N\,(\log_2 N+1) \quad 段$$

であり、一段当り、N／2個の比較エレメントを用いるので、全体で、

$$(N/2)\cdot \log_2 N(\log_2 N+1) \quad 個$$

となる。これは、ハードウェアの規模が、

$$O((\log_2 N)^2)$$

に比例する事を意味する。従って、Nの値をいくつに選ぶべきかは、応用との関係で決まるが、パターン認識の前処理の3×3の窓の隣接点のビットカウントに応用する場合は、8ビットだけでよい。

又、遅延時間も、段数に比例するため、

$$O((\log_2 N)^2)$$

となり、Nをむやみに大きくする事はあまり意味がない。

以上の理由から、N＝8のバブルロジックを設ける事にした。この場合、IC数は12個でよい。N＝16の場合は、40個になる。9ビット以上のデータの場合は、バイトスワップや、バイト単位のミラー操作、最左の1検出、バレルシフトの機能を用いて、掃き寄せる事ができる。また、16ビットのバブルロジックより、8ビットのバブルロジックを2組用意した方が、1マイクロサイクル余分に必要とする代りに、IC数が半分で済むので、16ビットのバブルロジックの使用頻度の多い応用に対しては、この方法が良いだろう。

4－4、プライオリティ・エンコーダ

プライオリティ・エンコーダは、最左端の0又は1を検出し、そのビットアドレスを出力する回路である。（図8参照）

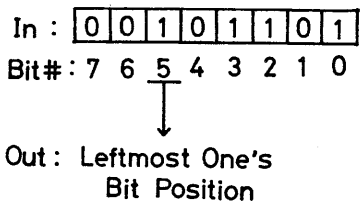


図8. Priority Encoder

このような回路は、通常の計算機で、割込優先順位決定回路として、非常によく用いられている。16ビットのプライオリティエンコーダを付ける事にしたが、このような機能を持つMSIがあるため、IC数は少なくて済む。16ビットの入力に対しては、4ビットの出力が得られる。

プライオリティ・エンコーダは、連想処理におけるコンテント・アドレッシングをする時に用いられ、検索条件を満たす語のアドレスを知る事ができる。他方、浮動小数点数の正規化の時には、バレルシフトと組合せて使う事ができるし、パターン認識の前処理では、連結成分の先頭ビットアドレスを調べたり、バブルロジックと組合せて、ビットカウントの処理が可能である。

Bit Count = Bubble Logic

+ Priority Encoder

特に、ビットカウントの機能の実現方法としては、従来、シフトカウントによる方法、加算器を継続接続する方法、等があったが、前者は多数のサイクル数を必要とするし、後者は早いが複雑なハードウェアを必要とする。バブルロジックとプライオリティエンコーダとを組合せる方式は、2ないし3マイクロサイクルで処理可能であり、ハードウェアも少なくて済み、かつ、それぞれの機能が、他の応用にも十分適用できるだけの汎用性を持っている。

なお、プライオリティエンコーダの優先順位を変えたい時は、ミラー操作を用いて、左右を入れ替えればよいし、最左端の1という指定を最左端の0という指定に変えたい時は、ALUで否定をとっておけばよい。

[5] マイクロ命令の構成

試作システムDREAM-Iのマイクロ命令は、1ワードが24ビットであり、図9に示すように、全部で11種類ある。

各命令の動作の説明と、データの流れについては、以下に示す。特に、前にふれたように、図2のモジュール内の全てのデータフローが許されるわけではない事に注意が必要である。

(1) NOP命令

モジュールは動作せず、I/Oポートを除くモジュール内のレジスタやメモリの値は保存される。この命令は、マルチモジュール化した時に、モジュール群を制御するシーケンスコントローラ用の命令として保留しており、各モジュール間の通信やステータスの収集等の機能を含む予定である。従って、下位20ビットのフィールドは、DREAM-Iがモジュール1台のシステムなので未定義となっている。

I/Oポートレジスタは、モジュールの外で共通バスに接続する予定であるが、このようなモジュールと外部の間の入出力命令も、このNOP命令中で定義できる。現在のDREAM-Iは実験システムであるので、データのロードは、LIM命令で行なっている。データ出力のモニターについては、次の節で述べる。

