総　論

並列処理マシン

1. 総　論†

田　中　英　彦††

1. まえがき

並列処理は古くからある問題である．機械式の計算機が研究された 19 世紀にはすでにそれが扱われていたし，電子式の計算機が登場してから間もない 1950 年代の始めにもビット処理から語処理へという形での並列処理が導入されている．現在，並列処理と言えばいくつかの演算が複数の ALU を用いて同時に実行されるとか，複数のタスクが複数のプロセッサを用いて実行されるとか，より高級なレベルを指すのが普通であるが，これらは広義の並列処理という観点から見れば同じである．処理の高速化を目指して，その時代時代の素子技術や要求に合わせてさまざまな並列処理が検討され導入されてきた．科学技術計算における高速化の要求はとどまるところを知らず，高性能な計算機の出現が新しい応用分野を開拓する形でますますその需要は広がるばかりである．高速化の手段は素子の高速化と並列処理であり，前者に限界がある以上，並列処理に期待がかけられる．特に VLSI の発達は同種の回路を数多く並べることを著しく容易とし 10^6 オーダの並列性が現実のものとして議論され始めている．

一方，第5世代計算機に代表されるように，処理のモデル自体を変革しようという動きは並列処理に新たなインパクトを与えつつある．知識情報処理がコンピュータアーキテクチャに要求する一つがその高速性であることはもちろんだが，さらに踏み込んで，並列処理を基本とし逐次処理をその特殊化として実現するという形態がそれである．すなわち，並列プログラミングの目的の一つに，現在の言語に見られる“不要な逐次性”をできる限り排除しようということがあるが，それをアーキテクチャから支援するのが並列処理マシンというわけである．

† Introduction: Parallel Processing Machine by Hidehiko TANAKA (Department of Electrical Engineering, The University of Tokyo).

†† 東京大学工学部電気工学科

並列処理は今まで，常に成功してきたわけではない．むしろ数多くの失敗が積み重ねられていると言った方が正確であろう．しかし，それらを乗り越えてまたもや並列処理に熱い期待が寄せられ，研究が活発になってきている．以前の研究とはどこが異なるのであろうか．それは，まず，VLSI に代表される素子技術・実装技術の著しい発達による環境の変化，次に，応用問題の動作を分析してアルゴリズムを確立し，それに適合した専用機を作るという応用指向研究の定着，そしてデータ駆動，リダクション，アクタなど，新しい計算モデルの成熟，があげられるように思う．

並列処理のサーベイ論文としては，村岡[1]，Kuck[2]らによるものがあるが，いずれも 10 年ほど前のものである．それらをこの特集と比較してみれば，この間の技術の動きが見えよう．並列処理は古くて，新しい問題なのである．

本文ではまず，2. で並列処理の歴史的な流れをみ，3. で並列処理技術の概説をし，4. で現在の研究動向に触れ，5. でまとめる．

2. 並列処理の歴史

並列処理の最近の動きは後で述べることとし，ここではまず，1980 年頃より以前の並列処理の流れを概観する．相互に関係を及ぼし合ったとはいえ，この流れは次のような6つに分けてみることができよう．

（1） マルチプロセッサ

複数のプロセッサが主記憶を共有するという形のシステムで，1960 年代初頭の UNIVAC LARC 辺りからその形が見られる．主として計算機システムのスループットや信頼性を向上させることを目的としており，現在の汎用大型計算機のほとんどがこの形態を取っている．プロセッサ台数としては性能上4台，多くとも 16 台程度までとするのが普通であり，高速機では cache の無矛盾性維持の回路がボトルネックとなっている．

（2） SIMD 型マシン

多くの演算器やプロセッサを用いて，異なるデータに対し同じ演算を施す形のマシンがこれである．空間を細かく分割し各格子点の演算を近傍のデータを用いて行うような偏微分方程式の解法が代表的であるが，1920 年頃 Richardson の考えた多人数の人手による気象計算，1960 年代の SOLOMON 計算機の設計と，その改良型実現機としての ILLIAC Ⅳ[3] などがその流れに沿うものである．制御が比較的容易で大規模な処理を行える特徴を持っており，これに合ったタイプの計算，たとえば拡散方程式，熱方程式，画像処理，空気力学などには威力を発揮するが，これに適合しない計算も数多く存在する．最近では 16,000 個の処理要素を持つ MPP が実用になっており Cray-1 で処理する場合よりも 3 倍の分解能での画像処理が可能と言われている．

