並列処理および並列処理計算機について*

加藤　満左夫**

はしがき

計算機が進歩すればするほど，計算機をもっと進歩させたい要求が常に生じてきた．その1例は数値計算による気象予測であって，全世界を 2° 間隔の格子点でおおって，実時間の100倍の速度で計算を行なうには，10^9 命令/秒の演算速度が必要であるといわれる．さらに，大気と海洋のエネルギーの授受を考慮すると，計算量は途方もない値になることは容易に想像できよう．

しかしながら，最近の論理素子の速度が向上したとはいえ 10^9 命令/秒，すなわち 1 ns で1命令を実行することは到底不可能である．そこで，プロセサの数を多くして並列に処理を行なうことによって，演算能力を高めようとする考え方が並列処理 (parallel processing) である．なお多重処理 (Multiprocessing) も同じく複数のプロセサを用いる処理の方式であるが，これはオンライン処理のように，1台のプロセサが故障しても残るプロセサで処理能力は下がるが，システムの運転を継続させ (Fail Soft) 正常な場合にはプロセサの負荷を能率よく分担させることを目的としている．したがって，並列処理で考えるプロセサの数は，10とか100のオーダであるのに対し，多重処理では実用上2ないし4程度が用いられる．

本文では，並列処理の研究の足どりを中心に，その二，三の具体例を紹介したい．

1. 研究の歴史

(1) 発　端

並列処理によって大規模な計算を実行しようという着想は，リチャードソンにさかのぼることができる．リチャードソンは1920年に数値積分による気象の予測を初めて試みた人であるが，彼の着想というのは，膨大な数値積分を実行するのに 64,000 人の人を円形劇場に座らせ，各人に各格子点の計算を割り当て，隣り同士でデータを交換しながら，1人の指揮者のもとに数値積分を演じようというものであった．

(2) ソロモン計算機[1),2)]

リチャードソンの着想を電子計算機によって初めて実現したのがソロモン計算機である．ソロモン計算機は1024個の演算装置を32×32の格子状に排列し，これをネットワークコントロールユニットが制御する．(**図1**) 各演算装置に格子点のデータを**図2**のように与えて，数値積分を高速に実行しようとするものである．

(3) Unger[3)] の計算機

Unger の計算機は2次元に配列した論理モジュールに図形を記憶させ，その拡大・縮小・移動などの処理を並列に行なわせ，パターン認識の処理を能率よく行なうことを目的としている．論理モジュールは1ビットのアキュムレータと3ビットのメモリを有し，隣り合うモジュールとの演算ができる．

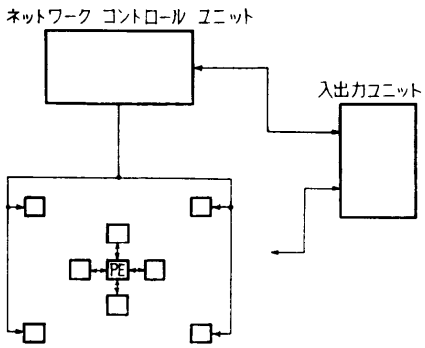

図 1　ソロモン計算機のブロック図

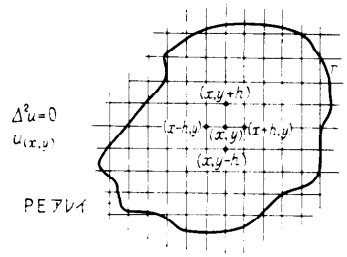

図 2　並列計算機によるラプラス積分

* Parallel processing and Parallel processing Computeres, by Masao KATO (Electrical Communication Laboratory)

** 日本電信電話公社研究開発本部

このように簡単な構造をもつ論理モジュールを空間的に配列して論理処理を行なう考え方には Lee[4], Gonzalez[5], Crane[6),7)], Fuller[8)] らの提案があり，また Illiac Ⅲ[9)] のバブル論理もその一例である．このような考え方は記憶と論理を一体化した分散論理型の計算機に発展しよう．

（4） Holland[10)] の計算機

Holland の提案する計算機は独立に命令を実行できる論理モジュールを次元に配列し，隣り合うモジュールを任意に結合し，命令やデータを交換して処理を行なう．すなわち Holland の計算機はソロモンや Unger の計算機のようにセントラライズされた制御装置をもたないところに特徴がある．IBM の Comfort[11)] は Holland の提案を若干具体化した A MODIFIED HOLLAND MACHINE を提案している．

また Univac の Mellen[12)] は 2^N 個の演算装置を N次元立方体の頂点に配置したシステムを提案している．すなわち，各演算装置はN個の他の演算装置と隣接し，かつ接続されている．$N=4$ の場合は**図 3** に示すごとくである．Mellen はこれを MDMP (Multi-demensional Multiprocessor) と称し，画像の静止した端，あるいは移動しつつある端を検出する処理の応用を提案している．

