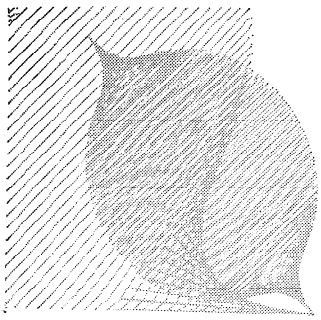
UDC 681.327



小特集

Ⅲ. 非フォンノイマン形計算機

樋口哲也
古谷立美
電子技術総合研究所

1. はじめに

計算機は 1950 年代に実用化が始まって以来，驚異的な性能向上を続けてきたが，それを支えてきたものは，主に半導体などの素子技術の進歩であった。この間，計算機アーキテクチャにおいても，幾つかの改良は成されたが，現在世の中で広く用いられている計算機のほとんどは，初期に提案されたフォンノイマン(von Neumann，以下 VN と略す）形と呼ばれる方式に基づいている。

これに対して，今日，半導体技術は理論的限界に近づき，更に性能を向上させるためには，従来の VN 形の枠を破る新しいアーキテクチャが必要と考えられるようになった。この新しいアーキテクチャは，VN 形と異なるものという意味で“非フォンノイマン形”と呼ばれ，現在，世界各所で盛んに研究が進められている。

VN 形の明確な定義はないが，一般に VN 形を特徴づけるものとしては，主に次のようなものが認められている。

（1） プログラムをメモリーに記憶する
（2） 逐次処理
（3） 線形アドレス記憶

非 VN 形を上記以外の性質をもつものとして広義に解釈すると，スタックやデスクリプタ方式，タグ方式などのような方式を非 VN 形の範ちゅうに入れることができる[1]。しかし非 VN 形を革新的なアーキテクチャと取れば，それらは VN 形の拡張と考えるべきであろう。本稿では，非 VN 形を Treleavan 氏の示した狭義の意味に解釈して議論する。Treleavan 氏は，計算機の計算機構を制御機構とデータ機構から分類し，VN 形の位置づけを明確にした[2]。表 1 はその分類である。制御機構とは，プログラムを構成する命令の実行順序の制御方式であり，データ機構とはデータの扱い方である。データ機構の“値によるもの”は，実行時に生成されるデータが，そのデータを使用する命令に直接，値として送られるのに対し，“リファレンスによるもの”は，変数のようにデータを格納する場所を用意し，命令からアドレスなどで格納場所を指定し，データにアクセスするものである。

表 1 計算機構の分類

		データ機構	
		値によるもの	リファレンスによるもの
制御機構	逐次	—	フォンノイマン形制御フロー
	並列	データフロー	並列制御フロー
	再帰	ストリング リダクション	グラフリダクション

以下，まず第 2 章で非 VN 形の必要性を述べる。第 3～5 章では，表 1 の分類に従って，非 VN の三つの形(並列制御フロー，データフロー，リダクション)について，その基本原理と世の中の研究開発状況を示す。

2. 非フォンノイマンアーキテクチャの必要性

非 VN 形の計算機アーキテクチャが必要とされる背景には，大きくわけて次の三つの理由が考えられる。

第一は，VN 形のプログラミング言語，および方法論の欠点が明らかになってきたことと関連する。VN 形言語ではメモリーセルの概念がプログラム上の変数に反映されているために，変数は計算状態の保存のために使われる。プログラマは，変数への代入でその状態を意図的に変化させることによって目的とする計算を達成する（つまり状態の変化による副作用を利用する)。従って，たとえ同一の変数であってもその意味はその時点の計算状態によって異なるため，プログラムの正しさの検証やプログラムの理解はシステムが複雑化・巨大化するにつれて困難になってくる。これを解決するには，関数形プログラミングなど，新しいプ

ログラミングスタイルが必要である。しかし，それらの計算モデルと従来の VN 形アーキテクチャとの間には意味的にギャップがあり，VN 形アーキテクチャによる実行では処理効率が落ちやすい。

第二は，VN 形マシンを複合化したマルチプロセッサの経験に基づくものである。従来の VN 形マルチプロセッサの中には，そのプログラム記述にあたって特定のアーキテクチャや副作用を念頭におかねばならず，そのために並列プロセス同期や資源管理などにおいてプログラマに負担を強いるものが少なくなかった。例えば，副作用による誤動作を避けるため，プログラマが問題に潜む並列性を調べて，それをプログラムに明示しなければならないこともあった。そこで，もっと純粋に，解法アルゴリズムに即した形でプログラムを記述し，実行時にそのプログラムのもつ並列性を機械的に検出，実行できるようなアーキテクチャが必要とされるようになった。第4章で述べるデータフローマシンはその代表例である。

