計算機アーキテクチャ 42－1
（1981. 7. 22）

データフロー計算機 D³P の実験システム

伊藤徳義　来住晶介　安原宏　河村保輔
沖電気　総合システム研究所

Ⅰ. はじめに

近年，国内外の研究機関を中心にデータフロー計算機の研究開発が進められている。[1]～[4]

データフロー計算機はデータの流れに依存して処理単位の起動制御を行うデータ駆動型の特性を有しているために，制御の分散化が容易であり，多数の処理装置を配することにより並列処理が実現できる。

一方，言語面から見た場合，一時に1語的に逐次処理することを前提とした従来のノイマン型言語に比して，関数型言語はプログラム構造が見易く，記述能力に優れ，さらにプログラム正当性の検証の可能性が生ずるという特長がある。データフロー計算機の実行制御は関数型の特性を有しており，このような関数型言語の実行に適していると思われる。

本論文は，著者等が開発を進めているデータフロー計算機 D³P（Distributed Data Driven Processor）[5]～[7]の構成を述べると共に，その中核をなす命令制御部を試作・評価したのでその結果を述べる。

Ⅱ. D³P の設計方針

D³P を開発するにあたって設定した設計方針は以下の通りである。

(1) 関数クラスタの導入

他のデータフローの動的アーキテクチャの研究では，関数やループの起動要求が発生する度に処理要素を割当てる方式を採用している。このような方式では，関数やループの起動毎にプログラムロードのためのオーバヘッドが伴う。従って D³P ではシステムを DFE（Data Flow processing Elements）と呼ばれる処理モジュール群で構成し，さらに関数クラスタの概念を導入し，この関数クラスタ毎に DFE を動的に割当てることにした。関数クラスタは関数，又は関数の集合であり，各関数は自クラスタ内に存在する関数を呼出すこと（内部関数呼出し）もできるし，他クラスタ内の関数を呼出すこと（外部関数呼出し）もできる。各関数クラスタに含まれる関数群の選択はクラスタのプログラムサイズ，及び関数呼出しの動的特性を考慮して決定される。

D³P におけるプログラムのローディングは外部関数呼出しの要求が発生したときのみ必要となり，そのオーバヘッドが軽減できる。

(2) 統合化された色

D³P は基本的に UI（Unfolding Interpreter）[2] の手法を導入している。即ち，ループや関数が起動された際，それらの起動要求毎にそれぞれ互いに異なる色の割当て制御を行い，プログラムを共有しながらループや再帰呼出し等の制御を可能としている。他の方式と異なるところは，ループ起動と関数起動を区別せずに，統合化した色（アクティベーション番号。以下 ACT# と略す）で制御し，この ACT# の割当てを動的に行っている。また，ループや関数が終了した際，それまで使用していた ACT# を解放するプリミティブも用意した。これによって，ACT# の空間を効率良く使用することができ，ACT# を表現するのに必要なビット数を減少できる。

また，ループや関数の起動をトリガ

ートークンを用いて行うようにすることにより，ループや関数の全引数が揃うまで待たずに先行して起動できるようにした。

(3) 命令レベルの並列実行

起動されたループや関数に内在する命令レベルの並列性も実現する。即ち，DFEを複数のPE(Processing Elements)から成る密結合多重プロセッサ構成とし，割当てられた関数クラスタのプログラムをそれらのPEに分散してロードし，命令レベルの並列処理も実現している。

(4) オペランド記憶域の動的割当て

オペランドのアドレス空間は，ACT#と命令アドレスを連結したアドレスで与えられる。一般に，2つの入力オペランドを有する命令が実行可能になったか否かの判定は連想メモリを用いる方式が提案されている。しかし，現状では連想メモリは高価であるためD^{3}Pではハッシュを用いた動的アドレス変換機構を用意した。

(5) 構造体データの非同期アクセス

D^{3}Pでは構造体データの非同期アクセス機構を用意する。即ち，構造体メモリ内の各記憶単位毎に，データが到着したか否かやデータ読出し要求があったか否かを示す情報を用意し，構造体への要素の書込みや読出しを非同期に行うことができる。

