Vol. 27 No. 5 情報処理学会論文誌 May 1986

実時間性を重視した並列計算方式†

徳山五郎††* 村上隆夫††

MIMD 型の並列計算方式の高速の限界を探る意図のもとに，高速性をもっぱら追求した並列計算の一方式が述べられる．一般に並列計算の高速性は，(1)並列化能力を高めることと(2)演算間隔を短縮することによって得られる．(1)については，十分な個数の演算器を用意し，データの揃うことを発火契機とすることの有効さは知られているが，その際(2)の条件に困難さがあった．本論文では，この(2)に関して，もし与えられた問題に必要な並列数と同程度の個数の処理ユニット (PU) と同程度の入・出線数を持つスイッチマトリックスが使用できるならば，演算間隔を非常に短くする方法の存在することが示される．その方法の要点は，(i)結合ネットワーク，すなわちスイッチマトリックスの接点を出線側から制御することと，(ii)一つの PU に割り当てる演算群をある条件のもとに作られる線形順序集合として定めることにある．これにより，PU 間のデータ転送における待ちが，緊急でないデータについてのみに生ずるように，いいかえれば，データ転送の輻輳による計算実行遅延が発生させないような制御が可能である．

1. まえがき

計算機応用のある種の分野では，大量の演算を厳しい実時間条件のもとで行いたいことがある．例えばロボットの制御において，ハンドの位置・姿勢・速度・角速度・トルク等を求めるには，三角関数・マトリックスの連乗積等を含んだ大量の計算を必要とし，その要求は，より大きな自由度と一般的な動きとともに増大し，より高速の MIMD 型の並列計算機が要求される．

計算機の処理能力を向上させるための方式面からの工夫として，並列計算が考えられ，種々の方式が形成された．アレイプロセッサ[1]は，アレイ演算を並列実行できる．そして，ある条件下の命令列において，先行制御を適切に用いることにより，演算間隔を非常に短くできる．主な制約は並列化能力にある．データフロー計算機は，最大の並列化能力を持ち，演算間隔については，データフローマルチプロセッサ[2]の方式において最も短いが，そのネックは結合ネットワークにある．したがって，データフローマルチプロセッサ方式において，もし，結合ネットワーク上の輻輳による性能低下を避けることができるならば，並列能力の最も高く演算間隔の非常に短い方式が，すなわち高速性に関して一種の理想形が得られるはずである．

本論文は，これを目標として得られた一案を述べるものであって，製造技術上のいくつかの問題が解決されたものと仮想して方式を組み立てることにより，速度向上可能性の限界についての見通しの一助とすることを目的としている．

本方式は，複数の処理ユニット PU とそれらの結合ネットワークによって一種のデータフロー計算方式を構成する．PU は演算器1個と，オペランドを収容するための記憶領域とから成り，PU ごとに発火検出機構を持つ．演算器の出力データは，それを次に必要とする PU に向かって，結合ネットワークを経由して直接送られる．その際，計算の並列数と同程度の個数の PU と同位の大きさの入・出線数を持つスイッチマトリックスが使用できるものと仮定する．そこで本論文の目的は，この仮定の下で，結合ネットワーク上での待ちが計算実行遅延を発生させないような制御が可能であること，いいかえれば準備のできた計算は，結合ネットワーク上で待たされることがないような方式が可能であることを示すことにある．

以下，第2章で，処理対象および考える方式の範囲が述べられ，速度向上の二つの要因，並列化能力と演算間隔の意味が説明される．第3章で各要因についてできるだけ有効な方法を選ぶことにより，一つの方式が構成される．第4章で本方式の評価が述べられる．

2. 計算時間とその短縮

2.1 計算グラフ

処理の対象，すなわち問題は四則演算と条件分岐の組合せとする．それを解く計算機として，四則および比較の演算器を備えたものを考える．演算器相互間の結合は，完全連結ネットワーク[3]でなく，結合ネットワークを用いるものとする．

回路・素子の差は度外視し，もっぱら方式について

† A Parallel Computing Method for High Speed Calculation by GORO TOKUYAMA and TAKAO MURAKAMI (NTT Communications and Information Processing Laboratories).