第三は VLSI 技術の進歩に起因するものである。VLSI 技術の進展によって大規模な回路を少数チップで実現できる見通しがつき，従来ソフトウェアで実現していたアルゴリズムを VLSI 化する動きが出てきた。また VLSI は同一回路の繰返し構成を実現しやすいという特徴があるため，それを生かした VLSI 向きのアルゴリズムの開発も行なわれている。そのようなアーキテクチャの中には，シストリックアレイなど，VN 形の枠を離れたものが少なくない。

3. 並列制御フロー

VN 形である単一制御フローとは，プログラムカウンタのように命令の実行順序を規定するものがあり，これに従って命令を次々と実行していく方式である。並列制御フローとは，複数の VN 形計算機を結合し，互いに交信しながら一つの仕事を行なう計算機と考えればよい。

このタイプは VN 形から非 VN 形への進化の第一歩であり，その提案はかなり古く，1970年代には多くの試作機が作られた。これらの並列計算機は並列プログラムの難しさから一時壁にぶつかったが，現在では並列処理用高級言語が開発され，それらの高級言語マシンという形で幾つかの並列計算機が提案され，製作されている。Transputer[3] は，occam という並列処理言語の高級言語マシンである。Intel iAPX 432 は並列処理言語 Ada をサポートできるように設計されている。

並列制御フローには，いろいろな変形が存在する。VN 形と並列制御フローの中間に位置するものとしては SIMD 形並列計算機がある。SIMD は Single Instruction Streams Multiple Data Streams の略で，多数のプロセッシングエレメント (PE) を用意し，それらに同一の命令を送り，それぞれの PE は自分のもつデータに対して演算を行なう。ILLIAC IV は最初の本格的 SIMD 形並列計算機として広く知られている。SIMD 方式ではシンプルで同一な PE を使用するため VLSI に適したアーキテクチャと考えられる。MPP (Massively Parallel Processor)[4] は 128×128 台の二次元プロセッサアレイで画像処理を行なうことを目的に開発された。また connection machine[5] は人工知能の応用で用いる意味ネットワークの1ノードを1台の PE に割り当て，高速な意味ネットワーク操作をサポートしようとしている。

データフローについては次章で述べるが，並列制御フローとデータフローの中間に位置づけられるものもある。これは，一般にマクロレベルのデータフローなどと呼ばれるもので，プログラムを逐次実行可能なマクロ命令の集合で表わし，それらの間の駆動をマクロ命令間のデータの転送で実現しようというものである。Illinois 大学の Cedar[6] はこの代表的なもので並列制御フローとデータフローの欠点を補なうアプローチとして注目されている。

4. データフロー

科学技術計算では，例えばベクトルプロセッサが取扱うアレイ計算にみられるような規則的な並列性が存在し，それらに対しては対象が限定される分だけハードウェアによるサポートも実現しやすい。しかし，適用する問題に特有で，しかも規則性を見いだしにくい並列性が存在することも数多い。一般の問題において並列処理の効果を上げるためには，そういった不規則な並列性についてもサポートできる必要がある。それも，サブルーチンといった大きな単位だけでなく，細かな命令レベルでも並列性を生かすことが望ましい。データフローマシンは，そのような並列処理をプログラムのしやすさを損なわずに実現するための方式の一つである。

データフローの実行原理は，必要な演算オペランドがそろった演算命令から実行が可能になることである。プログラムカウンタによる逐次制御の概念はなく，データの流れが実行順序を規定している。例えば図1(a)で，黒い丸印は利用可能なデータを示しているが，演算命令を表わす “+1”，“/” のノードは，それらを用いて直ちに並列に演算を開始できる。これに

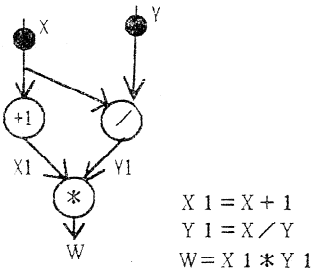


図 1 $W=(X+1)*(X/Y)$ のデータフロープログラム

対し“*”の実行は，“+1”，“/”命令の実行結果として $X1$, $Y1$ のリンクにデータが到着してからとなる。

データフローは，VN 形に比べ次のような特徴をもつ。

（1） データの流れが実行順序を決めるために各演算は他と独立であり，従って自然な形で並列処理が可能になる。

（2） メモリーセルによるデータ保存の概念がない。つまり，データは，ある演算に与えられるとそこで使いつくされる。このため VN 形言語で生じる可能性のある，副作用による並列処理の誤動作の危険がない。