(6) プリミティブノードの設定

ハードウェア実現上の観点から，すべてのプリミティブノード，即ち命令は最大2個までの入力アークを受取り，最大2個までの出力アークにトークンを出力できる構成とした。但し，特殊な命令では4個のトークンを生成できる。各ノードはそのオペランドとして定数を持つこともできる。

(7) 可変語長データ処理

低コストで，かつ汎用性のあるアーキテクチャを実現するために，データの基本語長を16ビットとし，この基本語長の4倍までのデータを処理可能とした。

III. D^{3}Pシステムの構成

前記のような設計方針に基づき，D^{3}Pを図1のように複数のDFEから成る構成にした。

各DFEはプログラムの一部，即ち関数クラスタ，の実行を負担し，必要に応じてDFE間ネットワークを介して相互に通信しながら処理を進める。このネットワークにはFCC(Function Cluster Controller)が接続されており，各DFEと交信できる。

FCCは(1)各DFEの監視や初期化，(2)DFEへの関数クラスタの割当てやプログラムロード，(3)構造体の初期化，及び(4)D^{3}Pと外界とのインタフェースの機能等を有する。

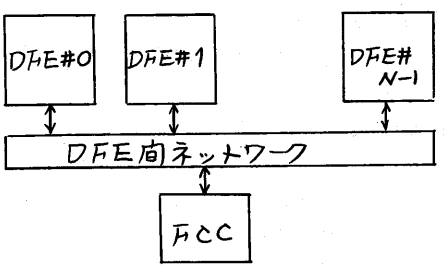


図1. D^{3}Pシステムの構成

以下，このD^{3}Pシステムのうち，DFEの詳細について述べる。

3.1 DFEの構成

3.1.1 概要

DFEは関数クラスタを実行する処理装置である。DFEは関数クラスタに内在する並列性を自動検出し，複数のPE(Processing Elements)で並列実行する。

各PE間は図2のように高速リング

バスで接続されており，相互にトークンの交信を行う。また，PE間で共有されるデータ構造を処理するために，共通バスを介して共有メモリへ結合されている。

各PEは，さらに同図(b)のように命令の実行制御を行うIM(Instruction Memory)と，実行可能命令を解釈し実行するPU(Processing Unit)から構成される。

DFE内の各PEのIMにはプログラムが分散して格納されており，IM内の命令のうちオペランドが揃い実行可能となった命令はPUに送られ，実行される。その実行結果は，命令中に指定されている目的地に応じて，自PEのIM，他PEのIM，あるいは他DFEへと送られる。

IM及びPUは互いに独立に動作し，命令の読出しと命令の実行はパイプライン制御される。

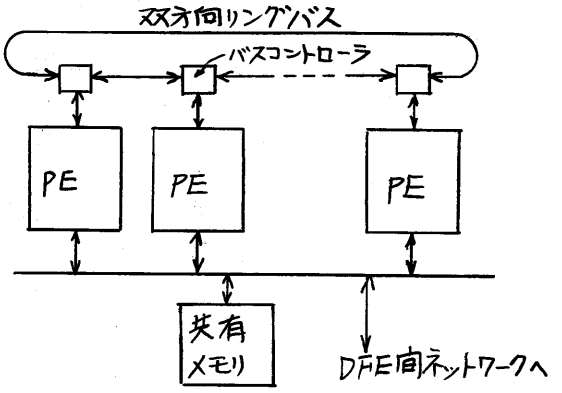
3.1.2 命令及びトークンのフォーマット

IMに格納される命令は図3(a)のように，基本的に2つのオペランド(OPR-L，OPR-R)を受取ると実行可能となる。但し，単一オペランド命令や2オペランド命令でもいずれかのオペランドが定数の場合は，1つのオペランドの到着で実行可能となる。

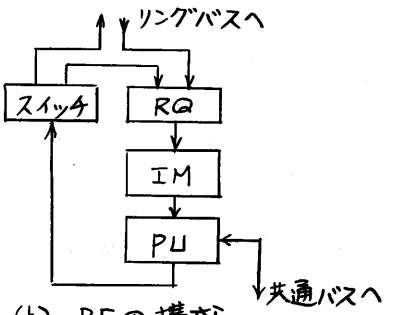
実行可能となった命令はPUで処理され，2つまでの目的地(DST1，DST2)へ送られる。