†† NTT 情報通信処理研究所
* 現在 図書館情報大学

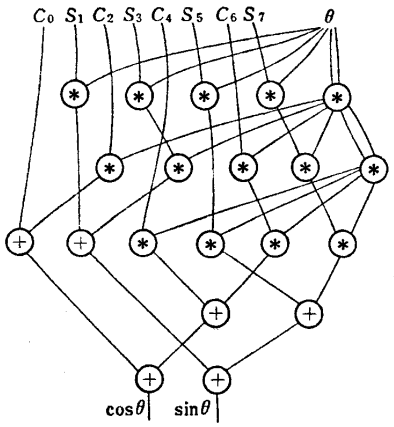

図 1 $\theta \rightarrow \sin\theta, \cos\theta$ の計算グラフ
Fig. 1 Calculation graph representation of $\theta \rightarrow \sin\theta, \cos\theta$.

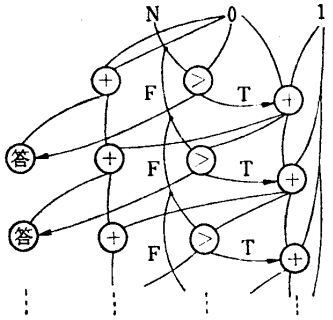

図 2 $N \rightarrow (1+\cdots+N)$ の計算グラフ
Fig. 2 Calculation graph representation of $N \rightarrow (1+\cdots+N)$.

本体系の中で考え，数値表現の方法・語長，入出力の方法等も考えない．

与えられた問題を解く手順は有向グラフで書ける．**節点**は入力データ・定数・演算および終点であり，それらを結ぶ有向の**辺**によって計算の流れが示される．これを**計算グラフ**と呼ぼう．**図1**に条件分岐を含まない例，**図2**に含む例を示す．演算節点の入力は2オペランド，1条件とし，ファンアウト数は任意とする．演算結果を前位の節点に戻してループを作ることをしない．これにより，節点間に準順序関係が定まる．

計算グラフは一般に有限でない．計算が実行されて正常終了であると，計算グラフの有限部分集合だけが実行され，その範囲は実行時に定まる．この部分グラフを**実行グラフ**と呼ぼう．そうすると**計算時間**は，実行グラフ上で入力データ節点から答の節点まで計算が流れるに要する時間と定義され，**高速**とは同じ計算グラフに対して計算時間が短いことと定義される．

2.2 演算間隔

実行グラフ上で計算時間を測る経路は一意でないが，その中のクリティカルパスを選ぶことにより，データ待ちがなくて時間の最長な経路が得られる．これを**最長経路**と呼ぼう．計算の流れが最長経路を通過する際に，一つの節点上での演算が終了してから次の節点で演算が始まるまでの時間を**演算間隔**と呼ぶことにすれば，前提により演算時間の差は考えないから，目標は，演算間隔のできるだけ短い計算機方式を求めることに帰着された．

演算間隔は次の成分からなる．：(1)データ転送動作の開始を待つ時間，(2)通路の空きを待つ時間，(3)データを転送する時間，(4)次の演算器にデータが送られてから，演算開始を待つ時間．

このうち(3)を特に**転送時間**と呼ぼう．

(1)は，制御が回ってくるのを待つ時間であって，制御の並列化によって短縮される．(2)は，転送の並列化能力を上げることにより短縮される．(3)は，転送の手続きを短くし，通路の設定に要する時間を減少することにより短縮される．(4)は，演算器待ちの時間を減らし，発火検出待ち時間を減らすことにより短縮される．

以上要約すれば，演算間隔を短くするには，演算の並列化能力を高くし，転送時間を短くし，転送動作の並列化能力を高めることが有効である．

そしてこのように考えると次のような指針が得られる．

1. 演算の並列化のためにデータフロー式の発火が，そして，制御の並列化のために個別の発火検出が有利である．

2. 転送の手続き段数を減らすために，演算をあらかじめ演算器に割り付けて，arbitration を省略することができる．演算器出力データは次の演算器に，直接，値で渡す方が，番地で渡すよりも段数が少ない．

3. Queue を発生しないことが望ましい．すなわち，演算器に対する Queue，演算器出力が転送を待つ Queue，結合ネットワーク上での転送要求の衝突，次の演算器に着信する時の Queue，発火検出を待つ Queue 等が発生しないような機構および制御方法であること．

このような目標で方式を構成するに当って，次の仮定を設ける．：

I)必要なだけの個数の演算器が使用できること．
II)必要なサイズ（入線数・出線数・fan-out）の並列スイッチを持った1段スイッチマトリックスが利用できること．

