54R—015

第５世代の電子計算機に関する調査研究報告書

—— アーキテクチュア研究分科会 ——

昭 和 55 年 3 月

JIPDEC

財団法人 日本情報処理開発協会

この報告書は，日本自転車振興会から競輪収益の一部である機械工業振興資金の補助を受けて昭和５４年度に実施した「第５世代の電子計算機に関する調査研究」の成果をとりまとめたものであります。

序

わが国における社会経済は、資源、エネルギー問題を始めとして、国際的な変動と、不確実性の流れのなかにある。同時に、的確な情報の加工利用が重要視される情報化社会の形成が指向されている。

コンピュータは、われわれの情報活用においてすでに不可欠なツールとなっているが、今後１０年間には多くの諸問題を解決するため更に高度な技術が要求され、新たな理論、技術にもとづくコンピュータ・システムの実現が望まれる。

このため当協会では「第５世代コンピュータ調査研究委員会」を設置し、１９９０年代に実用化されるべきコンピュータ・システム（第５世代コンピュータ）はどのようなものになるか、またその開発プロジェクトはどのように進めていくべきかについての調査研究を昭和５４年度から２か年の予定で開始した。

本年度は第１年目として、委員会の下部機関として社会環境、基礎理論、アーキテクチャの３分科会、ワーキンググループを編成したほか、大学・研究所への研究委託、米国からの外人講師招聘等により１９９０年の社会シナリオ、技術シナリオを作成するとともに第５世代コンピュータに必要とされる理論と技術について調査研究を行った。

次年度は、具体的な目標設定と開発計画、及び体制について研究を行う予定である。

最後に、調査研究にご協力を頂いた第５世代コンピュータ調査研究委員会委員を始め関係各位に厚く御礼申し上げる次第である。

昭和５５年３月

財団法人　日本情報処理開発協会

会　長　上　野　幸　七

アーキテクチャ研究分科会

	委員名	役職
主査	相磯秀夫	慶応大学工学部電気工学科教授
委員	飯川昭一	三菱電機㈱計算機製作所 計算機計画グループ　計画担当課長
〃	飯塚肇	電子技術総合研究所 電子計算機部計算機方式研究室長
〃	小高俊彦	㈱日立製作所神奈川工場開発部主任技師
〃	国井利泰	東京大学理学部情報科学科教授
〃	坂間保雄	日本電信電話公社　横須賀電気通信研究所 データ通信部 データ通信方式研究室 研究専門調査員
〃	坂村健	東京大学理学部情報科学科助手
〃	杉本正勝	富士通㈱開発技術部部長付
〃	武井欣二	日電東芝情報システム㈱　システム技術部主任
〃	田中英彦	東京大学工学部電気工学科助教授
〃	発田弘	日本電気㈱コンピュータ技術本部 方式技術部技術課長
幹事	元岡達	東京大学工学部電気工学科教授
〃	石井治	電子技術総合研究所ソフトウエア部長

デバイス・ワーキング・グループ

	委員名	役職
1	石井　治	電子技術総合研究所 ソフトウエア部長
2	内田俊一	電子技術総合研究所ソフトウエア部 情報システム研究室研究員
3	国分明男	電子技術総合研究所電子計算機部 記憶システム研究室主任研究官
4	山口喜教	電子技術総合研究所電子計算機部 計算機方式研究室研究員

目　　次

第１章　序論—調査研究活動の概要

第 1 章　序論 —— 調査研究活動の概要

1.1　活動の方針

第5世代アーキテクチャ研究分科会の調査研究活動の基本方針は次のとおりである。

(1)　１９９０年代に開発される計算機が具備すべき機能について，計算機アーキテクチャの観点から検討を加える。

(2)　初年度は次の点を留意して，問題の整理にあたる。

(i)　現在に至るまでの技術的な発展経過を調査する。

(ii)　現在の計算機が直面している問題点を把握する。

(iii)　新しい機能を求める要因ならびに必要性を理解する。

(iv)　問題解決にあたって，計算機アーキテクチャとの関係を明確にする。

(v)　新しい計算機アーキテクチャの意義ならびに効果について評価を行なう。

(vi)　新しい計算機アーキテクチャ実現の可能性とその問題点について検討する。

(vii)　将来の計算機を開発するために検討すべき研究課題について整理する。

(3)　実現すべき計算機アーキテクチャに関する研究課題の具体的な検討は次年度以降に行なう。

1.2　活動の方法

本分科会の今年度の活動方法は次のとおりである。

(1)　検討すべき研究項目について，学術文献，国内外学会発表成果，有識者インタビュー，調査研究委託，他の研究分科会との意見交換などを通して調査研究を行なう。

(2)　必要に応じて，出張調査，有識者招待，調査研究委託，文献購入などを行

ない調査研究の支援を行なう。

(3) 必要に応じて，作業グループを設け，調査研究を依頼する。

(4) 本分科会での調査研究成果を報告書の形にまとめる。

1.3 活動の経過概要

上述に示す本分科会の活動方針ならびに活動方法に従って，今年度は次のような調査研究活動を行なった。

(1) 本分科会は原則として，毎月2回委員会を開催することにし，実際21回の分科会をもった。

(2) 調査研究に際しては，関連検討項目についての有識者との討論を多くもつことに努めた。また，他の研究分科会との意見交換にも力を注いだ。

(3) 本年度は「デバイス技術予測」について作業グループによる調査研究を依頼した。

(4) 委託研究については以下の4件を行なった。

「ユーザ援助情報システムに関する調査」（東京大学元岡研究室）

「計算機の適応および学習機能に関する調査研究」（慶応義塾大学相磯研究室）

「データベースマシン・アーキテクチャの調査研究」，「設計仕様処理における要求工学技術の調査研究」（東京大学国井研究室）

(5) IBMのSRI（System Research Institute）からDr. Glenford J. Myers を招ねき，最近の計算機アーキテクチャの話題について討論を行なった。

(6) 報告書のまとめは3回にわたる長時間討論（合宿を含む）で行なった。

なお，本分科会ならびにデバイス・ワーキング・グループの委員は表1.1と表1.2のとおりである。

表1.1　アーキテクチャ研究分科会

	委員名	役職
主査	相磯秀夫	慶応大学工学部電気工学科教授
委員	飯川昭一	三菱電機㈱計算機製作所 計算機計画グループ　計画担当課長
〃	飯塚肇	電子技術総合研究所 電子計算機部計算機方式研究室長
〃	小高俊彦	㈱日立製作所神奈川工場開発部主任技師
〃	国井利泰	東京大学理学部情報科学科教授
〃	坂間保雄	日本電信電話公社　横須賀電気通信研究所 データ通信部 データ通信方式研究室 研究専門調査員
〃	坂村健	東京大学理学部情報科学科助手
〃	杉本正勝	富士通㈱開発技術部部長付
〃	武井欣二	日電東芝情報システム㈱　システム技術部主任
〃	田中英彦	東京大学工学部電気工学科助教授
〃	発田弘	日本電気㈱コンピュータ技術本部 方式技術部技術課長
幹事	元岡達	東京大学工学部電気工学科教授
〃	石井治	電子技術総合研究所ソフトウエア部長

表1.2　デバイス・ワーキング・グループ

	委員名	役職
1	石井　治	電子技術総合研究所 ソフトウエア部長
2	内田俊一	電子技術総合研究所ソフトウエア部 情報システム研究室研究員
3	国分明男	電子技術総合研究所電子計算機部 記憶システム研究室主任研究官
4	山口喜教	電子技術総合研究所電子計算機部 計算機方式研究室研究員

1.4　報告書のまとめについて

今年度における本分科会の調査研究成果を第2章以下にまとめて示す。

第2章は先ず将来の計算機開発における計算機アーキテクチャ研究の重要性，現在の計算機が直面する問題点とその解決アプローチについて述べている。次に、１９９０年代に開発されるであろう計算機 — System A — の機能イメージを紹介している。開発の哲学，個々の構成要素，あるいは新しい計算機アーキテクチャ技術との関係などについても概要を示しているが，ここで与えている将来の計算機像は必ずしも技術的な観点から述べられたものではなく，比較的概念的なものであることに注意されたい。

第3章および第4章は将来の計算機開発に関連する重要な技術について，最近の動向や問題点などを論じたものである。

第3章は作業グループに依頼した「デバイス技術予測」の調査成果をまとめたものである。１９９０年代に重要な位置を占めるであろうデバイスのいくつかについて，技術的な観点から位置づけを行っている。

第4章は計算機アーキテクチャに関連する重要な個々の技術の一部について技術の現状，問題的ならびに将来動向を論じたものである。

第２章　新しい計算機システム
アーキテクチャ

第2章　新しい計算機システム・アーキテクチャ

2.1　計算機アーキテクチャ研究の重要性

2.1.1　計算機アーキテクチャの性格

"計算機アーキテクチャ"という言葉が最初に用いられたのは，IBM 7030 (Stretch)の開発においてであろう。[*] そこでは「計算機の構造に関係するユーザからの要求事項を決め，その要求事項を経済的ならびに技術的な制約の下で可能な限り効率的に満足させる設計技術である。アーキテクチャは工学上の配慮をも含むべきであるが，むしろユーザの要求事項に力点が置かれている」と性格づけしている。この用語の正確な定義はIBM System/360の開発者G. M. Amdahl等によって初めて行なわれ，「プログラマから見たシステムの属性，すなわち，データの流れや制御の構造，論理設計および物理的実現方法とは別の概念的構造および機能的動作」を計算機アーキテクチャとしている。[**] しかしながら，現実的な立場では，計算機アーキテクチャの中に物理的構造に関する事項を含める必要が起きるので，より広義に解釈することもある。[***] このように計算機アーキテクチャのとらえ方は人によって若干の相異があるが，少くとも「機械語プログラマ，コンパイラあるいはオペレーティング・システム作成者などから見たハードウエアの論理仕様」という狭義の意

---

[*] W. Buchholg(Ed.): "Planning a Computer System—Project Stretch", Chapter 2, p. 5, McGraw—Hill, 1962

[**] G. M. Amdahl, et al.: "Architecture of the IBM System/360," IBM J. R&D, Vol. 8, №2, pp. 87～101, April 1964

[***] S. S. Reddi, et al.: "A Conceptual Framework for Computer Architecture," Computing Surveys, Vol. 8, №2, PP. 277～300, April 1976

味と「ハードウエアならびにソフトウエアに関する基本設計」という広義の意味があると見るべきであろう。

本報告書では，計算機アーキテクチャを広義の意味で一般的に解釈することにするが，ここで注意すべきことは，計算機アーキテクチャの意義は種々のユーザから要求される機能を価格性能比，処理速度あるいはメモリ要求量といった現実的な制約の下で可能な限り効率的に実現することにある。換言すれば，ユーザから要求される論理的な機能を実用的なレベルで実現するために計算機アーキテクチャが討議されるのであり，最初から計算機アーキテクチャが先導的な役割を果すことは少ないといえる。もちろん計算機アーキテクトの立場から，理想的な計算機アーキテクチャの追求も行なわれており，多くの面に大きなインパクトを与えていることも事実である。しかしながら，一般的に見た計算機アーキテクチャの立場では種々のユーザや応用面からの要求事項の解析と将来のハードウエアならびにソフトウエア技術の把握とが最も重要な課題である。

2.1.2. 新しい計算機アーキテクチャを求める要因

ソフトウエア開発の困難さ，膨大なソフトウエア資産の蓄積，ＬＳＩ（大規模集積）化に伴なうハードウエア設計・開発の制約などを考えると，新しい計算機アーキテクチャを大幅に導入することは容易なことではない。今までの計算機に見られるように，単に価格性能比の改善のために新しい機能を求めるとすれば，ハードウエアのＬＳＩ化に徹した方が得策であろう。それにも拘らず，最近新しい計算機アーキテクチャを求める傾向の背景には，

(1) 膨大な計算時間を必要とする大規模な科学計算問題や大量のデータを実時間で取扱う画像処理などの応用分野が急速に開拓されつつあり，超高速計算機能ならびに大幅な価格性能比の改善が求められている。

(2) 現在の計算機アーキテクチャとユーザが多用する高級プログラム言語との間には形式的というよりも概念的あるいは意味論的な相異，すなわちセマン

ティック・ギャップが目立ち，このままではソフトウエア開発の生産性改善は困難である。この問題を解決するためには，新しい概念に基づくプログラム言語の確立ならびに計算機アーキテクチャ・レベルでの基本的な機能支援が必要になっている。

(3) より自然な会話形式の処理あるいは複雑なグラフィック処理などを含んだ使い易いマンマシン・インタフエースを実現するための基本的な機能が求められている。

(4) 冗長技術あるいは可変構造技術を活用した高信頼性システムの実現が現実問題になっている。

(5) データベース，パターン処理，知識の蓄積・検索・利用を目的とした知識ベースあるいは人工知能といった全く新しい複雑な機能を要求する応用分野が展開している。

(6) 従来の計算機の使用法は起りうる全ゆる場合を想定した，いわゆる決定性（deterministic）プログラミングに基づいているが，マンマシン・インタフエースを含む多くの応用において，人間が通常とるヒューリスティックな非決定性（non-deterministic）データ処理手法が有用なことが認識されつつある。このような新しい問題解決手法には，例えば，あいまい性を伴なう連想処理機能あるいはプログラムの処理において任意の時点まで後戻りを可能にするバックトラック機能などが本質的な機能として望まれている。

(7) ハードウエア技術，特にLSI 技術の急速な進歩に伴って特殊ハードウエアの実現が容易になり，計算機アーキテクチャに必然的な変化が見られる。

(8) パーソナル・コンピュータあるいはインテリジェント端末など使い易さを求めた機器により，情報処理の普及が試みられている。

などの理由が考えられる。このうち(1)～(6)は計算機アーキテクチャに対する応用面からのニーズであり，(7)～(8)は技術進歩に伴うシーズといえる。特に，応用面からの要求は新しい計算機アーキテクチャなしに効率的かつ実用

的な情報処理が不可能なことを意味し，計算機アーキテクチャ研究の重要性を示唆している。

2.2 現状分析と新技術

2.2.1 ノイマン型計算機の性格と改良

これまでに開発され，実用に供されてきた計算機の多くは１９４６年のJohn von Neumann らの提唱する概念[*]に基づいたもので，一般にノイマン型計算機と呼ばれている。しかしながら，ノイマン型計算機を明確に定義することは困難であるが少くとも次のような性格を備えていると考えられる。

(1) プログラム自身を実行する前にメモリに記憶する，いわゆるプログラム記憶方式である。

(2) １命令ずつの逐次処理が基本で，その制御装置を一つ備える。

(3) メモリは一つで，線型アドレスをもつ。

(4) 命令語とデータ語を定義しているが，

(i) 命令語にはオペランドの所在を示すアドレスをもつ。

(ii) データ語処理の制御権は命令語がもつ。

(iii) 命令語とデータ語のメモリ内での区別はない。

(iv) データ語の種別を示すものはない。

など両者を明確に区別している面とあいまいな面の双方がある。

(5) ハードウエアとソフトウエアの役割を区別しているが，ハードウエアの論理構造は固定的である。すなわち，ハードウエア機能は動的に変えることができない。

(6) 決定性論理構造ならびに決定性プログラミングを前提としている。

---

[*] A.W. Burks, et al.: "Preliminary Discussion of the Logical Design of an Electronic Computing Instrument," IAS Princeton University, 1946

このような性格は今日の計算機にとっても基本的かつ常識なものであり，従来の機械には見られない数々の優れた機能を提供する基盤になっている。しかしながら，最近の応用分野の展開に伴ってノイマン型計算機に機能的限界が見られるようになり，本質的な改良が求められるようになった。このような改良の試みとして提案される新しい機能を総称して非ノイマン機能（Non － von Neumann Computing ）とここでは広義に定義したい。もちろん，最初のプログラム記憶方式計算機から今日の計算機に至るまでにも種々の観点から改良が加えられているので，非ノイマン機能は必ずしも新しいものとはいえないが，近い将来改良が予想される計算機アーキテクチャ・レベルの機能には質的に大きな変化が見られるものがある。

ノイマン機能に対する改良の試みのいくつかを挙げれば次のとおりである。なお，個々の計算機アーキテクチャ技術に関する最近の動向については第4章を参照されたい。

(a) プログラム記憶方式は計算機にとって最も基本的，かつ最も優れた機能である。この方式は記憶内容の読出し・書替えの考え方に基づく経過依存性（History Sensitive ）あるいは状態遷移を利用した高度な順序機械の実現を可能にしている。このような観点からは，将来の計算機においてもプログラム記憶方式の意義は変わることはないであろう。しかしながら，最近のソフトウエア工学などでは，むしろ経過依存性を否定する意見もでている。機能を本質的に単純化することによって，プログラミングの自動化あるいは新しいデータ処理方式の実現を容易にしようという考え方がある。

(b) 逐次処理に対する改良の試みは並列処理やマルチプロセッサ・システムにおいて見ることができる。

データ処理の制御を関連するデータ自身に行なわせる新しい方式として，最近データ・フロー・アーキテクチャが注目を集めている。この概念に基づけば命令語の逐次制御装置を必要とせず，最大限の並列処理が実現でき，しかも超ＬＳＩ 技術の有効利用，ソフトウエア開発の生産性改善ならびに自然言

語処理などに関連する新しい問題解決手法を支援できる可能性があり，今後重要な技術となろう。

(c) 現在のデータ処理においては，与えられた問題ごとにプログラムやデータは固有の構造をもつのが普通である。そのような構造を線型アドレスをもつメモリにマッピングするところに大きな問題が残されている。この問題を避けるために，メモリの分割，比較的単純なデータ構造を取扱えるディスクリプタ方式，あるいは制御スタックなどの概念が導入されている。

線型メモリ上で構造という概念をすてたハッシュ機構や連想メモリなども有用な機能を含んでいる。

(d) オペランドのアドレス指定を全く行なわず，レジスタの管理を自動化した演算スタックは複雑な構造をもつ数式の処理に便利な機構である。

データ側にデータ処理のための制御とデータ自身の種別を示すタグをもたせれば，実行すべき命令の機能をデータ側から修飾できる。その結果，命令語は単純になる。例えば，加算に関する機械命令は1種類でよく，データ側で浮動小数点，固定小数点，可変長，単精度，多重精度などの処理の詳細を指定できる。この考え方は通常の高級プログラム言語の形式とよく整合する。命令語が少なくなるためコンパイラも単純になり，処理効率もよくなる。更に，ファームウエア化にも適してくる。データをタグで区別できれば，プログラミング上の誤りも少くなり，加えて，命令実行前後のプログラム状態を効率よく把握でき，デバッギング機能も付加され，計算機アーキテクチャと高級プログラム言語との間のセマンティック・ギャップの縮少に役立つ。

プログラミングの誤りを防ぐという観点からは最近，種々の保護機構あるいはアクセス制御またはケイパビリティ制御機構をアーキテクチャ・レベルで備えることが提案されている。

(e) ハードウエアを可変構造にすれば，与えられた問題に対して最適な機能を自動的に合成する，いわゆる問題適応型計算機を実現できる可能性がある。そのためには，ダイナミック・マイクロプログラミング技術の活用が鍵になる。

同じような意味で，高レベル中間言語のファームウエア化，高級プログラム言語直接実行プロセッサあるいは特殊専用プロセッサの開発もハードウエアがもつ機能の可変構造性に依存しているといえる。

(f) 従来の計算機では，全ゆる論理判断あるいは状態遷移は入力変数のシーケンスで一義的に決まる。このような決定性論理あるいはプログラミングの概念に基づく問題解決手法は入力変数によって結果を得る過程が明確な問題に対しては有用であり，実際に今までの多くの問題はこれに該当している。しかしながら，複雑な確率モデルのシミュレーション，未知の要素を含む人工知能問題の解決あるいは意志決定問題の支援を計算機が行なう場合には必ずしも適していない。むしろヒューリスティックな非決定性処理方式に従う方が本質的な問題解決に結びつく可能性がある。このためには，バックトラック制御機構，推論機構あるいは学習機構など従来の計算機では見られない機構が計算機アーキテクチャ・レベルで備えられることが望まれる。自然言語処理などには必須な機構となろう。

以上，今までの計算機システムの中に見られるいくつかの非ノイマン機能を示したが，いずれも現在のノイマン型コンピュータに含まれる個々の機能を改善する立場で考案されている新しい機能と見ることができる。しかしながら，これらの非ノイマン機能をアーキテクチャ・レベルに組込むことによって，本質的にあるいは実用の面から解決困難であった多くの問題に対して有用な解決手段を与えることができると期待される。

2.2.2 現在の計算機の問題点

(1) ユーザの立場から

最初の計算機が開発されてから３０余年が経つが，その間計算機をとりまく技術進歩は著るしく，１０年ごとに約１００倍の価格性能比の改善を達成している。しかしながら，計算機の応用分野は計算機の能力をはるかに上まわる機能を要求する問題を生み出している。このため，計算機への要求も断

えることがないが，全ゆる要求を満足するような解を一つの計算機上に求めることは不可能であり，ある程度しぼった問題ごとに異なった解決アプローチをとるのが常識になっている。ここでは現在の計算機をとりまく情報処理の問題点と解決のための技術との関係を簡単に示してみたい。

現在の計算機が種々の応用分野で直面している問題点を列挙すれば，

(1) データ処理量の増大と質の多様化

(2) マンマシン・インタフェースの改善

(3) 信頼性の向上

(4) ソフトウエア開発の生産性改善

(5) システム評価の必要性

(6) 価格性能比の改善

などであろう。

このような問題を提起している要因には種々のものが考えられるが，その主な要因とそれを解決するために検討されている主要な技術との関係を表2.2.1に示す。

表2.2.1　現在の計算機の問題点と解決アプローチ

問題点	具体的な要因	解決するための技術またはシステムの例
(1)	大規模計算	超高速科学計算機・専用プロセッサ・データフロー・プロセッサ
	処理量の増大	マルチプログラム処理・マルチプロセッサ処理・分散処理システム
	ネットワーク化	コンピュータ・ネットワーク・ネットワーク・アーキテクチャ
	データベース化	データベース・マシン　・　分散データベース
	システム移行	エミュレーション　・　バーチャル・マシン
	新しい応用	画像処理・パターン認識・オフィスオートメーション・推論機構・知識情報ベース・人工知能

問題点	具体的な要因	解決するための技術またはシステムの例
(2)	音声・図形入出力	パターン認識
	使い易いプログラム言語・処理効率の改善	高レベル言語マシン。自然言語処理
	日本語データ処理	漢字処理
	使い易いファイルの実用化	単一レベル記憶 ・ ファイル計算機 データベース・マシン ・ 問合わせシステム
	現場でのデータ処理	分散データ処理
	ＯＳ機能の充実と使い易さの改善	ＯＳ核・ＯＳ機能のファームウェア化
	無人化・省力化	ＯＳ機能の充実・診断機能・保守体制・自動運用システム
(3)	ハードウェアの信頼性向上	冗長技術 ・ フォールト・トレラント計算機
	ソフトウェアの信頼性向上	構造化プログラミング ・ ドキュメント・システム
(4)	プログラム言語の改善	システム・プログラム言語 ・ 高レベル言語マシン
	プログラム開発手法の改善	構造化プログラミング ・ 開発管理技法
	プログラム・デバッグツールの開発	デバッグ・サポート・ハードウェア
	アーキテクチャ・レベルでの機能支援	並列処理制御機構・連想処理・バックトラック制御機構
(5)	性能予測	モデル化・解析手法 ・ シミュレータ・評価システム
	チューニング	モニタ機能 ・ 自動チューニング 適応化技術
(6)	コスト低減	ファームウェア化 ・ ＬＳＩ化 ・ 設計自動化
	性能向上	ファームウェア化・ＬＳＩ化

図2.2.1は現在の計算機の問題点と解決技術との関係を図示したものである。

なお，表2.2.1および図2.2.1に示した問題解決のために検討されている主要技術のいくつかは第4章において述べられているが，よりミクロなレベルでの計算機アーキテクチャとの関係をこれらの中から抽出し，その実現法を検討することが今後の重要課題といえる。

(2) 設計の立場から

表2.2.1に示した問題点と解決法は，主として計算機を使用する立場から見たものである。従って　"より高い機能を低価格で提供すること"がその基本的目的となっているが，現在の計算機は，この他にその設計側から見てもいくつかの問題点をかかえている。ここでこの点についても若干触れておきたい。

まず，最大の問題点はアーキテクチャの設計思想が明確でないことである。即ち，これまでの多くのアーキテクチャ技術は個々の問題を解決するために，その時のデバイス技術にあわせて考え出され，取入れられた結果，いわば，場当り的となり，混乱を招いているのである。このことは計算機がハードウェア素子が極めて高価であった時代から30年間に渡るゆるやかな改良の結果，現在に至ったという事情にもよるし，アーキテクチャがハードウェア素子と応用の間の一種のインタフェースであって，各時点での両者からの要求を妥協させるように設計せねばならなかったことにもよる。

しかしながら，デバイス側も，応用側もかなりの成熟度を見せると予想される第5世代ではアーキテクチャにも明確な思想を打出し，計算機に構造化した設計を取り入れることが必要であり，また，可能であると考えられる。

ここで，構造化した設計とは，ソフトウェア工学における構造的プログラミング(Structured Programming)とほぼ同様の意味であって，例えば，ハードウェアを明確に特性の記述されたモジュールに分割して，その組合せとしてシステムを記述したり，実際に構成したりすることもこの中に含まれる。更に言葉がソフトウェアからの類推であるのみならず，ハードウェア設計もソフト

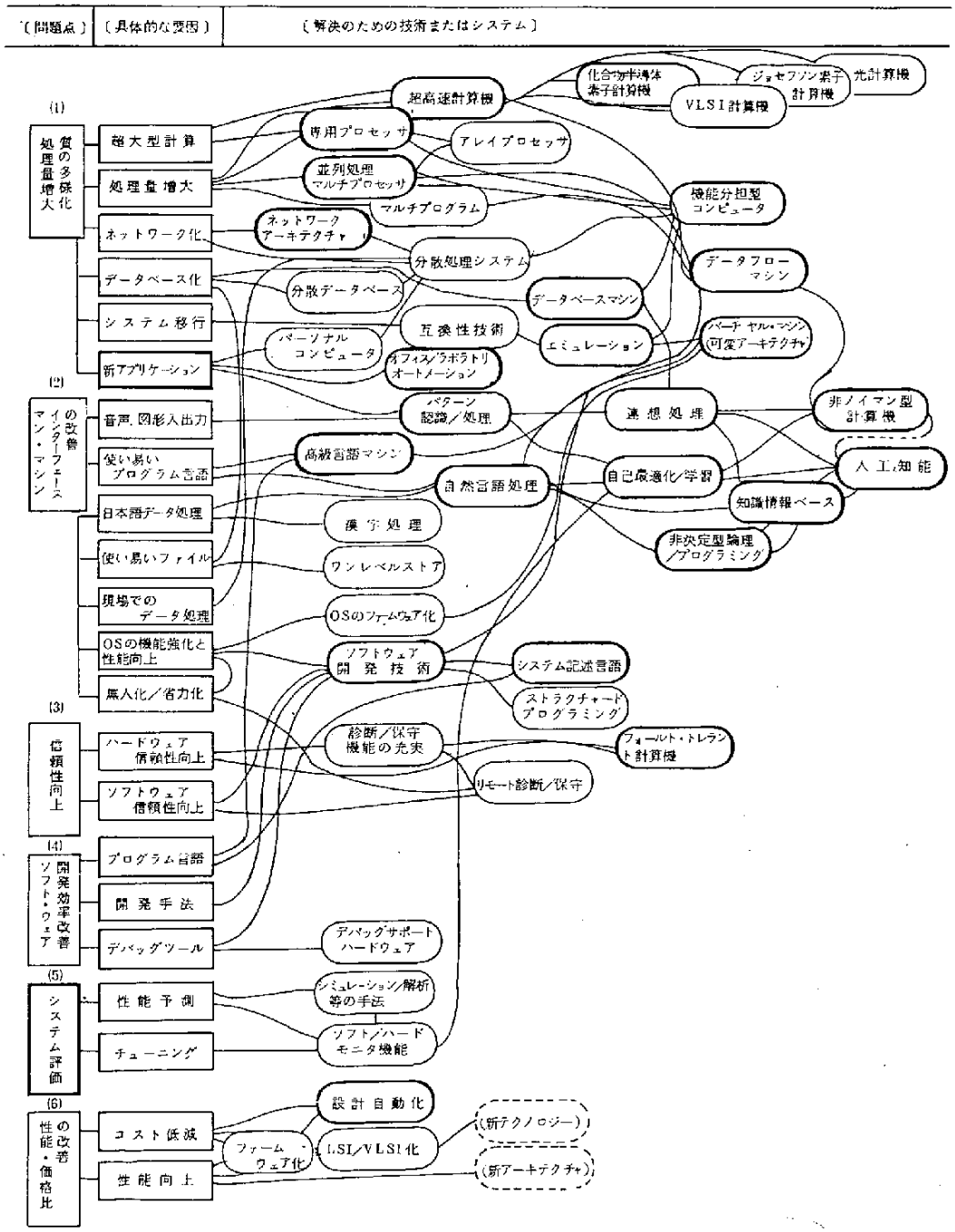


図 2.2.1　現在の計算機の問題点と解決技術との関係

ウェア設計もその本質はあまり変るものではないから，今後のハードウェア（アーキテクチャ）設計にはソフトウェア工学の成果を大いに取り入れるべきであるということもできる。