IMからPUへ送られる実行可能命令のフォーマットを同図(b)に示す。

同図においてOPCは命令コードを示し，TYPは左右両オペランドの語長等を示す。DST1及びDST2は，



(a) DFEの構成



(b) PEの構成

図2 DFE及びPEの構成

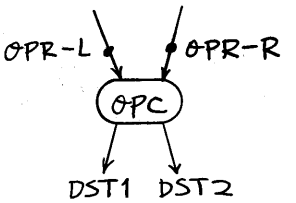


図3(a) 基本命令の構成

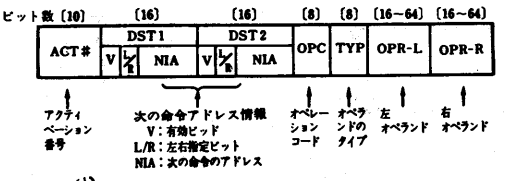


図3(b) 実行可能命令のフォーマット

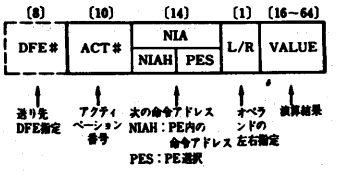


図4 実行結果トークンのフォーマット

それぞれ，次に結果を転送すべき命令アドレスNIA（Next Instruction Address），及びその結果が命令の左右いずれのオペランドとなるかを示すL/Rフィールドから成る。

命令がPUによって実行された結果，図4のようなトークンが生成される。このトークンはDFE#とNIAの指定に応じて次のEMへ転送される。DFE#はDFE間の通信，即ち関数クラスタ間における引数や結果の受渡し，の際にのみトークンに付与され，相手DFEの選択に使用される。

3.1.3 EMの構成

DFEは複数のPEを有し，プログラムは命令アドレスによりインタリーブされて各PEのEMに格納される。

EMのブロック図を図5に示す。EMはコードメモリ，オペランドメモリ，及びEMC（EM Controller）を有し，それぞれ，プログラムの格納，到着オペランドの一時記憶，及びオペランドの到着状態の監視と命令の実行制御用に使用される。EMに到着するトークンのフォーマットは，図4に示したフィールドのうち，DFE選択部（DFE#），及びPE選択部（PES：NIAの下位4ビット）を切捨てたものである。このトークンはFIFO（First-in First-out）キューであるRQ（Result Queue）に格納される。

EMCはRQからトークンを受取るとコードメモリの読出しを開始する。コードメモリの各エントリは図6(a)～(d)に示すように，定数の有無，及び定数語長により異なる。倍語長以上の定数は次アドレスに連続して格納される。図5の定数語長カウンタはこのためのアドレスを制御する。

コードメモリの読出しと同時にオペランドメモリの読出しも開始される。但し，命令コード中にSOE（Single Operand Enable）＝1の指定のある場合，即ち命令が単一オペランドの到着によって実行可能となる場合，オペランドの読出しは停止され，直ちに実行可能命令が生成される。このときTYPフィールドによる定数指定があれば，図6(b)～(d)に示される定数の内容が実行可能命令のオペランドの1つとして設定される。