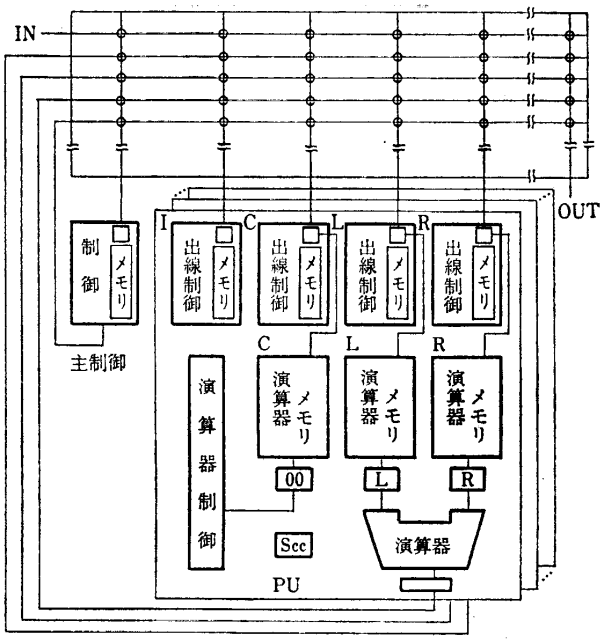

図 3 装置構成
Fig. 3 Block constitution.

3. 方式の構成

3.1 装置構成

装置は，複数の処理ユニット (PU) と，一つの交換回路と，一つの主制御ユニットとから成る．全体構成を図3に示す．

(a) 処理ユニット

処理ユニット PU は，演算部と，出線制御部とから成る．演算部には，演算器と，演算器メモリおよび制御回路がある．演算器メモリには，C, L, R の3区があって，格納のための個別のアドレスレジスタを持ち，読み出しは Scc による3区共通の番地によって行う．

出線制御部は，I, C, L, R の4区があり，それぞれ制御回路とスイッチ制御メモリを持ち，スイッチ制御メモリは4区個別に番地を持つ．

(b) 結合ネットワーク

結合ネットワークは一段のスイッチマトリックスである．入線は各 PU，入力 IN および主制御部からのデータ信号線であり，出線は，各 PU の出線制御部 I, C, L, R の各区，出力 OUT および主制御部へのデータ信号線である．

(c) 主制御部

主制御部は，計算動作の起動・進行・停止を制御するための部分で，制御のための回路と，計算のスケジュールを記憶するためのメモリから成る．

3.2 プログラミング

プログラム作成手順および計算動作の手順を，図1の計算を例に説明する．これは，条件分岐を含まない場合であって，分岐を含む場合については後述する．

(a) 計算単位への分割

図1は，C_0, S_1,… 等の定数をあらかじめ記憶し，入力 θ に対して $\cos\theta$ および $\sin\theta$ の値を出力する計算の，並列計算に適したアルゴリズムを示している．

与えられた計算グラフが複雑で大きな構造をしている場合に，計算グラフを適宜いくつかの計算単位に分割し，それぞれ名前をつけ，その実行順序を主制御部のメモリに格納する．図1の例は簡単であるから，単一の計算単位から成るものとする．名前をこの場合Tとしよう．

(b) 計算単位の鎖への分割

計算グラフ中の節点の集合は，準順序が付けられている．二つの節点 a, b の間で前後が定まらないとき，演算 a および b は並列実行の可能性があり，したがって，同じ演算器に割り付けると，並列化能力を損うおそれがある．逆に，a>b または a<b ならば，同じ演算器に割りつけても並列化能力は損われない．そこで図4のように，節点 A_0〜A_4 を同一の演算器Aに，B_0〜B_4 を演算器Bに，…, G_0, G_1 をGに割り付けることができる．こうして得られた節点集合 $\{A_0, \cdots, A_4\}$, $\{B_0, \cdots, B_4\}$, …, $\{G_0, G_1\}$ を鎖と呼ぶことにする．鎖の作り方については，それが線型順序集合をなすことのほかにさらに条件があるが，後述する．

(c) PU のメモリ内容

一つの鎖を1個の演算器に割り付けるので，鎖の名前と PU の名前とに同じ文字を使うことにする．計算単位Tは鎖 A〜G に分解された．鎖の節点1個

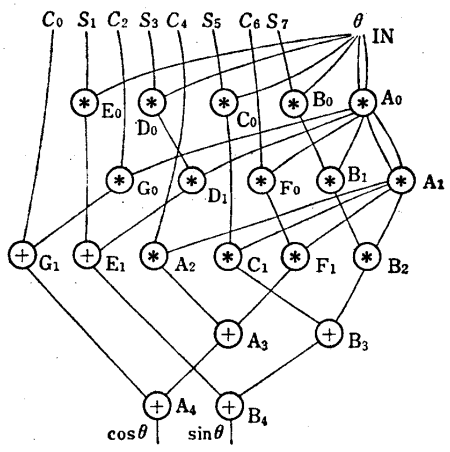

図 4 $\theta\rightarrow\sin\theta$, $\cos\theta$ の計算グラフ
Fig. 4 Source program of $\theta\rightarrow\sin\theta$, $\cos\theta$.