かくして，アーキテクチャやハードウェアの構造が1本の思想を持ったすっきりしたものとなれば，計算機の設計自動化も容易になり，ひいては低コスト化にもつながるし，信頼性や可用性を高めることにもなる。また，アーキテクチャとインプリメンテーションを分離し，第5世代から第6，7世代への移行に際しては，そのロスを最少に保って新しいハードウェア素子技術を活用することができるであろう。

2.3 将来の計算機像

2.3.1. 応用分野の背景とその特徴

将来の計算機は前節で示した諸問題を解決するための機能あるいは効果的な支援機能を計算機アーキテクチャ・レベルで積極的にとり入れる方向に進展するものと考えられる。しかしながら，問題の多様性に直接的な関係をもつ応用分野の広さから見て，将来の計算機が一機種に収約されるとは考えられない。少くとも以下に示すいくつかの代表的な応用分野ごとに特徴のある計算機が開発されるように思われる。それぞれの応用分野において重要と思われる計算機アーキテクチャ技術と計算機システムとしての将来像を簡単に示せば次のとおりである。

(1) 事務処理分野

この分野の計算機は計算処理そのものよりも大量データの蓄積・検索・管理に主眼を置いた，いわゆるデータベース指向のシステムになると思われる。計算機ネットワークを基盤とした分散処理システム構成が常識になり，現在のように事務用計算機が単体で用いられることはまれとなろう。また，高級プログラム言語，問合わせ言語あるいは自然言語に近い言語で計算機を利用

することになる。いずれはオフィス・オートメーションという形態をとるものと思われる。

(2) 公共システム分野

行政・医療・教育・広報など共益的な性格をもつ応用分野では大規模な計算機ネットワークの上に厳重なセキュリティをほどこしたデータベースが構築されるに違いない。この分野では分散データベースの管理，効率のよいアクセス法ならびに厳重なアクセス権の設定，一般市民とのインタフェース設計などが重要な技術課題となろう。また，経済ならびに社会問題のシミュレータなども重要な役割を果すと思われる。

(3) 産業システム分野

諸産業における設計・管理・経営などを支援する計算機システムは現在個別に活用されているが，将来は一貫した総合システムへと発展するものと考えられる。理想的には受注から設計・生産・販売に至るまでの作業は系統的に計画・管理されることになろう。この場合，高度のマンマシン・インタフェース機能を備え，新しい問題解決手法が容易に適用できる高機能シミュレータが主役を果すことになる。

(4) 科学計算分野

大規模なマトリクスやベクトル計算を要求する科学計算問題あるいは大量のデータを扱う画像問題の分野では汎用計算機では処理しきれない問題が続出する傾向にあり，科学計算指向の超高速計算機の存在が確立するものと考えられる。

(5) 制御分野

プロセス制御で代表されるこの分野の計算機は高度な計算機能が要求されないが，効率的な入出力制御や高い信頼性が要求される。多くの場合，分散処理システム構成をとることになろう。また，この分野には知能レベルの高い工業ロボットなども含まれることになる。

(6) 宇宙航空分野

特殊な環境の下で，軽量，小形，低電力消費，堅ろうであることが特に要求され，場合によって極めて高い信頼性を考慮する必要がある。この分野では，通常機器の制御が主な使用目的となるが，その場合は計算ならびにメモリに対する厳しい要求はない。しかしながら，最近では航空機側で複雑な信号処理を行うことが要求され，特殊な科学計算機能を実装することもある。

(7) 民生分野

家庭機器の制御などに用いられる民生用計算機はその性格からして安価なマイクロコンピュータが主流になろう。将来はホームコンピュータとして市民生活に不可欠な存在となろう。

2.3.2 技術的な背景

１９９０年代の計算機を支える技術のうちで最もインパクトの大きいものはやはり半導体ＬＳＩ技術であろう。ＬＳＩ技術はコンピュータの小型化・低価格化・低電力消費化・高信頼化をもたらし，機能的にも大規模化・高機能化に拍車をかけるものと思われる。その結果，現在多用されている大型商用計算機と機能的に劣らぬ小型ＬＳＩ計算機の開発が常識になる。このような高機能ＬＳＩ計算機が開発されれば，かなりのユーザが自分の手元に置いて，いわゆるパーソナル・コンピュータとして使うようになるものと予想される。少くとも計算処理のようなデータ処理はユーザの手元に分散し，今までのように計算センターに頼る必要はなくなろう。個人ベースのデータ処理の多くはパーソナル・コンピュータでローカルに行なわれ，計算センターはユーザ全体の共通の仕事場としての効率的なデータベースを提供すると同時に，特殊システムを設置し，全てのユーザが共用できるよう管理することが主な役割になる。

ＬＳＩ 技術はファイル・メモリにも大きな影響を与え，ＣＣＤ や磁気バブルによる電子ディスクの普及を促している。恐らく１００ＭＢ程度のパーソナル・ファイルは高速で信頼性の高い電子ディスクが主流を占めることになろう。

同様に主メモリの周辺においても論理メモリの開発が期待されている。

入出力装置への機能分散化も進んでおり，処理効率やマンマシン・インタフェースの改善などを目的に端末のインテリジェント化，パターン処理のため前後処理機能を端末側に盛込むことになる。

ソフトウェアに関しては，開発の生産性を改善するための手法のほか，ユーザが計算機システムを容易に使えるようなソフトウェア体系が確立されよう。問合わせ言語あるいは自然言語に近い形で計算機が使えるようになる。

2.3.3 System A 開発の考え方

上述のような応用分野ならびに技術的な背景から考えると，１９９０年代の計算機システム — ここではこれをSystem A＊と呼ぶ — は次に示すような考え方の下で開発されよう。

(1) 超ＬＳＩ 技術を基盤とした小型・高性能パーソナル・コンピュータが普及し，ユーザの手もとに分散される。多くのデータ処理はパーソナル・コンピュータで処理される。

(2) 計算センターには，パーソナル・コンピュータでは手におえない大規模な仕事を処理するための大型・高性能・汎用計算機，超高速科学計算機，高機能シミュレータ，高度なマンマシン・インタフェースと新しい問題解決機能を備えた計算機などが設置され，ユーザの共用に供される。

(3) ユーザのための共通の仕事場として，大規模なデータベースとそれを効率的に管理するデータベース・マシンが計算センターに設置される。

(4) パーソナル・コンピュータ，データベース・マシンならびに各種共用計算機は機能的には階層構造をなしている。

(5) ユーザと計算センター，あるいはユーザ間通信を可能にするため，計算機ネットワークを介して計算機同志が接続される。

＊ アーキテクチャ研究分科会での呼称であり，第５世代コンピュータ調査研究委員会全体での公式呼称ではないことをお断わりしておく。

(6) ユーザに質の高いサービスを供するため，プログラムあるいは計算機の実行状態を監視する機構，プログラムのデバッグや計算機の故障検出・自動修復を可能にする高信頼化機構，システムの最適制御を目標とするチューニング機構，無人運転・防災・自動運用を可能にする管理機構，ならびにドキュメント・教育システムなどが備えられている。

2.3.4 System A の構成

—— 第5世代コンピュータ・システムのアーキテクチャ・イメージ ——

(1) 概　　要

本節では，2.3.1節で述べた思想に基づき構築された第5世代コンピュータ・システムの構成イメージについて述べる。イメージは，主として機能的な観点から考察する。考察の方法は必ずしもトップダウン的ではない。

① とはいっても，イメージ構築は，ユーザの要求を知ることから始まった。まず，ユーザの要求を広く一般的にとらえる事から始めた。すなわち，ユーザの要求を2つに分類した。どの要求にも共通して現れるような汎用性のある要求と，ほとんど共通性がなく，要求に個有の特殊性の強いものの2つである。そして，ここでは個々の要求の中でも特に“何が特殊性の強い要求であるか”をできるだけ抽出することに努めた。

② また，これとは別に我々アーキテクチャ研究分科会では，分科会スタート当初より，現在のコンピュータ・システムを主としてアーキテクチャ面から分類を行ない，その構造にどの程度汎用性があるのか，また，特殊性が強いものについては，どのような応用に対して有効であり，実装技術とは関係なく，アーキテクチャ的にどの程度の可能性があるかを明らかにすべく作業を進めてきた。※

そして，以上 ①，②を融合させた。すなわち，ユーザの要求を広く一般的にとらえ，可能性のあるアーキテクチャの形態を元として，第5世代のイメ

※ この成果は第4章にまとめられている。

ージを機能的に外挿（エクストラポレート）しようとした。したがって，本節で示す図のほとんどは機能構成図であり，設計図とか，物理的な接続を意味するものではない。

また，第5世代コンピュータのイメージとは決して未来を予測しているのではない。未来とは広大，無限であり，我々には可能性を論じることしか出来ない。

我々が行なっている（行ないたい）のは，未来システムを想像することではなく，むしろ新しい一つのアーキテクチャの提示，構築なのである。よって，もしSystem Aを未来システムととらえるならば，それは単なる一可能性であることを強調しておく。

(2) イメージ構築の準備

イメージ構築の前に，新アーキテクチャに対するユーザの要求と，アーキテクチャ構築者，システム構築者から見た新システムへの要求をまとめておく。

① ユーザの要求

ユーザにとって，システムとはブラックボックスであり，それはどのように作られていてもよい。ユーザにとってみれば，解決したい問題が速く解ければ，それはどのような構成であろうとかまわない訳である。しかし，ここで注意しなければならないのは，「速く」という言葉には非常に多くの意味が含まれているという事であろう。「速く」という言葉を正確に解釈するならば，それは単に実行速度のみが速いということだけを意味しない。それは問題（要求）が提示されてから解決されるまでのトータルな期間の短縮を意味している。したがって以下に示すようないくつかの要求がアーキテクチャに対して出されていると解釈出来る。

(i) 問題（要求）が，容易に計算機に写像できるようなアーキテクチャ

具体的には計算機とユーザのインターフェースは，プログラミング言語であるので，使い易いプログラミング言語が作り易く，かつコンパイル処

理などの実行が十分高速であるアーキテクチャの要求。

(ii) 問題解決に当り十分高速であるようなアーキテクチャ

(i)で写像されたプログラムが十分な速度で実行出来るアーキテクチャの要求（通常の意味での高速性）。

(iii) 操作性のよいアーキテクチャ

(i)とも関係するが，計算機を機械と考えたときの操作性のよいことへの要求。

(iv) 高信頼性であるアーキテクチャ

ここで，(ii)，(iv)項については，計算機誕生以来多くの研究が成されてきたのに比べ，(i)，(iii)に対する要求を解決すべき研究は今までは比較的少なかったように思える。また，(iii)，(iv)の要求はどのような問題を解く場合にも共通しているのに比べ，(i)，(ii)は出される問題に依存していると思われる。したがって，問題により要求の程度は異なる。すなわち(i),(ii)を満足するためには目的指向，応用指向であるアーキテクチャの構築が必要と言える。また，問題に対しての適応性が高いアーキテクチャの要求が強いとも言えよう。

② アーキテクチャならびにシステム構築者からの要求

インプリメンタから出される要求はコスト・パフォーマンスの追求が最重要項目であるが，詳細は以下の通りである。

(i) インプリメントが容易であること。

ここで言うインプリメントとは，システムのインプリメントすべてを指す。したがってその中には，ハードウエア，システム・ソフトウエアなどを含む。いくらすばらしいアーキテクチャでも動かなければどうにもならない。当り前ではあるが，計算機は動くことが前提とされている。

(ii) 移行技術が確立していること。

新システムは，旧（現）システムのもとに存在する。旧システムから新システムへの移行は必須であり，きわめて現実的な問題として移行技術が確立していなければ，新システムの構築はありえない。すなわち，システムは

連続的に成長しており，これを止めることは実際問題として難しい。

(iii) メンテナンスの容易性を重視したアーキテクチャ。

基本的には高信頼性への要求であると思われるが，まったく故障しないということは考えられないので故障した場合にも直し易いアーキテクチャの要求は強い。

(3) System Aのイメージ

(2)の準備をもととして，System Aのイメージを固める。System Aは実現を前提としている。したがって，“いつ”実現するかが重要なポイントとなる。そこで，イメージ固めの前に，System Aとは１９９０年代に造られる全体システム*であることを再確認しておく。目的指向が強く，現在のシステムとの互換性の重視がイメージ造りの出発点となる。

① 機能イメージの特徴

ここでのポイントは，

- 適応化
- 階層化
- 分散化

の３点である。

(i) 適 応 化

(2)でも述べたように，新システムに対する要求の中でも，目的指向，応用指向の強い，いわゆるユーザから見た場合，システムがユーザの解くべき問題に適応しているようなシステムの要求は高い。そして，この要求を満足するようなアーキテクチャとして専用計算機（パーソナルコンピュータ**）を

(注) * システムであることに十分注意されたい。我々の頭にあるのは，常にあらゆる問題解決を考慮に入れたシステムであり，特定の問題を解決するためのものではない。

** これは，プライベートコンピュータとも呼べる。パーソナルコンピュータの我が国で通常連想されるイメージは貧弱であるので，特に，ここでは（個人）専用計算機をプライベートコンピュータと呼び，現在のパーソナルコンピュータと区別したいが，本報告書では慣例に従いこれもパーソナルコンピュータと呼ぶこととする。（意味の違いに十分注意されたい。）

各応用ごとに用意するようなイメージが浮んでくる。各問題はその問題を解くのに専用に計算機を与えられるので，適応化はやり易いと思われる。あくまで各応用ごとに1台の割り当てが原則であり，これにより，使い易さ，処理速度，システム全体の信頼性において各応用に対して，その要求を満足させなければならない。ここで問題となるのは，パーソナル・コンピュータとしてどの程度の能力を持つものが，System Aでは用意すべきかという事になるが，＊ 90年代の半導体技術・実装技術の能力からしてそのハードウェア能力はほぼ現在の大型計算機（たとえば，マシンサイクル50ns，主記憶1Mバイト，2次記憶100Mバイト）程度と考える。したがって，System Aでは，問題の多くは，このパーソナル・コンピュータだけで処理されてしまうと思われる。

(ii) 階層化

応用によっては，パーソナル・コンピュータだけでは処理しきれないものも出てくる。まず，処理能力的にパーソナル・コンピュータでは処理しきれないものがある。すなわち，パーソナルコンピュータはコスト的に強い制約を受けているので応用によってはそのコストで実現できる能力では満足できないものもでてくる。そのような応用は共通性の高いのが通常であり，多くの人々に共有されて使うことも十分考えられる。そこで，このような応用に対しては，多くの人々に共有して使われるという事を前提に，共通性のある特定の問題に密着した高性能計算機という形でイメージ化される。そこで，これを以後，特殊目的計算機と呼ぶ。興味深い点は，おそらく，90年代の汎用機（共有で使われるという意味で汎用という言葉を使っている。）は，現在の汎用機と異なり，構造的には汎用でなく，特殊目的であるという点であろう。また，パーソナル・コンピュータとの関係は階層的でありパーソナルで処理出来ない問題がここに送られてくるという事になる。（図2.3.1参照）

＊　これは，ほとんどコスト的因子のみによって決定されてしまう。

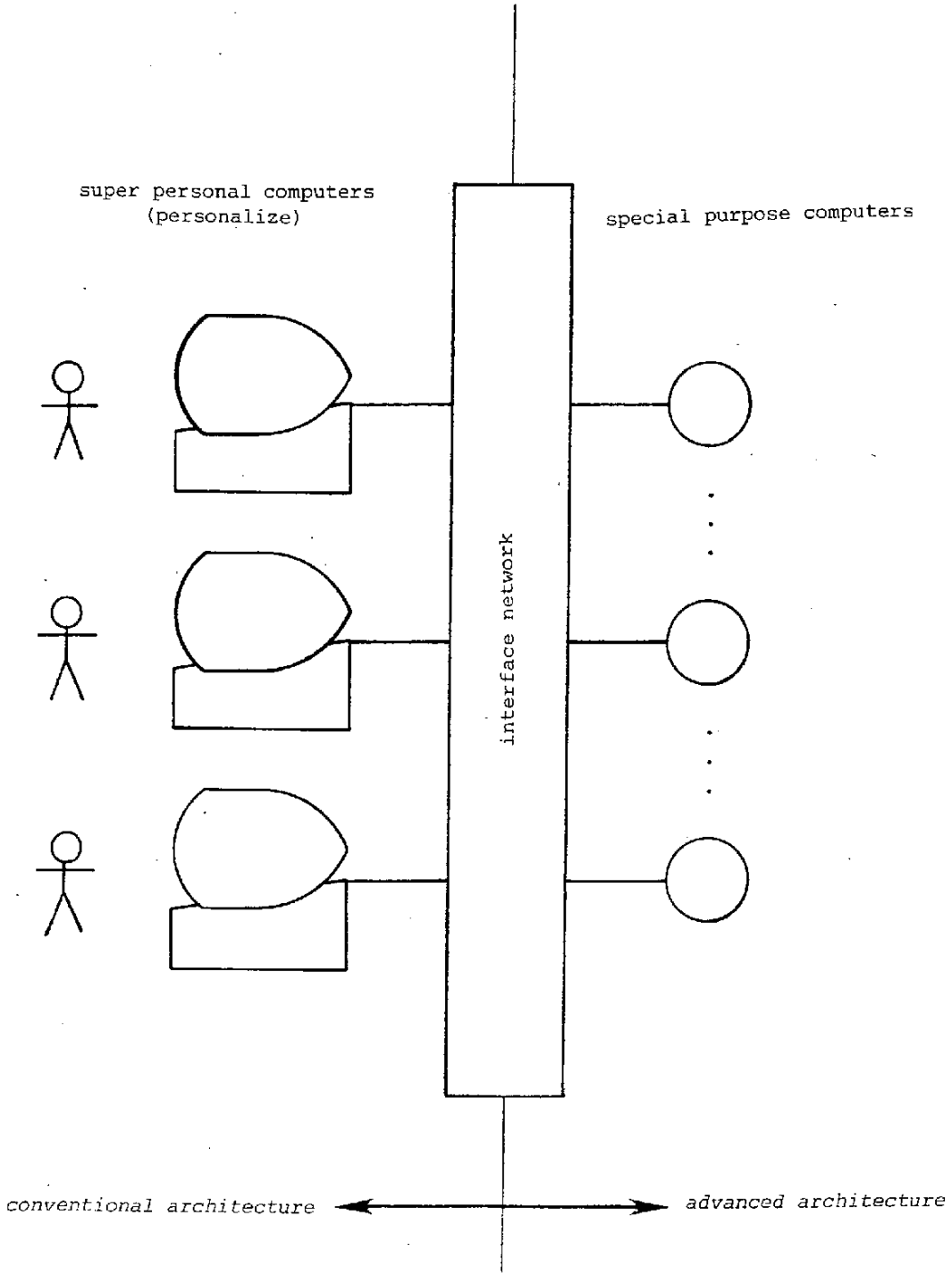


図2.3.1　System A のイメージ

(iii) 分　散　化

パーソナル・コンピュータだけでは処理出来ない問題のもう一つは，データが分散されている場合である。そして，このためにはパーソナル・コンピュータ間をつないでやる必要が出てくる。System Aにとって“計算機間通信”は重要で，分散された計算機は“通信路”によって積極的に交信を行なうものとイメージ化される。

② 実現にあたってのイメージ

次にSystem Aを実現面からイメージ化してみる。

(i) パーソナル・コンピュータの構造

“パーソナル・コンピュータ”は各応用専用なので，各個人の応用に適した構造をとる。したがってその構造も各応用によって異ってもよい。しかしながら，このレベルでのアーキテクチャを実現という点を考慮に入れ考察する場合，少なくとも，チップレベルアーキテクチャは同一で，その構造も現在の延長上にあろう。すなわち，プライベート化の前提の一つに低価格化があり，低価格化は大量生産の原理に基づき初めて実現される。プライベート化され，各応用ごとに専用計算機化するには，同一チップ構造は不可欠であり，ユーザには専用化されているように見えるであろうが構造的に一番下のレベルのチップレベル（部品レベル）では同一でなければならない。そして，専用化は，それ以上のレベル，少なくとも命令セット（ISP）[*]レベル以上で行なわれよう。したがって，アーキテクチャ的に重要と思われる技術は，マイクロプログラミング（特にダイナミックマイクロプログラミング）にあろう。また専用化するためのソフトウェア（目的指向ソフトウェア）の開発も重要となろう。専用化するための特殊入出力装置，入出力アーキテクチャもポイントとなろう。

(ii) 階層化のレベル

前述したように，“パーソナル・コンピュータ”とは汎用構造に基づく

＊ Instruction Set Processor の略。

専用マシンであり，原則的には何でもこなせる。ただ“要求”によっては，“パーソナル・コンピュータ”の能力では満足出来ないものがあるわけで，この場合，階層的に処理しようという訳である。では階層化はどの位のレベルを考えればよいのか。System A では3レベルとしている。すなわち，レベル1にパーソナル・コンピュータがくる。（この能力は現在の大型計算機程度）その一つ上のレベルにレベル2として現在の特殊計算機クラスの能力を持つ計算機を持ってくる。レベル2コンピュータはまた，分散されたレベル1コンピュータにとっては，クラスター的働きもする。レベル1コンピュータ間の制御を行なう。また，レベル1とレベル3の交信も受けもつ。このような意味でこれをサービスマシンと呼ぶ事とする。

レベル3は，非常に目的指向の強い応用に向いた，構造的にもかなり特殊な構造が要求されるようなクラスのコンピュータである。これはまた，特殊目的超高機能マシンと呼ぶ事とするが，これは現在のところ（現在の技術では）実現されていない程度のマシンを想定している。構造的に現在の計算機とまったく異なる事も十分考えられる。なお，おおよその機能比（速度比・容量比）は，

レベル1：レベル2：レベル3＝1：100：1000

程度と考えられる。

また，System A を一つのシステムと考えた場合のシステム構成上の量的な比率は，

レベル1：レベル2：レベル3＝1000：10：1

程度と考えられる。

(iii) 特殊目的超高機能マシン

パーソナル・コンピュータ，サービス・マシンでもなお処理しきれぬ分野とは何か。1990年代になっても，なおさらに，規模，能力に対する要求が拡大し続ける応用とは何か。

機能的には次の4つと考える。

(a) データベースマシン

大規模データベースの高速検索が出き，インテリジェンスレベルが高く高度なマンマシン・インタフェースを持つデータベースシステム。

(b) シミュレーションマシン

大規模実験設備に代るものとしてのコンピュータ・シミュレーション，CAD，経済社会環境問題などのシミュレーション，経営意志決定のためのシミュレーションおよび組込み型シミュレーションのためのマシン。

(c) 科学計算マシン

固体・流体力学，気象計算，原子力，画像処理，分子化学など巨大科学計算の高精度計算および高速数値計算処理のためのマシン。

(d) 高インテリジェンスマシン

画像処理，数式処理，知識処理，自然言語処理，文書作成処理，人工知能向特殊処理等の新しいタイプの処理のためのマシン。

(iv) レベル2，レベル3の構造

レベル2，レベル3は，かなり特殊性が強まるので（レベル3は特に）レベル1のように同一チップなどという構造をとることは不可能である。ここでは，実現面からの機能分散が積極的に行われよう。特にレベル2においては，“要求”に近いレベルでの分散（現在，一般に言われている機能分散）が行なわれる。しかし，レベル3になると特殊性が強まるためレベル2とは異なり，構造に近いレベルでの分散（要求とは関係のない構造上の問題）が必要とされると思われる。

(v) モニタ・コントロール機構

また，このような階層構造をベースとした大きなシステムでは，システム全体での最適制御や，運用上大きな比重を占る高信頼性という点から，動的なシステムモニタに基づくシステム制御は不可欠で，このための特別な機構が必要とされよう。ここでは，これをモニタ・コントロール機構と呼び，他

の機能ブロックとは明確に分離しておく。

③ イメージの詳細

機能的なイメージを図2.3.2に沿ってさらに詳しく述べると以下のようになる。

(i) パーソナル・コンピュータ（レベル１）

分散処理による処理効率の改善と高度なマンマシン・インタフェースの実現を図り，次のような機能をもつ。

(a) マシンサイクル：５０ｎｓｅｃ

(b) 主記憶：１Ｍバイト

(c) 高級プログラミング言語，強力なエディタを備え，日本語の取扱いをサポートする。

(d) ユーザ・マイクロプログラム可能

(e) 高分解能ＣＲＴディスプレイ

(f) １００Ｍバイト程度のパーソナル電子ファイル

(g) ハードコピー装置等の接続可能

(h) 単一ユーザ用オーペレーテング・システム

(i) 比較的単純な会話型グラフィック処理

(j) パターン処理の前処理，後処理

(k) 計算機ネットワークを介して，他のリソースにアクセス可能

(l) 高機能ＲＰＧ，問合せ言語などによる共有データベースへのアクセス

(ii) サービスマシン（レベル２）

パーソナル・コンピュータより，１桁以上の高い処理能力を提供するユーザサービスマシンと，他の計算機システム，プロセス制御系を管理するマシンサービスマシンおよび全体のスケジュールを司るコントロールマシンの３つのマシンに大別される。

(a) ユーザサービスマシン

・各種高級言語マシン群

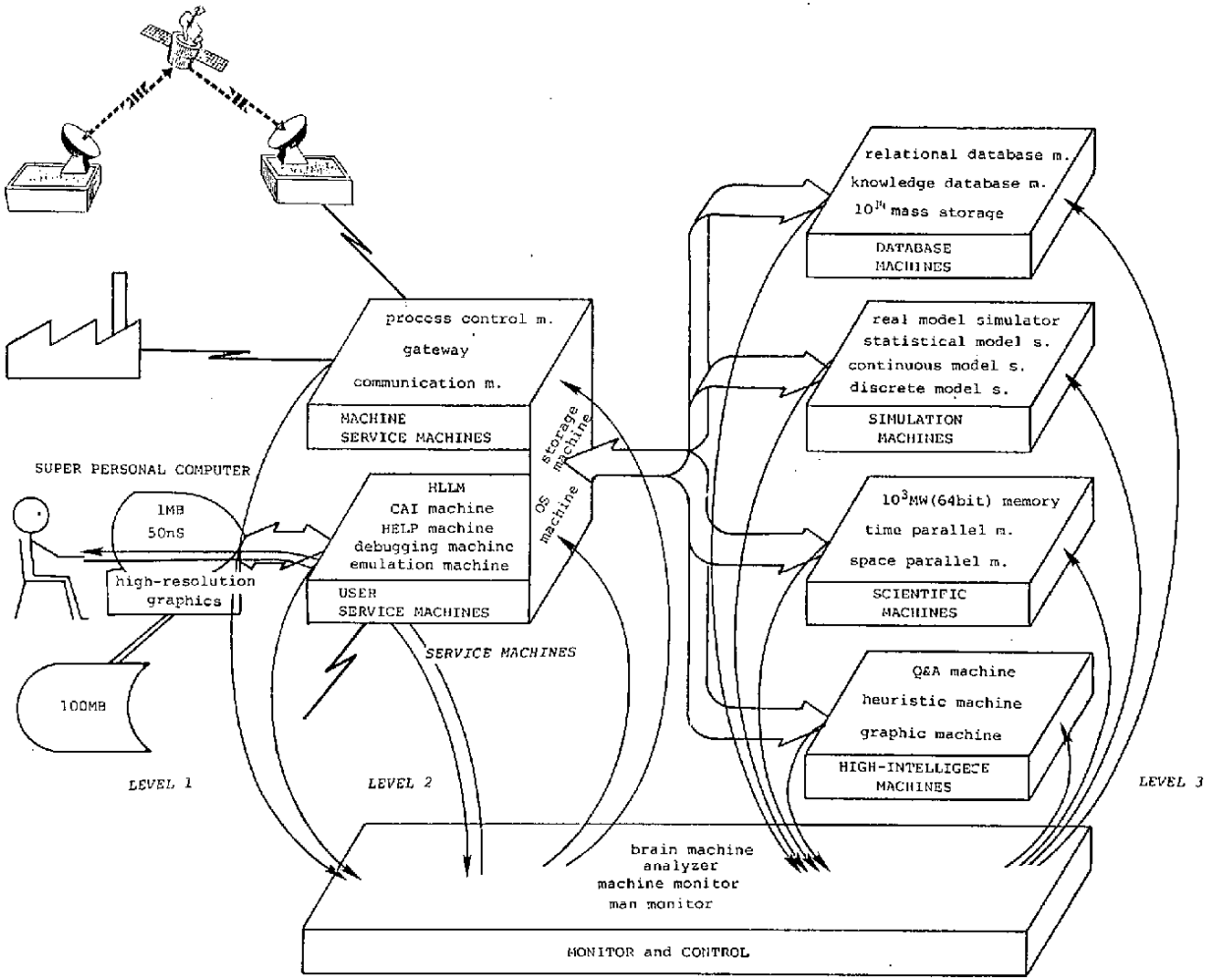


図 2.3.2　System Aの機能ブロック図

（APL, COBOL, PASCAL, GPSS, LISP, PL／I, ALGOL68, ADA, …）

・エミュレーションマシン

（第4世代との互換性，…）

・科学計算マシン

（現在の高速アレイプロセッサ並の能力）

・ファイルマシン

（10^{11} バイト程度のファイル管理）

・教育マシン（ユーザ教育用のマシン）

(b) マシンサービスマシン

・通信制御マシン

（10^{8} ビット／secのスループット，暗号化，高信頼性）

・変換マシン

（ファイル，プロトコルの相互変換）

(c) コントロールマシン

必要に応じてTMR※構成とする。

・スケジュールマシン

（システムリソースの全体管理）

・記憶制御マシン

（ケーパビリティ制御・アブストラクトデータ構造サポート）

・各種メディア制御マシン

（共用入出力機器の制御）

・通信制御マシン

（パーソナル・コンピュータ群の制御）

※ Triple Modular Redundancyの略。

(iii) 特殊目的超高機能マシン（レベル３）

(a) データベースマシン

(イ) 性　能：現在のデータベースの１００～１０００倍の処理能力

(ロ) 容　量：１０００Ｂバイト～１００,０００Ｂバイト（Ｂは10^9）

(ハ) 高度な問合せインタフェース

- 日本語／図表
- 連想型問合せ
- 集合型問合せ

(ニ) 各種データベースモデルのサポート

- 階層型データベース
- ネットワーク型データベース
- リレーショナルデータベース
- 可変データ項目数データベース

(ホ) パターンデータベースのサポート

- 画　像
- 図　形
- 音　声

(ヘ) 分散データベースサポート

(ト) きめの細かいセキュリティ機構

(チ) 超高信頼性

(b) シミュレーションマシン

(イ) 性　能：ＣＲＡＹ－１の１００～１０００倍のスピード

(ロ) 規　模：各種ニーズを十分に満足させる容量を有する。

(c) 科学計算マシン

(イ) 演算速度：10^3～10^4 MFLOPS（Million FLoating Operations Per Second），現在は10^1～10^2 MFLOPS