図1（a）のデータフローの図式表現は図1（b）の文章形プログラムに変換できる。これは各変数に対する代入は1回だけとする単一割り当て規則に基づく変換であり，この規則に従うことでプログラマは並列性を明示することなく，問題に内在する並列性をすべて表現できる。

このような計算モデルをサポートするデータフローマシンのアーキテクチャは，実行制御の面からみると次の二つに大別できる。第一はデータの到着待ち合せ場所として命令の中に格納場所をもち，すべて入力がそろった命令から実行するタイプである。これは静的アーキテクチャとも呼ばれ，例えば信号処理などの応用向きとして，MIT の Dennis 氏[7]，TI のマシンなどが提案，製作されている。但し，このタイプでは複数の入力トークンを演算ノードの入り口で待たせられないので，たとえ，そこで論理的には並列処理が可能でもそれを生かせない。

これに対し，第二のタイプはそのような場合でも演算ノードを複製することで並列性を生かす。動的アーキテクチャとも呼ばれ，データを待ち合せるためのマッチングユニットを，命令メモリーとは別にもたせる。このタイプの場合，再帰的あるいは並列性が動的

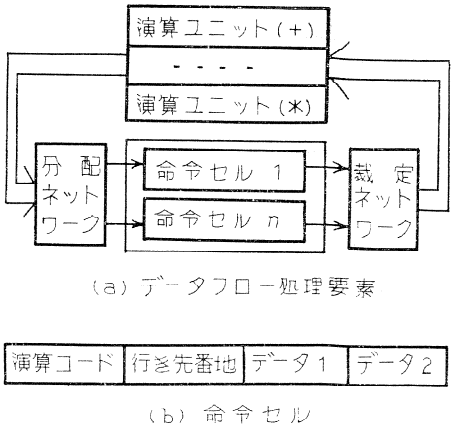


図 2 Dennis 氏のデータフローマシン

に変化するようなプログラムも実行することができ，数値演算のほかに記号処理など，非数値的応用にも適用できる。MIT の Arvind 氏[8]，ETL[9]，NTT，Manchester 大学などのマシンが提案，製作されている。

データフローマシンの例として，図2（a）に Dennis 氏のマシンの基本部分を説明する。データの待ち合せは中央の命令セル（図2（b））で行なわれる。命令セルは演算コード，結果の行き先番地，入力データ領域（待ち合せ用）から成る。ここで入力がそろった命令は裁定ネットワークに送られ，その命令に含まれる演算コードに対応する演算ユニットに送られる。そこでの演算結果は行き先番地とペアのパケットにして分配ネットワークに送られる。ここでは，行き先番地に従って，演算結果を必要としている命令セルの中にそれを書き込む。これによりまた別の命令セルが実行可能となり，同様のサイクルを繰返し，計算が進行する。並列処理能力を高めるためには，図2（a）のような基本データフロー処理装置を複数台組み合せ，マルチプロセッサ構成をとるのが一般的である。

5. リダクション

リダクションとは，与えられた文字列を，書き換え規則に従って変換する書き換え操作である。リダクションを繰返して行ない，それ以上変換できない文字列が得られるまでの過程が計算に対応する。図3に例を示す。枠で囲まれたのが定義体，つまり書き換え規則の集まりである。いま，プログラム中に現われる変数 W の値を求めるとする。W を求めようとすると，まず W についての書き換え規則が起動され，W が（*, $d1$, $d2$）で置き換えられる。これは W が $d1$ と $d2$ の積であることを示すが，$d1$, $d2$ の値が未定のため，更に $d1$, $d2$ についての書き換え規則が順番に適

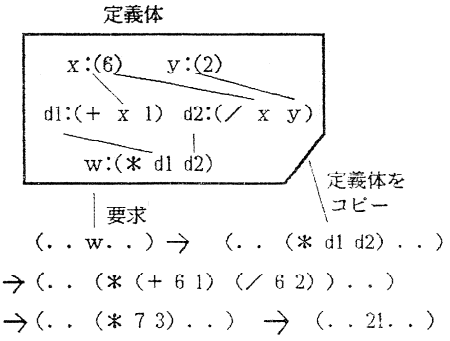


図 3　リダクションの例(ストリングリダクション)