表 1 演算器メモリ内容
Table 1 ALU memories for Fig. 4.

	OP	次番地	Acc L	Acc R	C	L_0	L	R_0	R
A_0	*	1	0	0	1	0	(In)	0	(In)
A_1	*	2	1	1	1	1	(Acc)	1	(Acc)
A_2	*	3	0	1	1	1	C_4	1	(Acc)
A_3	+	4	1	0	1	1	(Acc)	0	(F_1)
A_4	+	Nil	0	1	1	0	(G_1)	1	(Acc)
B_0	*	1	0	0	1	1	S_7	0	(In)
B_1	*	2	1	0	1	1	(Acc)	0	(A_0)
B_2	*	3	1	0	1	1	(Acc)	0	(A_1)
B_3	+	4	0	1	1	0	(C_1)	1	(Acc)
B_4	+	Nil	0	1	1	0	(E_1)	1	(Acc)
C_0	*	1	0	0	1	1	S_5	0	(In)
≈	≈	≈	≈	≈	≈	≈	≈	≈	≈
G_0	*	1	0	0	1	1	C_2	0	(A_0)
G_1	+	Nil	0	1	1	1	C_0	1	(Acc)

は演算器メモリの命令語1語で表現される．ここに命令語とは，C，L，R 3区の各1語を連結したものである．命令語は**表1**に示すような欄構成すなわち，OP，次番地，Acc 表示，C，L_0，L，R_0，R および再実行用のタグ C′，L_0′，R_0′ を持つ（ただし，表1では再実行用のタグは省略されている）．OP は演算コードであり，L は左オペランド，R は右オペランドで L_0，R_0 はその格納表示である．その他は後述する．

表1は PU-A～PU-G の命令語の初期値を示している．ただし，かっこ付きは空白を示し，計算の進行に従ってどこからデータが送られてくるかを見やすいように示したものである．計算単位Tに対する各 PU の“先頭番地”とは，鎖の先頭ノードに対応する命令語の番地である．この場合各 PU とも0番地であり，計算単位 T に対応する先頭番地 0 を各 PU の I 区のメモリに記憶している．

（d） スイッチ制御メモリ

鎖を構成している各節点の実行順序はあらかじめ定まっているので，PU の各出線制御部は，この実行順序に対応した順序で結合ネットワークの入線からデータを受信しながらスイッチを逐次切りかえる．その順序が**表2**（C区はall “Nil”のため，省略されている．）の発信元 ID および次番地の欄に記憶され，入力データの格納場所は演算器メモリ番地の欄に記憶されている．同じ入線から異なった情報が送られてくる時でも，情報には発生番地が付けられているので（発信元 ID は演算器 ID と命令実行番地から構成されている），

表 2 スイッチ制御メモリ内容
Table 2 Switch control memories for Fig. 4.

L区

	発信元 ID	次番地	演算器メモリ番地	残りのメモリ番地
A_0	In	1	0	Nil
A_1	G_1	Nil	4	Nil
B_0	C_1	1	3	Nil
B_1	E_1	Nil	4	Nil

R区

	発信元 ID	次番地	演算器メモリ番地	残りのメモリ番地
A_0	In	1	0	Nil
A_1	F_1	Nil	3	Nil
B_0	In	1	0	Nil
B_1	A_0	2	1	Nil
B_2	A_1	Nil	2	Nil
C_0	In	1	0	Nil
≈	≈	≈	≈	≈
G_0	A_0	Nil	0	Nil

受信 PU は識別できる．その他の欄は後述する．

3.3 動　　作

前項で述べたメモリ内容によって，計算の実行される順序を次に述べる．

（a） 起動・初期設定

主制御部が起動して，最初に実行すべき計算単位の名前，この場合はTを送出する．各 PU の出線制御部の I 区は，主制御部からの信号を常に受信可能の状態にしているので，信号“T”は直ちに受信され I 部はこれを自己 PU 内の各部先頭番地に翻訳して，C，L，R の各スイッチ制御メモリおよび演算器メモリの Scc にセットする．その結果，AL (PU-A の L 区)，AR，BR，CR，DR，ER のスイッチは“In”に，各演算器 Scc は0番地にセットされる（表1，表2）．