（3） MIMD 型マシン

それぞれ異なった処理を行う数多くのプロセッサからなるシステムで，マルチプロセッサの一種であるが，マルチプロセッサが異なるジョブを並列に行うことでスループットを上げることを主目的としているのに対し，一つのジョブを分割して複数プロセッサに分担させることにより高速処理を目指すところに力点がある．したがって，プロセッサ間通信機構もさまざまで，共有記憶のほか，明示的にデータの授受を行う機構も使われるほか，プロセッサ数も一般には多く，最近では 1,000 台規模のものが現れている．1970 年代中頃の C.mmp や Cm＊のほか，数多くのマイクロプロセッサを結合したいわゆるマルチマイクロプロセッサの多くがこれに相当する．SIMD 型マシンよりも柔軟な処理が可能なため，適用範囲は広いが SIMD 型処理に限れば能率は下がりやすい．この種のマシンの構造を特徴付けるのは要素プロセッサの結合トポロジであり，近傍間のみの接続とするか任意の要素間接続を可能とするかの選択がある．SIMD 型マシンでは能率が悪いが，MIMD 型によく適合する問題としては粒子モンテカルロシミュレーションがある．

（4） 連想プロセッサ

記憶素子それぞれに論理機能を持たせた，いわゆる Logic-in-Memory の系統に属するものである．1960 年代初頭から分散論理記憶 (DLM) や，連想記憶プロセッサとして検討されてきた．実用になったものとしては STARAN，OMEN，PEPE などがある．ビット処理，内容探索などに特徴がある．最近では VLSI 技術が発達してきたので，かなり大容量の連想メモリが手に入るようになっており，それを駆使したアーキテクチャを再検討する必要があろう．

（5） パイプライン方式

一つの処理を複数段の小処理段階に分けてハードウェアを用意しておけば，一つの小処理段階が済むとそのハードウェアが空くので新たな処理の受付けを開始できる．したがって等価的な処理時間はその小処理段階にかかる時間程度となる．これをパイプライン方式と呼ぶが，処理の高速化手法として非常によく使われている．これは，1960 年代始めの IBM STRETCH の先行制御方式辺りから，IBM 360/91 の演算回路内部のパイプライン化を経て一般に普及してきた．1970 年代初頭の ASC，CDC STAR-100 はそれを確固たるものにし，続く Cray-1 の成功は高速計算機といえば高度なパイプライン方式を指すまでに至らしめている．現在の汎用大型機はすべて数段～十数段のパイプライン方式を採用しているし，Cray-2 や SX-2 などのスーパコンピュータではパイプラインの 1 段が数 ns にまでなっている．

パイプライン方式の性能を定めるのは，この 1 段の所要時間と，パイプをどれだけ長く詰めておけるかであり，後者は対象問題によって大きく異なる．

パイプライン方式は，ほかの並列処理方式と独立して用いることができるのでほかの方式と共存して使われることが多い．

（6） データフローマシン

通常の計算機が，次に実行する命令を明示的に指示する（制御駆動と言う）のに対し，オペランドの到着によって実行可能となった命令から実行してゆく（データ駆動）のがデータフローマシンである．これは，可能なものはすべて並列に実行するという意味から並列処理の基本モデルであって，問題に内在する並列性を極力取り出すことができる能力を持っている．この考え方は 1970 年頃の並列プログラム研究に源を発し，単一代入の提案を経て，1974 年に J. Dennis がデータフロースキーマとしてまとめた．その後，MIT，Manchester 大学，UC Irvine，Toulouse 大学，テキサスインスツルメントなどにおいて試作や言語研究が進んだ．日本においても，武蔵野通研，東大，阪大，電総研，沖電気，ICOT，群馬大などで研究が進められ，今や我が国はデータフローマシン研究の最もさかんな国となりつつある．

3. 並列処理の技術

3.1 並列処理の諸レベル

ある応用をマシンの上で走らせる場合，当初の目的から何段かの変換操作を経て最終的に個々の電子素子の動作になるが，その変換操作に着目すると図-1 のような諸処理レベルを考えることができよう．図の各処理レベル相互間の変換に関係する技術が，その右側に示されている．もともとの応用に内在する並列性は，図のような変形を受けてゆく各過程で少しずつ失われてゆく．したがって，総合的な性能ができるだけ上がるようにこれら各技術を考える必要があり，一部のみの効率化は必ずしも全体の性能を上げていることにはならないことに注意する必要がある．