（5） Senzig のベクトル演算プロセサ[13)]

Seuzig は演算装置の空間的配列にはこだわらず，ベクトルの演算を能率よく行なうことを目的としたベクトル演算プロセサを提案している．

演算装置は n 個の演算器とこれを共通に制御する制御装置とから成り各演算器は共通の命令をおのおののデータに対して実行する．n 個の演算器に n 個のオペランドを供給するためのメモリ制御方式は**図 4** に示す

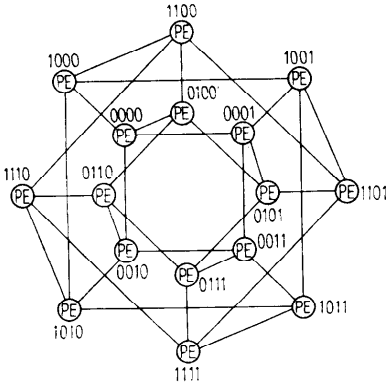

図 3　Mellen の MOMP ($n=4$)

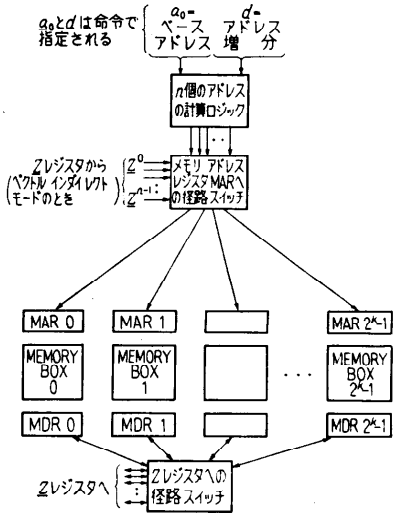

図 4　ベクトル演算装置のメモリ制御方式

ごとくである．メモリ全体は 2^k 個の装置から成り，全記憶容量は 2^m 語である．mビットのアドレスの中で下にビットが各メモリ装置を指定し，残りの $m-k$ ビットは各メモリ装置の中の語のアドレスを指定する．

Senzig の提案したシステムはソロモン計算機に比して記憶装置が一元化されていること，通常の計算機と同様の演算器を使用するために汎用計算機システムとの調和性がよいことが特徴である．

Senzig はこのようなベクトル演算装置の演算能力を高めるために，パイプライン式の高速演算器を使用することを提案している．

3. 並列計算機における命令とデータの制御方式

一般に並列計算機における命令とデータの制御方式は Flynn[14)] によれば，下記のごとく分類される．

単一命令ストリーム　単一データストリーム SISD
単一命令ストリーム　複数データストリーム SIMD
複数命令ストリーム　単一データストリーム MISD
複数命令ストリーム　複数データストリーム MIMD

SISD は通常の商用計算機の命令実行の制御方式である．SIMD 単一の命令ストリームが多数のデータストリームを制御するもので，ソロモン計算機，ILLIAC Ⅳ 計算機，Unger らの分散論理型の計算機がこれに属する．

ソロモン計算機は個々の演算装置のオペランドアドレスがすべて共通であるのに対して，ILLIAC Ⅳ 計

算機では演算装置がインデックスレジスタを有し，オペランドアドレスは独立でよいので，データ構造の融通性が大きい．この機能は後述するようにマトリクスを行単位に，あるいは列単位にアクセスするのに利用される．

MISD は単一のデータストリームに対して複数の命令ストリームが作用するもので，**図 5** のようにデータがベルトコンベアに乗っていくつかの命令実行ステーションを通り抜ける形を想定できる．MISD の例はあまりないが，Senzig のベクトル演算プロセサはその 1 例であり，後述する CDC 社などのパイプライン演算装置はその発展形と考えられる．

MIMD に個々の演算装置が独立した命令，あるいはプログラムとデータを有するもので，Holland の計算機，Mellen の提案するシステムなどがこの方式に属する．

4. 並列計算機の具体例

4.1 ILLIAC Ⅳ 計算機[15]

ILLIAC Ⅳ 計算機は**図 6** に示すように，演算装置(PE)および記憶装置(PEM)の組 64 に対して 1 台の制御装置をおき，これを 1 Quadrant と称し 4 Quadrant で 1 つのシステムを構成する．

すなわちシステムは 4 台の制御装置と 256 台の演算装置ならびに記憶装置より成る．各演算装置は 64 ビットの演算レジスタ 2(A, B) 一時記憶レジスタ (S), ならびにルーティングレジスタ (R) を有し，64ビットの浮動，固定小数点ならびに論理演算を高速に実行する約 1 万ゲートの論理装置である．各記憶装置は容量 2048 語 (64 ビット/語)，サイクルタイム 250 ns である．記憶装置は，最初薄膜記憶装置を使用する予定で設計をすすめたが，途中で半導体記憶装置が有望になり，電力，速度，価格，大きさの点で前者をしのぐものとなったので，これにおきかえている．