(2) BR命令

条件分岐命令であり、4ビットのNMIC(Next Microinstruction Control)フィールドで指定された条件が満たされると、Imm部で指定されたアドレスへ分岐する。現在、制御記憶は256ワードなので、Imm部は8ビットである。

(3) LIM命令

フロー：マイクロ命令 Imm部 → Rd

Imm部で示される16ビットのデータをRd(デスティネーション・レジスタ)で示されるポートアドレスのマイクロプロセッサ内のレジスタファイルへ格納する。マイクロプロセッサ外のTDAM、I/Oレジスタ等へのデータのセットはできない。

(4) ALU命令

フロー：Ra f Rb → Rb, f：ALU

RF(レジスタファイル)中のRa(Aポートのレジスタ)とRb(Bポートのレジスタ)に演算を施して、Rbに格納する。この命令では、マイクロプロセッサのみが動作している。しかし、(5)から(11)までの命令に対しては、マイクロプロセッサ外部の回路と並列に動作させる事ができる。

ALUフィールド中の各サブフィールドは、以下の意味を持つ。

Sh－算術シフト、論理シフト、回転シフトなどのモード指定

DC－ALU出力の格納先指定(Destination Control)

Func.－ALU演算の種類指定。加減算、OR、AND、EOR等が指定可能。

Cn－キャリーイン

SS－ALUの2つの入力のソース指定。(Source Select)RFのA、Bポート以外にも、直接入力、Qレジスタとの組合せも指定可。

なお、RFは16ワード×16ビットであり、Ra、Rbはそれぞれ4ビットのフィールドである。

（５）ＳＥＭ命令
（６）ＳＥＲ命令
フロー：Ｒｓ→ＳＥ→Ｒｄ
Ｒｓ（ソースレジスタ）のデータが、ＳＥ（シャッフルエクスチェンジ）回路で処理され、結果がＲｄ（デスティネーションレジスタ）へ格納される。ＳＥ回路に対する制御データ（例えば、バレルシフトのシフト量）を、マイクロ命令のフィールドから与えるか、レジスタから与えるかによって、２通りのマイクロ命令に分かれる。

前者のＳＥＭ命令の場合、Ｍｓ（モードセレクト）ビットで、バレルシフトかビットグループ交換のいずれであるかを指定する。Ｉｍｍ部は４ビットであり、バレルシフト動作の時は、シフト量を、ビットグループ交換の時は、スワップかミラーかの指定と、何ビット単位の交換であるかを指定する。

後者のＳＥＲ命令の場合、ＴＤＡＭＡＲから指定する。ＴＤＡＭＡＲは、１０ビットのレジスタであり、下８ビットは、アドレス部、上２ビットのうち、１ビットはビットスライスとワ