（b） 入　　力

数値 θ が入線から表2の演算器メモリ番地に従って演算器メモリの A_0L (AL の0番地)，A_0R，B_0R，C_0R，D_0R，E_0R に読み込まれる．受信を契機にスイッチ AL は IN→G_1，AR は IN→F_1，BR は IN→A_0，CR は IN→A_1，DR は IN→A_0，ER は IN→D_1 に切りかわる．

（c） 演算の発火

データの書き込まれた演算器メモリの格納表示 L_0/R_0 は 1 となる．演算の発火条件は，“$C \wedge L_0 \wedge R_0 = 1$

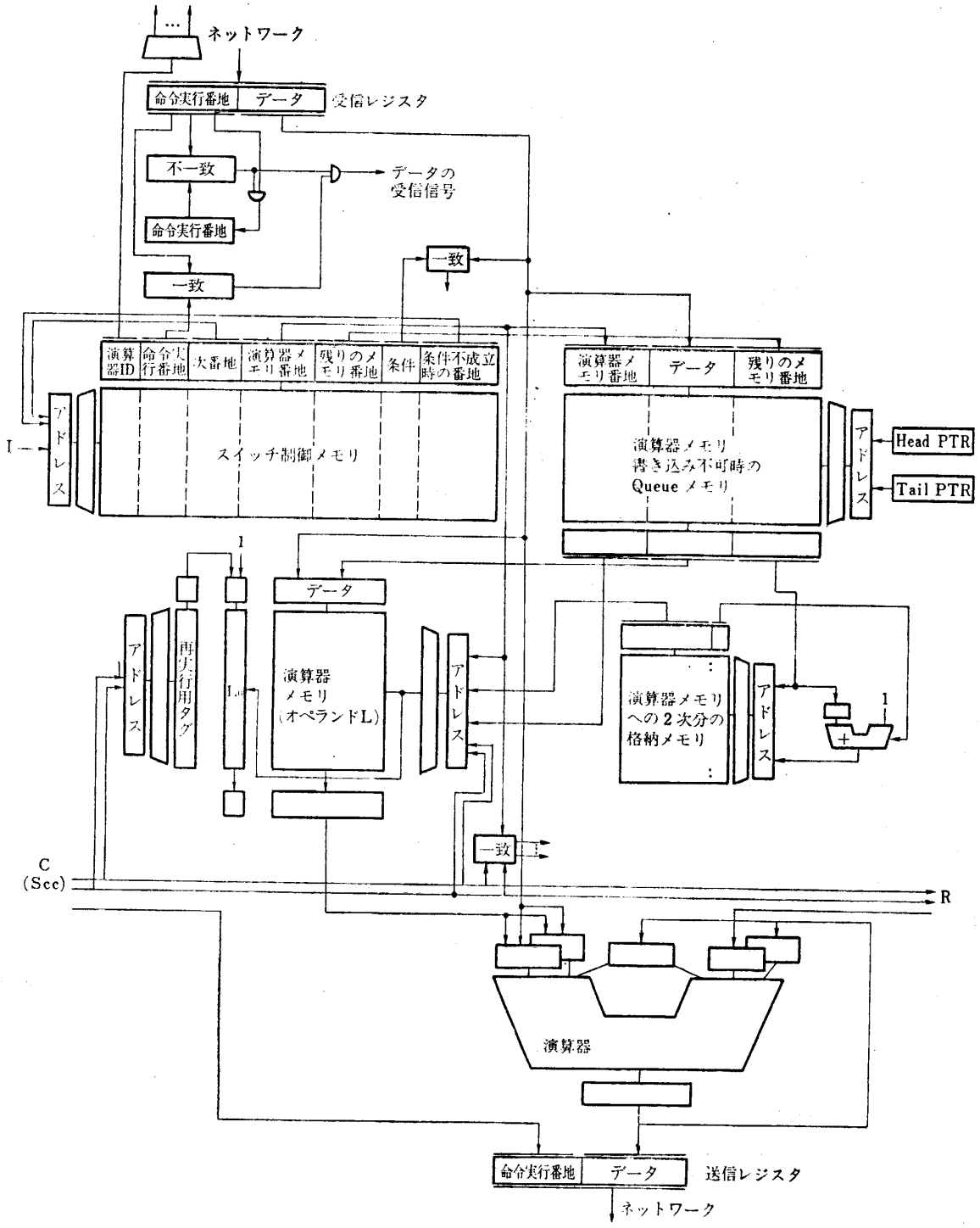

図 5 オペランドL部の出線制御部と演算部
Fig. 5 Overview of operand L part.