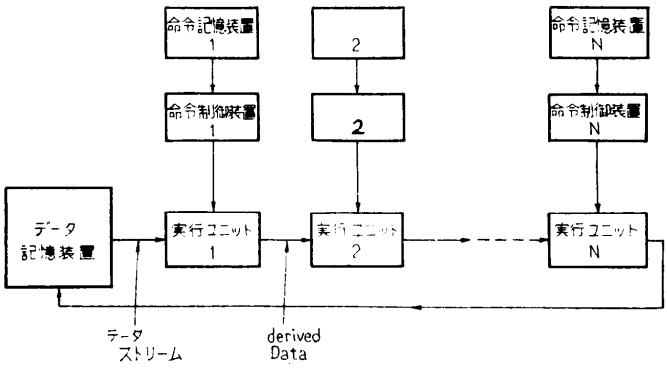

図 5　MISD 方式のプロセサの構成例

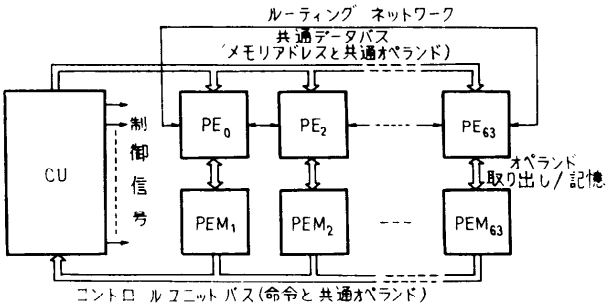

図 6　ILLIAC Ⅳ 計算機の制御の構成 (1 Quadrant)

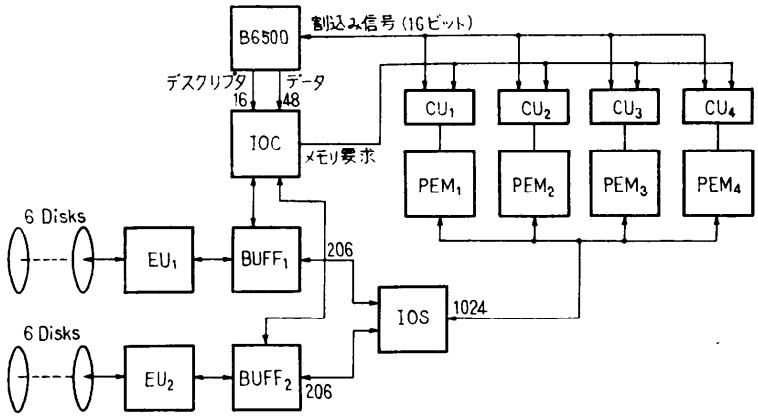

図 7　ILLIAC Ⅳ 計算機のシステム構成

制御装置は2つの 64 語のバッファ（命令と共通データ用），4個のセントラルインデックスレジスタ，ならびにファイナルキューと称する命令のバッファ（容量8）を主体として構成されている．

図7は ILLIAC Ⅳ 計算機全体のシステムを示したものである．ILLIAC Ⅳ 計算機は外部記憶装置として容量 10^9 ビット，転送速度 300 Mb/s の磁気ディスク装置を有し，その制御はシステム制御用計算機 B 6500 が行なう．B6500 はこの他各 Quadrant の結合状態，演算の異常状態，プログラムのローディングと実行，カード，ラインプリンタなどの通常の入出力装置の制御を行なう．このため B6500 の中に ILLIAC Ⅳ 計算機制御用のモニタをおく．

ILLIAC Ⅳ 計算機の i 番目の演算装置は $i\pm1$，$i\pm8$ 番目の演算装置と modulo 64 で相互接続されている．すなわち，接続は end around で端がない構造である．端がないのですべての演算装置を全く同一に作ることができる．ルーティングを行なうときには上記の4つの種類の距離を組み合わせて任意の距離の演算装置にデータを送る．