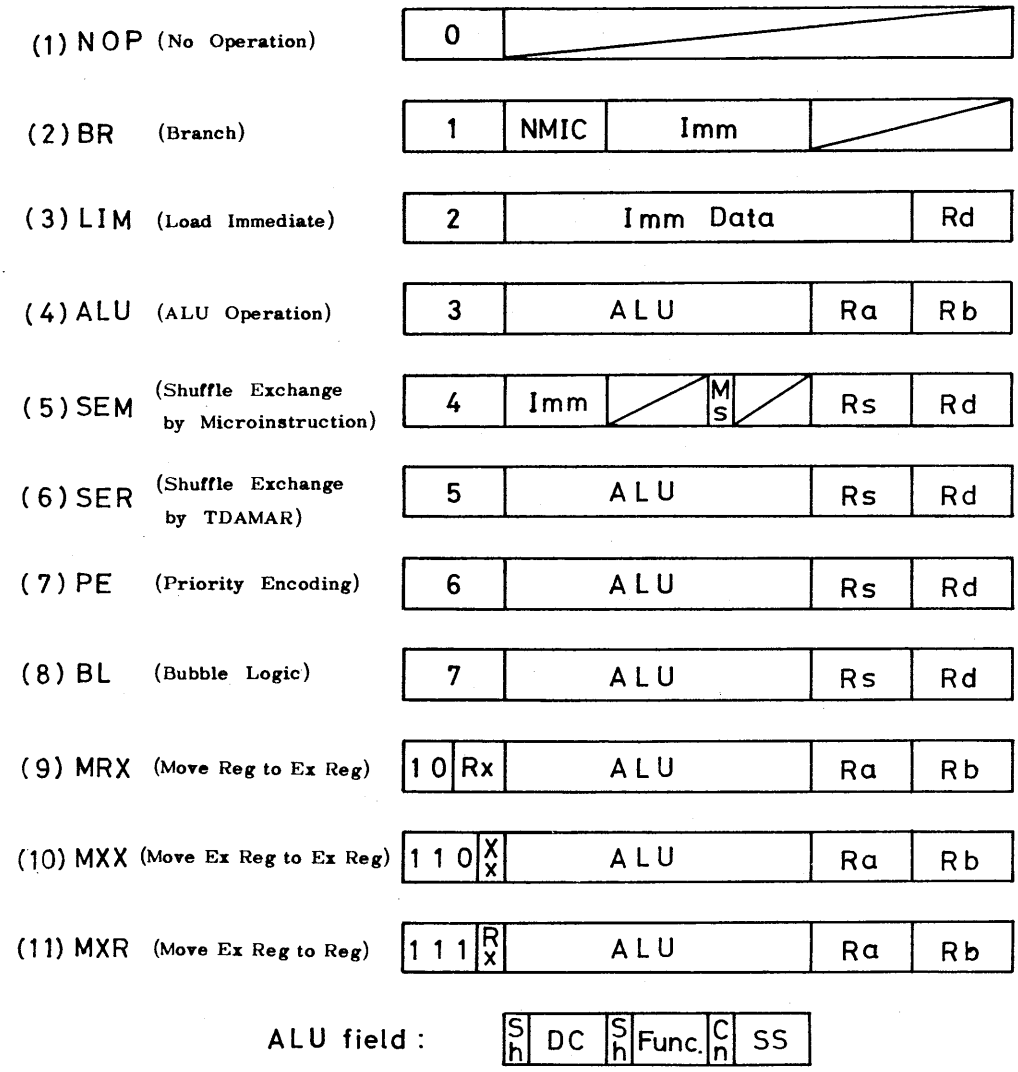


図９．ＤＲＥＡＭ－Ⅰのマイクロ命令の構成

ードスライスの指定で、他の1ビットは、SEM命令のMsビットと同じ指定を行なう。TDAMARの下4ビットが、SEM命令のImm部に相当する。

(7)PE命令

フロー：Rs→PE→ALU→Rd

演算：　g(Rs)fQ→Rd

g－PE、f－ALU

16ビットの入力データ中の最左端の1を検出し、そのアドレスを4ビットデータとして出力する。PEの出力は、ALUを経由して、Rdへ格納されるから、同時に、ベースアドレス(二次元記憶のプレーン番号)を加算してしまう事も可能である。Qはマイクロプロセッサ内のレジスタである。

検索処理の時は、条件一致の語を示すフラグビットをPEに通し、その結果をTDAMARへ転送する事によって、高速検索ができる。

(8)BL命令

フロー：Rs→BL→ALU→Rd

演算：g(Rs)fQ→Rd

g－BL、f－ALU

入力データの下8ビットのみを右へ掃き寄せ、出力する。これも、BLの出力はALUを通るので、結果にマスクをかける(Qレジスタをマスクレジスタとして使う)等の操作が同時に行える。

(9)から(11)までの命令は、マイクロプロセッサの外部レジスタ(External Reg.)とのデータのやり取りを行なうものである。内部レジスタ(R)と外部レジスタ(X)との組合せにより、MRX、MXX、MXRの3種類がある。

(9)MRX命令

フロー：　RF→ALU→Rd

演算：　RafRb→Rx(,Rb)

f－ALU

Rx(External Register)－

0 …… I/Oポート(E)

1 …… TDAM

2 …… TDAMAR

Rx＝1の時は、TDAMへのWriteに相当し、TDAMARのさしているアドレスへ、ALUの演算結果が出力される。ALUの演算と同時に実行可能であり、演算結果を内部レジスタRbに同時に格納する事もできる。