(ロ) 主記憶：10^2 ～10^3 Mワード（語長６４ビット以上）

(ハ) 処理能力：１０００×１０００×１０００程度のマトリックス演算が可能

(d) 高インテリジェンスマシン

(イ) 画像処理

・複雑な画像の大量データの〝解析〟，〝認識〟処理

(ロ) 数式処理

・並列処理に不向きな問題の超高速処理

(ハ) 知識処理

・専門家に対するコンサルテーション

・専門家の有能な助手機能

(ニ) 自然言語処理

・外国語／日本語の機械翻訳

(ホ) 文書作成処理

・文書，表，グラフ等の半自動作成

(ヘ) 人工知能向特殊処理

・発見的手法を利用した処理

(IV) モニタ・コントロール機構

動的モニタによる

・ 高信頼性の提供

・ 適応化の推進

を図る。

(a) 高度なデバッグサポート

(b) 高度なＨｅｌｐ 機能の提供

(c) ユーザ教育の高度化，自動化

(d) ロギングデータの収集

(e) チューニング・サポート

(f) 高信頼性の提供

(g) 最適システムの制御および動的再構成の実現

(h) RAS 機能の提供

(i) 無人運転，自動運用システムの実現

(4) システム構成とシステム利用形態の例

図2.3.2の機能イメージを具体化した一例を図2.3.3に示す。図では，「パーソナル・コンピュータ」が，光ファイバーリングネットワークにより接続されている。「ローカルメインコンピュータ」は機能的にはサービスマシンであり，したがって，パーソナル・コンピュータ間の通信制御，場合によっては，データ収集と，目的別にいくつかの機能を提供する。したがって構造的には機能モジュールの集合である。また「コミュニケーションプロセッサ」もサービスマシンの一種である。これはローカルメインコンピュータ間同士の通信，ローカルメインと中央に位置する特殊目的超高機能マシン間の通信を制御している。また，他のシステムとの交信制御も行なう。特殊目的超高機能マシンとしては，ここでは，データベースマシンと科学計算用スーパーマシンが用意されている。

注意すべきことは，System A とは中央集権制御型計算機ではないという点である。たまたま，特殊目的超高機能マシンは図の中央に位置しているが，これがパーソナル・コンピュータを制御しているわけではない。その位置は対等であり，それぞれのコンピュータは完全に分離され独立である。むしろ場合によっては，パーソナル・コンピュータのユーザからの指令により，それ以上のレベルのマシン群の動的再配置が行なわれ，ユーザ専用の要求適応型計算機と見えることすらある。

利用例の一例として，オフィスオートメーションシステムを取りあげ，System A の動作を見てみる。図2.3.4にこの場合の構成例を示す。ここで想定している組織体は，製造プラント，研究所，ヘッドオフィス，サテライトオフィス，海外サテライトオフィスから成る。それぞれのオフィスには，ロ

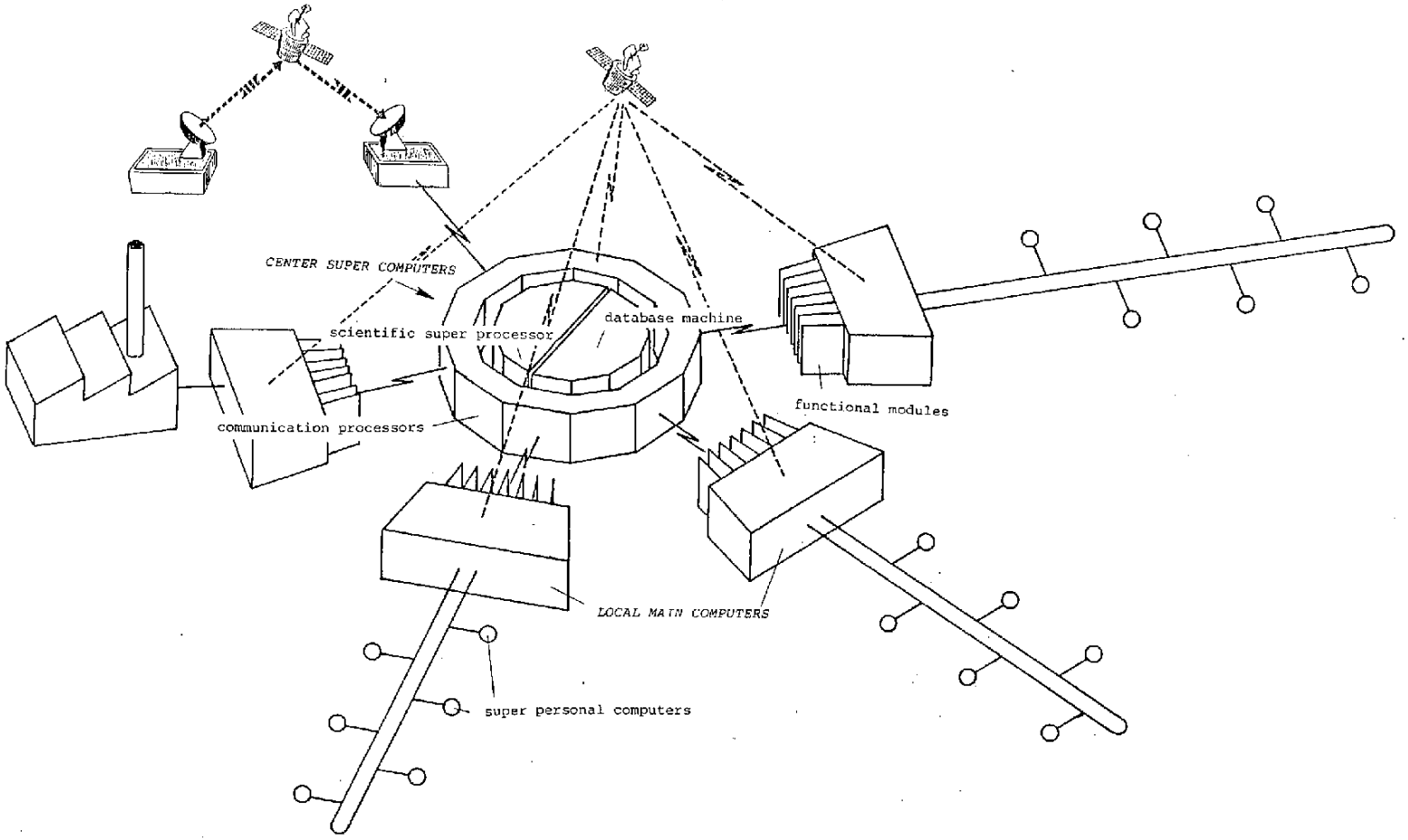


図 2.3.3　Ｓｙｓｔｅｍ　Ａの構成例

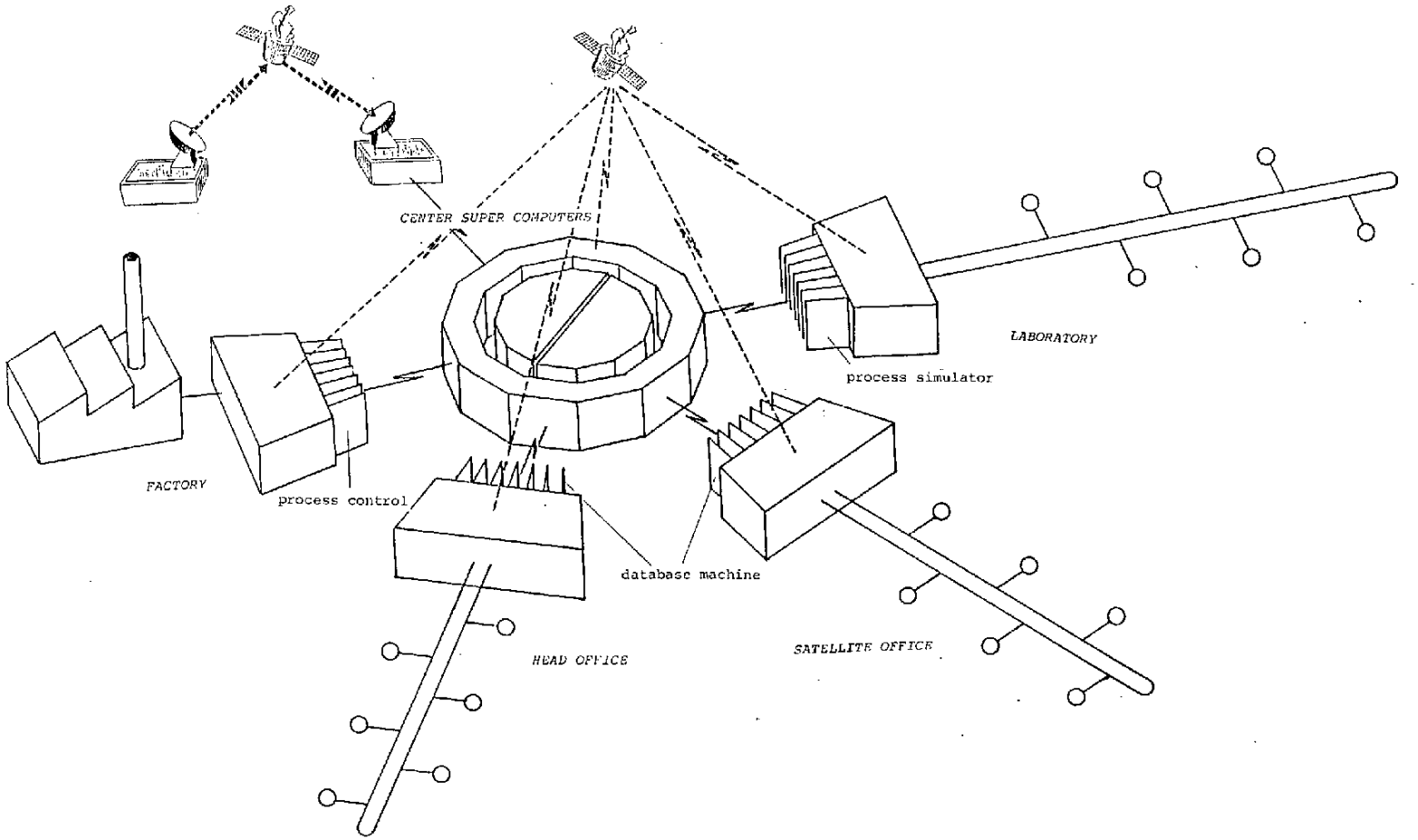


図 2.3.4　System A の利用例——オフィス オートメーション——

ーカルメインコンピュータが設置されているがその規模能力はオフィスにより異なる。特に，機能的には，まったく異なり，それぞれのオフィスに適応している。そこに接続されているパーソナル・コンピュータは更に千差万別であり，一オフィス内でも仕事により機能は異なる。

ここで，データの流れの一例を見るなら次のようになろう。※（図2.3.5 参照）

① サテライトオフィスは営業所であり，ここで受注が獲得されるとそのデータは一時ローカルサテライトのデータベースに格納される。これはサテライトオフィス内でも使われるが，

② さらに，センターデータベースに格納され，

③ ヘッドオフィスへ報告される。

④ また，このデータは自動的に工場へ送られ，

⑤ 工場でこのデータは自動的に製造計画に組込まれ，これに従ってプロセスコンピュータを動かし製造を開始する。

このように，すべてのオフィスで行なう処理は一貫して，かつ分散制御のもとで，総合的に行なうことが可能である。さらにトータル・システムとしての利点を生かすことにより，

⑥ 研究所で新プラント構築の研究をする場合，データとして実際に稼動しているシステムのデータの利用が可能で，

⑦ そのデータをもとに研究所ローカルコンピュータ内のプロセスシミュレータを動かし，高度な研究を可能とする。

以上は，ほんの一例であるが，このようにSystem Aでは，

- データの有機的利用を可能とし，
- 完全分散による適応型高機能処理を可能とする。

(5) System A 実現上の問題点

第5世代コンピュータSystem A はユーザ側から機能的に見れば，「問題指向型」であり「使い易く」「高信頼性」であるという事になるが，ア一

※ 以下の項目番号は図2.3.5のデータの流れの番号と一致している。

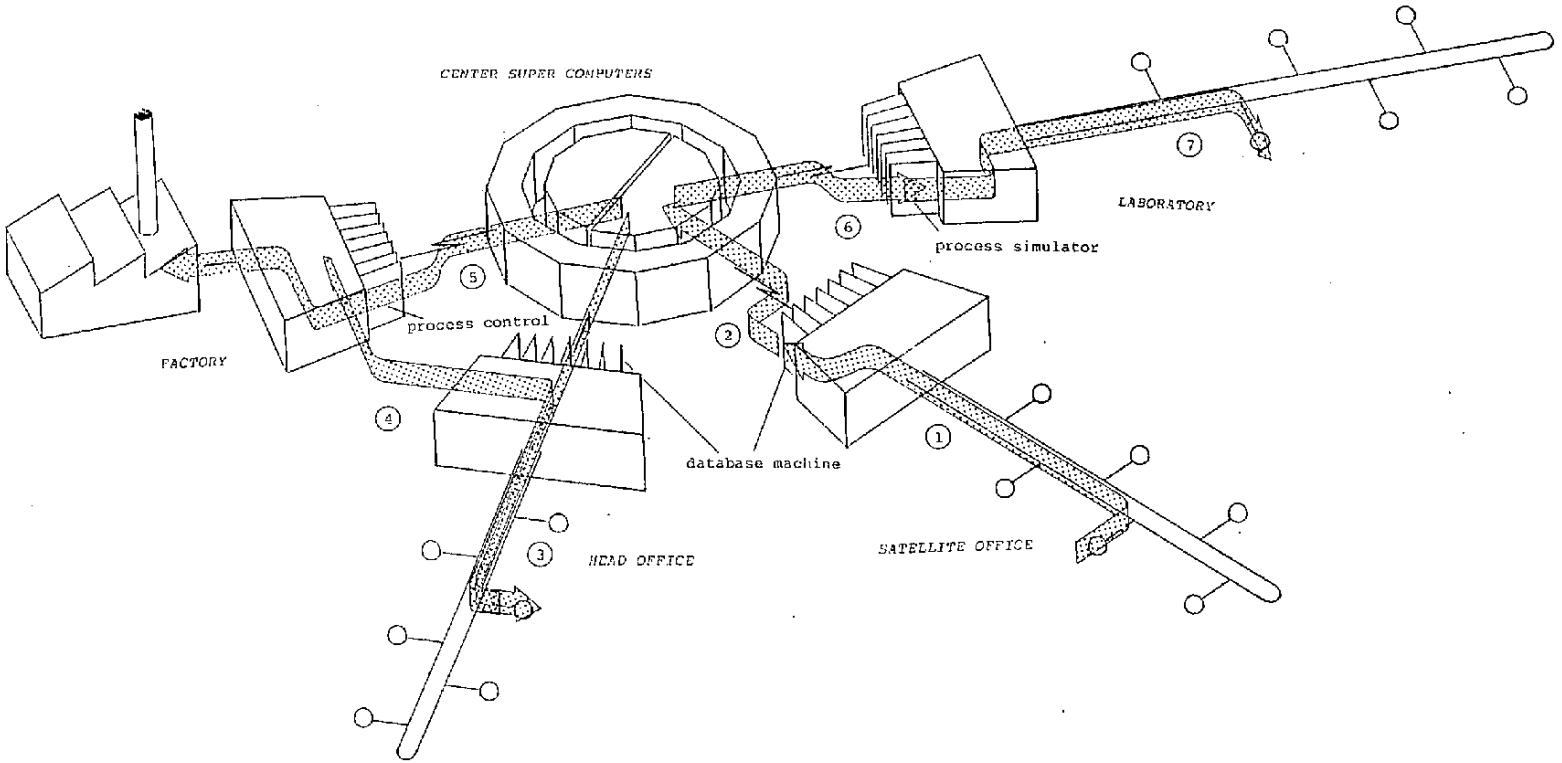


図 2.3.5　オフィス・オートメーションにおけるデータの流れの例

キテクチャ的に見た場合の特徴は次のようになろう。

① 「目的指向性」「使い易さ」のほとんどの要求はパーソナル・コンピュータで処理出来る。パーソナル・コンピュータは大量に造られる事を前提としているので既存アーキテクチャの延長上にあると考えられ，

・ 同一チップアーキテクチャ

であり，適応性はマイクロプログラミング技術により解決される。

また，オペレーティング・システムとしては，

・ 単一ユーザ用

で良く，それよりソフトウェア的には使い易さを重視したアーキテクチャが重要になると思われる。

② また，システム全体として，階層型アーキテクチャという事が重要になり，パーソナル・コンピュータで処理出来ないものは，汎用性の強い特殊目的計算機という形でまとめられ処理されよう。「サービスマシン」の位置付けは，「パーソナル・コンピュータ」の1ランク上のマシンとして定義される。そしてここでは，パーソナル・コンピュータで処理出来ない問題のほとんどを処理し，さらに，パーソナル・コンピュータ間同士の通信，プラント機械との通信，他システムとの通信を行なうクラスターマシン的特徴も備えよう。また，既存システムとの互換性もここで吸収されなければならない。これも大きな特徴である。

アーキテクチャを構造面から見れば，

・ 構造は「機能分散」されたマルチユーザ用計算機であり，

・ それぞれの機能モジュールは現在の計算機の進化した，一部非ノイマン機能を含む構造をとると思われる。

ここで重要な事は，どのようにして機能モジュール分けをするかという点と，いかにそのモジュールを機能的に高機能化するかにかかろう。そして，このためには，現在考えられているあらゆるアーキテクチャ技術が活用される可能性がある。

また既存技術とのつながりとしては，一つの機能モジュール内で新旧アーキテクチャを吸収するのではなく，

・　新旧アーキテクチャを明確に分離して，それを接続する技術で乗りきる事が重要と思われる。具体的には旧マシンが一つのモジュールとしてこのレベルで付加される形がとられると思われる。さらに，ソフトウェア的には，このマシンはマルチユーザにより共有されるので，

・マルチユーザ用構造をとる必要があり，

それに基づいて，

・　ケーパビリティ，セキュリティ機能が重要な開発ポイントとなると思われる。

③　更に，問題としては，少ないと思われるが処理速度，容量的に非常に高性能なマシンは必須で，その機能要求項目は現在のそれと比べるならば，2桁以上の開きを持ち，そのためアーキテクチャ的には，

・　現在の計算機構造とはかなり異なる構造をとる可能性が強く，

また構成素子も，

・　現在まったく使われていない素子が使われる可能性が大きい。

さらにこのレベルでは，

・　信頼性のファクターが最重要視される。

ソフトウェア的には，

・　使われ方は共有であるが，マシンによっては同時使用は許さず，能力的には単機能色の強いオペレーティングシステムが要求されよう。

④　また，システム全体の最適制御，高信頼性の保障ならびにユーザに対してさらに高機能なサービスを提供するため「動的な」システム情報に基づく再構成機能は重要な要素となろう。機能的には，「モニタ・コントロール機能」は単一と考えられるが，実現に当っては，これは

・　各レベルに分散実装

されると思われる。ここで重要な点は，

・　「何をモニタ」し，それを「どのようにフィードバックするのか」を明確にする事にある。また，この機能は，

・　アーキテクチャ構築上，初めから組込まれている必要があり，そうでないと十分その機能を達成するのは難しいと思われる。

以上，第5世代コンピュータシステムSystem Aにつきそのイメージをまとめた。総括するならば，System Aとは，能力的には現在のそれと比べ，はるかに高機能であり，現在の技術レベルからすれば単独のシステムとしてはきわめて「野心的」であり，また「夢のような」ものであろう。
しかし，アーキテクチャ構築者から見れば，構築の鍵のほとんどは現在の技術の延長上にあると考えられる。新しいシステムが夢のような技術で造られているという事は考えにくい。技術とは流れを持っており，新しいシステムもその延長上にある。我々がやらなければならないのは，いかに新しいシステムへ向けてその流れを変えていくかにある。このような意味で，その実現に当っては現在の計算機とは構造の異なる計算機の追求も重要であるが，同じく現在のアーキテクチャの基本技術を整理し直し，流れを持ったシステムアーキテクチャの構築が重要であると思える。

2. 4　新技術と研究課題

2.4.1　パーソナル・コンピュータ

(1) 概　要

2.3.2項で述べたように，半導体ＬＳＩ 技術の急速な進歩は計算機を次第にパーソナル化させつつある。実際に，今までにもメモリ素子は２倍／年，マイクロプロセッサ素子は２倍／２年の集積度の改善がなされている。

１９８０年代にもほぼ同じ技術進歩が期待できるといわれ，ここ１０年間で集積度は１００倍，価格は１／２０程度になると見られている。その結果，現在の大型計算機の処理装置はミニコンピュータＣＰＵ程度の感覚でとらえることができる。

以下に，パーソナル・コンピュータが備える機能イメージと技術的な問題点について述べる。

(2) 機能イメージ

パーソナル・コンピュータに期待できる機能を列挙すれば次のとおりである。

(i) 処理装置の機能は現在の大型計算機とほぼ同等で，マシン・サイクル時間５０ｎｓ，主メモリ１ＭＢを実装する。

(ii) 多様されている高級プログラム言語は利用できる。

(iii) 強力なエディタをもち，日本語が取扱える。

(iv) ユーザ・マイクロプログラム方式でエミュレータを実装できる。また，ファームウエアで機能の専用化もできる。

(v) 高分解能ＣＲＴ ディスプレイを備える。

(vi) １００ＭＢ程度のパーソナル磁気バブル・ファイルを備える。

(vii) ハードコピー装置などの入出力機器を接続できる。

(viii) 通信回線制御装置をもち，計算機ネットワークを介して他のリソースをアクセスできる。

(ix) 単一ユーザＯＳの下で動く。

(3) マンマシン・インタフェースとしての役割

パーソナル・コンピュータは次に示すマンマシン・インタフェースとしての重要な役割をも含んでいると考える。

(i) 比較的単純な会話型グラフィック処理を行なう。

(ii) 文字・図形・画像・音声などのパターン処理のための前処理・後処理を行なう。

(iii) 高機能RPG，問合わせ言語などで共有データベースをアクセスできる。

(iv) コンピュータ・ネットワークを介して相互に通信ができる。

(4) 基礎技術

パーソナル・コンピュータの実現に必要な重要な技術を列挙すれば，

(i) 超ＬＳＩ 技術の確立

(ii) 高性能３２／６４ビット・マイクロプロセッサならびに１～２Ｍビット／チップのメモリ素子の開発

(iii) １～１０ＭＢ／チップの磁気バブル素子の開発

(iv) 光ファイバ高速ネットワークまたは同軸ネットワークの構築

(v) パターン情報の伝送技術

(vi) 使い易いマンマシン・インタフェースの設計，日本語入力，限定された範囲内の音声認識を可能とする技術

(vii) 高分解能（６４×６４ビット／字）カラーＣＲＴの開発

(viii) ユーザの要望に適合するシステム作りの技術などであろう。

などであろう。

2.4.2 サービス・マシンとサブシステム・インタフェース

(1) 概　要

2.3節に述べられているように，System Aの中央サイトコンピュータは複数の機能マシンを結合して機能分散型に構成されており，この各機能マシンをサービスマシンと呼ぶ。サービスマシンの主な目的は遠隔地に存在するパーソナルコンピュータ（即ち，そのユーザ）の要求に応じてパーソナルコンピュー

タが持つものより少くも1桁以上高い演算，記憶，教育（情報提供）等のサービスを提供することと，他の計算機システムやプロセス制御系等と通信回線で結合されてその管理を行なうこと，の2つに分けられる。ここでは前者用のマシンをユーザサービスマシン群（ＵＳＭ′ｓ）と呼び，後者の機能を果すものをマシンサービスマシン群（MSM′s）と呼ぶことにする。

次項以下に述べられるデータベースマシン，シミュレーションマシン，科学計算マシン，高インテリジェンスマシンもユーザ側から見ればＵＳＭの一種でしかないが，これらは特に重要と考えられる機能なので，ＵＳＭとして提供されるものより更に1桁以上（パーソナルコンピュータからは2～3桁）機能が高いものを別に用意することにしているのである。

(2) 構　　成

サービスマシンは上記のＵＳＭ群，ＭＳＭ群及びこれら全体のスケジュール等の制御を行なうコントロールマシン（ＣＭ）の3部分から構成され，各部分には次のような機能マシンが用意される。

A) ユーザサービスマシン

① 各種高級言語マシン群

パーソナルコンピュータもファームウエアによる間接型高級言語マシンとして動作するが，ここに用意される高級言語マシンは実行速度が1桁以上高速であるか，あるいは言語仕様が高度でパーソナルコンピュータではサポートしきれない言語に対するものである。サポートする具体的言語としては，ＡＰＬ，Ｃｏｂｏｌ，Ｐａｓｃａｌ，ＧＰＳＳ等のシミュレーション言語，Ｌｉｓｐ，ＰＬ／Ｉ及び第2のカテゴリーとしてAlgol 68，Ada等が考えられるが，この他にも１９９０年までにある程度の普及を見た言語ができればこれに付加されることになろう。

このような高級言語マシンの実現手段としては機能レベルからしてファームウェアだけでは不可能で，専用のハードウェア及び高度並列処理技術が必要である。

② エミュレーションマシン

エミュレーションマシンは第4世代までに開発されたプログラムを実行するためのマシンであって，互換性を保つために含められる。代表的機種に対して用意され，ユニバーサルホストマシンを用いて1種のハードウェアで多種の機種をサポートできることが望ましいが，第4世代までの機種そのものを直接接続することも考えられる。

③ 科学計算マシン

科学技術計算をサポートするアレイプロセッサ，連想プロセッサ，信号処理プロセッサ等から成る。実行速度はほぼ現在の高速アレイプロセッサ程度で10M～100MFLOPS程度のものである。

④ ファイルマシン

ファイル管理用のマシンである。機能的には2.4.3項のデータベースマシンと変らないが，容量は10^{11}バイト程度とより小さい。

⑤ 教育マシン

ユーザに対してデバッギングやシステムリソースの使い方その他の教育をサポートするマシン（群）で，機能的には第4世代までに実施されているものより一段と高いインテリジェンスを備えている。ただし，デバッギングについては単独のマシンとして実施されずに，個々の機能マシンに内蔵される形で実現されることになろう。

B) マシンサービスマシン

① 通信制御マシン

他の同種のシステム（衛星経由の世界的ネットワークを含む）やプロセス制御系等の特殊システムとの通信制御を行なうマシンで，10^{8}ビット／s程度のスループットを持つ。また，暗号による通信のための機能を内蔵し，信頼性も1桁以上向上したものである。

② 変換マシン

他システムとのインタフェースとしてファイルのフォーマット，プロト

コル等システム間で異なる形式の相互変換を実行する。

C) コントロールマシン

システムの全体管理とＵＳＭとＭＳＭが共通的に使用するリソースの管理を行なう部分である。従って，ＵＳＭ群とＭＳＭ群が各システムの要求に応じて構成されるのに反し，ＣＭ群はどのシステムでも基本的には同じ形になる。また，この部分はシステム全体の中枢部であるから，特に高信頼性が必要で，必要に応じＴＭＲ構成が取られる。