SOE＝0の場合は，後述するアドレス変換を経て，オペランドの読出しが継続される。オペランド

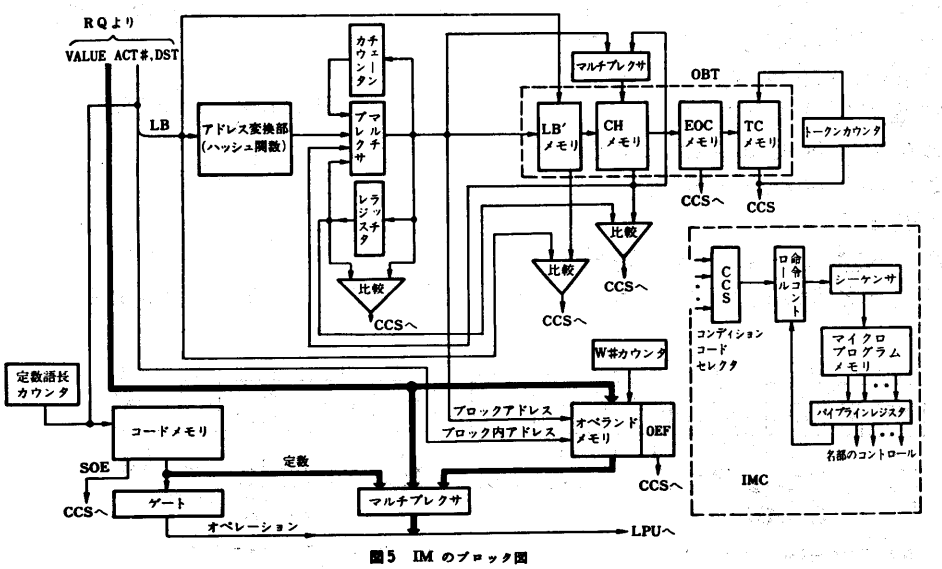


図5 IM のブロック図

の各エントリは4語長までのオペランド格納域と2ビットのOEF（Operand Enable Flags）から構成される。OEFはオペランドの到着状態を示している。即ち，左右オペランドがいずれも未到着の場合は00，左，又は右オペランドが到着した場合はそれぞれ10又は01に設定される。既にOEFの内容が10又は01のときに，それぞれ，右又は左オペランドが到着した場合，命令は実行可能となる。

一般的に，データフロープログラムで示されるプログラムの実行において，あるノードの命令が実行されたのち次に起動されるノードはその命令の近傍のアドレスに存在する確率が高い。しかも，オペランドを一時記憶する必要があるのは現在実行中のノード群の付近に偏る傾向がある。この性質を利用して，ACT#とNIAを連結して得られる論理アドレス空間を図7に示すように16エントリから成る論理ブロックに分割し，この論理ブロック単位に動的に物理メモリを割当てることにした。この割当てを行うアドレス変換部には図5に示すようにハッシュ関数を用いた。

アドレス変換部は，論理ブロック番号LBが与えられるとハッシュ値を求め，この値を物理ブロックアドレスとして管理テーブルOBTをアクセスする。ハッシュ変換により異なる論理ブロックが同一の物理ブロックに割当てられたときには，エントリのチェインが行われる。OBTの各エントリは論理ブロック番号LB，チェインポインタCH，チェインの終わりを示すEOCフラグ，及びブロック内に格納されている有効トークン数を記憶するTCの4フィールドから構成される。

図8にアドレス変換の動作フローチャートを示す。

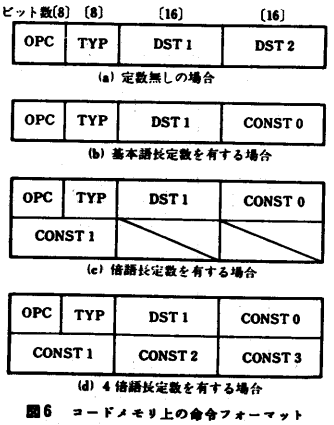


図6　コードメモリ上の命令フォーマット

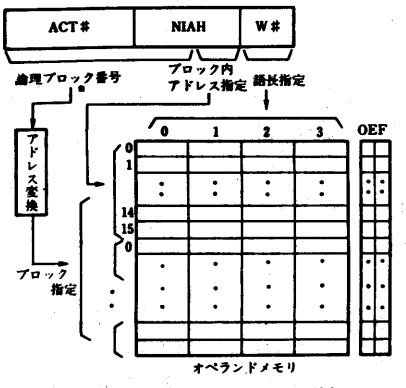


図7　論理アドレスとオペランドメモリの対応

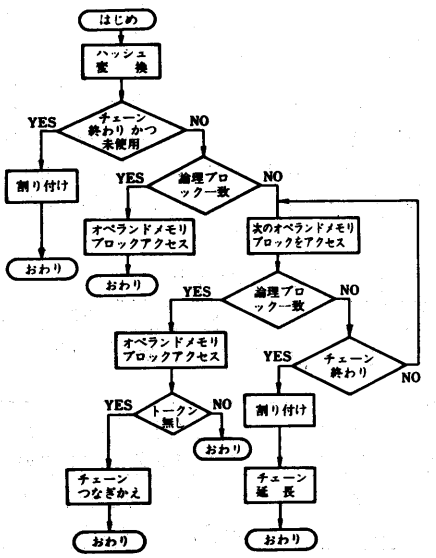


図8　アドレス変換操作のフローチャート

3.1.4 PUの構成

命令は表1で示すように大別して共有リソースをアクセスするものとそうでないものに分けられる。前者はさらに，ACT#制御命令と構造体データ操作命令に分けられ，いずれもDFE内の共有メモリを参照する。後者の命令はPU内でローカルに処理される。