図中，処理プリミティブとは，その計算機の機械語に相当するもので，用いられている計算のモデルから定められ，処理の進め方などを論理的に表現したレベルである．次の実行形式は，その計算機の物理構造に従って処理プリミティブを各処理装置に割り付けたもので，この実行はそのハードウェア特性に依存する．したがって，処理プリミティブから下の動きを定めるのが計算機のシステムアーキテクチャであり，実行形式の動きを定めるのがプロセッサアーキテクチャである．

以下，この図の用語を基に，代表的な技術ポイントを概観する．

3.2 並列処理問題とアルゴリズム

一般に膨大な計算を必要とする科学技術計算，たとえば，原子力，核融合，気象，衛星画像，航空機設計，分子科学，CAD，音声処理，信号処理などは，それ自体かなりの並列性を持っていると言われており，プログラムの分析が行われている．計算の類別化が，連続系，離散系，粒子系などとしてなされているが，プログラムは逐次処理を念頭に開発されたものも多く，せっかくの並列性を殺しているものも少なくない．当初から並列処理を意識してアルゴリズムを開発するという並列プログラミングパラダイムの構築が望まれる．また，アルゴリズムの優劣の評価はそれを走らせる計算機アーキテクチャや，処理の割当て方式にも強く影響を受けるので，現実的な評価にそれらの考慮は不可欠である．

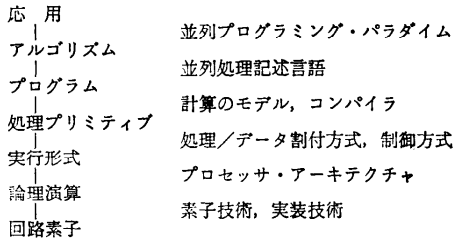

図-1　処理のレベルと関連技術

一方，知識情報処理もかなりの並列性を内在すると言われている．これは計算のモデルを変えれば，それによってアルゴリズムの記述法やその並列性が大きく影響を受けることを示唆しており，知識情報処理の枠組で今後一層の応用プログラムの開発と，その分析が望まれる．この分野の処理では発見的探索がその基本となることが多いが，その場合，単なるしらみつぶし探索では爆発的な場合数の増大に対処しきれず，なんらかのアルゴリズムを導入することが必要となる．実際，良いアルゴリズムの効果は絶大であって，少々のプロセッサ数を増やすことよりも有効である．アルゴリズムが重要なゆえんである．

3.3 並列プログラミング言語と並列性の抽出

並列プログラミング言語の役割は，アルゴリズムを具体的，形式的に表現することと，さらにそれをマシンの上で実行する場合のアドバイスを与えることである．前者は通常のプログラミング言語本来の役割であるが，特に並列処理という観点から見ると，並列性をプログラマが明示的に示すという役割も負うことになる．従来のスーパコンピュータの自動ベクトル化コンパイラのように，プログラマが明示しないでもコンパイラが自動的に並列性を取り出すことが望ましいのはいうまでもないが，それだけでは不十分であり，より多くの並列性を取り出すためにはプログラマの協力が必要である．マシン実行のアドバイスとは，プログラム内に出現する動的な型のサイズやストリームバッファのサイズとして予想される値や，コンフリクトが生じた場合の優先順を示したり，また複数プロセッサ上で実行する場合に各タスクを割りふる方法のヒントを与えたりすることで，能率良い実行には現在のところ不可欠である．一般的には，これらの指示をプログラマに強制することは望ましいことではなく，プログラマはアルゴリズムのみに関わるだけで済ませたい．しかし，並列実行の方法論が未成熟な現在，ある程度これを要求することは必要であるし，今後とも，あるレベルのプログラマには必要な機能であろう．

並列プログラミング言語としては現在，PL/I や Ada があり，これらはタスクレベルの並列処理記述を

与える．文レベルとしては以前 ILLIAC Ⅳ 用などに作成された言語があるほか，最近では CSP を基にした Occam がある．FORTRAN に並列処理用の記述能力を持たせる方向の検討もなされている．アレイ演算系やモジュール構造化の支援がそうで，並列性の解析やオブジェクトコード生成の容易化を目指している．

一方，言語や計算モデルから計算機システムを作っていこうというアプローチがある．関数型言語，論理型言語，プロダクションシステム，セマンティックネットワーク，オブジェクト指向言語などがそれで，これらはいずれも並列処理と密接な関係がある．これについては次節で述べる．

