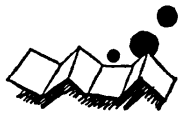
解　説

3. アーキテクチャ



3.7 データフローマシン†

島　田　俊　夫††

1. はじめに

気象，航空，原子力など科学技術計算の分野あるいは高度な知識処理の分野では問題に多くの並列性が内在することは明らかであり，その並列性を自然な形で利用することによりもっと高速な計算を実現したいという要求がある．さらに近年は VLSI 技術が発達し，プロセッサを1チップに納めることが可能となったので，プロセッサチップを多数使用して並列処理を行い，システムの処理能力と性能価格比を向上させようという考えはハードウェアの側からの要求でもある．

その結果，マルチプロセッサシステムの研究が非常に盛んになっている．マルチプロセッサシステムでは多数のプロセッサをいかに統一的に制御し効率を上げるかが大きな課題であり，これを解決するための理論的な裏付けが必要である．すなわち並列計算法を定義する計算モデルが重要である．計算モデルとは一連の計算に関する記法と規則を与え，計算過程を厳密に定義するものである．計算モデルがあればそれに基づく言語の意味を正確に示すことができる．また計算原理を理解することが容易になるとともに，統一的な視点からシステムを開発することができる．もし適切な計算モデルをもたずに並列処理システムを開発すれば，そのアプローチは場当り的になりユーザの負担が非常に重いものとなってしまい，使いやすく，効率のよいシステムを開発することはできない．

フォンノイマン方式がベースにしている計算モデルはチューリングモデルであり，これは逐次計算を定義しているが，並列計算については何も述べていない．したがってフォンノイマン方式に基づく限り，多数のプロセッサを効率よく制御する方法は方式の中にはなく，原則としてユーザが考えねばならない．

† Dataflow Computer by Toshio SHIMADA (Electrotechnical Laboratory).

†† 電子技術総合研究所

一方，データフロー方式は 1974 年に Dennis が提案した並列処理を容易に記述できる計算モデル[1]に基づいている．このモデルは，関数的言語との親和性がよく，関数的言語を使用すれば，プログラマは並列性を明示しなくても並列性が最大限に引き出せるという特徴をもつ．またモデルの基本概念から計算機を開発する場合どのような機能が必要であるかを容易に知ることができる．さらに並列処理の記述能力をモデル自体がもっているため，ユーザの並列処理制御に関する負担を大幅に減らすことが可能となる．

以上のように優れた性質をもつデータフロー方式であるがその普及は遅れている．その理由は従来のアーキテクチャと全く異なっているため，いくつかの問題点を克服する研究が必要であったためであろう．しかしそれらの問題点も多くは解決されており，商用機もいくつか登場してきており，今後の発展が期待される．

本稿は2章でデータフロー方式を支える計算モデルについて述べ，3章でそれらを実現するためのアーキテクチャの機能を述べ，4章ではいくつかの重要な実現例を紹介する．その後データフロー研究における新しい展開について触れる．なおデータフロー計算機の基礎的な事柄については本誌 85 年7月号に小特集があるのでそれらも併せて読まれることをお薦めする．

2. 計算モデル

データフロー計算モデルにおける計算は図-1 のように表現され，これをデータフローグラフと呼ぶ．データフローグラフで，ノード（以後アクタと呼ぶ）は加算，掛算などの演算を表し，アーク（以後リンクと呼ぶ）はデータの入（出）力線を示す．データフローグラフにおける計算は入力リンクにデータを表すトークンを置くことによって始まる．計算を実行する規則は

◆入力リンクのすべてにトークンがそろったらアク

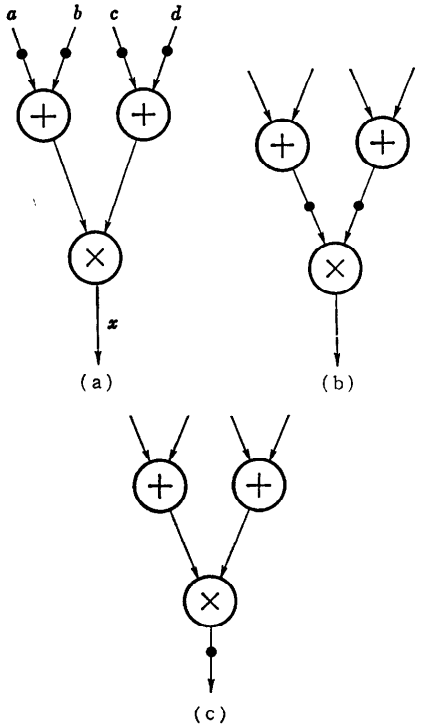


図-1 $x=(a+b)\times(c+d)$ のデータフローグラフ

タを発火する（演算を実行する）.