(10)MXX命令

フロー：　RafRb→Rb

{XX＝0 …… TDAM→I/Oポート(E)

{XX＝1 …… TDAM←I/Oポート(E)

この命令は、TDAMとI/Oポート(E)との間でデータの転送を行なう事を除けば、(4)のALU命令と同じである。つまり、この命令は、マイクロプロセッサ内での演算をしている間に、TDAMへの入出力を並行して行なう事を目的として設けられている。

(11)MXR命令

フロー：　RxfRa→Rb

{Rx＝0 …… I/Oポート(E)

{Rx＝1 …… TDAM

Rx＝1の時は、TDAMのReadに相当し、TDAMARのさしているアドレスのデータを入力し、ALUを経由して、Rbに格納される。したがって、Raを用いて、読み出したデータにマスクをかける事や、連想サーチにおいて、Raを比較レジスタとして使う事が可能である。これによって、マイクロプログラムのステップ数を減らす事ができる。

[6] 試作システムDREAM-Iの構成

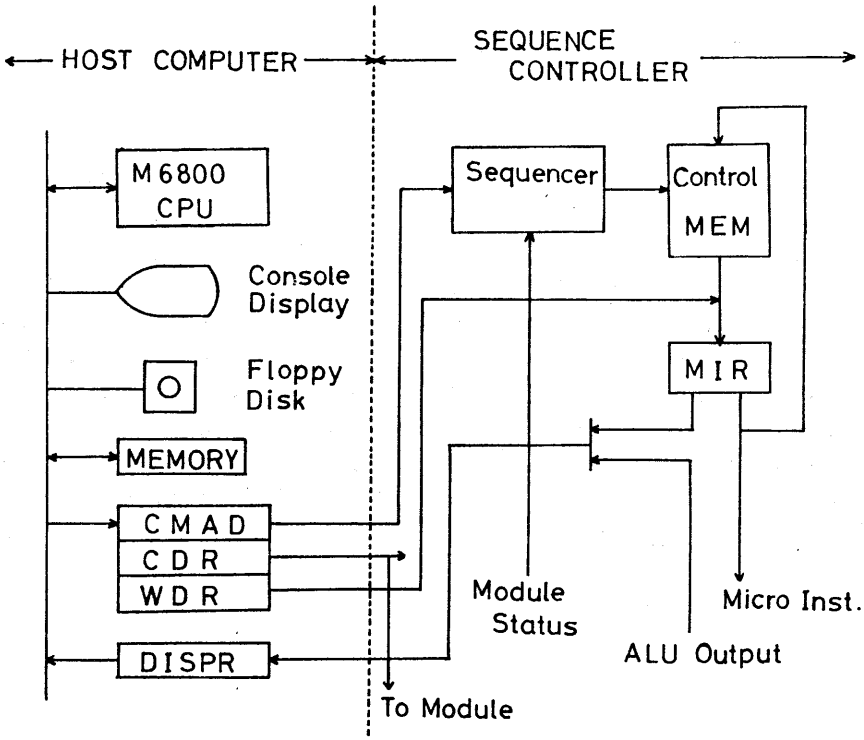
今回、試作した実験システムDREAM-Iは、次の3つの要素からなる。

- 1台の連想モジュール
- シーケンスコントローラ
- ホストコンピュータ

シーケンスコントローラは、シーケンサと制御記憶及び、ホストコンピュータとのインタフェースからなる。マルチモジュール用としては、設計されていないので、図10に示すように非常に簡単な構成となっている。

制御記憶(CM)は、256ワード×24ビットのWCS(Writable Control Storage)であり、アクセスタイム65nsのメモリーで

図10、DREAM-Iのコントローラ



CMAD: Control Memory Address
CDR: Control Data Register
WDR: Write Data Register
DISPR: Display Register
MIR: Microinstruction Register

ある。MIR(Microinstruction Register)は、24ビットのレジスタであり、パイプラインレジスタとして動作する。シーケンサは、直接入力の他に、CMのアドレスレジスタやスタック(サブルーチンジャンプ用で深さは4段)を持っている。シーケンサは、モジュールのステータス(キャリー、オーバフロー、ゼロ、サイン等)によって、次のCMのアドレスを決定する。