で，かつ，Scc が指していること" である．このため A_0, B_0, C_0, D_0, E_0 が発火し，演算が行われる．

（d） 結果の転送

演算結果のデータに，その命令の番地を付加して，転送データが作られ，送信レジスタにセットされる（**図 5**）．

転送先に対する結合ネットワークのスイッチは設定ずみであって，直ちに次の PU のC区・L区またはR区に送られる．受信側では，送られてきたメッセージの番地部分から，そのデータの格納すべき場所を知

ることができる.

一つのデータを同じ PU の複数箇所へ書き込む場合には，若番地から順に格納する．表2の演算器メモリ番地には，その最初の位置が示され，残りの演算器メモリ番地は，図5の演算器メモリへの2次分の格納メモリに格納されている．表2の残りのメモリ番地欄にはこの2次分の格納メモリ内の相対番地が格納されており，この値が Nil でない場合には，受信データと共に Queue（図5の演算器メモリ書き込み不可時の Queue メモリ）を作るが，発火は必ず若番地順であるから，実行を遅らせることはない.

受信データの演算器メモリへの書き込みの際に，格納すべき演算器メモリ番地と Scc のアドレスが一致した場合は受信データを直接演算器の入力レジスタに格納する．一致しないで，かつ，命令読み出しと衝突した場合は，図5の演算器メモリ書き込み不可時の Queue メモリに格納され，のちに，演算器メモリまたは直接演算器の入力レジスタに書き込まれる.

（e） 自 PU への転送

演算結果が，同じ演算器の入力となる場合は，結合ネットワークを経由せず，自 PU 内のパスで近道転送する．鎖の性質上，この頻度は高いので，高速化のために有効である．この場合の表示が表1のアキュームレータ表示ビット Acc L R であり，“1”の時 Acc からデータを入力することを意味している.

このようにして計算は進行する.

（f） 出力と終了

演算器 A_4 と B_4 の結果は出力線 OUT から送出される．このための制御機能は外部回路に用意されるものと想定する．また A_4 と B_4 の実行によって計算単位 T は終了する．この事象は主制御部に結合ネットワークを経て通知される．新しい計算単位が続いて起動される場合の動作は（a）と同様である.

3.4 条件分岐を含む場合

計算グラフ上の1節点を演算器メモリの1語で表す上述の方法は，条件分岐を含む場合にも同様に適用される.

条件分岐とその環境は，次の四つの要素に分割される.

1) 条件演算命令の節点

2) True 時のみ実行される鎖群（選択パスと呼ぶ.）

3) False 時のみ実行される鎖群（選択パスと呼ぶ.）

4) 条件分岐のあと共通の手続きに合流する鎖群（合流パスと呼ぶ.）

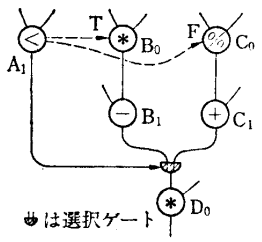

図 6 条件分岐のソースプログラム例
Fig. 6 Example of source program including a condition inatruction.

表 3 スイッチ制御メモリ内容
Table 3 Switch control memories for Fig. 6.

L区

	発信元 ID	次番地	演算器メモリ番地	残りのメモリ番地	条件	条件不成立時の次番地
D_0	A_1	1	Nil	Nil	T	2
D_1	B_1	3	0	Nil	Nil	Nil
D_2	C_1	3	0	Nil	Nil	Nil
D_3						

並列処理においては，1)と2)，3)とは並列に実行することが可能である．すなわち，選択パスの演算結果のデータが合流パスまたは外部（Output）に渡される時までに条件演算命令の結果が得られればよい．次にこの条件分岐の2形態について説明する.