◆アクタは発火すると入力リンクからトークンを取り除き，演算結果を示すトークンを出力リンクに置く．以上の規則にしたがってアクタは次々に発火し計算が進んでいく（図-1(a),(b),(c)）.

データフローグラフ内のアクタの発火時期，演算結果は入力データのみで定まり，他のアクタとは全く独立である．すなわちアクタ同士は互いに独立に並列実行することができる．さらにこのモデルのアクタは，その値が入力引数の値により一意に定まり他の関数の実行に影響されることはないという数学的な関数の概念に一致していることがわかる.

2.1 静的計算モデルと動的計算モデル

実は上のモデルにはもう一つの規則がある．それは「一つのリンク上にはただ1個のトークンしか存在できない」という規則である．上の議論ではデータフローグラフを1回しか使用しなかった．したがって暗黙のうちに規則が守られた．しかしこの規則に従えばループやリカーションは表現できないし，計算は常に一つのデータフローグラフで表されなければならないので，グラフが非常に大きなものとなり，実用的でない．この制限を除くため二つのモデルが考えられた.

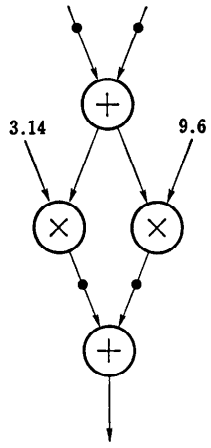


図-2 データフローグラフ上のパイプライン処理

静的計算モデルでは次の規則を追加する.

◆アクタが発火するためにはその出力リンクにトークンがあってはならない.

この規則は，静的計算モデルでは一つのリンクの上に2個以上のトークンがくる可能性があるのでそれを防ぎ，1本のリンクの上のトークンは1個以下であることを保証するものである.

この規則を導入することによりグラフを再使用することが可能となり，パイプライン計算を実現できる．図-2 はパイプライン計算の様子を示しており，第1段階と第3段階の計算がパイプライン並列で実行できる．静的計算モデルではループは容易に実現することができるが，リカーションは再帰回数だけグラフをコピーする必要がある.

動的計算モデルはトークンにタグをもたせ，一つのリンク上に1個以上のトークンを置くことを許す[2]．この場合トークンは $\langle\langle u,c,s,i\rangle,v\rangle$ と表される．ここで $\langle u,c,s,i\rangle$ がタグである．u はこのグラフの実行時のインスタンスを示し，c はそのグラフ内のループを示し，s は行く先のアクタを示し，i はループの繰り返し回数を示す．v はデータの値である.

発火の規則を以下のように変更する.

◆アクタは同じタグをもつトークンが入力リンクにそろったら発火する.

◆アクタは発火したらその入力トークンを取り除き，出力トークンのタグを計算し，出力リンクに新しいタグをもったトークンを置く.

図-3 にこのモデルの計算の様子を示す．最初に入力リンクに到着したトークンはタグが異なるためアクタを発火できない．次に到着したトークンのうち右側

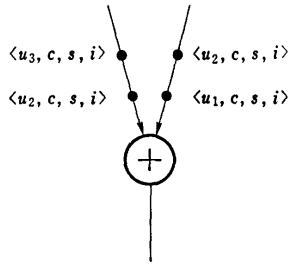


図-3 動的計算モデル

のトークンは左側のリンクの1番目のトークンとタグが一致するのでアクタを発火できる．入力リンクに残ったトークンはタグが一致するトークンの到着を待つ．このモデルはタグを操作する命令をもつことによって，ループ，リカーション，パイプライン計算を容易に実現できる．

3. アーキテクチャ

3.1 計算モデルに基づいたアーキテクチャに必要な機能

2章で述べた計算モデルに基づいたアーキテクチャを考えるとき必要となる機能について述べよう．

最初に静的計算モデルについて考察する．まず必要となるのは入力リンク上にトークンを保持する機能である．静的計算モデルでは一つのリンクの上に置けるトークンは1個だけである．またトークンはタグをもたないのでタグによる連想機能は必要がない．したがって静的計算モデルのトークン保持機能は普通のメモリでよい．