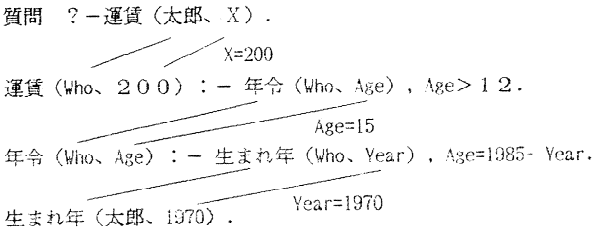


図 4　論理形プログラムの例

用され，図3にあるような変換過程を経て，最終的に W が 21 として得られる。このような書き換えシステムによって，すべての計算可能な関数が表現でき，十分な計算モデルとなることが知られている[10]。

リダクションマシンの研究動機としては，まず関数形言語の実行があげられる。関数形言語は，関数定義によりプログラム記述を行おうとするもので，その利点としては，

（1）　計算機構が代入操作でなく書き換えに基づくために副作用がなく，VN 形に比べて並列処理が実現しやすい

（2）　関数定義により変数が常に同一の値を表わすため，プログラムの了解性がよい

などがある。

最近では述語論理に基づく論理形言語の並列処理を目的としてリダクションマシンを構築しようとする動きも盛んである。これは論理形プログラミングでの命題の真偽の証明過程がリダクションとも見なせるからである。例えば，図4は論理形プログラムの例で，12才以上の運賃が200円であるときに太郎の運賃がいくらであるかを調べるものである。質問が“運賃（太郎，X)”としてなされると，運賃の節の証明にかかり，その中で年令の節の証明の必要が生じて年令の節に制御が移る。そこから今度は生まれ年の節の証明に入る。それが成功すると，逆もどりの形で年令，運賃の節へと制御が戻り，X に 200 が割りつけられて証明が終了する。この過程はリダクションと類似しているといえる。

リダクションマシンの機能は，通常の計算機でいえばプログラムに対応する書き換え規則集に基づき，与えられた文字列から，(ⅰ)書き換え可能な部分を見つけ，(ⅱ)書き換えによって新たな項を作る，ことである。リダクションマシンのアーキテクチャは，定義体のコピーをとるか，あるいはその共有化を図るかによって，ストリングリダクション，グラフリダクションに分けられる。前者には Berkling マシン[11]，Mago マシン，後者には Turner マシン，ALICE, SKIM マシンなどがある。これらは関数形言語の効率的実行を目的としている。ALICE については論理形言語もサポートされる。

国内でも，論理形言語の並列処理を目的に，東大の PIE，新世代コンピュータ技術開発機構（ICOT）の PIM/R が提案されている。なお本章にあげたマシンの概要については文献[12][13]を参照されたい。

6. おわりに

本稿では非 VN 形に属するアーキテクチャとその必要性について概説した。VLSI 技術の進展により任意の回路構成の実現性が高まり，今後知識情報処理など高度な応用が進むにつれて，非 VN 形アーキテクチャの比重はますます増えるものと思われる。

（昭和60年6月27日受付）

文　献

(1) 相磯，他：「新しい計算機アーキテクチャ」情報処理 19, 12, p. 1182 (昭 53-12)
(2) P. C. Treleaven, et al.: "Data-Driven and Demand-Driven Computer Architecture" *ACM Computing Surveys* 14, 1 (1982)
(3) Barron, et al. "Transputer Does 5 or More MIPS Even When not Used in Parallel" *Electronics* 17, p. 109 (1983-11)
(4) K. Batcher: The Massively Parallel Processor and its Application" *Proc. of the Computer Conf.* (1979-10)
(5) G. E. Hillis: "The Connection Machine" TR-646 (1980) Cambridge, MIT AI Lab.
(6) D. Gajski, et al.: "Cedar" *Proc. Compcon*, spring (1984)
(7) J. B. Dennis: "Data Flow Supercomputers" *Computer* 13, 11 (1980)
(8) Arvind, et al.: "An Asynchronous Programming Language and Computing Machine" TR 114 a, (1978) UC Irvine
(9) K. Hiraki, T. Shimada, & K. Nishida: "A Hardware Design of the SIGMA-1, a Data Flow Computer for Scientific Computations" *ICPP* (1984)
(10) 二木・外出：「項書き換え型計算モデルとその応用」情報処理 24, 2, p. 133 (昭 58)
(11) K. J. Berkling: "A Computing Machine Based on Tree Structures" *IEEE Trans. Computers* C-20, 4, p. 404 (1971)
(12) 田中：「並列リダクションマシン」No. 10, Computrol (昭 60) コロナ社
(13) 井田：「リダクションマシン」bit 16, 9 (昭 59)