ホストコンピュータは、以下の機能を持つ。

①連想モジュールに対する保守・診断機能
②マイクロプログラムのロード
③マイクロ命令の実行制御
④モジュールの演算結果のモニター

すなわち、ホストコンピュータは、DREAM-Iシステムのコンソールプロセッサとしての役割を持つ。ホストコンピュータ側で、DREAM-I用に作成したインタフェースは、I/Oポート用のLSI(PIA)とドライバだけであり、比較的容易に結合する事ができた。ホストコンピュータは、フロッピーディスクと、コンソールディスプレー付きのM6800(モトローラ社の8ビットプロセッサ)である。

ホスト側のインタフェース用レジスタとして、以下のものがある。

CMAD—ホスト側から、制御記憶の内容を読み出したり、書き込む時に、このレジスタによって、CMのアドレスを指定する。この時、シーケンサのアドレス出力には、このレジスタの値がそのまま現れる。

CDR—8ビットのうち5ビット使用。CMのRead/Write指定、ソフトウェアで制御可能なクロックビット、MIRのソースセレクト、CMアドレスのソースセレクト(ホスト制御/自走)、モジュール内のALUの動作制御(停止/動作)、等を指定

する。

WDR－8ビットで、MIRへのマイクロ命令のセットに用いられる。マイクロ命令は、24ビットなので、3回にわけて出力する。CMへのロードは、MIR経由で行われる。

DISPR－8ビットで、モジュール内のALUの出力と、MIRの出力を読み出す事ができる。さらに、CMの内容も、MIR経由で読み出す事ができる。いずれの場合も、バイト単位でホスト側に読み出される。

ホストコンピュータのコンソールから、コマンドをキーインする事により、以下のような操作ができる。

- MIRの表示及び変更
- CDRの表示及び変更
- CMのIMPL
- CMの内容表示

（この2つの操作後、MIRの内容はこわれる）

- クロックを1つだけ入れる
- ALUの出力を表示

これらの操作以外にも、種々の機能のコマンドを用意する予定である。DREAM－Iでは、コンソールパネルに相当するものは、一切なく、全てホスト側のコンソールディスプレーが、パネル代わりとなっている。

ホスト側からは、シーケンスコントローラと連想モジュールを1ステップずつ実行させる事ができる。すなわち、CMとMIRの内容を更新できるだけでなく、クロックも制御できる。ただし、ホスト側のクロックとは、全く独立の自分自身のクロックでも動作（自走）できるようにもなっている。

ホスト側から、データの供給（LEM命令による）や、マイクロ命令の供給と実行が可能なため、TDAMやSE回路の動作を診断する事ができ、このシステム製作時のデバッグにも、非常に役に立った。

なお、ホスト側で開発された診断・保守プログラムは、ホスト側のフロッピーディスクに格納されている。

［7］ パターン認識への応用

連想モジュールに用意された機能を用いて、いかに問題に適応した計算機（命令体系及び構造）を実現したらよいかの例として、パターン認識の前処理への応用を考えてみる。

図11に示したように、二値画像の細線化や雑音除去等の処理を効率良く実行可能な構造について考える。まず、3×3の窓に対する処理を、1ビットのビットシリアルな演算器で実行するものとする。このような処理の例として、8つの隣接点のビットカウントをした後、その個数によって中心点を更新する場合がある。