次にアクタが発火できるかどうかの検出機能が必要である．一番簡単な方法は入力トークンが到着する度に他の入力リンクにトークンがあるかないかをチェックすることである．各アクタに対して入力トークンの数を記録するカウンタを備えておき，カウンタが入力リンクの数と同じになれば発火するようにする方法もあり，リンクの数が多いとき効果的である．

実行に際して並列性が多ければ，発火するアクタがプロセッサの数より多くなることが予想されるので発火可能となったアクタを保持するキューが必要である．

このほかに演算実行，命令フェッチ，マルチプロセッサを結合するネットワークなどのマルチプロセッサに共通の機能が必要である．

静的計算モデルに基づくプロセッサのアーキテクチャの1例を図-4 に示す．このマシンのデータフローグラフに相当するプログラムを図-5 に示す．このプログラムの命令形式は，まず演算名があり，続く2語は入力トークンを置く場所である．右側のリンクは結果のトークンを送る命令のアドレスである．このプログラムは図-4 のプログラムメモリに置かれる．キューの中には発火可能なアクタのアドレスがある．このアドレスがフェッチユニットに送られるとフェッチユニットはそのアドレスの命令をフェッチし，演算ユニットに送る．演算ユニットは計算を行い，結果を更新ユニットへ送る．更新ユニットはプログラムメモリ内のトークンが指定する命令の対応するフィールドへ入力トークンの値を書き込み発火可能かをチェックする．発火可能ならそのアドレスをキューへ送る．次の命令が他のプロセッサの場合はトークンはネットワークを通じて他のプロセッサへ送られる．

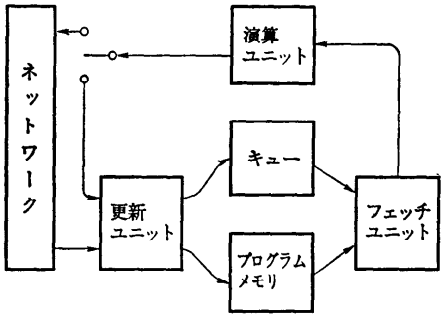


図-4 静的アーキテクチャの例

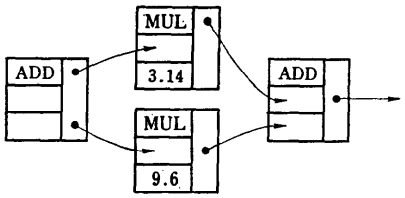


図-5 図-2 のデータフローグラフのプログラムの形式

静的アーキテクチャはフォンノイマン計算機に比べて必要なハードウェア量がそれほど多くないという利点をもっている．一方，このアーキテクチャは入力リンク上に1個しかトークンを置かないということを保証するために二つの問題点をもっている．一つは出力リンクにトークンがないことを確認するため，発火したアクタはその入力リンクとつながっているすべてのアクタに対し，出力リンクが空になったことを知らせる信号（アクノリジ信号）を送らなければならない．二つ目は同じサブルーチンを同時に2カ所でコールしたときにプログラムコードを共有することができな

い．この場合同じプログラムコードを二つもたなければならない．つまりサブルーチンコールの度にプログラムコードをコピーする必要がある．ループの場合にはループボディの最後で，すべての必要な演算が終了したことを確認する同期を取れば，コードを再使用できる．しかしこの方法ではループの各世代を並列に処理することはできない．

次に動的計算モデルについて考察する．動的計算モデルにおける入力トークンの保持機構は複雑である．このモデルでは入力リンク上にトークンを任意個置くことができる．したがってトークンを保持するためには静的計算モデルのように各アクタに定まったサイズのエリアを用意しておく方法では十分でなく，多数の入力トークンを保持するバッファが必要である．またトークンはタグをもっており，このタグを用いてペアとなる入力トークンを識別するので連想機能が必要である．並列性が大きい場合，保持すべきトークンの数はかなり大きくなるので，大容量の連想メモリが必要である．

そのほかに必要なものは静的アーキテクチャと同じである．動的計算モデルは連想メモリのハードウェアが重いが，アクノリジ信号を返す必要がない，同じプログラムコードを同時に多数のプロセスが共有でき，ループの繰り返しを時間方向に展開して並列に実行できるなど，静的アーキテクチャより並列度の高い実行が可能である．