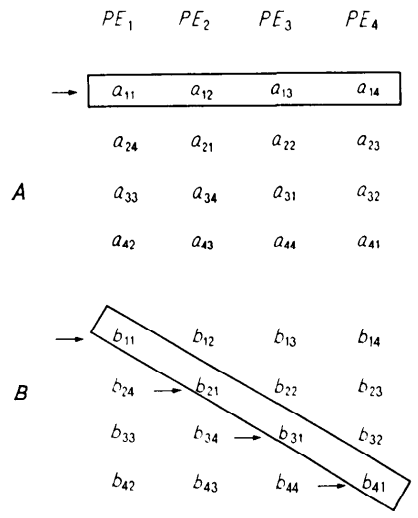

図 8　Skewed Storage

LOAD	b_{11}	b_{12}	b_{13}	b_{14}
BROAD CAST	a_{11}	a_{11}	a_{11}	a_{11}
MULTI-PLY	$a_{11}\ b_{11}$	$a_{11}\ b_{12}$	$a_{11}\ b_{13}$	$a_{11}\ b_{14}$
ROUTE	$a_{11}\ b_{14}$	$a_{11}\ b_{11}$	$a_{11}\ b_{12}$	$a_{11}\ b_{13}$
STORE	〃	〃	〃	〃
LOAD	b_{24}	b_{21}	b_{22}	b_{23}
BROAD CAST	a_{12}	a_{12}	a_{12}	a_{12}
MULTI-PLY	$a_{12}\ b_{24}$	$a_{12}\ b_{21}$	$a_{12}\ b_{22}$	$a_{12}\ b_{23}$
ADD	$\Sigma a_{1K}\ b_{K4}$	$\Sigma a_{1K}\ b_{K1}$	$\Sigma a_{1K}\ b_{K2}$	$\Sigma a_{1K}\ b_{K3}$

図 9　マトリクスの乗算

ILLIAC Ⅳ 計算機において，マトリクスを記憶させるのは**図8**の如くにする．これは行や列のどちらでも各 PE にとり出して並列に演算できるようにするためである．

マトリクスをこのように記憶させた場合，マトリクスの乗算は**図9**のように行なう．

一般に n 次元正方マトリクスの積は n^3 回の計算操作（1操作は乗算と加算）を必要とするが，上記の方法によって n 個の演算装置を有する並列計算機では n^2 回の操作で演算が終了する．

このようなベクトルにあるいはマトリクスの演算を基本として，ILLIAC Ⅳ 計算機による各種の並列処理のアルゴリズムならびにプログラミングは，a）固有値，b）リニアプログラミング，整数プログラミング，c）熱力学，d）Alternating direction implicit 法，e）テーブルルックアップ，f）大気大循環のモデル，g）時系列信号の処理，e）フーリエ解析などの分野を対象に研究がすすめられている[16]．

4.2 CDC の STAR 計算機

CDC の STAR (String Array) 計算機はパイプライン演算方式による超高速計算機である．

パイプライン演算装置は**図10**に示すように演算操作をいくつかの基本ステップに分解し，基本ステップごとに論理をおく．そして2つのオペランドベクトル $\vec{A}\vec{B}$ を対応する要素ごとに組にしてパイプラインに流し込む．オペランドの組は τ 時間ごとにロジックを

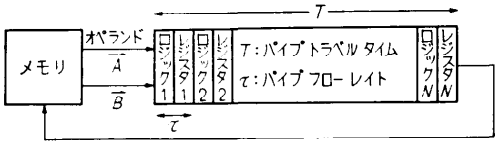

図 10　パイプライン演算装置

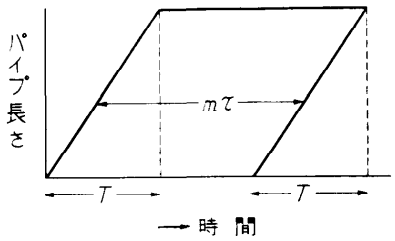

図 11　m 次元ベクトルの処理時間

1段ずつ進み，N 段のパイプラインであれば，$T=N\tau$ 時間後に最初の出力が現われ，以下 τ 時間の間隔で結果が出力される．T はパイプトラベルタイム，τ はパイプフローレートと呼ぶ．m次元のベクトルを処理する場合の時間は**図 11** に示すように $m\tau+T$ 時間である．

パイプライン演算装置には超高速度で出力される演算結果を受入れる記憶装置が必要である．このため最初 32 重，あるいは 23 重という高度の多重インタレースをかけた記憶装置の使用が考えられた．しかしながら，このような多重のインタレースをかけた記憶装置を使用すると取り出すオペランドの番地の不連続点でむだな時間を待ち合わせる．

最近の半導体メモリの発達はパイプライン方式のこのような欠点をとり除いた．CDC の STAR 計算機でも ILLIAC Ⅳ 計算機と同様，記憶装置を半導体におきかえている．サイクルタイム 40 ns，容量 100 億バイトをもち，近く完成して Lawrence Radiation Laboratory に納入されると伝えられる．**図 12** は STAR 計算機の演算器の概念図である．

4.3 IBM のアレイプロセサ

IBM のアレイプロセサは，ILLIAC Ⅳ 計算機や CDC-STAR のような超大型計算機ではなくもっと現実的に，ベクトルやマトリクスの演算を高速化するシステム 360 の付加機構[18]として現われた．現在 IBM 2938 アレイプロセサとして商品化されている．