① スケジュールマシン

ＵＳＭ's 及びＭＳＭ's のスケジュールを初めとしてシステムリソースの全体管理を行なう，また，モニタ・コントロール機構部とも通信して動的に最適管理を実施する。

② 記憶制御マシン

ＵＳＭが主に共通に利用する記憶とその制御マシンである。単なる記憶ではなく，ケーパビリティ制御及び各種アブストラクトデータ構造の処理機能を内蔵する。

③ 各種メディア制御マシン

各端末にはおけないので，共通に使用される特殊な入出力機器の制御用マシンである。いずれも単なる入出力制御装置ではなく，高い論理機能を有する高レベルのインタフェースによって他のＵＳＭやＭＳＭに接続される。機器としては高度の機能を持つグラフィックシステム，レーザプリンタ等大量の入出力データを処理する機器が対象となる。

④ 通信制御マシン

ＭＳＭとして述べたものと類似であるが，接続対象はこの系に属するパーソナルコンピュータ群であるから，スループットはより小さくてよい。反面ローカルネット等いろいろな仕様のものが考えられ，機能に多様性のあるものとなる。

(3) サブシステム間インタフェースと言語

上記のようにSystem Aは幾つかのサブシステム（マシン）の集合として構成されているが，このような場合，問題となるのは各々のサブシステムにどんな機能を持たせるかということと，各サブシステム間の情報授受形態である。これは即ち，サブシステムそれぞれの間の通信規約を定めることであり，サブシステム間の言語を定めることと見ることができる。従って，コンピュータシステムのアーキテクチャを定めることは，サブシステム分割というレベルで見れば，サブシステム間言語のセットを定めることに他ならない。各サブシステムをその外から見れば，その機能はサブシステム間言語で記述される事柄であり，それが如何に実現されているかについて他のサブシステムは知る必要がない。

例えば，データベースマシンの場合は，外部インタフェースとしてはVSAMのようなアクセス法を設定することもできるし，ＳＱＬ言語やＤＬ／Ｉ言語を設定することも考えられる。高速演算マシンに対しては配列操作文を含むようなＦＯＲＴＲＡＮをインタフェース言語として設定できようし，高級言語マシンやメディア制御マシンに対してはＰａｓｃａｌのような高級言語に近い形式で記述したジョブ制御言語が考えられるし，マシンサービスマシンに対してはVTAMをより高級化した言語が考えられる。

各サブシステムは又，幾つかのモジュールに分解することができるが，これについてはシステムのサブシステムへの分割と等価であり，何階かの階層を考えることができよう。

システムのハードウェアとしての実現は従って，各言語を内部コード（ビットパタン）に割り付け，言語で記述されたブロック（コマンド又はプログラム）を他サブシステム間で授受する機構をハードウェアに落とし，各サブシステムのモジュールにハードウェアを割付けることを実行することである。Ｓｙｓｔｅｍ Ａ のような大規模システムを実現してゆく場合，今後はこのような見方に立ってのシステム検討からシステムの性能を評価し，システムの

実現迄のステップを，システムに要求される性能・機能と密接に関連付けて実行してゆく手法の開発が必要であろう。この手法は正に，ソフトウェア工学に於ける構造化プログラミングであり，その意味から言えばハードウェアの設計とソフトウェアの設計は全く同一視することができよう。

(4) 技術的課題

前項では，System Aにおける各機能マシン間インタフェースの重要性について述べたが，以下では，各サービスマシンの実現において，考慮すべき技術的課題について若干のコメントを述べておきたい。まず，一般的にいうと，次項以下に述べられる4種の超高性能マシンを別にすれば，各サービスマシンの開発には今までに全く存在しないような革新的技術を開発するということよりは，むしろ基本的には既にある技術を高度化し，また，効率よく組合せることが重要であると考えられる。このことは必ずしも実現が容易であることを意味するものではなく，多くの地道な努力にもとづくゆるやかな（連続的な）改良という形をとるであろうということである。

以下に，重要と思われる主な技術を列挙しておく。

① 高級言語マシン作成技術

- 言語特性の把握
- 効率のよいユニバーサルホストプロセッサ
- 効率的専用ハードウェアの開発と効果的組込
- 言語の特性に合った並列処理方式
- ファームウェアコンパイラ

② エミュレーションマシン

- 効率のよいユニバーサルホストプロセッサ
- 各種エミュレータ
- 旧型機の接続技術

③ 科学計算マシン

基本的には2.4.5項と同じ。但し，性能／コスト比を高めるような設計が必要である。

④ ファイルプロセッサ

基本的には2.4.3項と同じ。但し，性能／コスト比を高めることとＤＢ方式の多様性に対処する設計が必要である。

⑤ 教育マシン

高度のインテリジェンスを持たせる方法の開発，アーキテクチャやハードウェアは大きな問題はないであろう。

⑥ 通信制御マシン

・高速化，スループット拡大

・暗号化技術

・高信頼化のための諸技術

⑦ スケジュールマシン

多数の高機能マシンをいかに効率よく制御するかは大きな問題である。この部分は従来ＯＳが処理していたわけであるが，本システムでは基本的フィロソフィーを変革し，新しく打ち立てる必要があろう。ハードウェアはあまり大きなネックにはなるまいが，新フィロソフィーに対する新アーキテクチャを検討する必要の生ずる可能性もある。

⑧ 記憶制御マシン

・抽象データ構造を効率的に処理するハードウェア方式

・ロジックインメモリの開発

・ケーパビリティ制御ハードウェア

⑨ メディアマシン

・高速回路（スループット大）

・処理機能よりも他のマシンとの高レベルインタフェースの確立等が重要である。

⑩ 統合制御と結合方式

スケジュールマシンと共にシステムの死命を制するものである。ハードウェア的には高速バス（～１０ｎｓ）の開発が必須であるが，制御方式のフィロソフィーの確立が最も重要であろう。

⑪ 構成の記述

各機能マシンの組合せによる全体構成を記述する方式と言語。またその記述から，実際のシステムの統合を生成し，管理ソフトウェアを生成する方式。

2.4.3 データベース・マシン

(1) データベースマシンの必要性

今後の大規模データベース・システムの研究課題はつぎの３点である。[1][2][3]

1) 大規模データベースに対する高速検索

2) より知能レベルの高いデータベース・システムの実現

3) 人間とインタフェースのよいデータベース・システムの実現

1)については自明のことではあるが，特に今後はオフィスオートメーション・システムの発展にともない，データベース・システムの利用者の数と利用頻度が飛躍的に増大すると予想され，データベース・システムは，現在の２桁以上の数のデータベース問合せに対処できなくてはならないと予想されている。[2]

しかも，データベース・システムの機能が高レベル化して行くので，更に大きな処理能力が必要になってくる。

また，データベースのアクセス・インタフェースの上にグラフィック・システムを構築する方式（リレーショナル・データベースを用いるＩＢＭ社のＰＢＳ，ＩＮＧＲＥＳシステム等）も今後利用されると予想される。この場合にもデータベース・システムに高い性能が要求される。

2）については，データの持つ意味情報もデータベースに格納し，ユーザに強力なオペレーションを提供したり，信頼度の高いデータベースをサポートする等がある。例えば，個々のデータ項目の制約条件をデータベースに格納したり，データ項目間の相互関係をデータベースに格納しておく方法がある。

3）はデータベースの問合せのインタフェースとしての問合せ言語，データベースのモデル，セキュリティ機能のレベル向上等である。

問合せ言語としては日本では日本語を使うのが最も自然である。また，IBM社のQBEのようにディスプレイ画面上でテーブルを表示し，所要の項目を選択させる方式も利用されて行くであろう。

データベースのモデルについては，従来のモデルの他に,最近はリレーショナル・モデルのデータベースも実用規模のものがある。

従来のデータベース・システムは，何んと言ってもEDP用のデータの格納場所としてのデータベース・システムであった訳で，データの構造の変更が頻繁ではなかった。しかし，今後のオフィスオートメーション・システムで新たに必要になるデータベースは格納されるデータが文字情報とか文章情報を主体とし，データの構造も頻繁に変わるものとなろう。この意味で新しいデータベースのモデルが必要になってくるし，データアクセスの高速化技術も必要になる。

データベース・システムのセキュリティ機能の充実にも，力を入れる必要がある。

オフィスオートメーション・システムでデータベースの中に日常業務で使用する大部分の文書ファイルとデータが格納されて，オフィスの誰もがオフィスオートメーション・システムの体系で日常の業務を行う時点ではデータベースのセキュリティの機能が充実していなくてはならならない。

オフィスの事務作業者が見てよいファイルもあるし管理職のみがアクセスできるファイルもある。オフィスの組織形態にあったセキュリティ機能を実現できなくてはならない。

この目的にはデータベースのあるまとまりごと，例えば一つの部門のデータ

とか，役職が課長以上の社員のといったまとまりごとで，データベースのアクセス権を設定できなくてはならない。

ここで注意しなくてはならないのは，セキュリティの機能を詳細なレベルで実現しようとすると，データベース処理側に大きなパフォーマンスが必要となることである。例えば，レコード単位でデータ項目の内容に従ってアクセス権を制御しようとする場合がこれである。

(2) 方　　式

図2.4.1に示すように半導体の二次記憶装置として，アソシアティブな論理を持ったものが１９９０年代のコンピュータには考えられる。

このマシンを実現する条件は超ＬＳＩである。例えば，１ＢＢ*の半導体の二次記憶に，アソシアティブ機能がつく構成が考えられる。

大容量記憶（ＭＳＳ）から二次記憶にステージングされたデータは二次記憶上でアソシアティブなアクセスがされる。

アクセスされたデータは，一般に集合であり，データベース問合せ条件を満たすように集合演算が行われた後，処理プロセッサに転送される。

セキュリティの検査もデータベース問合せ条件と同様に処理できる。

ＭＳＳは光記憶または磁気記憶を用いることになろう。

(3) 研究課題

１９９０年代のデータベースマシンのイメージは以下のようなものとなる。

1）性能：現在のデータベースの１００～1,０００倍の処理能力

2）容量：1,０００Ｂバイト～１００,０００Ｂバイト（Ｂは10^9バイト）

3）ハイレベルの問合せインタフェース

日本語と図表を用いてデータベースに対して問合せができる。

論理和とか論理積を用いる連想型の問合せと，集合型の問合せが標準機能として利用できる。

大規模データベースに対する問合せ時間を節約する目的で，Ｆｕｚｚｙのデータベース問合せを可能とする。

＊　Billion Bytesの略

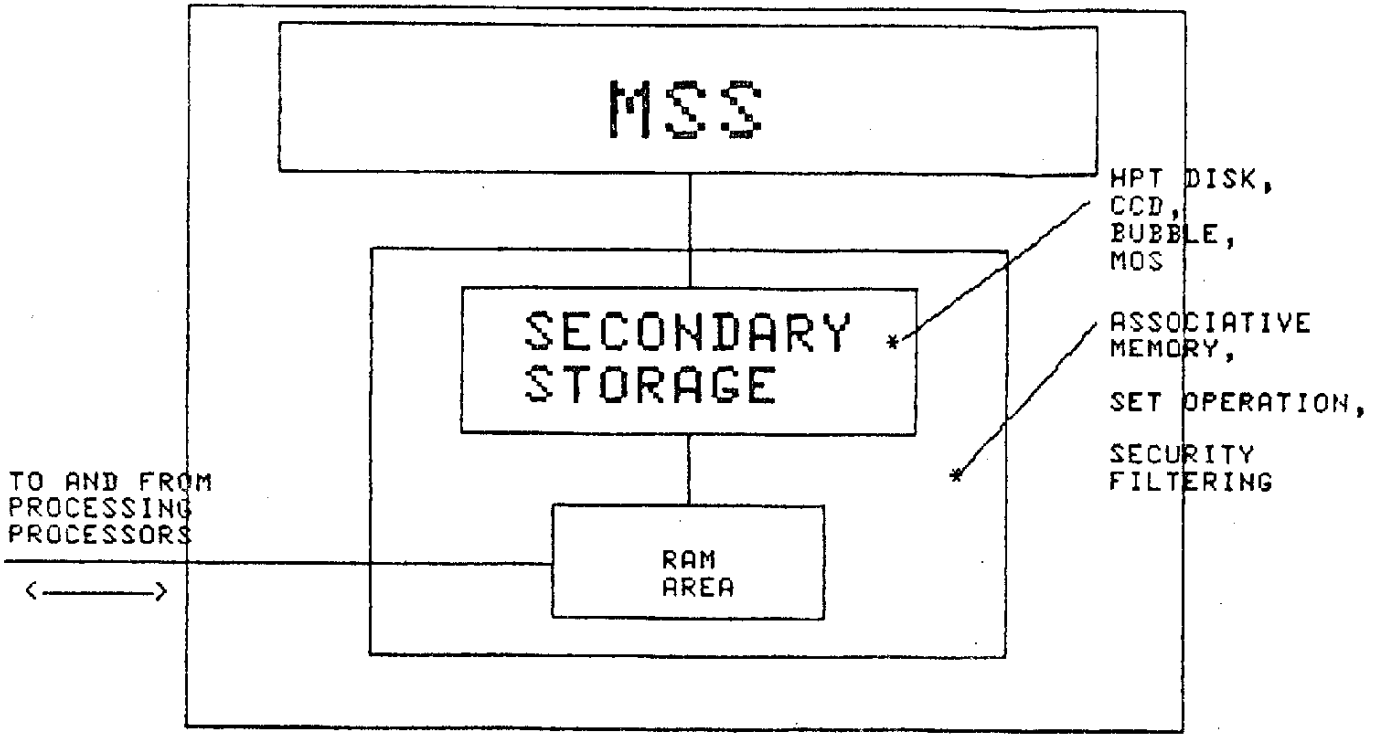


図 2.4.1　　D a t a b a s e　M a c h i n e

4）各種のデータベース・モデルをサポート

従来の階層型，ネットワーク型と共にリレーショナルデータベースをサポートする。更にオフィスオートメーション用の可変データ項目数のデータベースもサポートする。

5）パターン・データベースのサポート

画像データとか図形データのイメージ・データと，音声データからなるパターン・データも，このデータベースマシンで統一的に扱える。

6）分散データベースをサポート

データベースマシンを複数含むコンピュータネットワークとか，データベースマシンとパーソナルコンピュータとのネットワークに於ける「データの分散」の機能を持ち，ユーザは自分のデータがシステムの中のどの位置にあるかを意識しなくてもよいようになる。

7）ユーザの要求に応じた細かさでセキュリティの指定が可能

最も詳細なレベルとしてはレコード毎にそのフィールドの内容に従ってアクセス権を制御できる。

8）超高信頼度データベースシステム

単にシステムの誤りに対する考慮だけでなく，ユーザの指令の誤りに対するリカバリも考慮する。この配慮は特にオフィスオートメーション・システムで，エンドユーザがデータベースを利用するときに，大いに役立つと予想する。

上記のデータベースマシンを実現するための研究課題を以下に示す。

1）記憶域の階層管理：ステージング

大容量記憶装置（ＭＳＳ），ディスク装置，半導体の二次記憶装置等の間のステージング機能を検討する必要がある。

- データの使用頻度にもとずきデータを最適配置するアルゴリズム
- データのリカバリ技術（２重書き等）

2) 連想型アクセス技術

今後のデータベースでは，単一キーによるアクセスだけができるというのでは十分ではなく，多数のフィールドがサーチの対象になるので，連想型のアクセスが多く使われる。そこで，次の技術が必要になる。

・ 高速アソシアティブ・メモリ

Head-Per-Track ディスク，半導体のアソシアティブ・メモリ

特に半導体のアソシアティブ・メモリは超ＬＳＩ技術で高性能のものが期待できる。

3) 集合オペレーション

データベース問合せを満たすデータを求める過程で集合オペレーションが必要になるが，この操作を高速に行う必要がある。このオペレーションを記憶域の階層のどのレベルで行うかを定めたり，マークビットの利用法等研究する必要がある。

4) 超ＬＳＩの利用

2)のアソシアティブ・メモリと3) の集合オペレーションを実現するのに，Logic in Memoryを用いるというように，データベースマシンの中には超ＬＳＩを用いて性能を向上できる部分が多い。どの部分を超ＬＳＩ化するかを具体的に研究する必要がある。

5) データフローマシンとの関係

データベースマシンの高速化には，データフローの技術が利用できる。例えば，オハイオ大学のデータベース・コンピュータ（ＤＢＣ）はキーワード処理用とかインデックス処理用等の複数のプロセッサを用意し，その間をデータが流れるような構成をとっている。

6) Storage Processor との関係

各種の演算・操作を処理プロセッサで行うのではなく，データの格納されている近くで行ってしまう「Storage Processor」というアイデアがあるが，データベースマシンをこの観点からも検討する必要がある。

7) セキュリティ機能のサポート

レコード単位のセキュリティも実用になるように，セキュリティ機能の高速実行をはかる。このための検討は2)，3)，4)の検討とほぼ同じになると予想される。

なお，データベースのアーキテクチャ技術については第4.10節に述べてある。

参考文献

(1) Proc. 3rd Very Large Data Bases, 1977

(2) D. K. Hsiao and S. E. Madnick, "Data Base Machine Architecture in the Context of Information Technology Evolution", 3rd Very Large Data Bases, pp. 63-84, 1977

(3) G. A. Champine, "Current Trends in Data Base Systems", IEEE Computer, Vol. 12, No. 5, pp. 27-41, 1979

2.4.4 シミュレーション・マシン

(1) シミュレーションとは

シミュレーションとは，部分的には解析的に解けても，全体の動作を数学的に表わせない複合化したシステムをモデル化し，計算機を用いてモデルの動作を再現し，その挙動を観察することにより，もともとのシステムについての有効な知見を得る手法である。従来のシミュレーション手法は，大きく次の3種に分けられる。

- 連続系シミュレーション ： システムの時間的経過に伴う変化が連続的なもの，たとえば，流体関係，構造解析などを扱うシミュレーション。
- 離散系シミュレーション ： システムの中で起る種々の活動の開始，終了の2つの事象に注目し，その時瞬間的に変化が起きたと仮定してもよいものを扱うシミュレーション。

・ 確率系シミュレーション ： 社会モデル，経済モデルなど，上記のいずれの形態でも表わせないシステムを扱うシミュレーション。

(2) 今後のシミュレーション・ニーズ

技術の進歩と社会生活の複雑化に伴い，コンピュータを利用したシミュレーションのニーズは増大しつつあり，次に述べる各種アプリケーションが考えられてきている。

・ 大規模実験設備に代るコンピュータ・シミュレーション ： 風洞，水槽，土木システムなど大規模実験設備を作ってシミュレーションを行うのには，金額的，時間的負担が大きいものを，計算機を利用してシミュレーション・モデルを作り代用する。

・ 設計援助システム（ＣＡＤ）用シミュレーション ： 各種設計に際し設計対象システムの動作を計算機でシミュレートし，正しい設計を行えるようにする援助システム。

・ 社会・経済・環境などのシミュレーション ： 人間生活に密着したテーマを扱い，それの実生活に与える影響を事前にシミュレートし，具体的処置に伴う問題発生を回避するための意志決定援助システム。

・ 経営問題のシミュレーション ： 企業における各種意志決定のために，経済モデル，気象モデル，消費者行動予測モデル，経営管理モデルなどの各種モデルを連結し，総合的シミュレーションを行う。

・ 組込み型シミュレーション ： システムの中へ組み込んでおき，個々の操作または動作のシステムへ与える影響をリアルタイムでシミュレートし，問題のないことを確認してから実際の操作または動作を行うようにするためのEmbedded Online Simulation の実現。

(3) シミュレーションにおける課題

上述の各種シミュレーションは，最初に述べた３種のシミュレーション手法のいずれかあるいはそれらの組合せにより解かれることになるが，そこで解決されなければならない課題には次のものがある。

・ シミュレーション言語の確立 ： 個別のアプリケーションに対応する専用または汎用のシミュレーション言語の確立が第一に求められる。

・ シミュレーションのための特殊ハードウェアまたはアーキテクチャの確立 ： 上記シミュレーション言語により書かれたシミュレーション・モデルの実行を高速に行い，適正なる応答速度を保つことが必要とされる。このための専用のハードウェアおよび高速化ニーズを満たすアーキテクチャの確立が課題となっている。

・ シミュレーション経過または結果の表示技術の確立 ： シミュレーションの状況をリアルタイムに表示するグラフィック表示技術の確立が求められる。 たとえば，n次元の動的推移表示を，シミュレーション速度に遅れることなく，大容量のカラー表示で行うなどの技術である。

・ シミュレーション結果の正当性予測手法の確立 ： シミュレーション動作が，実際のシステムの動きを正しく反映していることを検証し，正しくなければモデルの修正を行うなどの自己学習手法を含んで正当性予測手法の確立が必要となる。

(4) シミュレーション・マシン

以上述べた各種課題を解決し，ニーズの項で述べた大規模シミュレーションなどに対応して行くためには，単に超高速科学計算マシンを実現するだけでは不十分と考えられシミュレーションのための専用マシンが求められよう。

たとえば，風洞シミュレーションのためのNavier-Stokes Equation Solver 専用マシン，原子炉シミュレーションのための専用マシン，気象モデル・シミュレーションの専用マシンなどの物理モデル・シミュレーションなど，現在の最高速のマシンの１０倍程度の演算能力では，不十分であるニーズは既に存在している。

このため，シミュレーション・マシンではCRAY-1の１００倍～1000倍程度の高速性を持ち，規模の面でも前述の各種ニーズを満すだけ大容量であることが求められよう。

いずれにしろ，この面では今後の研究に待つことが多い。

2.4.5 科学計算マシン

(1) 超高速科学計算マシンの必要性

科学技術の進歩に伴って科学技術計算用のコンピュータに対する要求は高くなっており今後この傾向は一層著しくなると予想される。たとえば応力解析へのコンピュータの利用は著しく進歩したが，精度をあげるためにモデルを実際に近くすれば計算量は飛躍的に増大する。一次元的モデルから二次元的モデルに変えることにより計算量は1桁以上増加し，三次元的モデルで考えるとなればそれよりさらに1桁以上増加するといわれ，従来と同じ計算時間で処理しようとすれば，１００倍以上速いコンピュータが必要になる。また，超音速航空機の設計に使われる超音速風洞はその建設費と運転費があまりにも莫大なために超高速コンピュータでその代わりをさせようという計画がある（ＮＡＳＡ等）。これは流体力学の3次元偏微分方程式をベクトル型演算に還元して数値計算化しようとするものでこれに必要なコンピュータは現在の汎用コンピュータの超大型マシンに比較して１００倍位の性能が必要だといわれている。この他気象の数値予報，原子力関係の研究，画像処理，分子科学，……など，この種の超高速コンピュータを必要とする分野は非常に多い。そして，これらの分野ではコンピュータの能力の限界から，計算の精度をあるレベルであきらめているのが現状で，精度を向上するためにはいくらでも速いコンピュータが必要だといわれている。したがって，超高速の科学計算用コンピュータを持たない事は将来多くの科学分野，特に巨大科学と呼ばれるような重要な分野において致命的になると予想される。

一方，従来技術の延長だけではこれらの要求を満たすような超高速コンピュータを実現することは困難で，アーキテクチャおよびハードウェア・テクノロジィ（素子技術等）の革新が必要である（図2.4.2 参照）。パーソナルコンピュータが高性能になったとしてもこの分野の要求を満たすことは

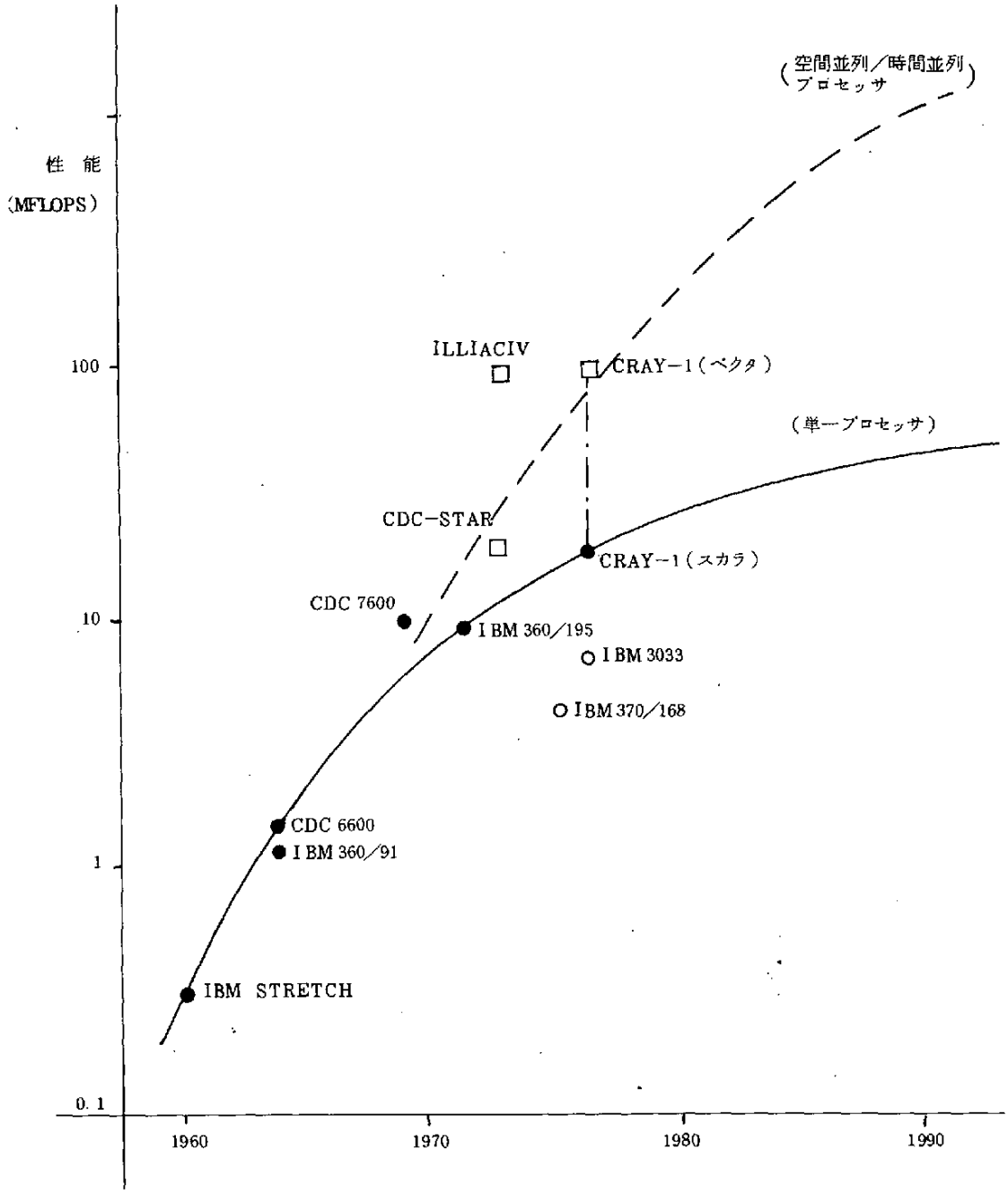


図 2.4.2　科学技術計算用コンピュータの性能推移

不可能であり，System Aにおいては専用マシンでこれに対処している。

(2) 方式と問題点

超高速科学技術計算用のアーキテクチャとして考えられる各種方式およびその特徴，問題点は以下の通りである。

1) 汎用アーキテクチャの改善

汎用コンピュータのアーキテクチャおよびハードウェアの実現手法（implementation）上での工夫によって高速化を図るもので，CDC 7600などにその例をみることができる。この方式では高速化のために，

(i) 演算回路を複数個持ち，それらを並列に動作させる。

(ii) そのためにプログラムにおけるデータの相互依存性などをハードウェアで解析し，場合によっては命令の実行順序を変えるなどして並列処理性を高める。

(iii) 高度の先廻り制御により演算回路をフルに動作させるよう命令，データを供給する。

などの工夫をする。

この方式のコンピュータは，汎用コンピュータと同じようにプログラムを作ることができてプログラムが作りやすく，また処理する問題によって性能が著しく異なることもないので使い易いコンピュータであるといえよう。

問題点としては，

(i) 今後この種の汎用アーキテクチャは画期的な飛躍が期待できないので性能向上は素子技術，実装技術の進歩によるしかない。素子技術の点では現在のシリコンLSIの飛躍的速度向上は期待できないので新しい素子たとえばGaAsやJJに期待する事になるが1990年頃の商用化は困難と予想される。

(ii) 汎用コンピュータに比較した場合，実現出来る性能は1桁程度以上にあげることは困難と思われる（同じ素子技術を用いた場合）ので汎用コンピュータで満足できないユーザの要求を満たすには不十分である。

などが考えられ，科学計算専用マシンは以下にのべる他の方式で実現されることになろう。

2) 空間並列化

多数（N台）のプロセッサを並列に動作させることによって高い性能（N倍の性能）を実現しようとする方法でIlliac IV などにその典型をみることができる（図2.4.3参照）。