3.2 アーキテクチャの例

ここでは実際にハードウェアが製作されたシステムをいくつか取り上げ，そのアーキテクチャを解説する．

3.2.1 マンチェスタ大学のデータフローマシン[3)]

動的計算モデルに基づくマシンで 1980 年から稼働しており，世界で最も早く実現されたデータフローマシンであろう．基本構成は 4 つのモジュールからなるリングで，これは I/O スイッチを通じてホストに接続されている（図-6）．リング内の各モジュールは独立にパイプライン動作ができる．

トークンキューは 96 ビットのトークンを 32K 個保持できる FIFO バッファで，プロセッシングユニットとマッチングユニットの速度差をこのバッファで吸収する．マッチングユニットはタグを識別子としてペアとなるトークンを見つける連想記憶である．連想方式は並列オープンハッシュでその並列度は 16 である．記憶容量は 1.25M トークンである．このマッチングユニットにはオーバフローを処理するマイクロプロセッサが付いている．命令メモリは 64K 命令の容量をもち，命令長はトークンを送り出す命令の数に応じて 32～70 ビットと変化し，200ns に 1 個の命令をフェッチできる．

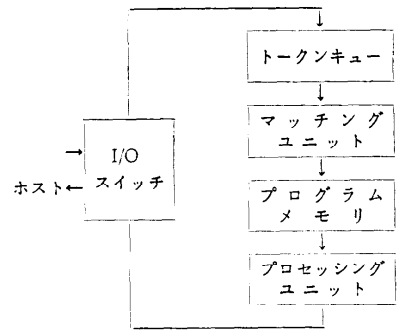


図-6 マンチェスタ大学のデータフローマシン

プロセッシングユニットは現在 14 台のファンクションユニットをもっており，互いに並列に動作する．ファンクションユニットは最大 20 まで実装できる．ファンクションユニット 1 台の性能は 0.27 MIPS で，全体では 3.7 MIPS である．

このシステムの特徴はマッチングユニットに並列度の高い方式を採用したこと，プロセッシングユニットのみを多重化したことなどであろう．このシステムの問題点の一つはマッチングユニットの方式にオープンハッシュ方式を採用したこと，ハッシュアドレスの衝突が並列度を越えるとオーバフロー処理に入るため処理速度が非常に遅くなることである．しかもこのような状況はかなり頻繁に起こるため実用的なプログラムの実行速度はかなり悪いようである．第 2 の問題点はプロセッシングユニットのみを多重化したため，他のユニットとのバランス上ファンクションユニットに高性能のものを採用することができず，高い性能を達成できない点であろう．

システムの評価は 29 個の数値計算のプログラムで行われ，平均約 2MIPS の性能を得ている．このシステムはデータフロー方式の高速性を示すことはできなかったが，データフロー方式が正しく並列性を引き出して動作することを世界で最初に実証した意義が大きい．

3.2.2 マサチューセッツ工科大学 (MIT) のデータフローマシン

MIT には静的アーキテクチャを研究する Dennis のグループと動的アーキテクチャを研究する Arvind の

二つのグループがある．Dennis のグループではデータフロー計算機のソフトウェアとハードウェアを評価するためにエンジニアリングモデルと称するマシンを製作した[4]．全体はプロセッシングユニット，ネットワーク，ホストの3者で構成されている(図-7)．プロセッシングユニットはルータネットワークによって接続されており，通常の実行の際のパケット通信はネットワークを通じて行われる．ホスト計算機 PDP11/40 のユニバスには8枚のプロセッシングユニットと6枚のパケットルータネットワークボードが接続されており，ホストはシステムの初期化，メインテナンス，モニタ，入出力などを行う．

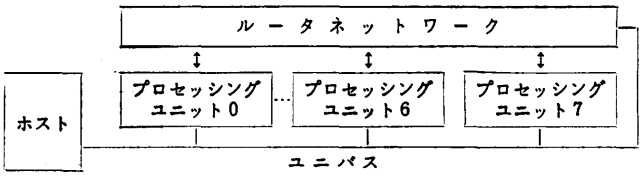


図-7 MIT データフローエンジニアリングモデル

プロセッシングユニットの構成を図-8 に示す．プロセッサは AM 2903, 2904, 2910 で製作されたフォンノイマン型のマイクロプロセッサである．データメモリ (64K バイト) 上に2章で記述した静的アーキテクチャのデータフローグラフを構成し，マイクロプログラムによるエミュレーション実行を行う．