いずれの命令も基本的には，命令のOPC，TYP，OPR-L，及びOPR-Rをもとに演算を行い，その結果及びDFE#，ACT#，DSTi (i=1, 2)とを組合わせてトークンを生成し，ネットワークを経て目的のEMへ転送される。

(1) ローカル命令

各種演算命令，ゲート制御命令，及びコピー命令等であり，PU内でローカルに処理される。

(2) ACT#制御命令

関数やループの起動制御を行う。図9にループの場合のデータフローグラフ例を示す。ループ，関数とも基本的には同一の制御法を用いる。即ち，それらの非同期起動を実現するために，ループ又は関数の起動条件が成立した時点でトリガートークンがCALL命令を起動する。この命令は共有メモリ中のACT#制御テーブルを参照し，新たなACT#をループ又は関数に割当てる。ACT#制御テーブルは図10に示すようにACT#の使用状態，及びループや関数が終了した時点で戻るべきDFE#，ACT#を管理する。

PUはCALL命令を受取ると有効ビットが0のACT#を割当て，そのビットを1にセットし，戻りACT#フ

表1 命令の分類

命令タイプ		機能
ローカル命令	演算	算術演算，論理演算，比較，シフト等。
	ゲート制御	Tゲート，Fゲート，スイッチ操作。
	コピー命令	入力値を複数の目的地へコピー。
共有メモリアクセス	ACT#制御	ループ制御，関数呼出し，及びACT#解放を行う。
	構造体データ操作	要素の読出し，書込み操作。領域の割当て，解放等。

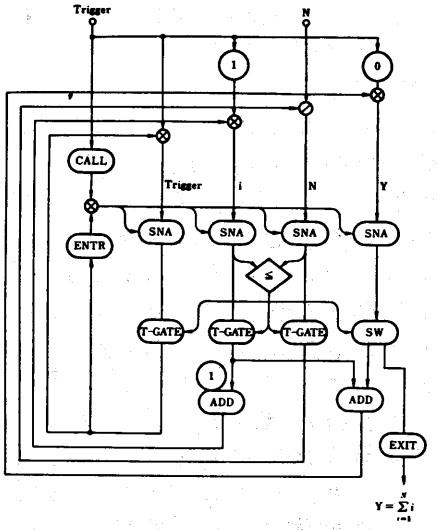
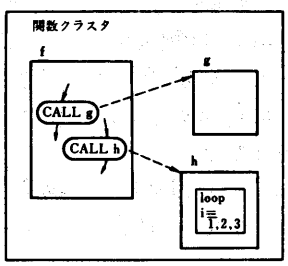
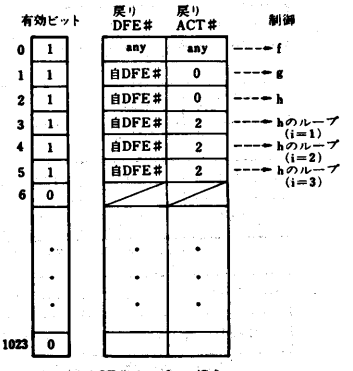


図9 ループのデータフローグラフ例



(a) 関数呼出しおよびループの起動例



(b) ACT#テーブルの構成

図10 関数およびループの起動制御

ィールドに現トークンのACT#をセットする。この命令の結果トークンのVALUE部には新たに割当てられたACT#がセットされる。このトークンはSNA(Set New Activation)命令群に渡されて，ループ又は関数に渡される引数のACT#が更新される。SNA命令は各PUでローカルに実行される。

ループが継続されるとき，ENTR(Enter Next Loop)命令が実行される。CALL命令と同様であるが，異なるのは，前のACT#における戻りACT#フィールドの値を新たな戻りACT#へコピーする点である。