方式上は，

(i) 各プロセッサが一斉に同じ演算動作をする（いわゆるSIMD型）ものと各プロセッサが独立に異なる演算を行える（いわゆるMIMD型）方式。

(ii) 個々のプロセッサとメモリとの接続方式（完全にメモリを共有する，あるいは各々のプロセッサが個有のメモリをもつなど）。

(iii) プロセッサ間の通信，全体を制御するプロセッサとの通信などの方式。

等の組合せで種々の方式があり得る。

この方式のコンピュータは今後LSI／VLSI の進歩によって高性能のプロセッサが小型，低コストで出来るようになったとき，それを活用するのに適した方式の1つと考えられる。

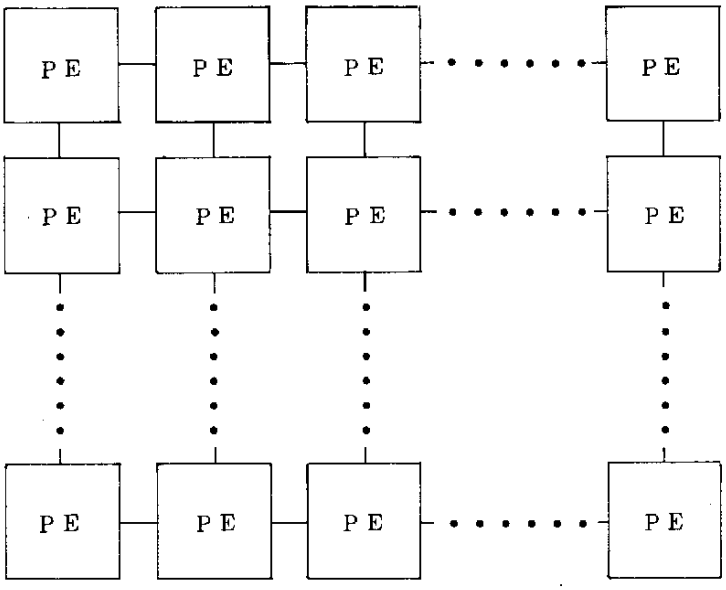
問題点としては，

(i) 並列処理可能な形に展開することがむずかしいプログラムもあるし，また並列処理を活かすようにプログラムを作ること自体汎用コンピュータに比較してプログラム作成が難しく，全体に使いにくいコンピュータになり易い。

(ii) プロセッサとメモリの接続方式が難しく，未だ方式が確立していない。プロセッサ台数が増えるに従って一層むずかしくなる。

(iii) 高速な演算処理に対応したデータの蓄積や供給をどうするか（バックアップメモリや入出力）。

などがある。



(注) PE：Processor Element

図 2.4.3　空間並列方式の例

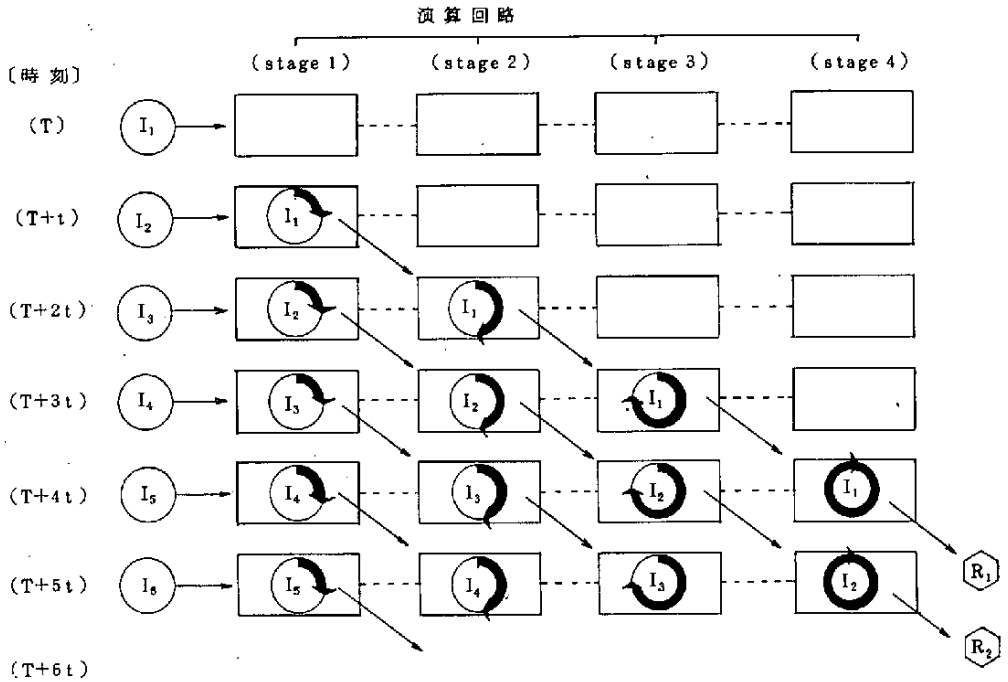
3) 時間並列化

製造ラインにおける流れ作業と同様に1つの演算動作を多段の演算に分解しその各々を独立の演算回路で処理するようにして（パイプライン化）高速化をはかったものである。たとえば，各段10nsで処理できる演算回路を直列に4段つないだとすれば，最初の演算については結果を得る迄に40nsかかるが次からの演算は10ns毎に結果が得られることになる（図2.4.4参照）。

この方式は超高速コンピュータの実現方法として現在迄に実用化された例も多く，CRAY-1などもこの方式を採用している。

時間並列化の方式としては，

(i) 1つのプログラムの実行に関してこれを上記のようにパイプライン化し，パイプライン化の効果を発揮させるためにベクトル命令を用いるなどアーキテクチャ上の工夫をする。



(注) In : 命令n
Rn : 命令nの実行結果
t : クロック

図 2. 4. 4 時間並列方式の例（パイプライン方式）

(ii) 複数の仮想のプロセッサを設け，各々が実行するプログラムを順番にパイプラインに送り込んで実行し，結果として複数プログラムを並行処理する。

がある。

時間並列化方式のコンピュータは空間並列化方式よりも一般に使い易いといわれているが，

(i) パイプラインの乱れによる性能低下。

(ii) ベクトル命令のような処理において高い効果を発揮するので，問題をベクトル命令で処理できるように変換すること。

(iii) 演算各段の速度（結局クロック・レート）で性能が決まるので性能をあげるにはクロックを高速化しなければならない。クロックの限界を超えた

高速化には空間並列化（パイプラインを複数設けるなど）が必要となる。

(iv) メモリから大量のデータを供給するためにプロセッサとメモリのデータ・パスが大変である。バック・アップ・メモリや入出力も性能に応じた大容量化，高速化が必要である。

などの問題がある。

4) データフロー・プロセッサ

データフロー・プロセッサは並列処理をうまく行って高速化するために考えられたもので，多数置かれたプロセッシング・エレメントによる空間並列化方式と考えることができよう。

データフロー・プロセッサはまだ研究段階で，言語（データフロー言語）が確立していない，汎用性に疑問がある等の問題はあるが，ＬＳＩ技術の効果的利用にも適しているといわれており，高度の並列性を要求する科学技術計算分野への応用は予想外に早く実現するかもしれない（たとえば偏微分方程式専用マシンなど）。

5) 数式処理

対象が複雑，精密化した科学技術において扱われる理論式も複雑なものになっており，いかにして正確かつ速くこのような理論式を誘導できるかが将来の科学技術の発展に大きくかかわっていると云われる。最近ではソフトウェアにより式の誘導過程における数式の操作を行うことが試みられているが，従来の科学技術計算用コンピュータは数値計算の高速化のみを目的に作られており，このような数式処理にはかならずしも適さない。このために数式処理にむいたアーキテクチャの研究，プロセッサの開発も行なわれている。

今後の科学技術計算用コンピュータとしては，数値計算の高速化を目的としたものだけでなく数式処理用のプロセッサ，あるいは数式処理に適した機能の導入などについても注目していく必要があろう。

(3) 研究課題

１９９０年代の科学計算マシンのイメージは以下のようである。

・ 演算速度 ： $10^3 \sim 10^4$ MFLOPS＊
（現在は $10^1 \sim 10^2$ MFLOPS）

・ 主記憶容量 ： $10^2 \sim 10^3$ MW＊＊（語長64ビット以上）

・ 処理できる問題の大きさ ： 1000×1000×1000位のマトリックスが扱える。

これを実現するための研究課題としては，

(i) プロセッサ単体の基本性能を向上するため，半導体素子の高性能化，新しい素子（GaAs，JJ など），実装技術（光の利用も含めて）など。たとえばJJを用いるとIBM370／168相当のアーキテクチャで70MIPS＊＊＊位のコンピュータが作れるといわれている（370／168は3MIPS位）（表2.4.1参照）。

(ii) 大容量・高速のバック・アップ・メモリ。1000×1000×1000のマトリックスを処理するには 10^{10} バイト級のバックアップメモリが必要であり，多量のデータを主記憶との間で高速に転送できることも要求される。

(iii) いずれの方式においても演算プロセッサと主記憶の間の接続方式（バス構造など）は性能実現上の重要なポイントとなるが，現在技術的に確立していない。特に空間並列で多数のプロセッサを設ける場合の最適な接続方式に関して問題が多い。

(iv) 科学計算マシンは演算主体であり，外界とのインタフェイスはホスト・プロセッサに頼るので，効率のよい接続方式，インタフェイスを研究する必要がある。これに関連するが，入出力の処理方式についても研究が必要であろう。

＊ MFLOPS ： 10^6 Floating Operations Per Second

＊＊ MW ： 10^6 Words

＊＊＊ MIPS ： 10^6 Instructions Per Second

表2.4.1　ＪＪで作ったコンピュータのイメージ

Performance	70 million instructions per second
CPU cycle time	4 ns
Cache capacity	32k
Main RAM capacity	16 Mbyte
Cache access time	4 ns
Main RAM access time	20 ns
I/O data rate (max)	360 Mbit/sec
Power at 4°K	7W
Power at 300°K	15 kW
Volume of Mainframe	4 liter
Volume of Cryostat	460 liter

(v) 超大型の科学計算マシンでは用いるハードウェアの量も膨大になるので，信頼性に関しても十分の改善がないと使いものにならない恐れがある。

(vi) データフローアーキテクチャ。前述の如く，科学計算マシン向きアーキテクチャとして可能性を持っているが未だ研究段階で今後の研究・開発成果に期待するところが多い。

(vii) 数式処理／記号処理にむいたアーキテクチャの研究と，それらを現アーキテクチャに導入して機能／インテリジェンスを高めること。

(viii) 対象の問題を並列処理を活かして解くアルゴリズムやそれに適した言語。

などがある。

2.4.6　高インテリジェンス・マシン

(1) マシンの目的と性格

本来のデータ処理の枠組を超えた，新しいタイプの処理に対する適用を目的としたマシンである。

このような新しいタイプの処理としては，つぎのようなものがある。

— 画像処理

— 数式処理

－　知識処理

－　自然言語処理

－　文書作成処理

－　人工知能向特殊処理

こうした処理を行なうために必要な論理機能については，かなり明確化されているものと，なお相当な研究を要するものとが相なかばしているのが現状である。しかし，最近の研究動向から判断すると，それら不明確な部分に対する究明は今後急速に進歩するものと見込まれる。

各タイプの処理は，互に同種の論理機能を利用する場合が多いことを考慮すると，必ずしも各タイプごとに専用のマシンを当てると考えるのが適切という訳ではないが，以下では便宜上各タイプごとの専用マシンという形式をとるものとする。

(2)　マシン各論

イ．画像処理マシン

複雑な画像を大量に処理，解析，認識するためのマシンで，論理的にはつぎのような機能部から成り立つものとしてよい。

1）雑音除去，スムージング，画像強調，微分処理などといった前処理部

2）大量の画像データを演算部に高速に流し込むためのデータ制御部

3）ｎ×ｎの範囲内での空間的演算と，二つの関連画像間の処理のための平面的分散演算（フレーム間演算）などを行なう演算部

4）人間との対話を行ない，それにもとづいて必要な制御を行なう対話制御部

ロ．数式処理マシン

論理深度が非常に大きくて，いわゆる並列処理の構造に適しない問題を，超高速で処理するためのもので，現在開発が進められているＦＬＡＴＳ計算機の成果を基礎にして，それの性能をさらに高度化するものである。

ハ．知識処理マシン

専門家に対するコンサルテーションを行なったり，あるいは専門家の有能な助手としての役割を果したりしうる程度の知能をもったマシンで，内容的には，いわゆる知識ベースと推論機構から成り立つものである。

知識ベースは，専門家のもつ専門的な知識を集大成した，一種のデータベースとみることができるが，それより一層高度な知的能力をもつために，つぎのような論理機能が必要である。

1) 連想的検索機能－手掛りとなる情報にもとづいて，それと強い関連をもつ情報やそれに類似のものを引出す機能。

2) 発達能力機能－情報を受け取ったとき，その真偽を判定する機能と，真の場合には，それをデータベースに同化的に追加する機能。

3) 推論機能－データベースに書かれた通りの情報を引出すだけではなく，ある程度の推論を行なうことによって，より高次の情報を導き出す機能。

4) 適応的検索機能－情報が入力された時に作られた構造とは異なる構造をもった検索要求を処理する機能。

つぎに，推論機構は，専門家の行なう推論過程を，計算機上に実現するものということができる。推論には，問題を一種の記号列と見なして，それに次々に変換操作を加えていく，記号処理としての側面と，多数の可能性を具体的に展開してみて，その中に必要な条件を満足するものを探し当てる，発見的なグラフ探索としての側面をもつ。したがって，推論機構に要求される論理機能はつぎの通りである。

1) リスト処理機能－ＬＩＳＰマシン相当の機能

2) バックトラッキング機能

3) 非決定性機能

ニ. 自然言語処理マシン

日本語から外国語へ，また，外国語から日本語への機械翻訳を行なう機能をもったマシンである。翻訳対象は，科学技術関係の専門家向文献を中心に，形式性の高いものに限定する。

必要な論理機能は，基本的には，知識処理マシンと同じであるが，性能的には機械翻訳マシンとしての性格からくる特異性が出ることも考えられる。

ホ．文書作成処理マシン

雑多に与えられた情報をもとにして，文章，表，グラフなどさまざまな形式の文書を，半自動的に作成するマシンである。オフィス・オートメーションの中枢部に位置づけられることを想定したマシンである。

このマシンに必要な論理機能は，知識処理マシンと画像処理マシンのもつもので十分カバーしうるものと考えられるが，文書作成処理の内容として，何をどの程度まで期待するかによって，必要な論理機能に大きな変動がありうる。

ヘ．人工知能向特殊処理マシン

パターン認識などのように，発見的手法を本質的に利用した処理を行なうためのマシンである。これに必要な論理機能としては，高度なパターン抽出機能，パターン変換機能，パターン・マッチング機能，バックトラッキング機能などが基本的である。ただし，これらの具体化については困難な問題が多い。

(3) 研究課題

前項に列挙した各マシンの処理内容そのものが，現在最先端の研究課題となっているもの，あるいはそうなろうとしているものばかりである。しかも，基礎研究の段階にあるものが多い。しかし，これらはいずれも極めて実用価値の高いもので，いったんブレークスルーが達成されれば，それを実現したところが一挙に市場優位性を確立してしまうことが予想される。その意味で，これらの研究は必須である。

アーキテクチャに特定化した立場からの研究課題としては，各マシンに必要な論理機能に対する実現方式の開発を挙げることができる。これらの論理機能は，これまでほとんどがソフトウェア的に実現されており，これらを有

効にサポートするためのハードウェアのあり方は，ほぼ白紙の状態にある。

さらに，これらの論理機能の多くは，データおよびその表現に関する知識に基づく意味処理に基礎を置いており，意味処理の複雑さからいって今後とも多量の試行錯誤を必要とするものと考えられる。

2.4.7 モニタ・コントロール機構

(1) 概　要

モニタ・コントロール機構は，「動的な」システム情報に基づき，（コンピュータ）システムを制御したり，人間にその情報をフィードバックする事により，ユーザに従来の計算機では実現できなかったような高度なサービスを提供する事を目的としたシステム機構である。

2.3.4項で述べたように，第5世代コンピュータに対するユーザの要求は，強い「目的指向型アーキテクチャ」と「高信頼性アーキテクチャ」にあるが，

① 目的指向型アーキテクチャ実現に当り静的な問題の解析だけではどうしてもその最適構造を知ることが出来ない本質的に動的な解析を必要とする問題がある。

② 目的に応じて最適なアーキテクチャの構造は一定でなく，そのため，問題に応じて何らかの方法で構造を変えてやる必要がある。

③ 高信頼性を保障するには，コンピュータがコンピュータ自身で動作状態をモニタし，その結果，故障を認識し，さらに自動的に復帰させる必要がある。

④ ユーザに対するサービスの中でも，ユーザの動作をモニタしないと行なえないようなものがある。

⑤ コンピュータ自身にコンピュータ室やその周りの世話をやらせるためにも動的モニタが必要である。

などの理由により，その実現に当ってはモニタ・コントロール機構をシステム・アーキテクチャレベルで備える必要性が高い。この機構をシステム構築

後に付加する事は難しく，実現に当ってはシステム・アーキテクチャレベルで初めから備えておかなければならない。したがって，System Aでは，モニタ・コントロール機構は本質的に重要な機能要求として，アーキテクチャ構築時に取り入れられた。

モニタ・コントロール機構を備えた事の利点（効果）としては，システムの最適制御による実行速度の向上，システム・リソースの有効利用などの直接的な効率向上も大きいと思われるが，使用感の向上，サービスビリティの向上，ソフトウェアの生産性の向上，将来（第6世代，第7世代…）のコンピュータ設計のためのデータ提供などの間接的な利点も大きいと思われる。

(2) 機能イメージ

モニタ・コントロール機構に期待できる機能について機能イメージをまとめると以下のようになろう。

① パーソナル・アーキテクチャ化

- 目的に応じて，動作状態（たとえばワークロードなど）をもとに最適な構造を予測し，システム・パラメタを最適にするようなオペレーティング・システム（dynamic adaptive operating system）アーキテクチャの実現。
- 命令レベルでの動作状態をもとに問題に適応した命令を自動的に生成し，自動効率向上を行なうアーキテクチャ・チューニング機構。

② 最適アーキテクチャ構造

- 目的に応じ最適構造をとるためのPMS[*]レベルでの再構成機能（仮想最適マシン化）。
- 階層化によるオーバーヘッドを最小に抑えるため，レベル間通信を常にモニタし，動的情報を収集し，レベル間の情報送受の最適化を行なう機能。
- 仮想メモリシステムなどにおけるプログラムの再配置による効率向上。

---

[*] Processor Memory Switch レベル

③ 高信頼性アーキテクチャ

・ 動的モニタをもとにして，ダイナミックマイクロプログラム技術によりマイクロプログラムを再構成し，高信頼を実現する機能。

・ 動的モニタをもとに故障ハードウエア・モジュールの切り離しを行ない高信頼性を実現する機能。

・ RAS のための機能。

④ ユーザサービス

・ プログラムの事象モニタによる動的モニタを核とした高度なデバッグのための機能。

・ ユーザの動作をモニタし，その結果をもとにユーザに対して援助を行なう。たとえば，動作状態の説明などの高度なＨｅｌｐ機能。

・ 動作状態をもとに，個々のユーザに応じた教育を行なう機能。

・ マシン効率を落さず個々のロギング情報をユーザならびにシステム管理者に提供できる機能。

⑤ 無人運転，自動運用機能

・ 防災システムと連動させた高度な無人運転機能。

・ コンピュータ室の備品管理をも含めた自動運用機能。

(3) 実現に当ってのイメージ

実現に当ってのモニタ・コントロール機構のイメージは以下の通り。

① モニタ・コントロール機構はシステムとして一つということはなく，各レベルに分散され実現される。

② レベル間通信の最適制御は，サービス・マシンにおいて行なう。

③ また，モニタとしては，

・ ハードウエアモニタ

・ ファームウエアモニタ

・ ソフトウェアモニタ

が必要に応じて使われると思われる。

また，可変にする（コントロールする）ための技術としては，

・　ハードウェア・モジュールの切換え（スイッチ）技術

・　高速バスアーキテクチャ技術

・　モジュール接続技術

などがあり，ダイナミック・マイクロプログラム技術とともに使われると思われる。

(4)　研究課題

モニタ・コントロール機構実現に当って必要な重要な技術を列挙すれば以下のようになろう。

①　適応システムのモデル化

(2)で述べた機能の実現に当り共通しているのは，動的モニタにより系を変え最適化するということであり，これは適応システムの実現に他ならない。そこで，適応化の目的ごとに（(2)で述べた項目ごとに）何をモニタするか，その結果をどのように解析するか，またそれをもとにどうやって系を変え適応させるかという適応システムのモデル化をまず明らかにする必要がある。さらに，

②　モニタする事をあらかじめ考慮したハードウェア設計技術，コンピュータ内データ構造，ソフトウェア作成技術の研究。

③　ハードウェア再構成技術の研究，たとえば，再構成を意識したモジュール分割の方法なども必要であると思われる。

2.4.8　制御・宇宙航空・民生用計算機

図2.3.2に示すように，System A には制御用計算機，宇宙航空用計算機ならびにホーム・コンピュータが含まれる。以下に，それぞれの計算機の特徴と技術的な課題について簡単に述べる。

(1)　制御用計算機

プロセス制御計算機に要求される機能のうち重要なものは実時間処理のた

めの高速応答，多数のセンサに対する効率のよい入出力制御，温度・湿度・振動・圧力・電気ノイズなどの特殊環境下の運転，長期間の連続・安全運転に伴う高信頼性システム構成に関するものであろう。

(2) 宇宙航空用計算機

制御用計算機に比らべ，更に高い信頼性が要求される。例えば，１０年間連続運転が要求されることがある。また，重力・放射能・温度差に対する条件も極めて厳しい。結局，この種の計算機は冗長技術を駆使した高信頼化と特殊環境に耐える実装技術が最も重要な課題である。また，複雑な科学計算を主体とした高速信号処理装置のオンボード化などもこれからの課題である。

(3) 民生用計算機

家庭を中心としたホームコンピュータの役割は電子郵便で代表される情報交換，生活映像情報サービス，自動モニター，自動管理，自動制御あるいは教育・娯楽などに関するものである。一般に，これらの役割を１台の計算機で集中的に制御することは考えられず，むしろ個々の役割を単純な１チップ・マイクロコンピュータで果すのが普通であろう。したがって，機能の高い・高集積度の一般ＬＳＩ素子および各種センサの開発，実装技術などが重要課題といえる。

2.5 計算機アーキテクチャの検討課題

前節までに概要を示したSystem Aを開発する際に考慮すべき重要課題について，計算機アーキテクチャの立場から簡単に触れてみたい。

2.5.1 新技術の検討

System Aは１９９０年代に標準的に使われる計算機であり，しかも計算機開発に関連する諸技術が急速に進歩していることを考えると，System Aの開発に際しては今までの計算機アーキテクチャにとらわれることなく，現在の計算機がかかえる諸問題を効果的に解決する新しい技術を計算機アーキテク

チャ・レベルで積極的に導入することが大切である。

新技術検討において特に考慮すべき点は，

(1) 関連する諸技術の発展過程を十分理解し，その延長上で将来動向を予測する。

(2) 社会環境分科会ならびに基礎理論分科会の研究調査成果を計算機アーキテクチャの観点から解析し，要求事項を整理した後その解決策を検討する。

(3) 問題点の整理を関連技術の分類あるいは応用面からも徹底的に行なう。

(4) 新しい技術の提案に際し，その実現可能性を検討し，実現のための課題（条件）を整理しておく

ことなどであろう。

2.5.2 検討時の留意点

新しい計算機アーキテクチャの検討には次のような点を留意することが望ましい。

(1) 今後に残された研究課題の徹底調査

(2) ハードウェアからソフトウェアにいたる幅広い関連技術ならびに応用分野との関係の明確化

(3) 検討すべき計算機アーキテクチャをよりミクロなレベルで具体的にとらえる

(4) 新しい計算機アーキテクチャがもたらす効果の評価

(5) 繰返し試行による効果の改善

(6) 実現のための技術的条件の整理と解決の見通しの把握

(7) 蓄積された経験・知識・資産の有効的な活用

なお，実際の検討実施に際しては，

(8) 有識者ならびに技術者による組織的な研究体制

をとることが肝要であり，対外的には，

(9) 国際的な場における研究調査成果の発表を通して，将来の計算機技術に対する我国の姿勢を諸外国に理解させる努力をすべきと考えている。

第３章　デバイス技術

第 3 章　デバイス技術

3.1　概　　説

将来のコンピュータのアーキテクチャを考える場合，デバイス技術の進歩が大きな影響を与えることは改めて述べるまでもない。過去の10年間を振り返れば，集積回路技術の進歩がコンピュータとその関連技術に与えたインパクトは，それ以前には誰も予想しなかったくらいに大きい。シリコンを基材とする集積回路技術の進歩は，今後少なくとも10年間は継続すると考えられ，そのアーキテクチャに与える影響は極めて大きいと予想される。また，シリコンＬＳＩのほかにもアーキテクチャにインパクトを与えるハードウェア要因は多いが，この調査で扱った分野は概略つぎの通りである。

(1)　シリコンを基材とする超ＬＳＩ

(2)　シリコン以外の超高速デバイス ― 化合物半導体素子，超電導素子など

(3)　ファイル・メモリのハードウェア

(4)　光関連技術

つぎにこれらの概要を述べる。

(1)　超Ｌ Ｓ Ｉ

デバイス技術そのものとしては，シリコン集積回路の限度追及が今後10年にわたって継続される。性能項目としては，集積度，遅延・電力積，信頼度などが挙げられ，そのいずれにもかなりの進歩が期待できる。図3.1.1は集積度の増加予測の１例であるが，増加そのものは継続されるけれども，増加率は1980年代に入ってそれまでよりも低下するとみられる。その原因としてはデバイス技術そのものの困難性が増加することのほかに，集積度の増大を量産の矛盾（メモリよりもとくにプロセッサに関して），初期コストの増大などが挙げられる。初期コストの増大のうち，製造設備の高度化以外に，

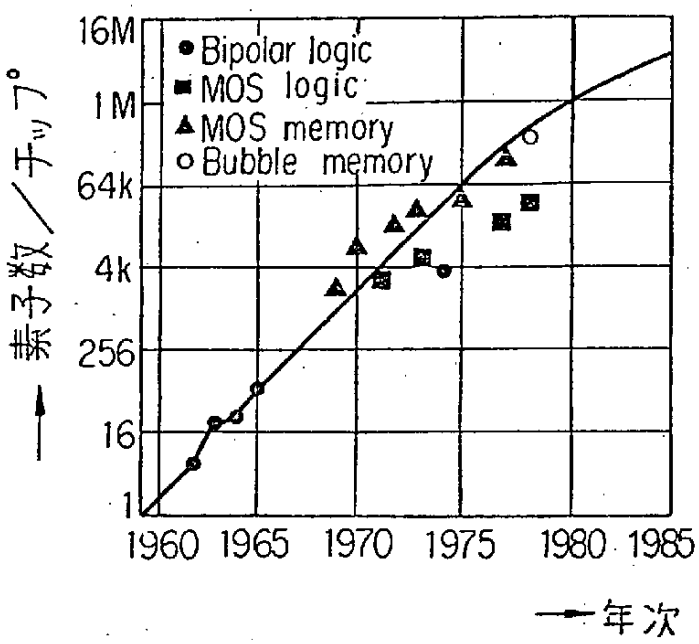


図 3. 1. 1　集積度の増加

ＬＳＩの規模の増大にともなって，要求機能の仕様決定と設計に要するコストが急増することは見落せないポイントとなるだろう。図 3. 1. 2 はこれを示したものの１例である。これに対して，論理設計，レイアウト，マスク設計などのＣＡＤの改良強化が当面の課題となるが，この問題の本質はソフトウェア工学における要求仕様，ソフトウェア作成技術などと同じものであり，今後長期にわたる研究課題となるだろう。これらの問題の進展は超ＬＳＩの入手可能な多様性を支配する要因となるだろう。

(2)　シリコン以外の超高速デバイス

集積回路の基材はシリコンに限るわけではない。シリコンは実用的に最もよく研究されているが，その他の基材にはそれぞれの利点が考えられる。その例として本調査では化合物半導体と超電導素子をとりあげた。シリコンよりも電荷担体の易動度の大きなガリウム砒素などの基材は高速動作に適しており，また独特のガン効果などを論理動作に適用することも考えられる。この種デバイスの具体化は，実現性に問題があるのではなく，産業的評価によ