（a） 合流パスが存在する場合

選択パスからのデータを制御するために選択ゲートを合流パス上に設定する．この選択ゲートはあらかじめ閉じられており，条件演算命令の演算結果の受信を契機に，演算結果の真・偽に対応した一方のゲートを開くことにより選択パスからのデータの通行を制御する．この選択ゲートは出線制御部で制御されるため，スイッチ制御メモリに条件（True, False, 強制 on の内容を持つ）の欄と条件成立時の番地（次番地の欄で代用する）および条件不成立時の次番地の欄を設けている．すなわち，**図6**に示す例では，合流パスを構成する演算器 D の L 区の出線制御部で，**表3**に示すように条件演算命令の演算結果の受信を契機に DL は $A_1 \rightarrow B_1$ または $A_1 \rightarrow C_1$ に切りかわる.

（b） 合流パスが存在しない場合

選択パスを構成する命令語のうち，特定の命令語が発火しないようにあらかじめその命令語のC欄に0をセットし，条件演算命令の結果とスイッチ制御メモリの条件欄の条件指定とが一致した時にC欄に1をセットすることによって命令の発火を制御する.

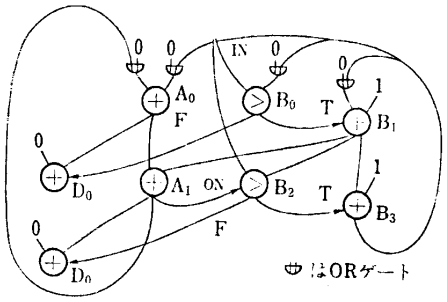

図 7 $N\rightarrow(1+\cdots+N)$ のソースプログラム
Fig. 7 Source program of $N\rightarrow(1+\cdots+N)$.

3.5 ループを構成する場合

もし，計算グラフに同形反復，すなわちループがある時は，1語で複数の節点を表すことができる．条件分岐を含む計算は無限グラフとなる場合があるが，それは通常ループによって形成されるので，同形反復を利用してプログラムを有限にすることができる．その方法を，図2の例について述べる．

1語-1節点の場合は図2のままでよいが，1語-複数節点の場合は，一つの演算節点の入力元は最初と2回目以降とは異なるので，**図7**のように OR ゲートが必要になる．この場合，二つの辺の入力順序はあらかじめ定められているので，スイッチ制御メモリの次番地で対処することが可能である．複数節点の場合は，同一命令語を複数回使用するという意味であるから，命令語の C，L_0，R_0 に対して2回目以降のためのタグ C′，L_0′，R_0′ が必要となる（ただし，表5においては省略している）．さらに，追い越しを防ぐために，ループ処理の最後の演算節点から条件演算命令に，すなわち，図7のように A_1 節点から B_2 節点への演算済みを通知する辺を加える必要がある（これを，スイッチ制御メモリの条件の欄では強制 ON で表示している）．演算器メモリ内容を**表4**に，スイッチ制御メモリの内容を**表5**に示す．

3.6 鎖の条件

鎖の設定条件として，線型順序性に加えて必要なのは，放送および条件分岐，ループに関する条件である．

放送の場合，送信レジスタの内容は，同じ演算器から次回の出力があった時に更新されるので，放送すなわち複数の転送先がある場合は，全部の転送が終るまで次の演算を行わないようにする必要がある．そこで，順序関係のある放送受信節点の集合に対しては，“放送節点の次の節点は放送受信の最も遅い（早くない）節点であること”という条件を付加する必要がある．一方，順序関係のない場合は，放送節点を1 PU に割り付けられる鎖の最後のノードとするか（放送元は受信されるまで常に送信しているか），受信先が必ず放送を受信できるように PU に割り付けられなければならない．

条件分岐の場合は，3.4 節に述べたように条件演算命令の節点と選択パスは並列に処理される必要があるので，各々別の鎖とする．一方，合流パスは，条件演算命令の結果の受信を契機に動作可能な状態になるので，条件演算命令と合流パスを同一の鎖にすれば

表 4 演算器メモリ内容
Table 4 ALU memories for Fig. 7.

	OP	次番地	Acc		C	L_0	L	R_0	R
			L	R					
A_0	+	1	1	0	1	1	0(Acc)*	1	0(B_3)*
A_1	+	0	1	0	1	1	(Acc)	0	(B_1)
B_0	>	1	0	1	1	0	(In)	1	0(Acc)*
B_1	+	2	0	0	0	1	0(B_3)*	1	1
B_2	>	3	0	1	0	0	(In)	1	(Acc)
B_3	+	0	0	0	0	0	(B_1)	0	1
D_0	+	0	0	0	0	1	0	0	(A_0, A_1)

*：（ ）内は，ループの2回目以降の入力元

表 5 スイッチ制御メモリ内容
Table 5 Switch control memories for Fig. 7.