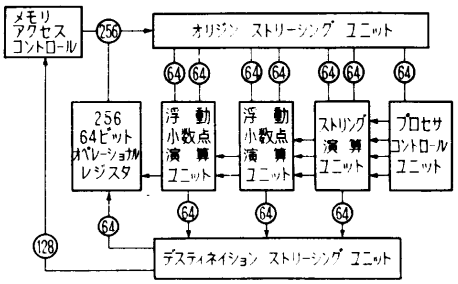

図 12　CDC-STAR の演算器の構成

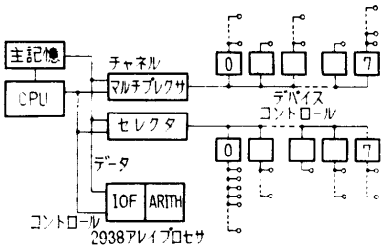

図 13　IBM のアレイプロセサ

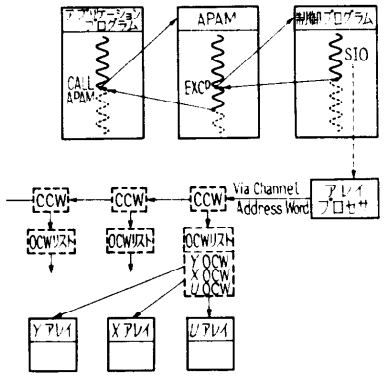

図 14　アレイプロセサの制御方式

IBM 2938 アレイプロセサはシステム 360 モデル 44, 65, 67, 75 の CPU に**図 13** に示す形で付加され，アレイプロセサの制御は**図 14** に示すようにチャネルコマンド語 (CCW)，オペランドコントロール語 (OCW) によって行なわれる．アプリケーションプログラムはアレイプロセサアクセスメソッド (APAM) を介

表 1　IBM のアレイプロセサの演算操作

演　算　名　称	略　号	演　算　操　作		
Vector move/convert	VMC	$Y_i=X_i$	$i=1\sim N$	$N=\min$ (CTY, CTX)
Vector-float-to-fixed pt	VFX	$Y_i=X_i$	$i=1\sim N$	$N=\min$ (CTY, CTX)
Scaler multiply	SMY	$Y_i'=Y_i+X_iU$	$i=1\sim N$	$N=\min$ (CTY, CTX)
Vector element-by-element sum	VES	$Y_i=U_i+X_i$	$i=1\sim N$	$N=\min$ (CTY, CTX)
Vector element-by-element multiply	VEM	$Y_i'=Y_i+U_iX_i$	$i=1\sim N$	$N=\min$ (CTY, CTX, CTU)
Sum of vector elements	SVE	$Y=Y+\sum_{i=1}^{N}X_i$		$N=$CTX
Sum of squares	SSQ	$Y'=Y+\sum_{i=1}^{N}X_i\|X_i\|$		$N=$CTX
Vector Inner product	VIP	$Y'=Y+\sum_{i=1}^{N}(U_iX_i)$		$N=\min$ (CTX, CTU)
Signed squared array	SSA	$Y_i'=Y_i+X_i\|X_i\|$	$i=1\sim N$	$N=\min$ (CTY, CTX)
Partial matrix multiply	PMM	$Y_i'=Y_i+\sum_{i=1}^{N}(X_{jU(i-1)n+j})$	$i=1\sim N$	$n=$CTY, N=CTX
Convolving multiply	CVM	$Y_i'=Y_i+\sum_{j=1}^{N}U_jX_{i+j-1}$	$i=1\sim m$	$m=$CTY, $N=$CTU
Convolving addition	CVA	$Y_i'=Y_i+\sum_{i=1}^{N}\|X_{i+j-1}+U_j\|$	$i=1\sim m$	$m=$CTY, $N=\min$ (CTX, CTU)

CTX, CTY, CTU は アレイ X, Y, U のエレメント数

してアレイプロセサを使用する．アレイプロセサは高速の演算器を有し32ビットのオペランドの積和を 200 ns ごとにつくる．

CCW, OCW は各アレイの初期アドレス，アドレスの増分（あるいは減分），アレイの要素の数を指定し，これによって**表1**に示すごとき 12 種の演算[18]を行なう．演算速度は IBM システム 360-65 を使用したマトリクス乗算の場合，アレイプロセサをつけると 40～50 倍に向上するとのことである．

ILLIAC Ⅳ 計算機のような並列計算機と，CDC-STAR のようなパイプライン計算機を比較すると，まず並列計算機が有利な点は次の如くである．

（a） 並列数Nを大きくとりうる．

パイプライン方式では演算器の並列処理段数Nは，たかだか 10 ないし 20 程度で，それ以上分割することはむずかしい．格子配列方式ではハードウェアの実現の限界内で，理論的には並列数はいくらでも大きくとりうる．

（b） 設計製造コストが安い．

格子配列方式では同一プロセサを数多く作るので設計が容易であり，また製造コストが安くつく．パイプライン方式では，超高速の論理素子を多量に必要とするが，超高速の論理素子はまだ市場が小さく，したがって，ハードウェアが高価である．