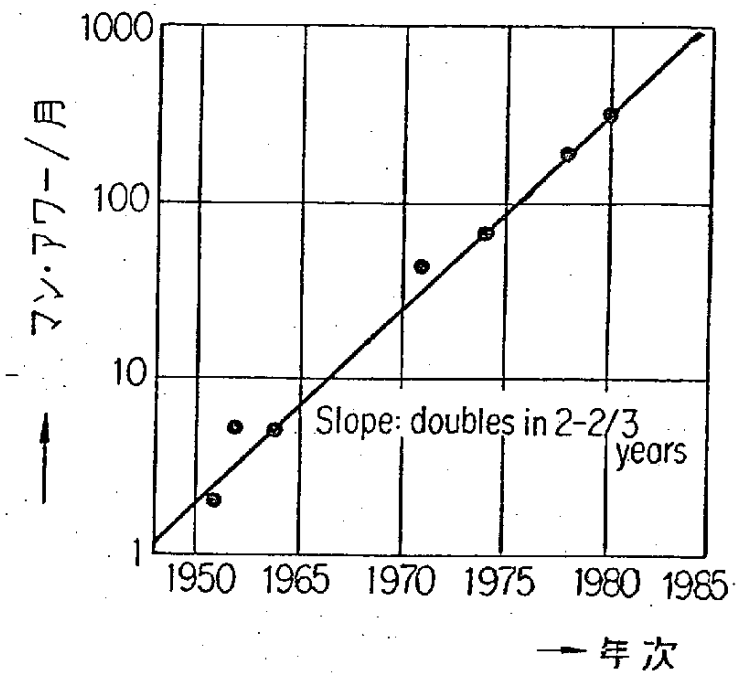


図 3.1.2　ＬＳＩの仕様決定と設計，レイアウトに要するマン・アワーの増加

るものと考えられる。

超電導素子は20年以上前から幾つかの形式が検討されてきたが，ジョセフソン効果を動作原理とするもの以外は性能的にみて常温デバイスに対する優位が認められない。ジョセフソン効果を用いるデバイスは，論理素子，メモリ素子としてそれぞれ数種のものが提案されており，超高速動作のほかに低電力の利点による小形，高集積という特徴も期待できるが，実用化のためには未知の問題がかなり残されている。

(3)　ファイル・メモリ

今後のコンピュータ技術の進展にともなって，大容量ファイルの重要性はますます大きくなると考えられる。この分野は現在磁気ディスク，磁気テープなどの磁気記録技術が主な部分を占めており，過去における技術の進歩はきわめて著しい。けれどもその記憶密度に関する進歩はかなり限界に近いところに達しており，今後10年間に期待できる進歩は１桁程度と考えられる。

一方光メモリなどの革新的技術は実用までになお多くのブレークスルーを必要とする。中でも情報の書きかえが可能な光メモリ材料は現在も２，３の候補があるが特性的に不満足なところが多い。この面でのブレークスルーがあれば実用的に広い応用を見出すだろう。

(4) 光関連技術

光情報処理技術は20年くらい前から有望といわれながらまだ確実な適応分野を模索している状況である。その中で通信分野だけが数年以内に実用の範囲に入るものとみられる。そして10年くらい後にはかなりの部分がこの技術によって置換されるだろう。けれどもこの技術によって通信コストが低下する割合は，ＬＳＩ技術によってメモリのコストが低下するほどの量的革新は期待できない。通信以外の分野ではＩ／Ｏおよびメモリが光技術にとって有望と思われるが，光技術を主要な要素として従来の技術に対する優位を主張できるためにはなお多くの技術的ブレークスルーが必要であり，その時期の見通しを得ることは困難である。

3.2 シリコンＶＬＳＩ

3.2.1 まえがき

ＶＬＳＩはＩＣの思想の延長線上にあり，集積度が10^6 素子程度のものをいう。集積度は現在までのところ年率２倍で増加してきており，1980年代なかばまでには１Ｍbits のメモリチップ，1990年代には数Ｍbits のメモリチップの出現が予想され，2000 年頃までは集積度の増加が続くと予想される。以下ではシリコンＶＬＳＩ技術の可能性と問題点についての概要を述べる。詳細については下記の文献1)，2）を参照されたい。

1） 超ＬＳＩ特集，電子通信学会誌，Ｖol．６２，№４，昭和５４年４月

2） 高集積技術の将来とそのシステムへのインパクト，電子材料マネージメントボード報告書，５２－Ｍ－１１８，日本電子工業振興協会，昭和５２年３月

3.2.2 ＶＬＳＩ技術の可能性と問題点

(1) デバイス技術

年率２倍の集積度の増加率は図3.2.1に示されるように，①シリコンチップの大形化，②微細寸法化，③デバイスと回路の改良によって達成されている。80年代になると③の要素が限界となり，２年で２倍から３年で２倍程度に増加率が落ちる。しかし2000 年頃までは増加が続くであろう。

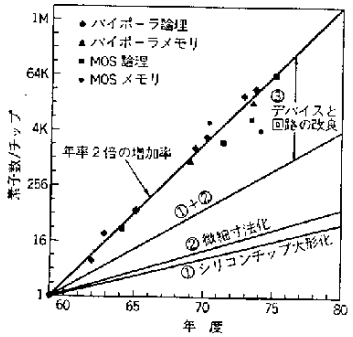
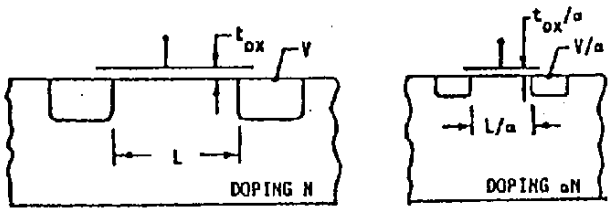


図 3.2.1　素子数の１チップ当りの年次変化

素子寸法の微小化には図3.2.2に示すようなスケーリングルールが成立する。スケーリングルールに従い，寸法を1/2倍，電圧を1/2倍，不純物濃度を２倍にすると，ＭＯＳトランジスタの性能は図に示すように改善される。

SCALING THEORY



RESULTS

DENSITY	D	a^2D
DELAY	τ	τ/a
POWER	P	P/a^2

図 3.2.2　ＭＯＳトランジスタのスケーリングルール（理論）

微小化が進むことによってどの位の集積密度が実現されるかは図3.2.3から明らかになる。この図は最近の文献における例をプロットして，これを2乗に逆比例する直線で外そうしたものである。この図によると、たとえばチャネル長が1μmの場合は密度が約$10^4/mm^2$，したがって，10㎜角のチップでは1Mbits程度の値となることが分かる。また，ゲート酸化膜が50Åとなったところを限界と仮定するならば，現状の約100倍の集積密度となることが分かる。

微小化には微細加工上の限界とデバイスの限界がある。微細加工は従来は光を用いたリソグラフィによって行なわれてきたが，さらに微細なパターンの形成のために電子ビーム，Ｘ線を用いた方法の開発が進められており，近い将来に0.1μmオーダの最小線幅は実現されると考えられる。デバイスを微細化していくときに到達する各種の電気的，物理的限界を表3.2.1に示す。実際のデバイスの最小寸法は熱放散で決まると見られ，その限界はおよそ2～3μmではないかと考えられている。

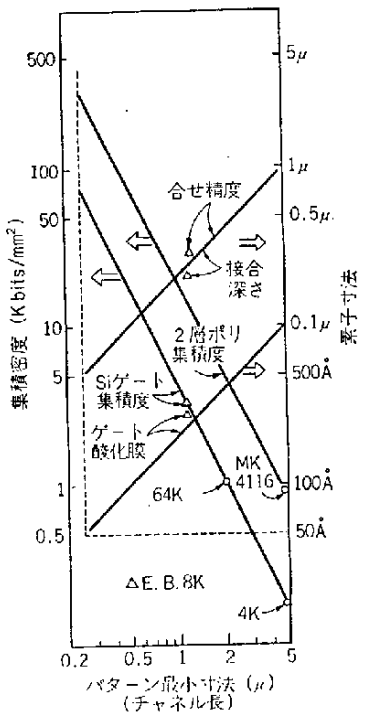


図3.2.3　素子寸法および集積密度とパターン最小寸法

表3.2.1　各種の限界値

種　類	限　界　値	備　考
SiO_2膜	50Å	
エレクトロマイグレーション	10^6 amp／cm²	
飽和速度	10^7 cm／sec	バルク
〃	6.5×10^6 cm／sec	表　面
pn接合濃度	1.3×10^{19} cm^{-3}	
接合端降伏	1.5×10^6 V／cm	
放熱限界	1W／cm²	空　冷
〃	20W／cm²	液　冷

微小化に伴う問題点として，①信号電圧の減少，②放射線によるソフトエラーなどが指摘されている。①はセル面積が小さくなるための出力低下と電源電圧の低下の両方に起因する。②はパッケージ材料から放射される微量のα線がセル中で電子－正孔対を発生することに起因する。今後も新たな問題点が出てくることが考えられるが，これらすべては克服されていくものと期待されている。

(2)　設計技術

ＶＬＳＩはＬＳＩの延長線上にあり，設計に関して本質的に現在と異なるわけではない。しかし，図3.2.4に示すように論理ＬＳＩではゲート数の増加が著しい。また，論理ＬＳＩは回路の構成が不規則でメモリＬＳＩのように規則的なくり返しが少ないために集積規模の増大につれて，設計の複雑さが増加している。これらのことから，ＶＬＳＩにおける設計の困難さは充分に予想される。この困難に対するブレークスルーとして活発に研究されなければならないのは，各種の自動設計（ＣＡＤ）の技術であろう。

自動設計のためには，まず要求仕様の明確化がなされなければならない。この点はソフトウェアにおいて仕様作成コストがかなりの部分を占めていることから類

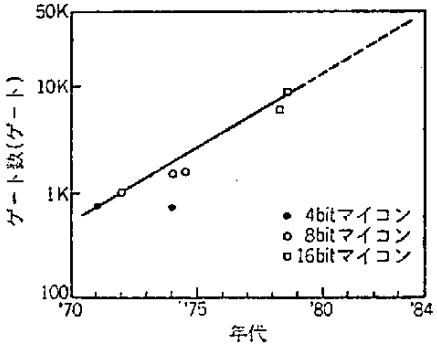


図3.2.4 論理LSIのゲート数の推移

推すると，ＶＬＳＩにおいても今後の見落せないポイントとなるであろう。

自動設計の段階は，①論理設計，②レイアウトに大別することができる。①では論理ミスの除去，ピン数の節約，テスタビリティに対する配慮などが重要視されてくるものと考えられる。ＬＳＩでは論理ミスが発見されると，ミスの程度によってはＬＳＩ開発の出発点にもどされると言われている。ＶＬＳＩではシミュレータや設計言語を充分に用意すると同時に，修正のターンアラウンドを短くするために機能をブロック化してミスが全体に波及しないようにすることや，試作工程における電子ビーム露光装置によるウェーハへの直接露光なども取り入れていく必要があろう。

ＶＬＳＩになるとシステム全体が１個のチップに収容される場合が考えられる。システム全体を１個のチップに収容してしまえば，マイクロコンピュータなどに見られるようにピン数はむしろ少なくてすむ。しかし，大形計算機では当分はＶＬＳＩチップに全体を収容することは無理であろう。ピン数の制限はピンゲート比（ゲート数とピン数の比）を大きくする。ゲート数とピン数の関係を各種のシステムについて示すと図3.2.5のようになる。ピンゲート比が大きくなることは，ピンから内部が見えにくくなることであり，それだけ試験が難しくなる。

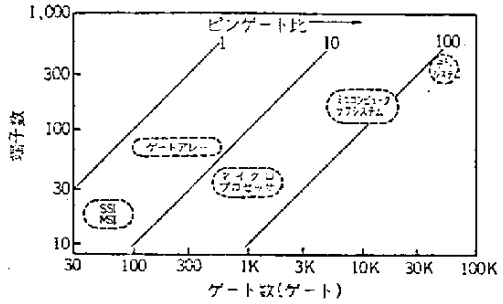


図3.2.5 ゲート数と端子(ピン)数の関係

集積度が増大し，内部が複雑になると共に，ピンゲート比も大きくなっていくことは，試験の困難さを増大させている。したがって，テスタビリティに対する配慮も重要である。

②のレイアウト段階では要求機能に対してチップ寸法の制限があり，一定面積のチップ上にできる限り高密度に各種の機能をレイアウトする必要がある。このために人手に比べて集積密度の点で劣る自動レイアウトは敬遠されていたようであるが，ＶＬＳＩでは自動化は避けられないであろう。上記の点を解決するために，階層構造をとり入れたレイアウト手法の提案がなされている。

表3.2.2　マスタスライス方式と標準セル方式の比較

		マスタスライス方式	標準セル方式
レイアウト	最小単位	セル	セル
	配置	配置できる場所は，あらかじめ決められている	列状の配置から全く自由な配置まで各種
	自動化	完全自動のものもある	種々の方法が開発中
開発期間		短い	長い
レイアウトの自由度		小	大
集積密度		少し劣る	高い
特性		少し劣る	高い
製品単価		少し高い	安い
用途		少量多品種	大量需要品種，高性能品種

現在のＬＳＩの自動設計法として，回路素子をアレー構造に固定して配慮する方法，回路とレイアウトパターンを標準化したセルを自由に配置する標準セル方式がある。アレー構造の代表的方法であるマスタスライス方式と，標準セル方式の概略の比較を表3.2.2に示す。

(3)　ユーザとの関係

集積度の増加もユーザから見れば，機能あたりのコスト低下や信頼性の向上という利益が期待される。メモリＬＳＩのビットあたりコストは今まで2年で1/2に低下するという実績を残しているが，今後はその割合は減少するものと考えられる。なぜならば，集積度の増加を実現するための製造設備の高度化による初期コストの増大，高集積になることによる歩留りの低下が考えられるからである。ユーザから見た利益がどう展開していくかが，歩留りを向上させるために，チップ内の全素子が良品である場合にそのチップを良品とする100%良品主義から，欠陥を含んでいてもシステム的に救済する方向に変わることも考えられる。

次に重要な点は，集積度の増加と共に試験が難しくなっているので，メーカ側でどの程度の試験をしてユーザ側でどの程度の試験をしなければならな

かという点である。試験のコストは上昇していくのでユーザ側により多く負担させる方向へ行くことも考えられる。ユーザにとっては試験が難しいということは，故障診断も難しいということである。したがって，設計段階ではこれまで以上にテスタビリティに対する配慮が必要となろう。

3.2.3 むすび

ＶＬＳＩはＬＳＩの延長線上にあり，今後約20年間はコンピュータのハードウェアの中心的存在であり続けるであろう。同時にＶＬＳＩの技術は化合物半導体デバイス，磁気バブルデバイス，集積ヘッドのような他のデバイスへの波及効果を持ち，それらの進歩をうながすであろう。

今後重要と考えられるものはＶＬＳＩの設計技術である。

3.3 化合物半導体デバイス

3.3.1 GaAsを中心とした化合物半導体の動作原理

- 半導体は自由電子やホールの動きを利用してダイオードやトランジスタ効果に使うことができる。

  半導体内における電子の移動速度をmobilityと呼んでいるが，GaAsやインジウムアンチモンなどの化合物半導体では，このmobilityがSi半導体に比べて相当大きい（表3.3.1参照）。

  したがって，GaAsなどの化合物半導体はゲートの遅延時間を小さくできる可能性があるため超高速のデバイス素材に適している。

- GaAsはSiやGeに比較してエネルギー・ギャップが大きい（表3.3.1参照）。これは絶縁物の程度が良いことを示し，ＬＳＩや超ＬＳＩ化において，半絶縁性の基板上にゲートを集積するため寄生容量が小さくなる。

表 3.3.1　各種素材のMobilityとEnergy gap

素　　材	Mobility	Energy gap
Ge	3,500 cm^2/Vsec	0.7　eV
Si	1,300 cm^2/Vsec	1.1　eV
GaAs	8,000 cm^2/Vsec	1.4　eV
InSb	70,000 cm^2/Vsec	—

☆化合物半導体によるＦＥＴ素子

GaAs などの化合物半導体によるＦＥＴの種類としてはシリコンによるものと同様に次の３つが考えられる。

(1)　ＪＦＥＴ　（接合形ＦＥＴ）……PN接合を利用

(2)　MESFET（Metal-semiconductor FET）…ショットキーゲートを利用

(3)　MOSFET（Metal oxide FET）……MOS反転層を利用

GaAs の場合，(3)のMOSFETは酸化膜の製造技術上の問題などからあまり研究は進んでおらず，(2)のMESFETが広く研究されている。

◦ＦＥＴの性能の基準としては，ゲートの電界に対して電流が大きく変化する程良い性能であるといえる。

GaAsのMESFETとシリコンMOSFETの電界に対する電子速度の関係を図 3.3.1 に示す。これによれば将来のＶＬＳＩにおいて１μ程度のゲートを作成する場合，電界は１０KV／cm程度となり，このような情況ではGaAs はシリコンに比べ約６倍チャネル電流を大きくとれることになる。

◦GaAsにはシリコン半導体にはないガン効果と呼ばれる特有の現象がある。通常の半導体ではオーム性の領域，すなわち電流が電圧に比例する領域が用いられている。物性的にはこれは電子の速度Ｖが電界Ｅに比例することを意

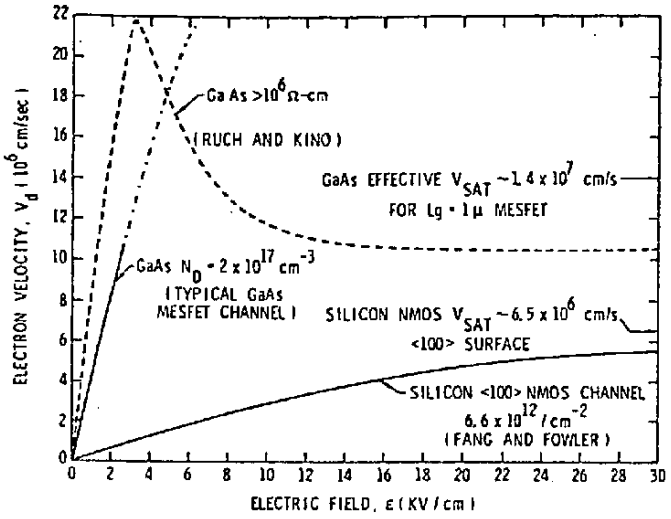


図 3.3.1　GaAs MESFETとシリコンMOSFETの特性比較

味するが，GaAsなどのいくつかの半導体では，電子の速度が電界の増加とともに遅くなる現象が現れてくる（図3.3.1参照）。これはGaAsなどの化合物における結晶のバンド構造が特殊な形をもっているためによる。

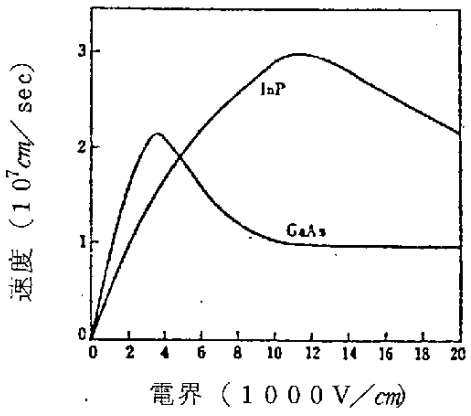


図3.3.2　GaAsとInPの電度電界特性

◦図3.3.3に示すように，GaAsの両端にオーム性電極をつけ直流バイアスを加えてゆくと，電圧がある臨界直Vthをこえると，電流が発振を起こすようになる。これは図3.3.2による微分負性移動度現象によって内部に高電界ドメインが発生し，この発生・消滅のくり返しが連続して起こる。これを利用して固体のマイクロ波発振器が実用化されている。

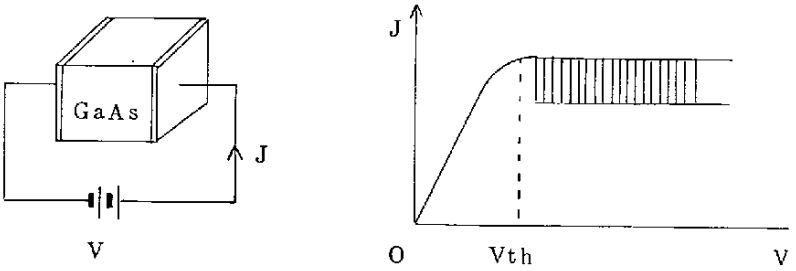


図 3.3.3　ガン効果素子の基本構造と特性

◦高電界ドメインの移動速度は約10^7 cm／secと高速であるので，このドメインの発生や検出を制御できる素子を構成できれば高速度の論理機能デバイスとなる。

実際，図3.3.4に示すように，2ないし3の信号電極を素子に設けると，動作条件によりANDやORの演算を100ps程度の速度で行うことが可能である。このような原理を応用して，シフトレジスタや高速キャリー発生器などが考えられており，このような素子を機能デバイスまたはTED (Transfered- Electron Device) と呼んでいる。

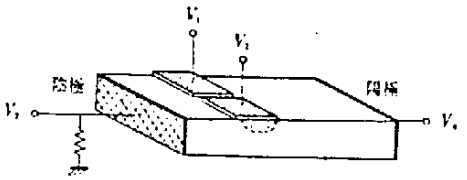


図3.3.4　ANDおよびOR素子

3.3.2　化合物半導体による論理素子の得失

(1)　FET素子

◦GaAsのFET素子における論理素子あたりの遅延時間と消費電力の関係は図3.3.5に示される如くであり，遅延電力積の値はシリコンにおけるCMOSとほぼ同等か，下まわる値であり，この面からはLSI向きであると言える。

◦GaAsは熱電導率がシリコンの1／3程度であり，集積度を高めた場合の冷却方法に問題が生じる可能性も考えられる。

◦GaAsにおけるMESFETにはノーマリオン型＊とノーマリオフ型がある。

◦ノーマリオン型のMESFET素子は論理振幅が比較的大きくとれるため高速素子に向いている。しかしながら，この素子は電源が2ついることや電圧のレベルを変換するためのレベルシフタが必要であることといった欠点を持ってい

＊ ゲート電圧ゼロの時に電流が流れる素子をノーマリオン型（MOSFETのデプリーションに対応）流れない素子をノーマリオフ型（MOSFETのエンハンスメントに対応）と呼ぶ。

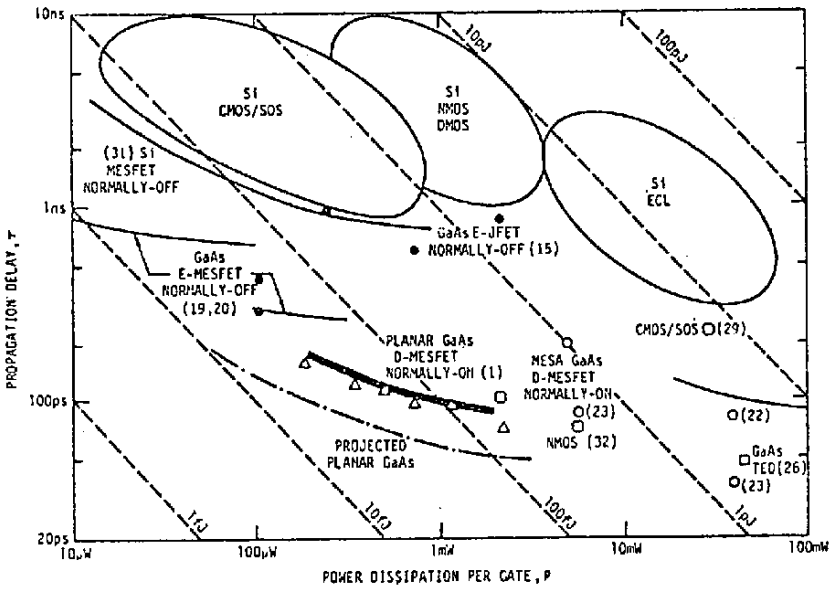


図3.3.5　ゲートあたりの遅延・電力の関係

るため将来のＶＬＳＩに必ずしも向いているとは言えない。ただし高速化を必要とする特殊な目的としては十分な用途が考えられる。

◦ノーマリオフ型のＦＥＴは低消費電力，回路構成が簡単であるなどの特長を持つため，ＶＬＳＩ向きであると言える。しかし論理振幅がショットキー・バリアのポテンシャルである0.6－0.8Ｖしかとれないため，均一で薄い活性層を作成する必要があり，この点は製造技術面での問題として残る。論理振幅の低下はシリコン素子においても検討されており，また製造技術の進歩も著しいことなどから以上の欠点は克服される可能性は大きい。

(2)　ガン効果機能素子

◦ＧａＡｓはガンダイオード，ＬＥＤなどの分野ですでに実用化されており，ガン効果を利用した機能素子の研究も実用化へ向けて進められている。この素子は１ゲートあたり５０ｐ～１００ｐｓｅｃの遅延という高速の機能が期待される。

◦回路面から見た場合，負性抵抗を利用した素子はトランジスタやＦＥＴによるものと比較して扱いにくいという欠点があり，互換性や他の論理素子てのイ

ンタフェースなどを考慮した場合大きな問題となる。

◦また電力消費が大きいため高密度化は困難であり，ＬＳＩには不向きと言える。ただし高速Ａ／Ｄ変換用デバイスや高速キャリー発生器などといった特殊な目的としては大いに実用化の可能性が考えられる。

3.3.3　技術・研究面での評価と問題点

(1)　材　　料

◦現在，化合物半導体の中で最も広く研究されているものはＧａＡｓである。その他，化合物半導体として用いられる可能性のある材料としてはＩｎＰ，ＩｎＳｂ，$Ga_x\ In_{1-x}Sb$などという化合物が挙げられる。

◦化合物半導体の場合，候補となる材料自体の研究がこれを用いる素子の特性に大きく影響すると考えられるので，新たな材料の研究や材料特性の研究が重要である。

(2)　集積技術

◦将来の計算機システムはモジュール化がさらに進むと予想され，その部品としての論理素子も集積化に適することが必要条件となる。

◦ＧａＡｓはシリコンによる集積回路技術の大部分が適用でき，集積化に適した素子である。

◦ＧａＡｓとシリコンを比較した場合，ＧａＡｓでは均一な基板を作成する技術が確立していないこと，活性層の厚さの制御技術が確立していないこと，大口径のウェーハができないことなど主として製造技術に多くの問題をかかえている。

◦基板製造技術の問題はノウハウ的な要素が多いため革新的な改善は望めないが，現在ＧａＡｓのガンダイオードやＬＥＤ素子がすでに実用化の域に入っていることなどから，次に述べる特殊目的の素子としての需要が確立されれば，これによって製造技術の進展に拍車がかかることも予想される。

3.3.4　応用面での予測

◦メモリシステム

計算機用のメモリシステムとしては，現在シリコンのＭＯＳによるものが大勢を占めている。ＭＯＳのダイナミックメモリはチップあたりのメモリ容量を大きく採れるため主メモリとしての需要が大きいが，ＧａＡｓの場合は前に指摘したように酸化膜によるＭＯＳ反転の形成が困難であることから考えて，高速用のバッファメモリなどの用途以外にはＧａＡｓによるメモリ素子の一般化は困難であると予測される。

◦レーダー及びセンサーシステム

レーダーやリモートセンシングに用いられる素子は，処理すべきデータが２～３次元のイメージデータであるためそのバンド幅が大きく，これを処理するためには数ＧＨｚ程度のＡ／Ｄ変換機能や乗算，比較機能が要求される。通常のシリコンＩＣでは，数１０ＭＨｚ乗算器がせいぜいであり，ＧａＡｓのＦＥＴでも１００ＭＨｚ程度である。ＧａＡｓを機能デバイスとして用いると，さらに高速度の処理が期待でき，これを並列化することによって実時間処理も可能となる。したがって高速素子としてのＧａＡｓに対する要求は切実であり，実用化の見通しも高い。

◦通信システム

将来の通信システムにおいて，そのバンド幅が高まり通信容量が増大するに従い，これを有効に利用する手段として通信のＰＣＭ化が進むことが予想される。これに伴い，高い周波数帯域における変復調器が必要となり，この目的のために高速のＧａＡｓを用いた論理素子ならびに機能素子が実用化される可能性は高い。

参考文献

1)　B．G．BOSCH,"Gigabit Electronics－A Review," Proceedings of IEEE, Vol.67.№3，pp.340－379,Mar.1979

2) R. C. EDEN, et al," The Prospects for Ultra high — Speed VLSI GaAs Digital Logic," IEEE Journal of Solid — State Circuits, Vol. SC-14, №2, pp.221—239, April 1979

3) 片岡照栄，“機能デバイス,”電子材料，pp83—87, April 1977

4) 片岡照栄，“超高速機能デバイス,”電子材料，pp.52—53, April 1978

5) 津田建二，“高速，低消費電力の魅力で開発に拍車がかかり始めたGaAs論理IC,”日経エレクトロニクス，pp.64—77，1979年4月2日号

6) ソリッドステート回路ハンドブック，丸善発行

3.4 ジョセフソン接合素子

3.4.1 はじめに

ジョセフソン接合素子（以下，ＪＪ素子という）を，電子計算機の論理回路，記憶回路に用いる場合の利点および解決すべき問題点を示すとともに，現在行われている研究動向について述べる。

3.4.2 素子の材料と動作原理

ＪＪ素子は，図3.4.1に示すような，２つの超伝導体の電極と，これにはさまれたごく薄い絶縁層から成っているトンネル型のほか，ブリッヂ型，ポイント・コンタクト型，近接効果型がある。トンネル型は，接合の出力電圧が確定すること，出力電流を大きくとれること等，論理素子として適した性質を有しており，以下，トンネル型を中心に述べる。

ＪＪ素子の，超伝導体としては，鉛（Pb），ニオブ（Nb）などが有力材料とされ，絶縁層としては，PbやNbの酸化膜が使われるほか，半導体（CdS，CdSe，Ge，Si，InSbなど）が使われている。絶縁層は，トンネル型には，数10Å（１～３㎜）におさえる必要があり，この安定な層を作る技術が，一つの問題点となっている。半導体では，この接合の障壁ポテンシャルを小さく