並列性の検出については，従来の自動ベクトル化コンパイラでは DO ループの最内側のみを対象としていたが，更なる並列性を求めて DO ループ間や，サブルーチン間の並列性を並列タスクとして取出す研究が行われている．グラフ変換によって冗長な依存関係を減らしループ構造を変えて，ループ間の並列性を高めるという検討もある．

並列処理の支援ソフトウェアとしては，スーパコンピュータを対象としたプログラムの作成・チューニング・実行の各段階を支援するものが開発されている．作成に対してはプログラムの分割やデータ配置法の指示を可能とすること，チューニングに対してはデータ参照／定義関係やサブルーチン従属関係などの解析をとおしての並列性認識や動的データフロー解析のツールなどがあり，実行支援としては並列性認識コンパイラのほか，実行時ライブラリ，タスク間分割／同期やデータ転送の制御用ソフトウェアなどがある．

3.4 計算モデル

計算モデルやプログラミング言語の研究から，それを支援するアーキテクチャを検討するという，1970年中頃から始まった動きがある．本節では，並列計算モデルの一般論はさておき，より具体的なものを取り上げる．まず，並列処理の理論的モデル研究の集大成として現れたデータフローモデルは，プログラムに含まれる最大限の並列性を抽出しやすいという利点から多くの研究が行われている．特に，不規則な形をしたプログラムからうまく並列性を取り出すことは，パイプライン方式やアレイ計算機の不得意とするところで，ほかの方式では得がたい特徴である．最近は，単に並列性を多く出すのではなく，必要に応じてそれを制御し，マシン資源の有効利用をはかる機構も工夫されるようになった．

データフローマシン用言語は基本的に関数型であるが，関数型計算モデルからの研究も多い．パターンの書換えを中心として処理を進めるリダクションマシンがそれで，Berkling マシン，Mago マシン，ALICE などがあげられる．これも並列性の観点からみれば性格の良いモデルである．

述語論理を計算モデルに設定した研究は，第5世代計算機プロジェクトとして有名である．言語としては Prolog のほか，いくつか提案されているが，最近では並列処理を処理の基本とする新しい言語 Guarded Horn Clauses も現れた．これら言語の処理プリミティブの検討も進んでおり，PIM，PIE，PARK，KPR などかなり並列度の高い並列推論マシンの研究が行われている．このモデルは，知識をベースとした処理形態に相性が良く，知識獲得，学習，自動プログラミングといった機能への接近が試みられている．

Expert System の構築にプロダクションシステムを用いることが多いが，それをアーキテクチャ側から支援しようとする研究がある．OPS5 を対象とした DADO，PSM などがそうで，パターン探索を多くのプロセッサで並列に行うことにより高速化を指向している．

知識表現手法の一つとしてのセマンティックネットワークを直接ハードウェアで支援しようとする研究もある．Connection Machine がそれで，10^4〜10^6 台ものプロセッサを用いることが検討されている．

さらに，ソフトウェアの能率良い作成を支援するパラダイムにオブジェクト指向計算がある．この基本概念は，actor と呼ばれる主体がメッセージを交わし合うことで処理を進めるというモデルである．これは並列処理のモデルとしても整合性が良いと考えられており，並列オブジェクト指向言語やそれに向いたコンピュータアーキテクチャの研究がある．また，オブジェクト指向の考え方をほかの言語に持ち込み，オブジェクトという自然な単位をモジュラプログラミングのモジュールに用いるとか，並列タスクの単位に用いるなどの研究も行われている．プログラマに不自然な形で並列処理のヒントを強要するのではなく，プログラマにとって自然な単位であるオブジェクトをこれに用いることはきわめて自然な手法であり，より一層の研究が望まれる．

以上述べたように，計算モデルを中心とした並列処理の研究は数多く存在する．しかし計算モデルを定めても，それをマシンの上で実行するプリミティブ操

作の選択（処理の粒度：granularity）には自由度があり，そこにマシンアーキテクチャ研究の意味がある．計算モデルを変えることは，アルゴリズムに大きな影響を与え，したがって並列性にも大きな変化を与える．本節で述べたマシンの研究は，このようにラジカルなものでいまだ研究段階のものがほとんどであるが，今後の大きな発展が期待される．並列処理を従来の von Neumann の枠内に閉じて行うことは一般になり手間がかかりやすく，並列処理の利益を相殺しかねないのである．