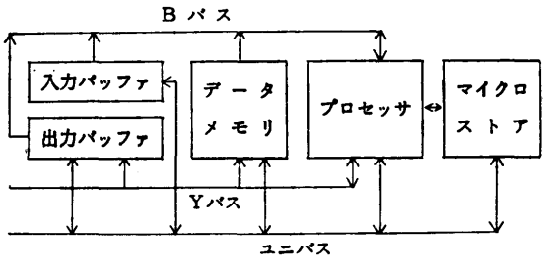


図-8 プロセッシングユニット

ネットワークは 2×2 ルータ 12 個を使って 8×8 のネットワークを構成している．

高級言語 VAL のコンパイラは DEC2060 上に実現されており，コンパイルされたプログラムはローカルネットワークを通してホストにロードされる．このシステムの実行速度はデータフロー命令で 1.3KIPS である．

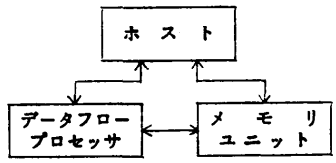
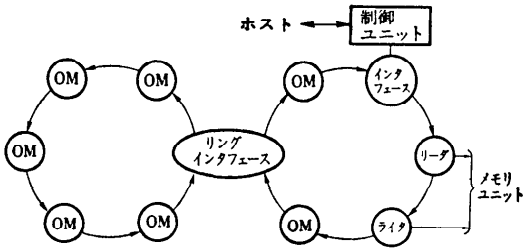


図-9 NEDIPS の全体構成

Dennis は現在 Dataflow Technology Corporation という会社を主宰して商用のデータフロー計算機を開発している．そのアーキテクチャは静的で，パイプライン演算器をデータフロー方式で制御する点を特徴としている．



OM: オペレーションモジュール
図-10 NEDIPS データフロープロセッサの構成

Arvind のグループは数10台の LISP マシンと MEF (Multiprocessor Emulation Facility) と呼ばれるマイクロプログラムドプロセッサをローカルネットワークで接続し，並列アーキテクチャのさまざまなテストができるテストベッドを製作中である．このシステムでは高級言語 Id のコンパイラが LISP マシン上に実現されており，Id で書かれたプログラムをデータフローグラフに翻訳し，LISP マシン及び MEF 上で解釈実行できる．

3.2.3 日本電気の NEDIPS[5]

世界最初の商用データフロー計算機で静的アーキテクチャである．システム全体の構成を図-9 に示す．データフロープロセッサは制御ユニットを介してホストと高速データパス (80 MB/sec) によりメモリユニットと接続されている．ホストはプログラムの起動や結果の受取，入出力などを受け持つ．

データフロープロセッサは制御ユニット，算術ユニット，アドレッシングユニットで構成されており，後二者はオペレーションモジュールをリングバスで接続した構造である (図-10)．アドレッシングユニットのオペレーションモジュールは固定小数点演算，メモリインタフェースなどであり，算術ユニットのオペレーションモジュールは浮動小数点演算とリングインタフェースである．算術ユニットは最大3台まで拡張できる．

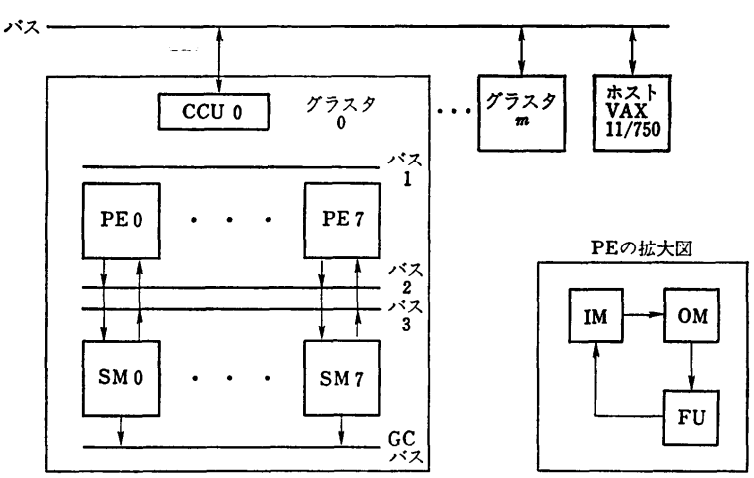
オペレーションモジュールには単項演算モジュール

と2項演算モジュールがある．単項演算モジュールはバスインタフェース及び単項演算カードからなる．2項演算モジュールはバスインタフェース，待ち合わせ記憶，2項演算カードからなる．