一方，ループ又は関数の結果が得られると，それぞれ，EXIT又はRET命令が実行される。いずれもトークンのACT#を戻りACT#に回復し，結果を親へ戻す。

(3) 構造体データ操作命令

構造体への要素の書込みや読出しを行う。構造体データは共有メモリに格納される。

構造体領域は要素への非同期アクセスを実現するために図11のように構成されている。即ち，各語毎にその内容が有効であるか否かを示すW(Write)ビット，及び語の読出し要求待ちがあるか否かを示すR(Read)ビットが用意されている。Wビットがオンの場合は対応する語に値が格納されていることを示し，Rビットがオンの場合は対応する語にWQT(Wait Queue Table)のエントリアドレスが格納されていることを示す。WQTの各エントリには要素を転送すべき目的地(ACT#, DSTi)，及び次のWQTエントリへのポインタが格納される。

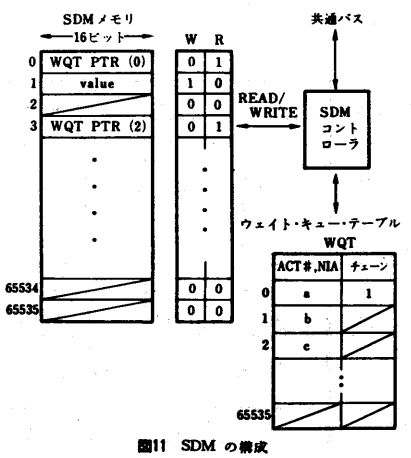


図11 SDMの構成

構造体への要素の書込みの場合，対応する語のRビットが調べられる。このときRビットがオンであれば，WQTポインタを辿り，待ちとなっているエントリ中のACT#, DSTiからトークンを作成する。Rビットがリセットされ，実際の書込み動作が行われて，Wビットがオンにされる。

構造体からの読出しの場合，対応する語のWビットが調べられる。これがオンのときは語の内容によりトークンを作成する。オンでないときは，WQTチェインに待ちエントリを追加する。

3.2 DFE実験システムと評価

D3Pシステム全体の評価を行う前に，著者等はまずEMのハードウェア規模やスループットを検証するためにその詳細設計を行い，図12のような実験システムを試作して，その評価を行った。

表2に試作EMのハードウェア諸元を示す。

マイクロコンピュータ上には，サポートソフトウェアシステムとしてEM監視用のモニタ，データフロープログラムのクロスアセンブラ，及びEMから出力されるデータフロー命令のシミュレータ等を開発した。

表3にいくつかのサンプルプログラムに対してEMの処理時間を計測した結果を示す。表3のEM時間は，マイ

クロコンピュータがEMへトークンを送ったのち，そのトークンをEMが処理し終えるまでの時間を観測したものである。

付録1及び2にサンプルプログラムとしてAckermann関数のデータフローグラフ及びそのアセンブルリストを示す。

3.3 今後の方針

前述のように，著者等はまずEMを試作し，その評価を行った。本試作EMではマイクロプログラムの最適化等が行われていないので，トークンあたりの処理時間は3μs程度と当初の目標値（1.5μs以下）に比して劣る。このため現在EMの見直しを行っている。

今後，この結果をもとにまず，4台程度のPEから成るDFEを試作し，マルチPE環境におけるデータフロープログラム実行の評価を行う予定である。

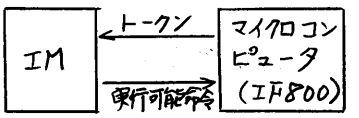


図12 EM評価システム

る。これに伴い，PE間のリングバスの設計やPUの設計も行っている。

Ⅳ おわりに

本論文はD³Pの構成要素となるDFEの基本アーキテクチャを中心に述べた。このDFEの中核部となるPEの制御方式は大規模多重処理においても有効と思われる1つの現実解を与えており，さらに検討を進めていく予定である。

今後，DFE及びD³Pシステム全体の評価を行っていくと共に，数値処理や非数値処理を含めた応用領域への適応性を検証していく予定である。

本システムの開発にあたって様々の御支援をいただいた沖電気 総合システム研究所 安楽所長他の関係者に感謝する。

表2 EMのハードウェア諸元

マシンサイクル	250 ns
制御記憶容量	20bit×256word
コードメモリ	48bit×1Kword
オペランドメモリ	64bit×4kword
ハッシュ表	33bit×256word