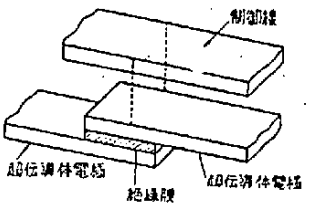


図 3.4.1　ＪＪ素子の構造

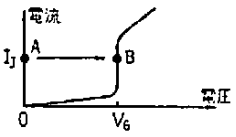


図 3.4.2　ＪＪ素子の電流電圧特性

でき，この層の厚みを増すことが可能で，接合容量も小さくできる利点があるが，その形成には蒸着法を用いるため，この絶縁層にピンホールができやすい等の問題がある。

ＪＪ素子は，その電極材料が，超伝導状態を示す極低温（鉛の場合 7° K以下）で用いられる。その接合の電気的特性は，図 3.4.2 のようになり，接合を流れる電流が，閾値 I_J 以下では，接合間に電位差を生じない。これを零電圧状態（図中，Ａで示す）と呼ぶ。電流が閾値 I_J を越えると，接合間に，その材料によって決まる接合電圧 V_G が現れ，有限電圧状態（図中，Ｂで示す）となる。ＡからＢへのスイッチング速度は，素子の電気容量で決まり，１０ps 程度の高速なものとなる。ここで，発生する電圧値 V_G は，鉛の場合約 2.6 mV，閾値電流 I_J は，電極の重なり面積や絶縁層の厚さなどにより決まり，通常，数 mA以下となる。この結果，消費電力は，数 μW以下となり，スイッチング素子として，極めて，優れた性質を示す。

3.4.3　スイッチング特性

スイッチング素子としての構成は，ＪＪ素子に，制御線を追加したものとなり，いくつかの方法が考えられる。ＪＪ素子は，零電圧状態から有限電圧状態へのスイッチングは，１０～１００ps の遅れしかなく高速であるが，有限電圧状態へのスイッチングは，接合の抵抗が大きく，放電に時間がかかり，1ns以上かかることもある。これについては，いくつかの回路中の素子をまとめて，リセットする等の工夫により，解決できる。

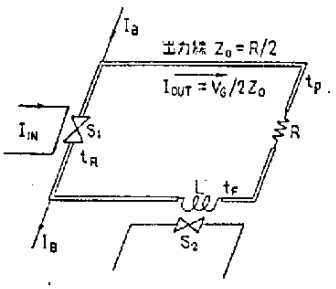


図 3.4.3　スイッチ回路の構成

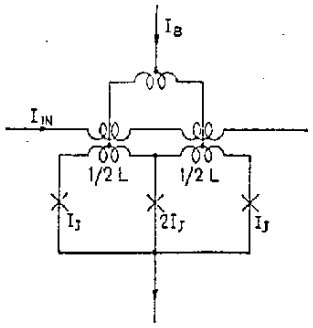


図 3.4.4　3素子干渉形スイッチ素子

このような前提のもとに，1つのスイッチング回路を考えると，まず，素子単体をスイッチに利用するものが考えられる。これは図3.4.3に示したように負荷抵抗Rと，次に続くスイッチの制御線を結ぶループを形成するものであるが，この構成では，スイッチングの感度が低く，遅延時間200ps，消費電力20μW程度となる。この改良型としては，3個のＪＪ素子を用いた，図3.4.4のような構成のものがある。ここでは，素子と制御線は，インダクタンスにより電磁誘導的に結合されている。これを用いた場合は，遅延時間50ps，消費電力2μW程度の値が得られ，スイッチング素子単体としては，半導体素子に比べ，電力遅延時間積において，2桁以上の改善が期待できる。

(2)　論 理 回 路

上で述べたようなスイッチング素子を用いて，ＯＲゲートや，ＡＮＤゲート，ＮＯＴゲートが構成でき，これら基本ゲートにより，種々の演算回路をつくることができる。図3.4.4に示した3素子を用いたスイッチング回路によって構成したＯＲゲートとＡＮＤゲートを図3.4.5に示す。この場合，ＡＮＤゲートは，スイッチング素子のカスケード接続となるため，遅延時間が増え，2入力ＡＮＤゲートで，100ps 程度となる。このような問題点の解決も含め，回路構成の研究が行われている。

演算回路の試作研究においては，2素子干渉型スイッチを用いた全加算器

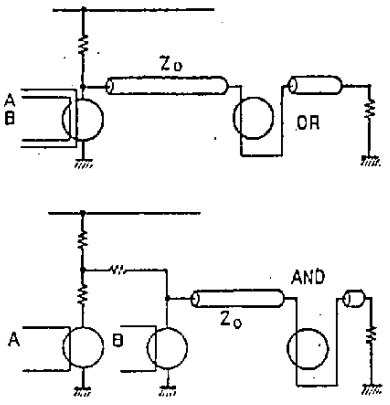


図 3.4.5　干渉型スイッチによる論理ゲートの構成

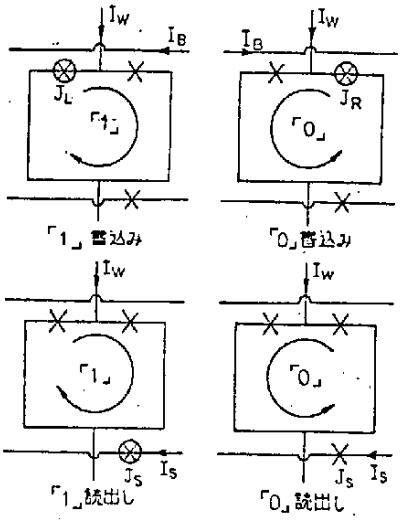


図 3.4.6　3素子形記憶セルの原理

乗算器なども作られており，4×4の乗算器に関しては，4ビット並列加算器を用い，4回のくり返しによって実行する場合，27nsという値を得ている。このときのゲート数は，45個であった。

このほかにも，いろいろな回路が試作されているが，本格的な規模の大きな回路ができるまでには至っていない。

(3)　記憶回路

記憶回路の構成は，超伝導状態における電気抵抗ゼロの性質を利用し，永久に流れ続ける電流のループを用いるものが考えられる。これを循環電流と呼ぶが，この電流のループ中の方向を，1と0に対応させる。電流のスイッチには，JJ素子を用い，図3.4.6に示すような構成をする。読み出しは，JJ素子Jsによって検出する。書き込みは，I_B による磁束を用いる。このほか，JJ素子の数を減らし，チップ上の面積を縮小したものとして，図3.4.7に示すようなものがある。これは，磁束の量子化現象を利用し，立体的なリング中にとらえられている磁束が，ゼロであるか，Φ_0 であるかを，0と1に対応させている。読出しは，バイアス電流 I_B に対し，ワード電流

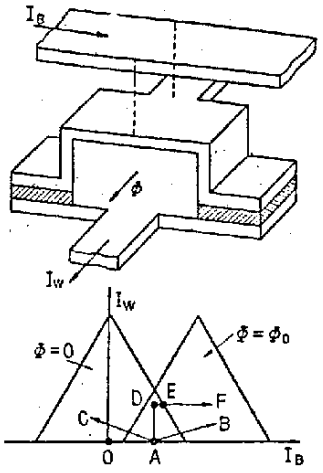


図 3.4.7　量子化磁束記憶セル

I_W が流れるか否かで，検出する。このため破壊読出しとなる。

このような構成による記憶のアクセス時間としては，３素子型のもので，６４ビットの容量の場合，2.3ｎｓ程度，２素子型のもので，１６Kビットを想定した場合，７ｎｓ程度の値が得られている。

以上のように，記憶回路についても，小規模な構成ではあるが，種々の試作が行なわれている。

3.4.4　ジョセフソン計算機の可能性と問題点

(1)　利　　点

ＪＪ素子を用いて，計算機を構成することを想定した場合の利点としては次のような点が挙げられる。

1）素子単体のスイッチング速度が，半導体等に比べ極めて高速である（１０～１００Ｐｓ）。また，消費電力も小さく，高集積化に有利である。

2）回路の実装寸法が，小さくできることにより，配線上の遅延時間も小さくできる。

3）記憶素子などとして用いる場合，情報を保持している間は，電力消費が無

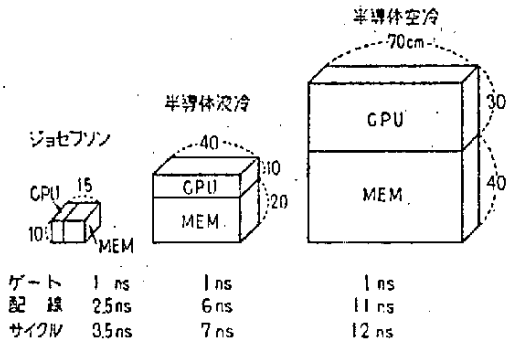


図 3.4.8　メインフレーム体積の比較

い等のメリットがある。

4）集積化技術としては，半導体に対して用いられているものが，そのまま生かせる。

5）回路を実装する場合のプロセスが，簡単であり，高集積化に適する。

以上のような利点のうちでは，1)と2)は，特に魅力的であり，半導体素子の速度が限界に近づきつつあるといわれ始めた現在では，特に説得力がある。小型化の程度については，メインフレームを，すべてＪＪ素子で実現すると考え，その規模を，ＣＰＵが 10^6 ゲート，記憶容量64MBの大型機を想定した場合，図3.4.8に示すような割合になるといわれている。消費電力も，半導体の場合1.5KWのものが，ＪＪ素子のものでは，1.3Wに減らすことが可能であると考えられている。

これらの議論は，半導体とＪＪ素子を，素子レベルで比較し，ＪＪ素子特有の回路構成上の問題，実装，保守等の問題は考慮されていない。しかしながら，その理論的上限は，半導体に比べ，非常に優れており，多くの可能性を持っていると言えよう。

(2)　問　題　点

ＪＪ素子を用いた計算機の研究は，まだ多くの問題をかかえており，今後の研究の困難さを示唆している。

以下，それらを挙げる。

1）素子単体での物性面の問題

① ＪＪ素子のふるまいに関する物性面の理解が十分でない。

② 安定なＪＪ素子の製造法が確立しておらず，歩留りが悪い。

この内容としては，以下のような点がある。

- 素子に用いる材料として何を選ぶか
- 絶縁層の材料の問題。また，ヒルロックの防止法など，絶縁層の安定化の問題
- 常温と極低温を往復する際の温度サイクルに対する安定性の問題
- 素子間干渉による誤動作の防止や雑音に対する耐性の問題

2）集積化技術，製造技術

① きわめて薄い酸化膜の製造技術の改良

② 信頼性，歩留りの確保

③ 回路の構成方式，チップ用実装方法，試験方法として，新しい技術開発が必要

3）アーキテクチャ，システム実装について

① 冷却技術

② 極低温系と外部系との間のインタフェイス

③ 保守，修理，改良などのための技術開発

4）その他の周囲状況

① 半導体（シリコン）素子との市場的，価格的競合

② 既存のハードウェア技術や，種々の装置との整合性，既存のものとの混用の可能性

以上のような問題点のうち，冷却技術など，すでに解決の見通しの立つものがあるほか，物性面における温度サイクルの問題など，改善の著しいものもある。しかし，多くの困難な問題もあり，基礎的な分野での研究がまたれるのが現状である。

3.4.5 ま と め

ＪＪ素子は，その基本的な物性面での性質により，多くの困難な問題をかかえてはいるものの，非常に魅力ある研究テーマといえよう。これまで，シリコン等の半導体のために開発された諸技術の多くが，役立つ可能性が大きい。しかしながら，物性面での安定な材料の探求や，その加工技術の見通しについては，必ずしも楽観できない点が多い。また，たとえ，製品としての計算機が技術的に実現可能となったとしても，その時点において，従来のシリコンを中心とする半導体技術との価格的競合が考えられ，現在の計算機の代替としての実力を備えるには，かなり長い年月を必要とするのではないかと考えられる。

参 考 文 献

1) ジョセフソン・ディバイス調査報告書，日本電子工業振興協会,昭和52年,pp. 107－168

2) 原宏,"ジョセフソン素子の問題点," 電子通信学会誌，Vol.62，№6，pp. 677－680（1979年6月）

3) Bosch B. G,"Gigabit Electronics—A Review," Proc. IEEE, Vol. 67, pp. 340－379 (Apr. 1979)

4) 石田晶,"ジョセフソン素子とその応用," 情報処理，Vol. 20，№9，pp. 779－784，(Sept. 1979)

3.5 フィルム・メモリ

3.5.1 まえがき

将来のファイルメモリとしては，磁気記録技術をベースとする現在のファイルメモリの延長線上にあるものと，光や電子ビームを用いた新しいメモリが候補としてあげられている。以下では，最初に現在の技術の延長がどこまで続くかという点を明らかにするために磁気記録技術の限界について述べ，次に新しいメモリの可能性と問題点を述べることにする。

3.5.2 磁気記録技術の限界

(1) 磁気ディスク装置

磁気ディスク装置の記録密度は，図3.5.1に示すように2年で2倍近く向上してきている。このような実績の外挿を今後どこまで続けることができる

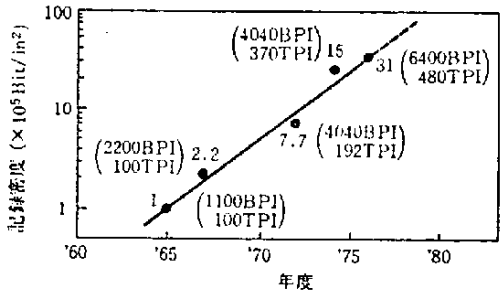


図3.5.1 塗付形ディスクにおける記録密度の進歩

かは，技術的要素以外のことも考慮しなければならないので明確にしにくいが，記録密度の限界がどこにあるかは図3.5.2から明らかになる。[1)]

図3.5.2は，現在までのトラック密度と線密度の進化過程と実用の観点からの限界を示している。xの線は現在まで用いられてきた塗付形媒体とフェライト形ヘッドの場合の推定限界を示し，yからzまでの間にリング形ヘッドによる記録方式の実用限界があることを示している。図から，現在では10^7

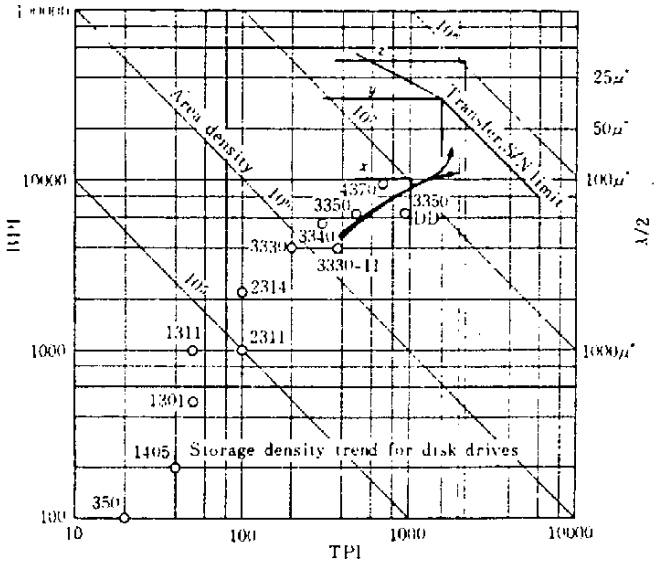


図 3.5.2　トラック密度（ＴＰＩ）と線密度（ＢＰＩ）の進化過程と実用限界

bits／in^2 近くの面密度が達成されており，限界は１０^8 bits／in^2 までの間にあることが分かる。すなわち，現在の技術の延長では磁気ディスク装置の記録密度は高々１桁の増加しか期待できない。

図 3.5.3は,記録密度と装置の大 きさを関連づけるために，市販の磁気ディスク装置の単位体積あたりの記憶容 量を面密度と共に示したものである。[1)]現在の磁気ディスク装置は，数１０MBのラック形と数１００MBの自立形に大別される。ラック形は自立形と比べて，単位体積あたりの記憶容量が同一の面密度ならば幾分大きい。これはラック形には大きさの制限があるために周辺部をコンパクトにするための種々の工夫がなされるからであり，自立形では大きさの制限が必ずしも強くないことからそのような工夫がなされないからである。１ドライブあたり数１０００MBの記憶容量を実現しようとする場合に，自立形ならば従来技術の改良と装置の大形化によって対処できるであろうが，ラック形ならば面記録密度を現状より１桁上げるためには，図 3.5.2の実用限界まで現在の技

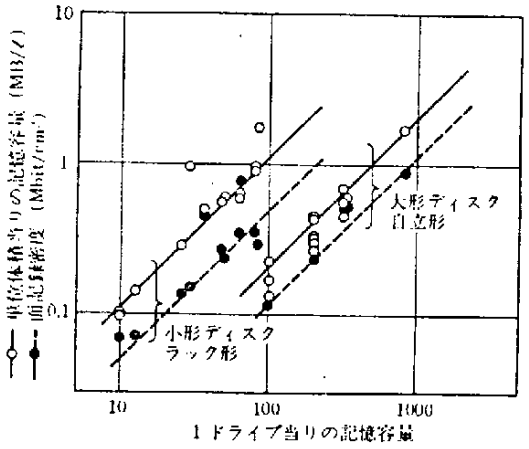


図 3.5.3　単位体積あたりの記憶容量，面記録密度と1ドライブあたりの記憶容量の関係

術を改良していく外に，何らかのブレークスルーが必要である。ブレークスルーとしては，垂直磁気記録方式や光磁気再生方式が考えられ，記録媒体の記録能力としては将来の記録密度の増加に対しても十分な潜在能力があると考えられている。

(2)　磁気テープ装置

IBM3850MSSやAmpex Terabit Memoryに見られるように超大容量を実現するためにVTR技術を用いた磁気テープ装置は，データベースの大容量化に伴なって今後ますます重要になるものと考えられる。

VTR技術の最近の進歩は，図3.5.4に示すように著しいものがある。図3.5.4は，VTRにおけるTV番組1時間あたりのテープ消費量がどのように減少してきたかを示すものである。[2)] 家庭用VTRの進歩は特に著しく，この技術が近い将来にコンピュータ用磁気テープ装置に転用されていく可能性は非常に大きいと考えられる。

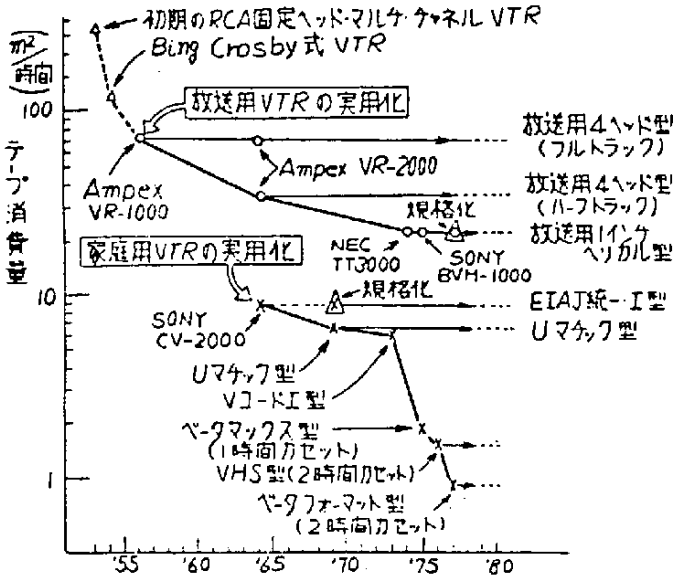


図 3.5.4　ＶＴＲにおけるＴＶ番組１時間あたりのテープ消費量の減少

ＴＶ１フレームあたりの情報量をディジタル換算で3×10^6 bits，１秒あたり約３０フレームと仮定すると，１時間あたりでは324×10^9 bits の容量になる。これだけの容量が１m^2に記録される場合，記録密度は３２.４Mbits／cm^2になる。この値は前述の磁気ディスク装置の面密度より１桁以上大きな値である。ＶＴＲはコンピュータ用磁気テープ装置よりも誤り率に関して条件がゆるやかであるし，アナログ記録方式なので，そのままの転用は無理であるが，この技術が低価格の大容量ファイルメモリに結びつく可能性は非常に大きいと考えられる。

3.5.3　新ファイルメモリの可能性と問題点

既に述べたように，磁気記録技術の今後の可能性はそれほど大きくなく，何らかのブレークスルーが必要と考えられている。このようなことから磁気記録以外の新しいメモリに期待が寄せられるのも当然である。

新しいファイルメモリとしては，光や電子ビームを用いたメモリが検討されている。光を用いたメモリはレーザ光の高干渉性を利用するホログラフィック方式と，レーザビームの収束性を利用するビット・バイ・ビット方式に大別される。電子ビームを用いたメモリはビット・バイ・ビット方式である。

図3.5.5は，ビット・バイ・ビット方式の光メモリと磁気記録メモリに関する面記録密度の将来予測の1例である。[3] 光メモリの記録密度の限界が磁気記録メモリの限界よりも2桁高いところにあるという予測は，極めて楽観的であり，ブレークスルーが実現されるという前提で考えられているものと思われる。実用という観点からは，光メモリの記録密度の限界は磁気記録メモリの限界よりも1桁程度高いと考えるのが妥当ではなかろうか。図3.5.5の予測でさらに注意しなければならない点は，磁気記録メモリが書きかえ可能であるのに対して，現在のところ光メモリは書きかえが困難であるという事実である。光メモリの研究が長期間にわたって行なわれたにもかかわらず，コンピュータ用としての実用化が進まないのは，この点にあるのである。もちろん，書きかえ可能な記録媒体の研究は今日まで続けられている。しかし，決定的なものは将来に期待せざるをえない状態にある。

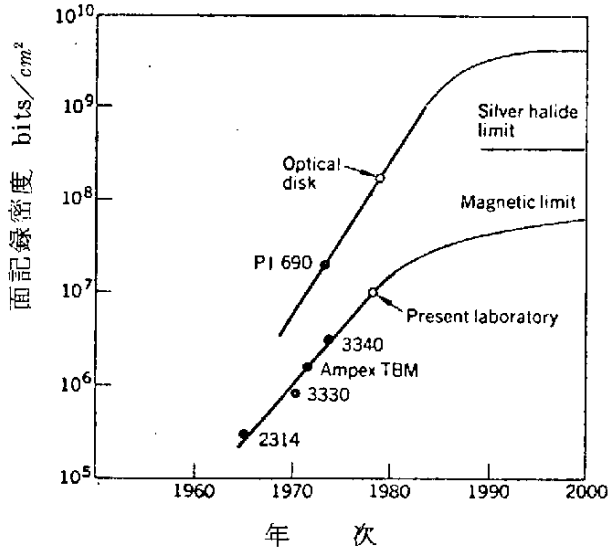


図3.5.5　記録密度の将来予測

ホログラフィック方式の光メモリの面記録密度は，ぼかし（defocusing）をとり入れたフーリエ変換ホログラムで10^6 bits/cm^2程度である。ランダム・フェーズ・シフタを用いると10^7 bits/cm^2程度になる。図3.5.5に示される記録密度と比べると，ホログラフィック方式の記録密度がビット・バイ・ビット方

式よりも必ずしも高いものではないことが分かる。ホログラフィック方式の利点は記録密度よりはむしろ，連想処理や多重蓄積が可能なところにあると考えるのがよいではなかろうか。[4]

電子ビームを用いたメモリは，記録媒体として連続的なＭＯＳ構造を持った読み書き可能なメモリと，[5] 記録媒体として電子ビーム用の感光材料を用いた書きかえ不可能なメモリ[6]が試作されている。電子ビームは光ビームよりも原理的には微小なスポットに絞ることができるので，高密度記録の場合には 記録媒体の問題は別にして，光ビームより有利であると考えられる。しかし，電子ビームは偏向の視野が狭く真空でなければならないので，それらを克服するために今後は記録媒体の研究に加えて実用をめざした装置化の研究も必要である。

3.5.4 むすび

将来のファイルメモリとしては，磁気記録技術をベースとする現在のファイルメモリの延長線上にあるものと，光や電子ビームを用いた新しいメモリが候補にあげられている。磁気記録技術をベースとするメモリの記録密度はあと１桁近くの向上を期待でき，ビット・バイ・ビット方式の光メモリは10年後から20年後の時点で磁気記録メモリよりも楽観的に見て２桁，実用的な観点からは１桁大きいという予測を述べた。

新しいメモリの場合，最も大切な点は実用に耐える書きかえ可能な記録媒体の開発にあると考えられる。

参考文献

1) 「現在の磁気ディスク技術はどこまで高密度化できるか」，応用磁気学会誌，Vol. 3, No. 2, pp. 50－55, 1979

2) 横山，「録画技術におけるマグネティクス」，応用磁気学会第６回研究会資料，pp. 41－48, 1978

3) G. C. Kemey et al,「An optical disk replaces 25 mag tapes」,

Spectrum, Vol. 16, №2, pp. 33−38, 1979

4) 石井，国分，「光メモリー光学技術を用いた大容量記憶装置の現状」，機械の研究，Vol. 25, №2, pp. 263−268, 1973

5) W. C. Hughes et al,「A Semiconductor Nonvolatic Electron Beam Accessed Mass Memory」, Proc. IEEE, Vol. 63, №8, pp. 1230−1240, 1975

6) S. Bousky et al,「Characteristics of Scanning Laser and Electron Beams in Bulk Data Storage」, IEEE Trans. Magn., Vol. MAG−7, №3, pp. 594−598, 1971

3.6 光関連技術

3.6.1 はじめに

光を用いたデータの伝送や，処理，記憶などについては，古くから，いろいろな可能性が検討され，レンズ系を用いた光学的変換などが研究されてきた。レーザの利用が可能になってからは，レーザ・ホログラムによる記憶や，連想処理が，活発に研究されるようになった。光を用いる通信については，昭和41年に，低損失の光ファイバが開発されてから，急速に研究開発がすすみ，実用化の段階に至っている。この光ファイバ通信は，光関連技術のうちでも，将来のコンピュータシステムに与える影響が，最も大きくなることが予測される。

このような理由から，本調査では，種々の光関連技術のうちから，光ファイバ通信をとりあげ，そのコンピュータシステムの影響について，調べることとした。

3.6.2 光伝送技術の特徴と現状

光伝送を支える技術は，従来の銅媒体に代る光ファイバケーブル技術と発光受光素子や光ファイバの接続に関する周辺技術に大別される。

(1) 光ファイバ技術

光ファイバを銅ケーブルと比較した時の利点を列挙すると以下のようになる。

① 低損失のため中継距離を長くできる。

② 広帯域のため伝送容量の向上がはかれる。

③ 電気・磁気的妨害（ノイズ）に対して強いため高信頼。

④ 軽量（比重が銅の１／４）。

⑤ 資源問題がない。

一方，光ファイバは細くもろいため被覆による保護が必要であり，ファイバの切断・接続には高精度の技術がいること，また光ファイバ通信には発光受光素子という新たな素子が不可欠などの欠点もあるが，技術開発によって解決される可能性が高い。

◦光ファイバの原理と分類

光ファイバは中心部にコアと呼ばれる高屈折率の部分とそれを取りまく低屈折率の部分から成っており，ファイバ内の光は全反射の原理によりコア内に閉込められて進む。特定の反射角を持った光線の組（モード）が１組のものをシングルモード，多数のモードを伝搬させ得るものをマルチモードファイバと呼んでいる。後者においては屈折率分布の形によりステップ形とグレーデッド形に分けられるが，後者の方が広帯域であるためマルチモードの場合の開発はほぼこの形式に絞られている。シングルモードは極めて広帯域であり，中継回線を対象としている場合には長距離の局間伝送に適しているが，ファイバへの光の入射やファイバ相互の接続が困難であるという欠点を持っている。図3.6.1に光ファイバの分類を示す。

(2) 周 辺 技 術

(i) 発光受光素子

光ファイバ用の発光素子は伝送路の一部として組み込まれることを考えて小型軽量である必要があり，半導体を利用したＬＥＤ（Light Emitting

	シングルモードファイバ	マルチモードファイバ	
		ステップ形	グレーデッド形
屈折率分布と光の伝搬の様子	屈折率分布 クラッド コア		
コア径	数 μm	40－100μm	40－100μm
比屈折率差(注)	0.1－0.3%	0.8－3%	0.8～1.5%
伝送損失	伝送損失は本質的には光ファイバ材料の組成成分により決り，上記分類には依存しない． 石英系光ファイバでは　～3dB/km（0.85μm），　～0.5dB/km（1.3μm）　である．		
伝送帯域	10GHz·km 以上	10－50MHz·km	数百M－数GHz·km
接続	困難（0.1μmオーダの精度が必要）	比較的容易（1μmオーダの精度）	比較的容易（1μmオーダの精度）

（注）　コアの屈折率(n_1)とクラッドの屈折率(n_2)の差を相対評価するときに用い，比屈折率差 Δ は次のように表される．

$$\Delta = \frac{n_1 - n_2}{n_1}$$

図 3.6.1　光ファイバの分類[1)]