バスインタフェースは演算にともない増減するデータ量を吸収するFIFOバッファである．

パケットは64ビットで上位16ビットはモジュールのアドレス，次の8ビットは命令のアドレス，残り40ビットがデータである．

メモリユニットは3バス構成で，高速バス2本はデータフロープロセッサが使用し，150 nsでデータのリード/ライトが行える．低速バス1本はホストが使用する．メモリボードは128 KW/枚の容量でどのバスにも接続でき，最大64 MWまで拡張できる．



CCU：クラスタ制御ユニット FU：演算ユニット
IM：命令メモリ SM：構造メモリ
OM：オペランドメモリ

図-11 プロトタイプDFM

データフロープログラムはテンプレートアセンブラで記述し，FORTRANプログラムのサブルーチンとして使用する．プログラムメモリはリング上のオペレーションモジュール内にあり，データフロープログラムがホストからダウンラインロードされる．ホスト上のFORTRANプログラムがデータフローサブルーチンをcallすると起動データがホストからデータフロープロセッサに送られデータフローの実行に入る．

本システムはプログラムメモリのサイズがあまり大きくないこと，タグの幅が狭いことのため，プログラムメモリにストアできるサイズのルーチンを実行中ずっと使用する応用，たとえば画像処理などに適しているが，科学技術計算全般に使用するのには困難があろう．

日本電気ではNEDIPSのほかに，IMPPという1チップの画像処理専用データフロープロセッサ[6]も実現している．

3.2.4 NTTのDFM[7]

信号処理を目的とした動的アーキテクチャのデータフローマシンで，高速リスト処理，lenient consなどを特徴としている．試作されたものはプロトタイプでその構成を図-11に示す．8台のプロセッサ(PE)と8台の構造メモリをまとめてクラスタと呼ぶ．クラスタ相互及びホストとの接続はバスである．クラスタ内のPE間は1本のバス，PEとSEの間も別の2本のバスで接続されている．SEのガーベッジコレクション用にGCバスがSE間を接続している．

PEはIM, OM, FUの三つのユニットで構成されている．IMはデータフロープログラムをストアし，データを運ぶ結果パケット(RP)がくると，そのRPが指示する命令と運んできたデータをオペランドメモリ(OM)に送る．

OMは待ち合わせ記憶で1,024のCAMブロックからなっている．各関数のインスタンスに対して一つのCAMが割り当てられる．CAMのサイズは32ワードで，これが8つの組に分けられており，組の識別はパケットの運ぶタグで行われる．各組の中の4ワードは同時にアクセスされペアとなるデータを探す．この方式により待ち合わせ記憶のハードウェアの簡素化を計っている．

FUは命令を実行し，RPを生成し，それをIMに送る．一つの関数は1台のPEに割付けられ，実行される．したがって関数レベルの並列実行はプロセッサ間でなされ，命令レベルの並列実行はPE内のモジュールのパイプラインにより行われる．

SMはリスト命令を実行する構造メモリであり，lenient consの制御も行う．また要求駆動型の評価も行うことができるようになっている．

現在プロトタイプ DFM は，2台の PE と2台の SE が稼働中である．単一代入型の高級言語 VALID がホスト上に実現されている．性能評価はレジスタトランスファレベルのシミュレータで行われ，1台の PE-SE システムでは VAX11/750 の 5〜8 倍，DEC 2060 の約半分の性能が得られている．また 32 台の PE-SE システムで1から n までの整数の総和を分割統治法で計算した場合，ほぼリニアな速度向上が予測されている．

3.2.5 群馬大の DFNDR[8]

強力な連想記憶をもった動的アーキテクチャのマシンで，1981年より稼働している．

全体はトークンメモリ (TM)，プログラムメモリ (PM)，実行ユニット (FU)，ホスト (HC) の4つの部分からなる (図-12)．このマシンの特徴は，トークンの待ち合わせを行う TM がマルチポートの連想記憶となっており，すべての FU が TM の任意の語をアクセスできることである．どの FU でも TM 上で発火可能となったノードを取ることができるため，FU 間の負荷分散の問題は起こらない．ただ TM のハードウェアが複雑となること，TM の連想アクセスの速度と FU の速度の関係などから非常に多数の FU を接続することは難しいであろう．

システムはアーキテクチャ実験のために製作されたので，速度は速いものではないようである．最近はフォンノイマン型とデータフローの中間的性格をもったコントロールフローの研究が行われている．