3.5 実行割付けと制御

論理的に数多くの並列性が得られても，それを物理的な並列マシンの上でうまく走らせて最終的な並列利得を得るためには，実行割付けの能率が問題となる．すなわち，各並列アクティビティをどのように各プロセッサに割り付け，データをどこに配置すれば最も能率が良いかを決定する必要がある．この決定には，各 activity の処理時間，所要データ，計算機内の通信オーバヘッドなどが関係するが，さらに，その activity の処理結果として次に新しい activity をできるだけ多く生成するようなものを先に処理することも必要であろう．使えるプロセッサ台数が限られている場合，それらをできるだけ有効に使うためには，処理の開始時点ではできるだけ急速に全プロセッサを稼動状態に持ってゆき，飽和した時点では各プロセッサが遊ばないように activity を割り付けるなどの工夫が要る．

activity の割付手法としては一般に3種ある．一つはプログラマが指定する方法，二つ目はコンパイラで割付けを決定する方法，三つ目は，各時刻における各プロセッサの稼動状況に応じて動的に割付けを決める方法である．動的手法は一見確実に見えるが，一般にその決定のための負荷モニタリングや判定手続きのオーバヘッドのため，より容易な静的割付けを採用する例が多い．また，仮に各 activity の処理時間や所要資源がすべて既知であっても，全実行時間を最小にする割付けパターンの決定には非常に多くの計算時間がかかるため，なんらかの発見的手法の導入が不可欠である．またもともと，その前提条件としての情報が不明であることも多い．したがって実際上は，処理開始時の activity 割付けを静的に行い，後は粗い稼動状況に基づいて簡単な修正をするなどが現実的な解であろうが，この辺りはいまだ多くの具体的な研究が必要である．

タスク間同期を取るなどの実行制御はできるだけオーバヘッドを減らす必要があり，共有メモリへのアクセスを少なくする工夫や，ハードウェアで同期通信を支援するなどの研究も行われている．

3.6 結合網と共有メモリ

数多くのプロセッサを結合して大規模システムを作り上げる場合，その性能に重要な影響を与えるのが結合網である．網としては，できるだけ伝達遅延が小さく，かつ転送スループットの大きいものが望ましいが，網が大規模になるとその双方を経済的に満たすことは難しくなるため，通信の局所性を取り入れた網のトポロジがよく用いられる．網の設計上重要なのは，このほか，スイッチング制御速度，障害時の代替ルート機能，網内デッドロックが発生しないことなどである．

最近は網内にかなり高級な機能を埋め込もうという研究もある．たとえば，マシン PIE における各プロセッサの負荷状況の帰還機能や，自動的にすいたプロセッサを選択する機能，さらに，Cedar プロジェクトにおける同一宛先へのアクセスを整理する機能などがあり興味深い．

数多くのプロセッサがあるとき，それらから共有される資源があれば，そこへのアクセスが集中する結果，それが全体性能のボトルネックになりやすい．これは高並列システムを設計する場合の常識である．しかし，共有メモリはプロセッサ間の通信機能として非常に優れているため，できることなら用いたい．これらのジレンマを解決する手段として考えられているのが，共有メモリの多バンク分割による競合の減少策や，マルチリード・ワンライトメモリである．後者は，共有メモリに対する読出し要求が，書込み要求よりも何倍か多いという特性を利用したもので，同じコピーを持ったメモリを複数備えることにより読出し時の競合を避けている．神戸大の PARK や，慶大の $(SM)^2$ などで使われている．

3.7 デバッグ及び障害対策

並列プログラミング言語でプログラムを作成してもそのデバッグをいかに行うかは難しい問題である．計算の流れが複数の流れとなり，また相互関係が非決定的であったりする．従来のステップ実行だけでは苦労するのは目に見えている．より抜本的なデバッグ支援が望まれる．たとえば，データフロー方式では関数としての枠組みの利用，オブジェクト指向システムではメッセージと応答の利用などがあげられよう．

システム内の要素数が増すと，当然各要素の障害を考慮した構成とする必要がある．障害要素の同定，切

り離し，システム内再構成などの処置が行いやすい形でなければならず，高並列マシンが実用になるためには Fault Torelant Computer の技術が必要であろう．