C区

	発信元 ID	次番地	演算器メモリ番地	残りのメモリ番地	条件	条件不成立時の次番地
B_0	B_0	1	1	Nil	T	Nil
B_1	A_1	2	2	Nil	ON	Nil
B_2	B_2	0	3	Nil	T	Nil
D_0	B_0	1	0	Nil	F	Nil
D_1	B_2	0	0	Nil	F	Nil

L区

	発信元 ID	次番地	演算器メモリ番地	残りのメモリ番地	条件	条件不成立時の次番地
B_0	In	1	0	0	Nil	Nil
B_1	B_1	2	3	Nil	Nil	Nil
B_2	B_3	1	1	Nil	Nil	Nil

R区

	発信元 ID	次番地	演算器メモリ番地	残りのメモリ番地	条件	条件不成立時の次番地
A_0	B_1	1	1	Nil	Nil	Nil
A_1	B_3	0	0	Nil	Nil	Nil
D_0	A_0	1	0	Nil	Nil	Nil
D_1	A_1	0	0	Nil	Nil	Nil

よい.

ループを構成する場合は，3.5 節に述べたようにトークンの追い越しを防ぐためにループの最後の節点の実行が終らないうちに次のループ制御の条件演算命令を実行しない方式，すなわち次のインクリメントのトークンを発生しない方式をとっている．このためループ制御の条件演算命令とインクリメントの選択パスを同一の鎖とすることも可能である．

4. 評　　価

(1) 性能の検討

この方式は，スイッチについての大きな仮定があるので，性能推定の意味も，条件付きのものとなるが，可能性の見通しのために，一応の推定を試みた．

マシンサイクルをτとし，メモリへの書き込み，読み出しの時間を2τ，結合ネットワークからPUにデータを受け入れる動作が間隔3τのタイミングで行われるものと仮定し，演算器の入力レジスタを2組用意して命令語読み出し時間の短縮を考えた結果，演算間隔は1τ～5τと推定された．演算に要する時間が，たとえば乗算等において10τの桁であるとすれば，これは並列性の効果を十分に発揮できる値と考えられる．

(2) PU 数

必要となる PU 数を例題で調べてみると，図1の例題では最大並列数6に対して PU 数は7であり，図2の例題では最大並列数2に対して PU 数は3である．このように必要とする PU 数はほぼ最大並列数に等しい．

5. む　す　び

四則演算と条件分岐の組合せを，基本演算器の並列使用によって解く計算方式を，高速性のみをもっぱら追求して構成することを試み，並列数程度の個数の PU と，それと同数の入線数，4倍数の出線数を持つ1段スイッチマトリックスを仮定して次の結論を得た．

(1) 計算グラフ中の節点の線型順序集合(鎖)を単位として演算器に割り付けること

(2) スイッチマトリックスの接点を出線から入線を選択する形で制御すること

により，クリティカルパス上にある演算の実行が，データ転送要求の輻輳のために遅延することがないように制御することが可能になる．

謝辞 東海大学の中川教授にはロボット制御の計算方法を教示いただいた．NTT 基礎研究所雨宮情一研究室長にはデータフロー計算機の諸方式について教示いただいた．同社情報通信処理研究所の伊吹特別研究室長はこの研究の基本方向を示され指導・推進を与えられた．これらの方々に深くお礼申し上げる．

参　考　文　献

1) 山田　博：コンピュータ・アーキテクチャ，p. 316，産業図書株式会社，東京 (1976).
2) Dennis, J. B.: Data Flow Supercomputers, *Computer*, *IEEE*, Vol. 13, No. 11, pp. 48-56 (1980).
3) 四本善一郎：並列処理システムの性能を左右する相互結合ネットワーク，*Nikkei Electronics*, 1981年12月12日号，pp. 88-109 (1981).

(昭和60年4月2日受付)
(昭和61年2月20日採録)

徳山　五郎（正会員）

昭和7年生．昭和29年東京大学理学部数学科卒業．同年電電公社入社．昭和60年図書館情報大学教授．工学博士．情報処理機能の計量的評価法に関心を持つ．電子通信学会，日本オペレーションズ学会各会員．

村上　隆夫（正会員）

昭和19年生．昭和42年金沢大学工学部電子工学科卒業．同年日本電信電話公社入社．現在，NTT 情報通信処理研究所において，情報システムの基礎に関する研究に従事．