表3. サンプルプログラム実行結果

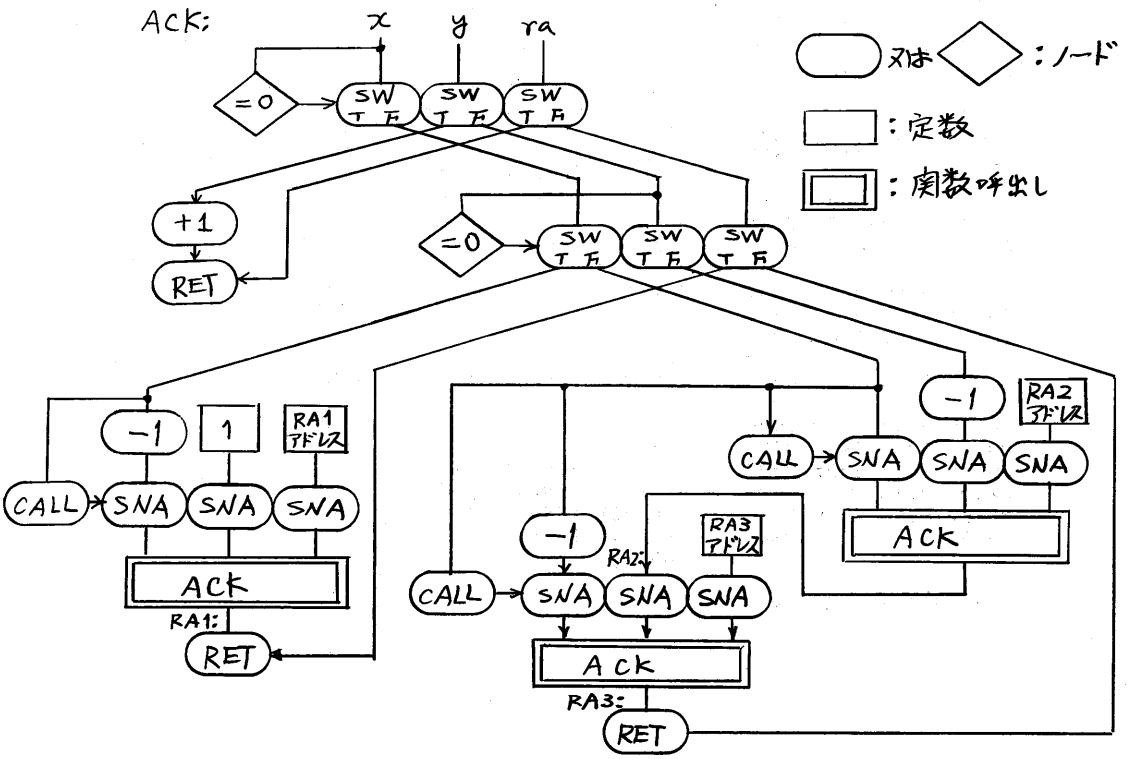
プログラム名		全EM時間	処理トークン数	全実行命令数	時間/トークン	時間/命令
Ackermann関数	A(2,1)	454μs	385	272	3.1μs	4.5μs
	A(3,3)	217,342	69,505	48,804	3.1	4.5
$\sum_{i=1}^{N} i$	N=5	438	150	91	2.9	4.8
	N=100	7,486	2,529	1,520	3.0	4.9
マトリックス乗算	3×3配列	7,095	2,187	1,382	3.2	5.1
	5×5〃	27,090	8,059	5,020	3.4(3.0)	5.4(4.8)
Quick Sort	10個の要素	5,239	1,720	1,084	3.0	4.8
平均		—	—	—	3.1(3.0)	4.9(4.8)

( )内はハッシュ関数を改善したとき。

[参考文献]
(1) J. B. Dennis, "The Varieties of Data Flow Computers". Proc. of 1st Int'l Conf. on Dist. Comp., Vol. 1, pp. 430~439, 1979.
(2) Arvind, K. Gostelow, and W. Plouffe, "An Asynchronous Programming Language and Computing Machine". Information and Computer Science, Univ. of California, Irvine, TR114a, Dec., 1978.
(3) J. Gurd, I. Watson, and J. Glauert, "A Multilayered Data Flow Computer Architecture". Univ. of Manchester, 1978.
(4)「国内でも研究・試作が増えるマルチプロセッサ・データフローマシン」, 日経エレクトロニクス, No. 246, 1980.
(5) 安原, 伊藤, 瀬賀, 上原,「データフロープロセッサ D^3P の設計思想」, 情報処理学会, 計算機アーキテクチャ研究会, No. 37, 1980年1月。
(6) 安原, 伊藤, 来住,「データフロー計算機 D^3P のアーキテクチャ」, 沖電気研究開発, Vol. 47, No. 2, 1980年12月.
(7) 来住, 伊藤, 安原, 河村,「データフロー計算機 D^3P の実験システム」, 情報処理学会第22回全国大会, PP. 57~58, 1981年3月.