4. 研究動向

4.1 大規模アレイコンピュータシステム

数十台から数千台のプロセッサを結合したシステムで，並列処理の研究や科学技術計算を目的としたものが多く現れている．筑波大の PAX，Calitech の Cosmic Cube，Mosaic，ITT の CAP のほか，商用としても Sequent 社の Balance など多くのシステムがある．32 bit マイクロプロセッサの出現によって一段とシステム構成が容易となったもので，その結合形態はマトリクス，ハイパーキューブ，バスなどである．

4.2 スーパコンピュータ

Cray-1 などのいわゆる商用のスーパコンピュータは，パイプライン演算器を中心に構成されてきたが，単一ステージが現在 5 ns を切るようになりそろそろ Si の限界に近づきつつある．したがって，素子を GaAs などのより高速なものに変えてゆくことのほか，別の方向としては，パイプラインを複数設けることと，アレイプロセッサなどの並列アーキテクチャを採用することであろう．ここ数年来の大規模科学技術計算の分析結果によって，応用がかなり類別されてきており，またそれらに適合する並列アーキテクチャが明確になってきている．すなわち，パイプライン方式，SIMD 方式，MIMD 方式の分化であり，問題によってこれらを選択するという行き方，ないしはアーキテクチャを柔軟にして SIMD/MIMD の選択を動的に再構成可能とする行き方である．さらに，より高速化を狙って，単なる最内 DO ループやベクトル化にとどまらず，外側ループ相互間の並列性や手続き間の並列性を利用しようという動きである．我が国のスーパコンピュータの大型プロジェクトのみならず，アメリカの Cedar などにもそれが見られる．Cedar では大きな単位の並列処理をマクロデータフローと呼んでいる．現在，このようなスーパコンピュータのプロジェクトは数多く，世界で数十存在する．上のほか，ローレンスリバモア研究所の S-1，IBM の RP 3，ITT の CAP，CDC 社の Cyberplus など枚挙にいとまがない．

4.3 VLSI 指向アーキテクチャ

これからの計算機はほとんど VLSI を用いることが考えられるが，中でも特に VLSI 指向の強いシステムがいくつか現れている．VLSI に望ましい特徴としては，単純な繰返し構造，ソフトウェアのハードウェア化，要素間通信コストが小さいことなどで，したがってメモリ構造を中心とした設計や，高度並列処理が向いている．

代表的なものはシストリックアレイである．いまだ実用規模のものは作られていないが，徐々に実用化されている．CMU のコンピュータビジョン用のシストリックアレイコンピュータ Warp がその例である．シストリックアレイをフルウェファを用いて実装しようという試みも行われている．

厚木通研の AAP-1 は，256×256 アレイプロセッサであり，その規則的な構造を用いて VLSI 化し小さくまとめている．

その他，3 次元 VLSI の研究も行われており，1990 年代には 10〜20 層のものが手に入ることが予想される．発熱やプロセス技術等に問題があるが，さらに多くの素子が手に入ることに加えて，高集積によるコンパクト化や，3 次元化による垂直配線で配線遅延が短くなり高速化が計れることなどが期待される．

4.4 データフローマシン

データフローの概念が明確になって以来，ここ 10 年くらいの間に多くの実験機が作られた．Manchester 大のマシン，Toulouse 大の LAU，TI の DDMI，東大の TOPSTAR，武蔵野通研の DFM，沖電気の DDDP，MIT の MEF，電総研の EM-3，SIGMA-1，ICOT の PIM-D などである．その結果，評価も行われているが，一般に並列性のある問題に対しては非常に優れた並列利得が得られている．必要なゲート数はコンベンショナルな計算機に比して多く，またメモリも多く必要で，ベクトル演算のような定形的な処理に対してはパイプライン方式の計算機の方が優れているが，非定形的な並列性を有する問題に対してデータフローマシンは実に強力である．パイプライン方式やアレイ計算機ではほとんど並列性を出し得ないが，データフローマシンはコンスタントに力を発揮する．

最近はデータフロー型の信号処理用チップがあるほか，商用のマシンも出始めた．データフロー方式が未だ十分他のアーキテクチャを越えないまでもなんらかの形でコンピュータの中に取り込まれることは十分考えられ，手始めとしてその最も容易なところがマクロデータフローであろう．