このような処理をする時、従来のプロセッサで問題となったのは、

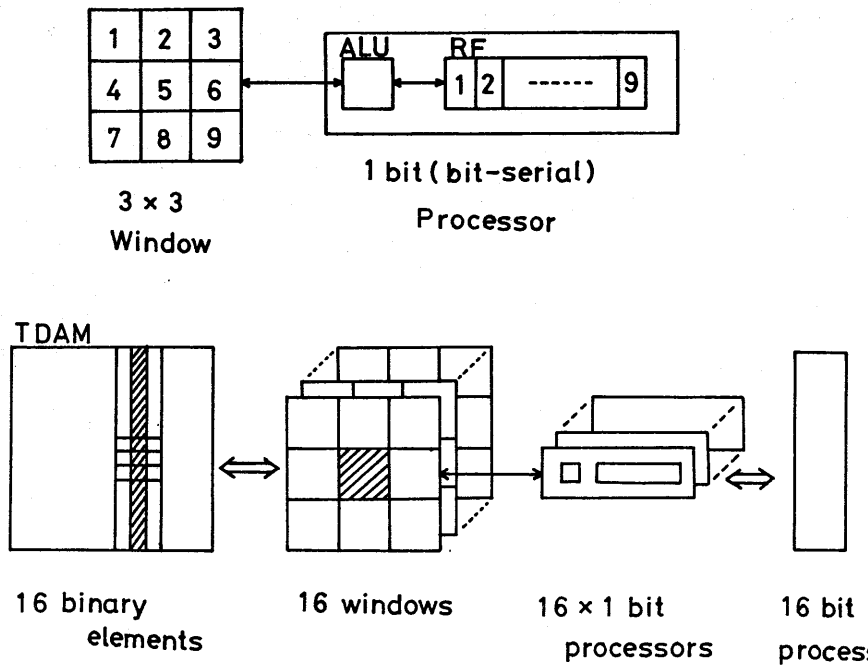
- 3×3の窓で各画素を取り出すために、アドレス計算やシフトに余分な時間を費していた事（アクセスのオーバヘッド）
- 1ワードに1画素（1ビット）だけしか入れない方法では、冗長で並列性が使われていない事
- ビット処理機能の不足

等の点である。

第1の問題点は、図11の下図で示されるように、比較的容易に解決される。すなわち、図11の左下図において、斜線部分の16個の画素に対し、16組の3×3の窓が対応している。各画素の隣接点の相対位置番号を、1から9のように（図11の左上図）定義すると、中心点自身は5番となる。1から9までの各相対位置に対し、やはりそれぞれ16個の点が対応するが、これはビットスライスアクセスとシフトによって取り出せる。従って、図11の上図のように、中心点のまわりの8つの隣接点のデータは一次元に展開した形へ変換される。この変換が16個の中心点の全てに対し、同時に並列に実行され、あたかも、16台の1ビットシリアルプロセッサが並列に動作しているように見える。実際には、16ビットのプロセッサで、ロジカルな演算をしているに過ぎないが、これによって、16ビット全ての演算機能を用いており、第2の問題点も解決されている。

第3の問題点は、［4］節で述べたような各処理機能によって、かなり改善されている。

図11. APPLICATION FOR PATTERN PROCESSING



他方、このような応用に対しては、境界点の処理がさらに問題になる。ＴＤＡＭでは、ビットスライスアクセスの方は、０から２５５番地まで連続しているので、左右の境界はあまり問題にならない。むしろ、上下方向の境界は、別のモジュールにまたがるので問題となる。これの解決法としては、１６語中の上下１語ずつを上下の境界点用としてデータを配置すればよい。並列度は、１６倍から１４倍に低下するが、境界点の処理がそれほど複雑にならずに済む。

[８] まとめ

以上、連想モジュールをベースにした試作システムＤＲＥＡＭ－Ｉの概容を述べた。現在、システムはほぼ稼動しており、ソフトウェアの開発とシステムの評価を行なっている。近い将来、マルチモジュールのバージョンも製作する予定である。

終わりに、ホスト側の制御プログラム開発に対し、協力された大学院生の三坂敏夫君に感謝する。なお、本研究の一部は、昭和５２年度科学研究費補助金（一般研究）によった。

[参考文献]

(1) 元岡、田中、上森「パターン処理用プロセッサのアーキテクチャ」情処全大第17回

(2) 元岡、田中、上森、鈴木「二次元記憶を用いた連想処理システム」信学技報 EC76-80

(3) 上森、田中、元岡「連想処理システムのモジュール化」情処全大第18回

(4) L.C.Higbie,"The OMEN Computers:Associative Array Processors,"COMPCON pp.287-290,1972

(5) K.E.Batcher,"STARAN parallel processor system hardware,"NCC,pp.405-410,1974

(6) B.H.McCormick,"The Illinois pattern recognition computer ILLIAC III," IEEE EC-12,pp.791-813, 1963

(7) D.E.Knuth,"The Art of Computer Programming ," vol.3,Addison-Wesley,1969