3.2.6 電子技術総合研究所の SIGMA-1[9]

科学技術計算用スーパコンピュータにおいてデータフロー方式の可能性の実証を目指したマシンで，動的アーキテクチャを採用しており，その最大性能は約 380 MFLOPS である．プロセッシングエレメント (PE) 1台のプロトタイプが 1983 年に，LSI 版の PE 4台のシステムが 1986 年に稼働しており，全システムは 1987 年に完成予定である．

全体は図-13 に示すように階層構造のネットワークで接続されている．4台の PE と4台の構造メモリ (SE) を局所ネットワーク (クロスバー) で接続したものをグループと呼び，32 グループを大域ネットワーク (10×10 のルータによる多段ネットワーク) で接続している．PE の構成は図-14 に示すように5つのユニットからなり，2段のパイプラインを形成している．パイプラインの段数を少なくし，並列性が小さい場合のパイプラインの遅れを小さくしている．FU はプログラムをストアし，入力トークンが指示する命令をフェッチし，EU に送る．待ち合わせ記憶 MU は連鎖ハッシュ方式を採用し，平均アクセス時間を短縮している．EU は演算を実行し，DU は EU の結果をパケット形式に直す．

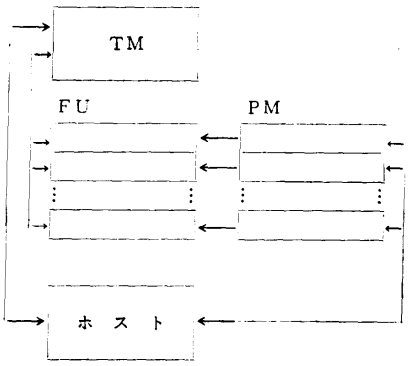


図-12 DFNDR

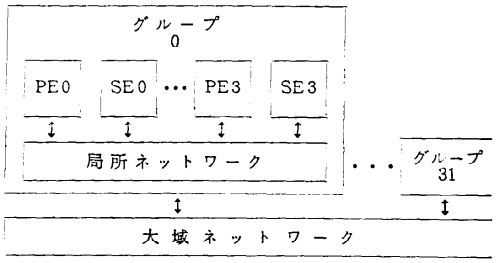


図-13 SIGMA-1 の全体構成

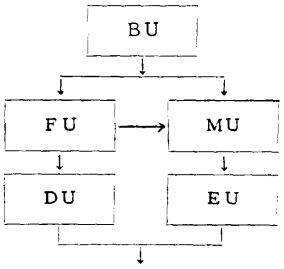


図-14 SIGMA-1 の PE の構成

高級言語は単一代入型でC言語との互換性を考慮した DFC がホスト上に実現されている．性能評価はプロトタイプの PE 1台に対しては実機でなされ，sinx の計算で 2 MFLOPS，ベクトル計算のベンチマークで平均 0.3 MFLOPS を記録している．プロトタイプ PE の最大性能 2 MFLOPS に対してベクトル計算の平均性能が低いのは，命令のオーバヘッドが大きいこと，構造データのパイプラインが十分でないためで，

LSI 版ではこの点を改良中である．また多数台のシステムに関しては，スカラレベルの並列性の大きいモンテカルロ計算においては VP100 の 10 倍程度の性能が予測されている[10]．

4. 最近の話題

本章ではデータフロー研究における最近の話題をいくつか取り上げて紹介する．

4.1 マクロデータフロー

この言葉はもともとイリノイ大学の Kuck らが FORTRAN をデータフロー解析して並列実行部分を抽出し，フォンノイマン型のマルチプロセッサで並列実行する方法に付けた名前であったが，ここでは並列性の粒度を大きくした，たとえば関数レベルデータフローなどを意味するのに用いる．

粒度を大きくする理由はいくつかある．データフローモデルでは状態という概念がないため，履歴依存性のある計算を記述できない．またデータフローモデルではあるリンク上にトークンがあるかどうかをテストすることはできない．このため同期はすべて受身となり，ある時点で積極的に同期を取ることが難しい．具体的にはあるまとまった計算の終了判定などのオーバヘッドが大きくなる．