一方，パイプライン方式の有利とするところは

（a） 判断が高速である．

格子配列型計算機は空間的に大きい拡がりをもっているので，全体を見渡した分岐の判断に時間を要する．

（b） 記憶装置の使い方に融通性がある．

格子配列型計算機では演算装置ごとに記憶装置が分けられているので，データ構造の各記憶装置へのマッピングに工夫を必要とし，また記憶容量の使い方にむだが生ずる．

5. 数式の並列処理

アレイプロセサによって並判処理のハードウェアが具体化する一方，計算手順の並列化の研究も盛んになりつつある．これはたとえば

$$A+B+C+D*E*F+G+H$$

なる式が与えられ，計算を並列に行なうプロセサが必要なだけ利用できるとした場合に何回の操作で結果を得られるかという問題である．

これには Hellerman[19], Squire[20], Stone[21], Baer-Bovet[22] らの方法が考えられている．

LSCOP	ITEM	SCOP	IS	STACK(1.IS)	STACK(2.IS)	中間結果	出力ストリング
+	A	+	1	A	+		φ
+	B	+	0	φ	φ	$T_1=A+B$	T_1+
+	C	+	1	C	+		T_1+
+	D	*	2	D	+		T_1+
				C	+	$T_2=D*E$	T_1+T_2*
*	E	*	1	C	+		T_1+T_2*E+
*	F	+	1	C	+		
+	G	+	0	φ	φ	$T_3=C+G$	$T_1+T_2*F+T_3+$
+	H	;	0	φ	φ		$T_1+T_2*F+T_3+H$

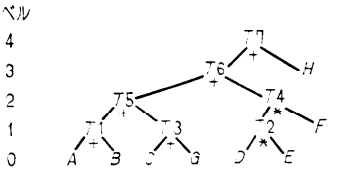

図 15 Hellerman の方法による並列処理

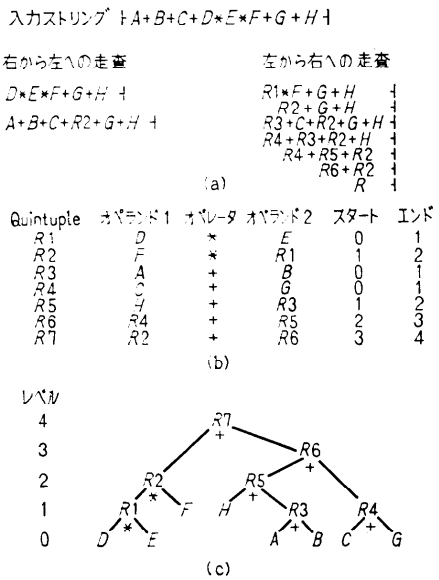

図 16 Squire の方法による並列処理

Hellerman は式を左から右へ走査して逆ポーリッシュ記法に変換する．2つのオペランドを結合するオペレータを中間結果におきかえる．中間結果におきかえたオペレータは同時に演算することができる．中間結果を含む式の走査を繰り返して並列処理の木が完成する（**図 15**）．

Squire の方法は式を左端と右端の両方から走査する．すなわち，式を右から左に走査し，オペレータのプライオリティが低くなるところで止め，$D*E*F+G+H$ ⊣ を作る．次いで，このサブストリングを

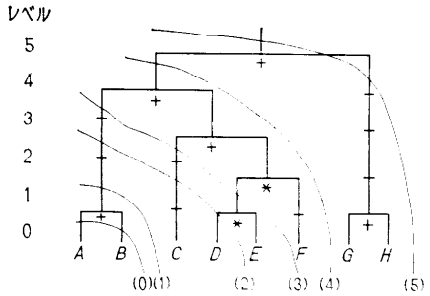

図 17 Stone の方法による並列処理

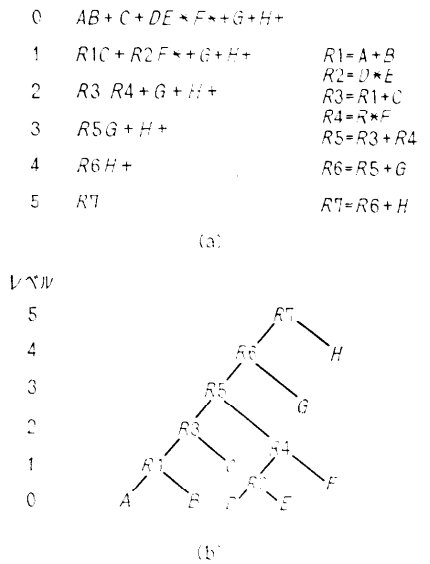

図 18 Baer-Bovet の方法による並列処理

左から右へ走査してオペレータのプライオリティが異なるところで止め，$D*E*F$ を得る．このような走査を繰り返し，**図 16** に示すように中間結果を作ってゆく．Hellerman の方法では 5 レベルであるが，Squire の方法では 4 レベルで演算が終了する．