4.5 知識処理マシン

第 5 世代コンピュータプロジェクト以来，我が国では述語論理を対象とした並列推論マシンの研究が多く

行われている．ICOTのPIM-P, PIM-R, 東大のPIE, 京大のPR, 神戸大のK-Prolog, 電総研のDialog, RITのOR Parallel Token Machine, BullのMulti SCHUSS, ECRCの並列シンボリック・マシンなどである．当初pure Prologを対象とするものが多かったが，ストリーム並列などでAND並列を取り込む動きがあり，今後GHCのサポートが話題となるであろう．評価に対しては適切なベンチマークプログラムの蓄積が必要である．

関数型のマシンについては1986年2月動き始めたImperial College of LondonのALICE，製作中のMagoマシンなどのほか，MITではMultilispマシンが作られている．

プロダクションシステムの支援ではColumbia大のDADO, CMUのPSMがある．これら両者は対照的で，DADOが多数台低速プロセッサを指向するのに対し，PSMでは少数高速プロセッサシステムである．また，シュランベルジュ社のFAIM-1もAI処理支援を目的としたものでオブジェクト指向の枠組にいくつかの言語様式を入れている．

その他，セマンティックネットワークを直接アーキテクチャで支援するThinking Machine Corp.のConnection Machine, CMUのBolzmann Machine, 電総研のIXなどがあり，これらは10^4を越える大規模プロセッサ指向のシステムである．これらは別の観点から眺めればLogicin-Memoryそのものであり，大規模素子技術が手に入った現在，連想メモリに近いこのような構成も考えてみる必要があろう．

4.6 専用コンピュータシステム

最近の動向の一つは多くの専用マシンが開発されていることである．きちんと専用分野をしぼって応用を分析し，それに合ったマシンを作るという行き方は，素子コストが安い今日，確実な方法である．その分野としては，画像処理，図形処理，信号処理，CAD，データベースなどがあり多くのマシンが作られている．スーパコンピュータ大型プロジェクトでは，衛星画像処理，三次元図形処理を目的としたマシンが検討されているし，Goodyear AerospaceのMPPは画像処理用，IBMのYSEやNECのHALは回路／論理シミュレーション専用機，Britton-LeeのIDM, ICOTのDelta，東大のGRACEなどはデータベース処理専用のデータベースマシンである．

これらは問題をしぼっただけハードウェア化による利得は大きく，たとえば富士通の分光相関システムFXの場合，FFTなどを主体とする処理で120 Giga Operations Per Secondに達すると言われている．この速度は通常のスーパコンピュータではとても達成不可能で，専用機の作られるゆえんである．

5. むすび

本文では，並列処理の歴史，技術，及び動向について概観を行った．並列処理は，計算機に対するあくなき処理能力の需要と，著しい素子技術の発展によるシーズ技術（素子はいくらでもあります．うまく使いこなして下さい）の両面に支えられて今や発達せざるを得ない段階にある．研究の成果は着々と蓄えられつつあるが，要約すれば次のように言えようか．

まず，成功する一つは，目的をうまくしぼり，応用をきちんと分析して処理パターンを見極めて専用システムを作ること，それに使いやすい支援ソフトを付けることである．次に，汎用並列処理マシンは，今道具立ても揃いアタックするに面白いテーマであるが，そこで銘記すべきは計算モデルと言語パラダイムの重要さである．それに並列アクティビティの割付等制御手法をアーキテクチャと密接な形で確立することであろう．

いずれにせよ並列処理は，すべての状況をシミュレーションで調べることが難しく，試作してみることが大切である．

従来，計算機システムは，チャネル，SVP, CCPなど次々と機能の分散化が行われてきたが，これらも広い意味の並列処理であり，本論の並列処理が密な並列であるのに対し粗な並列と言えよう．並列マシンの商用機も次々と現れているが，本格的な発展は正にこれからである．並列処理が特殊なシステムでなくなり，その特殊な実行形態が逐次処理となるという形態はまだとしても，専用システムから徐々に並列処理が流布するのは確実である．今後の発展が期待される．

参考文献

1) 村岡洋一：並列処理概論（1）～（3），情報処理，Vol. 16, No. 1～3, pp. 65-72, pp. 137-144, pp. 239-245 (1975).
2) Kuck, D. J.: A Survey of Parallel Machine Organization and Programming, Computing Surveys, Vol. 9, No. 1, pp. 29-59 (1977).
3) 加藤満左夫，苗村憲司：並列処理計算機－超高速化へのアーキテクチャー，オーム社 (1976).

（昭和61年5月12日受付）