付録1. Ackermann 関数のデータフローグラフ

```
Ack(x,y) := if x=0 then y+1
            else if y=0 then Ack(x-1, 1)
                 else Ack(x-1, Ack(x, y-1))
```



付録2. Ackermann 関数のアセンブルリスト

```
ADDR: CODE DST1 DST2  SOURCE-STATEMENT

                      ; DATA FLOW SAMPLE PROGRAM
                       ; ACKERMANN FUNCTION
    :                  MAIN:              FUNC
0000: 8060 8401 0003   X:                 CALL     ACT
0001: 8200 8005 8006   X:         ACT:    SNA      ACKFN.XX
0002: 8200 8007 0000   Y:         ACT:    SNA      ACKFN.YY
0003: 8260 8008 8004   A'RA':     ACT:    SNA      ACKFN.RRA
0004: 4660 0000 0000   RA:        0:      APP
                       ; ******************
                       ; * FUNCTION BODY *
                       ; ******************
    :                  ACKFN:             FUNC
0005: 2060 8020 0000    XX:       0:      TEQ      XZERO
0020: 6040 8406 8021
0021: 6040 8407 8408
0006: 5100 800C 0000    XX:       XZERO:  F        FXX
0007: 5200 8009 8022    YY:       XZERO:  SW       TYY,FYY
0022: 6040 800B 800D
0008: 5200 840A 800E    RRA:      XZERO:  SW       TRA,FRA
                       ;  ENTRY AT X=0
0009: 0240 800A 0000    TYY:              INC      RES1
000A: 8700 0000 0000    RES1:     TRA:    RET
                       ; ENTRY AT X<>0
000B: 2060 8023 0000    FYY:      0:      TEQ      YZERO
0023: 6040 840C 8024
0024: 6040 840D 840E
000C: 5200 8025 8026    FXX:      YZERO:  SW       TFXX,FFXX
0025: 6040 800F 8010
0026: 6040 8016 8027
0027: 6040 8017 8028
0028: 6040 801A 801B
000D: 5100 8015 0000    FYY:      YZERO:  F        FFYY
000E: 5200 8414 841F    FRA:      YZERO:  SW       TFRA,FFRA
                       ;  ENTRY AT X<>0 & Y=0
000F: 0340 8011 0000    TFXX:             DEC      TFXX1
0010: 8060 8411 0003    TFXX:             CALL     TFACT
0011: 8200 8005 8006    TFXX1:    TFACT:  SNA      XX
0012: 8260 8007 0001    1:        TFACT:  SNA      YY
0013: 8260 8008 8014    A'TFRET':TFACT:   SNA      RRA
0014: 8700 0000 0000    TFRET:    TFRA:   RET
                       ;  ENTRY AT X<>0 & Y<>0
0015: 0340 8018 0000    FFYY:             DEC      FFYY1
0016: 8060 8417 0003    FFXX:             CALL     FFACT
0017: 8200 8005 8006    FFXX:     FFACT:  SNA      XX
0018: 8200 8007 0000    FFYY1:    FFACT:  SNA      YY
0019: 8260 8008 801D    A'FFYYY':FFACT:   SNA      RRA
                       ;
001A: 0340 801C 0000    FFXX:             DEC      FFXX1
001B: 8060 841C 0003    FFXX:             CALL     FFACT1
001C: 8200 8005 8006    FFXX1:    FFACT1: SNA      XX
001D: 8200 8007 0000    FFYYY:    FFACT1: SNA      YY
001E: 8260 8008 801F    A'FFRET':FFACT1:  SNA      RRA
001F: 8700 0000 0000    FFRET:    FFRA:   RET
    :                   END

*** ASSEMBLE END ***
```