Stone の方法は 1 パスで木を生成しようとするもので，まず，A と B を結合したサブトリーを作りこれをレベル 1 とする．次に C と D をみるがこれらは結合できないので，レベルを 1 つ上げ，レベル 1 のサブトリー $D*E$ を作る．そして**図 17** に示したように木を生成する．

Baer-Bovet の方法は式の走査は一方向にして，レベルの少ない木を生成することをねらったもので，1 回の走査でできるだけ数多くの中間結果を作ることを目的としている（図 **18**）．

Ramamoorthy[23] はプログラムの流れを状態図に書き表わし，その遷移マトリクスの変換によって並列処理の手順を求めることを提案している．同じような考え方は CDC の Thorlin[24] が CDC 6600 の並列処理演算装置を有効に利用するプログラムのコンパイル方法に関連して提案している．

Kuck[25] は $a_0+a_1X+a_2X^2+\ldots a_nX^n$ なる多項式を計算するのに**図 19** に示す接続関係を有する並列処理システムを考えている．同図は 16 個の PE（演算装置）を有するシステムの場合の接続構成であり，○で囲った数字は接続関係を有する PE の番号である．

最初全 PE に X が与えられている．まず X^2 を作り，これを 2 箇所の PE に送って X^4 と X^3 を同時に作る．次のステップで X^8, X^7, X^6, X^5 が得られる．これを繰り返して $\log_2 N$ ステップで X^N のすべてのべきが得られる．

次に X の各べきとその係数を並列に計算し，その和は 2 つずつ同時に加え合わせると $\log_2 N$ ステップで終了する．すなわち，$2\log_2 N+1$ ステップで N 次の多項式の値が計算できる．

Kuck はさらに数多くの X について，その多項式の値を計算する場合には，各演算装置に独立に動作する乗算器と加算器を備えれば，2 つの X の値の計算

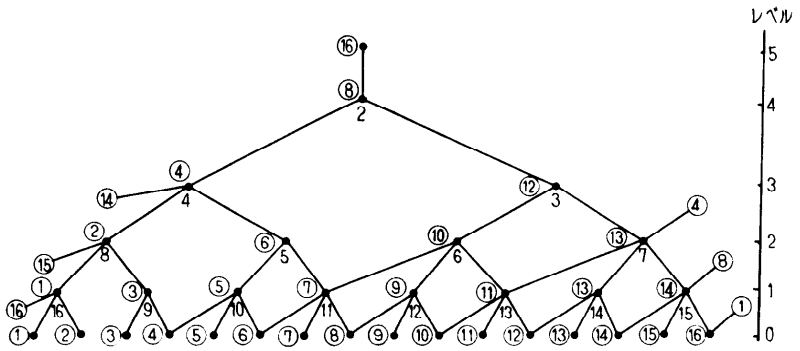

図 19 多項式を計算する演算装置の接続関係

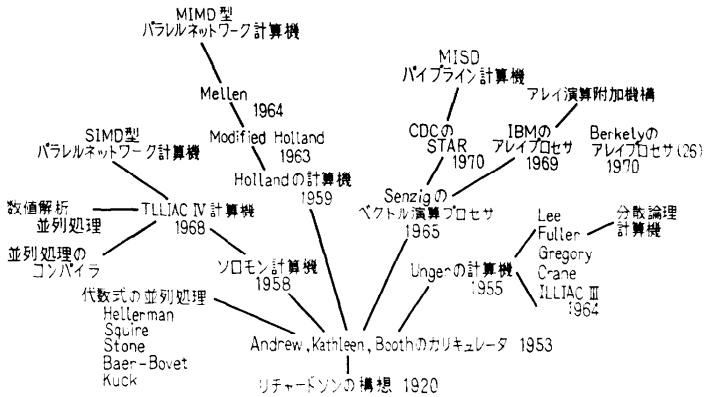

図 20 並列処理ならびに並列処理計算機の変遷

を同時に進行させうることを提案している.

ま と め

以上述べた並列処理ならびに並列処理計算機の変遷を図示すると**図 20** の如く考えられる. なお, 同一の論理モジュールを接続して, 任意の論理関数を合成する cellular logic の問題は, 本文の範囲に入れなかった. また, 計算機の処理方式としてユニークなものは並列処理以外にも多々あり, その方面については参考として文献 27, 28, 29, 30 をあげておく.

電子計算機の進歩は, 一方においてオンライン化に代表される使いやすさを指向している. 他方においては, 並列処理に代表される高度の機能を指向し, 公害の解決や環境の制御を初めてとして社会, 科学の発展に役立つものと考えられる.