さらに初期のデータフロー研究ではプログラムからいかに多くの並列性を引き出すかが焦点であったが，最近はあまり多くの並列性を引き出すと待ち合わせ記憶やカラーなどの資源が足りなくなり，実行が継続できなくなるので，並列性をいかに制御するかが焦点になっている．これらの問題は実行になんらかの順序性をもたせれば比較的容易に解決できる．このため並列性の粒度を関数レベルとし，関数内の実行に逐次処理のような順序制御を導入する方法が研究されている．

4.2 アーキテクチャ

ハードウェア側からの要請として，実行を高速化するためレジスタを使用したいという問題がある．データフローグラフ上で1本の糸につながれた部分をレジスタで実行する機能をデータフロー方式にもたせることは可能であるが，関数内の実行に順序制御を導入すればより容易にレジスタが使用できる．

世界のデータフロー計算機研究のほとんどは動的アーキテクチャを採用している．その理由はより多くの並列性を引き出せること，データフローグラフを多数のプロセスにより共有できるためメモリアクセス回数が少なくなり高速化できるためであった．しかしハードウェアを製作してみると連想記憶部分が非常に重いので，ここを簡素化するため，静的アーキテクチャを改めて見直す動きもある．静的アーキテクチャではデータフローグラフを1回しか使用できないので，同じ関数を何回も実行する場合，実行における命令アクセスの比重が非常に高くなり実用的でないとされていた．しかしこの点を改良すればハードウェアが簡素化され LSI の製作に適している．

データフローがスカラーレベルの並列性のある計算を高速に実行できるということはこれまでの研究からほぼ明らかとなった．今後はパイプラインをデータフローの中に取り込み，いかに効率よく制御するかも面白い問題である．

7. おわりに

データフローの将来はどうなるのであろうか．この点に関してデータフローモデルの考案者である Dennis 教授は次のように述べている[11]．

1990 年代のスーパコンピュータは現在のものと非常に異なったアーキテクチャとなるだろう．その第1の点は，高性能を達成するために 1,000 以上の超並列を追求する．第2は CMOS のカスタム LSI 数種類で全体を構成する．第3はプログラミングに関数型言語が使用される．第4はフォールトデテクション技術が大いに使用される．

この予測が当たるかどうかはともかく，この言葉の中には今後のデータフロー研究の方向が示唆されていると思う．またデータフローの商用機が日本，米国で登場し，1970年代後半から盛んになったデータフロー研究がそろそろ収穫期を迎える．今後は 1,000 台以上の超並列を目指した研究とともに，より実用性を指向した研究も盛んになることであろう．

参考文献

1) Dennis, J. B.: First Version of a Data Flow Procedure Language, Lecture Notes in Computer Science, Vol. 19, Springer Verlag, pp. 362-376 (1974).
2) Arvind, Gostelow. K. P. and Plouffe, W.: An Asynchronous Programming Language and Computing Machine, TR114a, Univ. of California, Irvine (1978).
3) Gurd, J. R., Kirkham, C. C. and Watson, I.: The Manchester Prototype Dataflow Computer, Comm. ACM, Vol. 28, No. 1, pp. 34-52 (1986).
4) Dennis, J. B., Lim, W. Y. P. and Ackerman,

W. B.: The MIT Dataflow Engineering Model, Proc. IFIP, pp. 553-560 (1983).
5) 能美，本田，九十歩，天満：データフローコンピュータ NEDIPS のアーキテクチャ，情報処理学会第 30 回全国大会論文集, pp. 235-236 (1985).
6) 松本他：画像処理分野をねらったデータフロー型プロセッサ LSI，日経エレクトロニクス (4. 8, 1984).
7) Amamiya, M., Takesue, M., Hasegawa, R. and Mikami, H.: Implementation and Evaluation of a List-Processing-Oriented Data Flow Machine, Proc. ISCA, pp. 10-18 (1986).
8) Sowa, M. and Murata, T.: A Data Flow Computer Architecture with Program and Token Memories, IEEE Trans. Comput., Vol. C-31, No. 9, pp. 820-824 (1982).
9) Shimada, T., Hiraki, K., Nishida, K. and Sekiguti, S.: Evaluation of a Prototype Data Flow Processor of the SIGMA-1 for Scientific Computations, Proc. ISCA, pp. 226-234 (1986).
10) Shimada, T. and Sekiguti, S.: SIGMA-1-A New Generation Data Flow Supercomputer, in Supercomputers, Univ. of Texas Press (1986).
11) Dennis, J. B.: Data Flow Computation, NATO ASI Series, Vol. F14, Springer-Verlag (1985).

(昭和 61 年 11 月 14 日受付)