参 考 文 献

1) Gregory J. & McReynolds R.: The SOLOMON Computer, IEEE Trans on EC 12, 6, pp. 774-781 (Dec. 1963)
2) Slotnick, D. L., Borct, W. C., McReynolds R. C.: The Solomon Computer, Proc. FJCC 22, pp. 97-107 (Dec. 1962)
3) Unger, S. H.: A Computer Oriented Toward Spatial Problems, Proc. WJCC 13, pp. 234-239 (May 1958)
4) Lee, C. Y.: Intercommunicating Cells, Basis for a Distributed Logic Computer, Proc. FJCC 22, pp. 130-136 (Dec. 1963)
5) Gonzalez: A Multilayer Iterative Circuit Computer, IEEE Trans on EC, 12, 6, pp. 781-790 (Dec. 1963)
6) Crane B. A. & Githens, J. A.: Bulk Processing in Distributed Logic Memory, IEEE Trans on EC., 14, 4, pp. 186-196 (April 1965)
7) Crane, B. A. & Laane R. R.: A Cryoelectronic Distributed Logic Memory Proc. SJCC 30, pp. 517-524 (April 1967)
8) Fuller, R. H.: Achieving Large Computing Capabilities through Associative Parallel Processing, Proc. SJCC 30, pp. 471-476 (April 1967)
9) McCormick B. H.: The Illinois Pattern Recognition Computer ILLIAC Ⅲ, IEEE Trans. on EC 12, 6, pp. 791-813 (Dec. 1963)
10) Holland J.: A Universal Computer Capable of Executing an Arbitrary Number of Subprograms Simultaneously, Proc. EJCC 16, pp. 108-113 (Dec. 1959)
11) Comfort, W. T.: A Modified Holland Machine, Proc. WJCC, 24, pp. 481-488 (Nov. 1963)
12) Lewis D. R. & Mellen G. E.: Stretching LARC's Capability by 100—A New Multiprocessor System, 1964 Sympon Microelectronic and Large Systems, Washington D. C.
13) Senzig, D. N. & Smith R. V: Computer Organization for Array Processing, Proc. FJCC, 27, pt 1, pp. 117-128 (1965)
14) Flynn, M. J.: Very High-Speed Computing Systems, Proc. IEEE pp. 1901-1909 (Dec. 1966)
15) Barnes, G. H., Brown, R. M., Kato, M., Kuck, D. J., Slotnick, D. L. & Stokes, R. A.: The Illiac Ⅳ Computer, IEEE Trans on C, 17, 8, pp. 746-757 (Aug. 1968)
16) Kuck D. J.: ILLIAC Ⅳ Soffware and Application Programming, IEEE C, 17, 8, pp. 758-770 (Aug. 1968)
17) Graham W. R.: The Parallel and the Pipeline Computers, Datamation, p. 68 (April 1970)
18) Ruggiero J. F. & Coryell, D. A.: An Auxili-

ary Processing System for Array Calculations, IBM Sys J. p. 118, No. 2 (1969)
19) Hellerman, H.: Parallel Processing of Algebraic Expression IEEE Trans on EC 15, 1, pp. 82-91 (Feb. 1966)
20) Squire, J. S.: A Translation Algorithm for a Multiprocessor Computer, Proc 18th ACM Natl Conf. (1963)
21) Stone, H. S.: One-pass compilation of arithmetic expressions for a parallel processor, Comm. ACM Vol. 10, No. 4, pp. 220-223 (April 1967)
22) Baer, J. L. & Bovet, D. P.: Compilation of arithmetic expressions for parallel computation Proc. IFIPS, 68, B4-B10.
23) Ramamoorthy, C. V. & Gonzalez, M. J.: A Survey of techniques for recognizing parallel processable streams in computer programs, FJCC 1969, pp. 1-14.
24) Thorlin, J. F.: Code generation for PIE (parallel instruction execution) computers, SJCC, pp. 641-643 (April 1967)
25) Kuck D. J. & Muraoka Y.: A Machine Organization for Arithmetic Expression Evaluation and an Algorithm for Tree Height Reduction, Dept of Computer Science University of Illinois.
26) Dere, W. Y. & Sakrison, D. J.: Berkeley Array Processor, IEEE Trans. on C. p. 444-447 (May 1970)
27) Alt, F. L. & Rubinoff, M.: Advances in Computers, VOLUME 7, ACADEMIC PRESS (1966)
28) Bell, C. G., & Newell A.: Computer Structures, Readings and Examples, McGraw-Hill Book Company (1970)
29) 棟上, 山口: 図形処理および連想処理のための高度並列演算装置, 情報処理, Vol. 11, No. 2, 1970年2月, pp. 70-76.
30) Booth, A. D. & Booth, K. H. V.: Automatic Digital Calculators, Butterworths, Washington (1953)

(昭和46年2月3日受付)