Diode ）やＬＤ（ Laser Diode ）の研究開発が行われている。ＬＥＤは安価で，直線性が良く長寿命で信頼性が高いが，コヒーレンスに乏しく速度限界も低いため低速，短距離の用途に適している。一方ＬＤはレーザ発振によりコヒーレントな光が得られ出力が大きいが，平均寿命がほぼ１万時間と実用には不適当なため長寿命化が最大の研究課題となっている。表 3.6.1 に発光素子の現状値を示しておく。

表3.6.1　ＬＥＤとＬＤの比較

項　目	ＬＥＤ	ＬＤ
出　力	≃１mW	≃数mW
ファイバ入力	≃－15dBm	≃－３dBm
寿　命	≃100万時間	≃１万時間
消費電力	≃300mW	≃300mW
スペクトル幅	≃400Å	≃20Å
変調速度	数10Mb/sまで	数Gb/s

受光素子はホトダイオードが一般的であり，0.85μm帯においては既に実用段階にある。発受光素子については長波長帯域での開発が中心になっている。

(ii)　光コネクタ

光ファイバの実際的な応用においてはファイバ相互間の接続を行うコネクタ技術も見逃してはならない。光コネクタの低損失化はファイバのコアをコネクタの中心に設定する技術にあり，精密な機械的技術を必要とすることから種々の技術的改良が必要とされている。

3.6.3 将来の可能性と問題点

(1) 光ファイバ通信によるデータ伝送

光ファイバを利用した通信を通信距離の長さによって分類すると次のようになる。

(a) 遠距離データ伝送（大陸間）

(b) 中距離データ伝送（都市間）

(c) 近距離データ伝送（都市内）

(d) ビル内，構内データ伝送

これらは実用化の程度にもよるが，ほぼ現在の技術の延長上で達成可能であろう。個々の問題点としては(a)，(b)に関しては，シングルモードの開発や減衰をおさえるための長波長帯の利用，また海底ケーブルなどに関しては発光受光素子の寿命や信頼性を高める必要がある。(c)に関しては現在実用化段階にあるマルチモードファイバの技術を高めることによって１０Km程度の近距離においては価格や信頼性その他の面で銅ケーブルに十分対抗でき，この技術開発の程度により実用化のスピードアップが早まろう。(d)については光データリンクとしてすでに実用化段階にあり，種々の伝送速度と長さを持ったものが開発されている。この応用例としてはモデムインタフェースを介して各種の入出力端末を接続したり，構内ネットワークなどへの利用形態が考えられる。ただしネットワークを目的として利用する場合には，コネクタやタップをどのようにするかという問題があり，これは残された研究課題となろう。

全般的にみて光伝送技術は通信の手段としてみた限りでは既に実用化の一歩手前にあり，10年以内に種々の分野に普及していくことは確実であろう。

(2) コンピュータシステムへの応用

コンピュータシステムにおける光ファイバ通信の利用形態としては，広域ネットワーク，分散処理システムのような，伝送距離の大きなものと，メインフレーム近傍のごく短い区間の伝送に大別でき，前者については，(1)

で述べたように，公衆回線または，これに準ずる専用線として，実用化が確実視されている。後者については，将来のコンピュータシステムを考える上で重要な役割を果す可能性もあり，以下のように伝送距離により，分類してその実現可能性と問題点を検討する。

(a) ビル内，構内での分散処理システム（数十～数百メートル）

(b) 計算機室内，同一フロア内の密結合コンピュータコンプレクス（数メートル～数十メートル）

(c) メインフレーム内のＣＰＵ，メモリ，チャネル等のユニット間（数メートル以内）

(d) ＰＣボード間やＶＬＳＩとＰＣボード間

これらの用途については，現在の光ファイバ通信の研究主流からはずれているため十分検討されているとは言いがたいが，次のような現在の銅を線材とする方式に比べ原理的に優れた点が見出せる。

(i) 軽量（銅に比べ比重１／４），小型（細径で0.2㎜以下）

(ii) 無誘導，無漏話である

(iii) 広帯域（十数MHz～数＋GHz）

これらは，メインフレーム周辺の配線を考えるとき，ケーブルの高密度，高集積化や，直列転送によるケーブル数の低減，ケーブル・スペースの縮少など，種々のメリットを有する。しかしながら，同時に，解決すべき点も多く，以下のようなものが考えられる。

① ケーブルの高集積化に伴う，コネクタの高集積化

② 発光，受光素子などの高信頼化，長寿命化

③ 電気系と光学系間の高速変換技術の開発

④ 光学系内部での分岐，スイッチングなどの論理機能の実現

以上のような一般的な長所，短所をもとに，(a)～(d)の応用について，検討してみる。

(a)については，現在すでに，耐雑音性，低減衰性などの点で，同軸ケーブ

ル等より有利となっており，今後，計測，制御システムなどを中心に，急速に浸透していくものと考えられる。図3.6.2，図3.6.3，表3.6.2に光モデムの開発例を示す。

(b)の形態としては，周辺機器との接続やマルチプロセッサにおけるプロセッサ間接続が考えられ，このような用途には，①および②が不可欠である。①の問題に対する試みは，通信の分野でも一部試みられており，ベル研究所では図3.6.4に示すような高密度構造ケーブルを開発している。これは，12心の光ファイバから成るテープを１２層重ねたもので，合計１４４本のファイバからできている。その接続は，多心一括接続方式を採用することを前提としている。コネクタについては，図3.6.5に示すようなＬＥＤアレイをＤＩＰ素子に実装したものなどが提案されている。これらの多心のケーブルやコネクタは，まだ，伝送損失や信頼性，その他の問題が未解決であり，実用には至っていない。しかしながら，この種の高密度化，高信頼化がすすめば(b)のような用途には，十分使用でき，利点が多いものと思われる。

(c)のレベルにおける利用は，(a)，(b)に比べ，さらに高伝送速度を要求されることや，パケット転送などとは異なるバイト単位の非同期転送が要求されることもあり，③および④の，光素子技術や，光一電気系の変換技術が問題となる。転送速度については，ファイバ自体は，数十ＧＨｚの帯域を有し，直列転送におけるビットあたり転送は，１ ns／bit 以下も可能であり，実際，公衆通信の上限では，数Ｇ bits／sec も，検討されている。

現在のシリコン素子を用いた回路では，このような高速動作の実現は難しく，直列一並列変換を複数段挿入した，光一電気系変換などのバッファリングが必要である。このような用途に合った素子や技術は，実験室レベルでは，既に，伝送実験が為されているものの，これらは，長距離通信を目的としており，超小型化低価格化などの計算機用としての目的とは，必ずしも一致していない。また，高速の応答を必要とするような用途に対しては，変換系の時間遅れの問題もあり，今後の課題である。

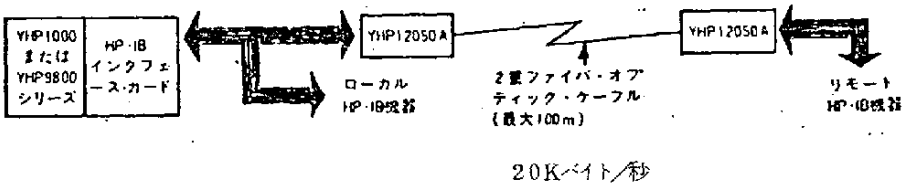


図3.6.2　YHP12050Aファイバ・オプティック　HP－IBリンク

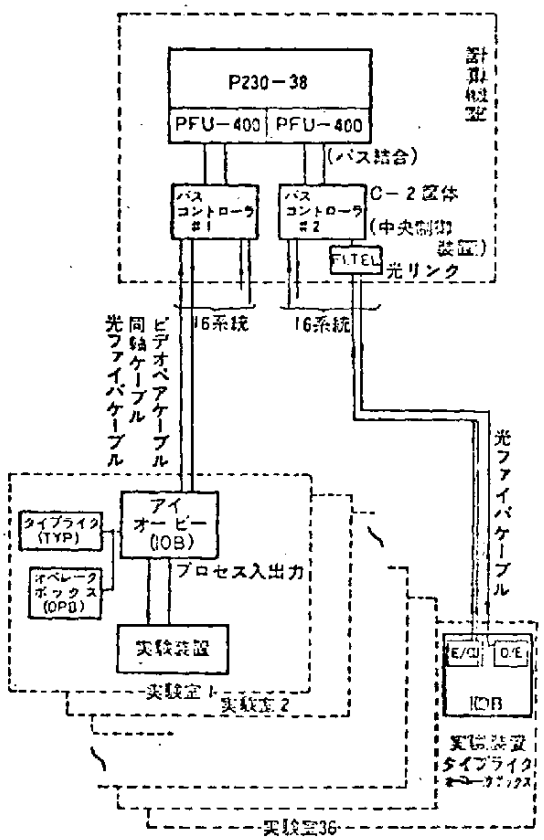


図3.6.3　光ファイバーによる実験装置の接続

表 3. 6. 2　光モデムの開発例

メーカ	製名	データ伝送速度	リング長	インタフェース	符号誤り部	電源	動作温度
富士通	FOPIC15-1Q10A	同期1.2～19.2kビット/秒 非同期20kビット/秒以下	2km	V-24, V-28	10^{-9}	AC100V, 50/60Hz	0～40°C
	FOPIC15-1Q21A	48kビット/秒	4km	V-35	10^{-9}	AC100V, 50/60Hz	0～40°C
古河電気工業	DIOS-50シリーズ	同期48kビット/秒 非同期20kビット/秒	4km	V-35 V-24, V-28	10^{-9}	AC100V, 50/60Hz	
三菱電機	A8903	2.4～48kビット/秒	3km	JIS C6361, V-35		AC100V, 50/60Hz	0～40°C
日本電気	DATAX OPT20	同期1.2～19.2kビット/秒 非同期20kビット/秒以下	500m, 2km, 4km	V-24, V-28		AC100V, 50/60Hz	0～45°C
	DATAX OPT120	48～128kビット/秒	500m 2km, 4km	V-35		AC100V, 50/60Hz	0～45°C
沖電気工業*	OIDM-002	同期/非同期 2.4～19.2kビット/秒	2km 4km	V-24			
住友電気工業	SUMILINK-K	同期/非同期 200ビット/秒～19.2kビット/秒	500m	V-24		AC100V, 50/60Hz	0～40°C
東京芝浦電気	TOSFIC FB20	1.2～19.2kビット/秒	5km			AC100V	
Valtec	RSK	20kビット/秒	1km	EIA RS-232C	10^{-9}	AC115V, 60Hz	0～55°C

*同様のものを藤倉電線(03)647-1111で取扱

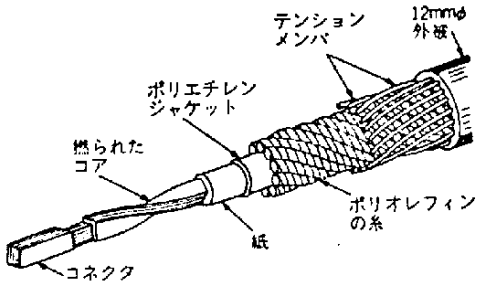


図 3. 6. 4　ベル研究所の高密度構造ケーブル

将来のコンピュータの構成が高度並列化の方向へすすみつつあることからも，④の分岐やスイッチング機能を光学系内部で行うことは，その速度などを生かす上で，魅力的である。この方面の研究も，通信における交換機などの実現を目ざして，いろいろ行われており，コンピュータシステム内で用いられる可能性も高い，しかしながら，現在の研究が，スイッチングに際し，レンズ系を機械的に移動させるものであるため，高速化には，限界があると考えられ，これに代る電気光学素子等の開発が期待される。

(d)のレベルの利用については，まだ未開拓の分野であるが，ＶＬＳＩのピンリミットの解決策としての，高速直列転送の利用や，ＰＣボード内へ光導

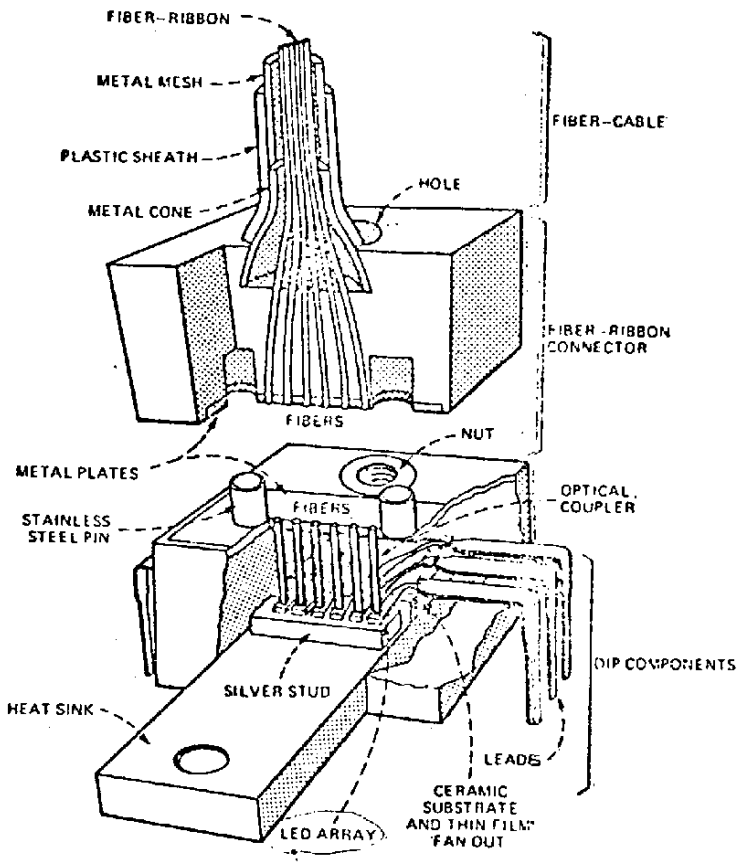


図 3.6.5　ベル研究所のＬＥＤアレイを用いたコネクタ

波路を構成する研究など，いまだ，アイデアの段階である。これらが現実のものとなるには，受光素子，発光素子や高速の光一電気系変換素子などの，高集積化が不可欠であるとともに，シリコンＬＳＩなどとの整合性の問題を解決する必要があり，このためのインパクトが，通信の分野における実用化から，どの程度得られるか不明である。通信の分野における実用化がすすめば，研究対象になってくるものと思われる。

3.6.4　ま と め

光ファイバ通信のコンピュータシステムへの応用として，通信回線とメインフレーム周辺に分けて調査した。通信回線については，１０年以内で，確実に

実用化が進んでいくと思われ，分散処理ネットワークなどは，この恩恵を受けるものと思われる。計算機室内や，メインフレーム周辺については，現在のブロック転送やパケット転送方式の使用できる範囲では，十分実用化され，さらに，銅を材料とするケーブルより優れたものとなろう。メインフレーム内部については，ケーブルやコネクタの高密度実装が鍵となるほか，光学一電気系間の変換素子などに，優れたものができれば，一部実用化の可能性もある。しかし，高速応答を必要とする非同期データ伝送等には原理的に不向きであり，その使い方が限定されると思われる。この点については，パケット交換を基本とするデータフローマシン等では，一応の可能性があるかもしれない。変換には，バルク効果素子の使用もあり得る。

また，光学系内でのスイッチングや論理機能の実現が期待される。これらについては，いまだ未知数であるが，交換機等通信における需要の影響如何では研究の進展の可能性も考えられる。

いずれにしても，光ファイバ通信は，原理的に秀れた点を有し，距離の長いものから短いものへと順々に，銅線を用いた伝送に，置きかわっていくものと考えられる。

参考文献

1) 「光ファイバ通信特集」，施設，Vol．31，№1，日本電信電話公社

2) 島田，三木，「光伝送方式」，テレビジョン学会誌，Vol．32，№4，pp．277—285，1978

3) 「応用範囲が広がり，需用が立上り始めた光ファイバ伝送」，日経エレクトロニクス，№220，p．84，1979

4) Albanke A, et al, 「LED Array Package for Optical Data Links」, Bell Syst. Tech. Jour., Vol. 58, №3, pp. 713, 1979

5) 西地，「光I／Oインタフェイスの検討」，信学会通信方式研資，CS78—128，1978

6) 中村，「光IC技術」，電子技術，Vol．21，№1，pp．75，1978

3.7 まとめ

１０年後のコンピュータ・アーキテクチャに対するデバイス技術からのインパクトは以下のように要約される。

(1) 最も大きなインパクトを与えるものは広義のＬＳＩ技術（シリコンだけでなく，ガリウム砒素，磁気バブル，ジョセフソン素子等を含む）である。

(2) 中でもＲＡＭ，ＲＯＭ，ＰＲＯＭ等メモリの低価格化，大容量化によりＣＰＵの構成にインパクトが与えられるだけでなく，周辺機器その他にも広くメモリが用いられるようになり，機能的な高度化が容易となる。

(3) 論理装置においてはデバイス技術の提供する集積度を十分生かすためには方式技術，設計技術の進歩が必要である。設計のためのコストがデバイス製作のコストにくらべて大きな部分を占めるようになり，このことが多種のＬＳＩを作ることの制限要因となるだろう。

(4) ファイルメモリの性能は漸次改良進歩するが飛躍的な進歩は期待できない。むしろこの種の装置にＬＳＩ技術を組合せることにより，機能的なシステム構成を実現することが具体化するだろう。

(5) 光情報処理技術はまず通信の分野で実用化する。さらにインハウスのコンピュータ間接続に大幅に用いられるようになるだろう。このためにシステム構成にかなりのインパクトを与えると考えられる。

第4章　計算機アーキテクチャ技術

第4章　計算機アーキテクチャ技術

将来の計算機の開発に際しては，過去の計算機アーキテクチャにとらわれることなく，現在の計算機がかかえる諸問題を効果的に解決する新技術を積極的に導入すべきである。新しい計算機アーキテクチャの具体的な問題点は以下に例を示す個々の技術に関する検討項目に含まれている。それらは，

(1) 新しいデータ処理に関する技術

(i) パターン処理および画像処理

(ii) 人工知能および知識工学

(iii) 問合せ言語および自然言語

(2) 新しい考え方に基づく技術

(i) データフロー・マシン

(ii) 適応計算機ならびに学習機構

(3) 最近の研究課題

(i) データ構造とデータベース

(ii) マルチプロセッサと並列処理

(iii) 連想処理

(iv) 分散処理

(v) 計算機ネットワーク

(vi) 高級言語マシン

(vii) エミュレーションと仮想マシン

(viii) スタック・マシン

(ix) タグ・マシン

(x) ハッシュ機構

(xi) フォールト・トレラント計算機

(4) ソフトウェアに関する技術

(i) オペレィティング・システム

(ii) システム記述言語

(iii) ソフトウェア開発技法

(5) 応用分野に関する技術

(i) 超高速科学計算機

(ii) 記号処理向き計算機

(iii) 特殊計算機および高機能シミュレータ

(iv) オフィス・オートメーション

(v) 新しい応用—宇宙航空・ホームコンピュータなど

(6) 関連技術

(i) 新しいデバイス

(ii) セル構造論理および論理記憶

(iii) 設計自動化

(iv) システム評価

(v) RAS

(vi) オペレーション・運用

などである。

本章では今年度討議を行なった検討項目について，技術的な発展経過，将来展望ならびに実現に要する技術課題などを述べる。

検討した技術課題は大別して，

(a) 並列処理技術

(b) 機能分散技術

(c) その他の技術

の3分野である。

(a)について

(ア) 従来の逐次制御方式に基づいた，いわば古典的な並列計算機の代表で

あるマルチプロセッサ

(イ) データ駆動制御方式に基づいた新しい並列計算機の代表であるデータフロー計算機

(ウ) 並列処理技術応用の代表である科学技術計算機

について検討を行っている。

(b)について

(エ) 今後の計算機技術の中核になると考えられる機能分散型計算機

(オ) 各種の機械語命令セット・アーキテクチャを実現するユニバーサル・ホスト・プロセッサ

(カ) 高級プログラム言語処理を指向したアーキテクチャをもつ高級言語マシン

(キ) データベースの効率的な処理を指向する専用計算機，データベース・マシン

(ク) 機能的に分散した専用計算機あるいは汎用計算機間の接続技術であるコンピュータ・ネットワーク

について検討を行っている。

(c)については種々のものが考えられるが，今年度は

(ケ) 与えられた問題に対してより適合したアーキテクチャを自動的に生み出す適応計算機

(コ) ソフトウェア工学の立場から見たアーキテクチャのあり方

などについて検討を行っている。

なお，今後の課題としては，今年度検討できなかった技術課題についての徹底的な調査，よりミクロな計算機アーキテクチャ・レベルでの共通重要課題の抽出，実現のための具体的な方法などの検討が重要である。

4.1 並列処理

4.1.1 概　説

並列処理とは，１つのＪＯＢを，複数個の独立実行可能な処理単位に分割し，これらを複数個の装置を使用して，同時に並行して処理することを言う。１時点で１つの処理しか実行しなければ，計算機システムの処理速度は素子の動作速度を越えることができない。よって，高速処理実現のためには，何らかの形で並列処理を行う必要がある。

並列処理の計算機システムへの導入は，次のようにとらえることができる（表4.1.1参照）。

(1) 複数の処理装置による並列処理

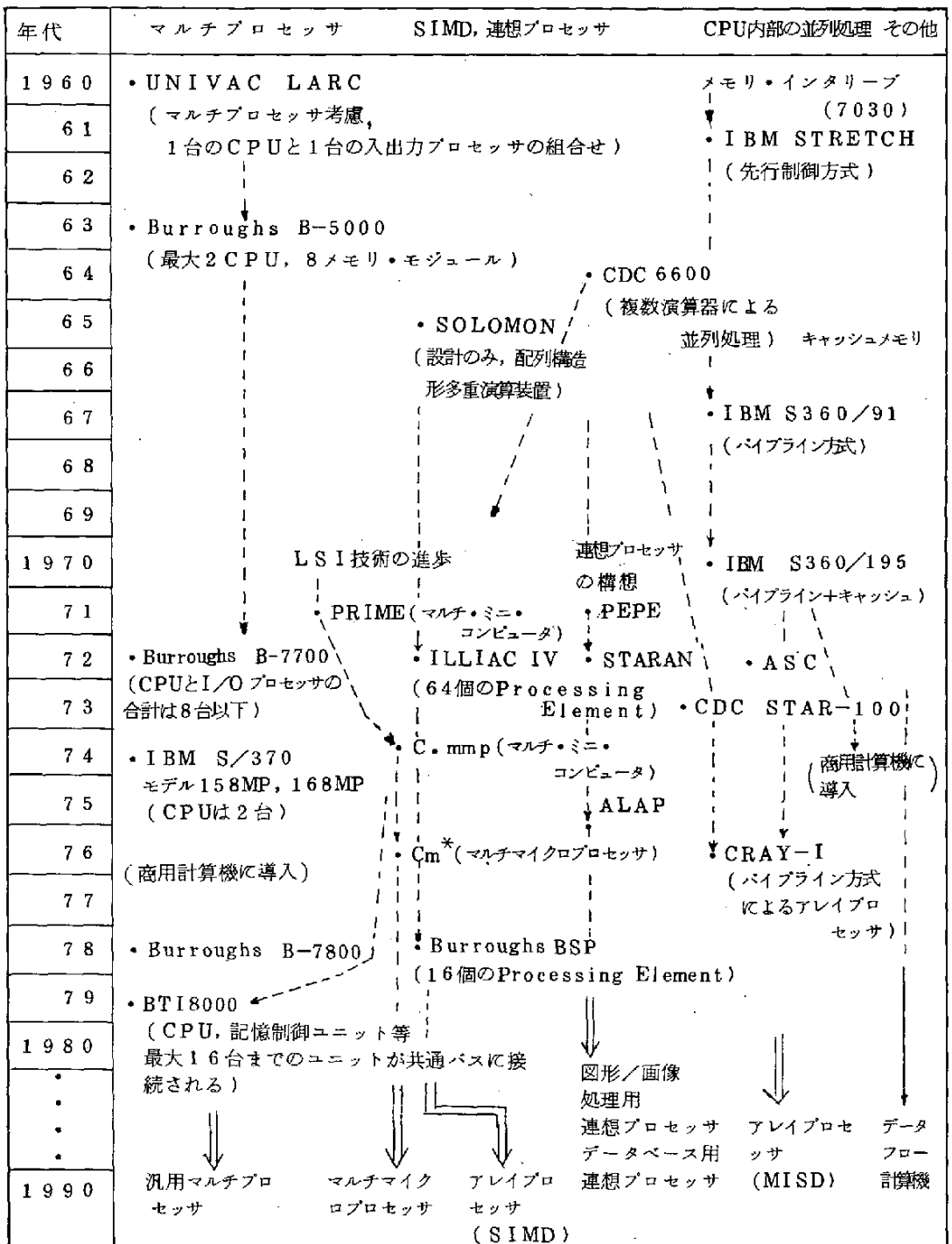
この考え方は，入出力装置を中央処理装置（ＣＰＵ）から分離し，ＣＰＵ処理と入出力処理を同時に処理することから始まった。その後，これに引続いて複数の処理装置を密結合した各種のマルチプロセッサシステムが出現した。

(2) ＣＰＵ内部の並列処理

① 処理装置の空間的配列による並列処理ではなく，時間的な並列処理を狙い，命令フエッチと命令実行をオーバラップさせる先行制御の技術が導入された。先行制御は，高速キャッシュメモリの進歩と共に，より高度なパイプライン方式へ発展した。現在は，命令フエッチと命令実行のオーバラップだけでなく，命令実行で使用する演算器での内部処理のオーバラップも行われており，パイプライン方式を積極的に利用した高速科学計算用プロセッサも実用化されている。

② ＣＰＵ内部に複数個の演算器を持つ方式も考え出された。例えば，加算器，乗算器を複数個持ち，ｃ＝ａ＋ｂ，ｄ＝ｅ＊ｆのような処理を同時に

表 4.1.1　並列処理システムの歴史〔参考文献 1), 2) 等による〕



実行するものである。これを更に発展させ，行列や偏微分方程式等の計算を効率的に行うために，多数の演算装置（ＰＥ：Processing Element とも呼ぶ）を２次元に配列したシステムが出現した。この代表例がＩＬＬＩＡＣ　ＩＶであり，このシステムでは，６４個のＰＥを１個の制御装置で制御する方式を採用している。

(3)　記憶装置内部での並列処理

記憶装置内にあるデータを番地で指定してＣＰＵでシーケンシャルに処理するのではなく，与えられたデータの内容と，記憶装置内の全データ内容を同時に比較し，条件を満した全データに対して共通の比較的単純な処理を行う方式が考え出された。これは，連想処理と呼ばれ，ＣＰＵで行う処理機能の一部が記憶装置の中に入り込んだものである。

(4)　多数のＬＳＩプロセッサによる並列処理

２～４台の処理装置から成るマルチプロセッサや，ＣＰＵ内部のパイプライン方式の技術は，１９７０年代には，ほぼ確立された。その後，ＬＳＩ技術の発展に伴ない，同種のマイクロプロセッサを数十～数千個結合したＬＳＩ指向の並列処理システムが経済的に可能である見通しが得られ，現在研究実用化が進められている。

(5)　データ駆動による並列処理

これまでのシーケンシャル制御方式の代りに，データの流れに着目しデータ駆動による新しい制御方式を採用したデータフロー計算機が，現在，新たな並列処理システムとして脚光を浴びている。

並列処理を求める要因は，次の２点に存在する。

(1)　処理速度の絶対的向上

限られた時間内での処理が要求される場合（例：ミサイルの弾道計算，天気予報，多量のデータの中からの情報検索等）は，ハードウェア素子の動作速度以上の高速処理を行うために並列処理技術が必要となる。

(2)　価格性能比の向上

ＬＳＩ技術の進歩により出現した，安価ではあるがそれほど高速でないマイクロプロセッサ等を構成要素とし，並列処理技術により高性能なシステムを作成する。

4.1.2　並列処理の原理と問題点

(1)　アーキテクチャからの分類と原理

並列処理システムのアーキテクチャは，Flynnによれば，次のように分類できる。[3)]

①　単一命令ストリーム・複数データストリーム　　ＳＩＭＤ

②　複数命令ストリーム・単一データストリーム　　ＭＩＳＤ

③　複数命令ストリーム・複数データストリーム　　ＭＩＭＤ

ＳＩＭＤは，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（データの同時処理），図4.1.1のＩＬＬＩＡＣ　ＩＶに見られるようなアレイプロセッサに相当する。また，多数のデータの中から，ある特定の条件を満たすデータを同時に発見し，そのデータに対して比較的単純な処理を行う連想プロセッサも，この分類に入る。

ＭＩＳＤは，単一のデータストリームに対して複数の命令が作用するもので，図4.1.2のように，データが流れ作業で処理される形式を想定できる。また，図4.1.2のパイプライン・プロセッサは，それぞれ特定の機能を処理する複数の装置が直列に結合されたものである。

ＭＩＭＤは，複数個の独立した命令が，それぞれ異なるデータストリームを処理するもので，図4.1.3のマルチプロセッサが代表例である。データフロー計算機もこの分類に入る。

(2)　ソフトウェアからの分類と原理

上で述べた並列処理アーキテクチャ上で実行されるプログラムの並列処理のレベルは，次の3レベルに分類できる。[4)]

①　命令またはステートメント・レベル

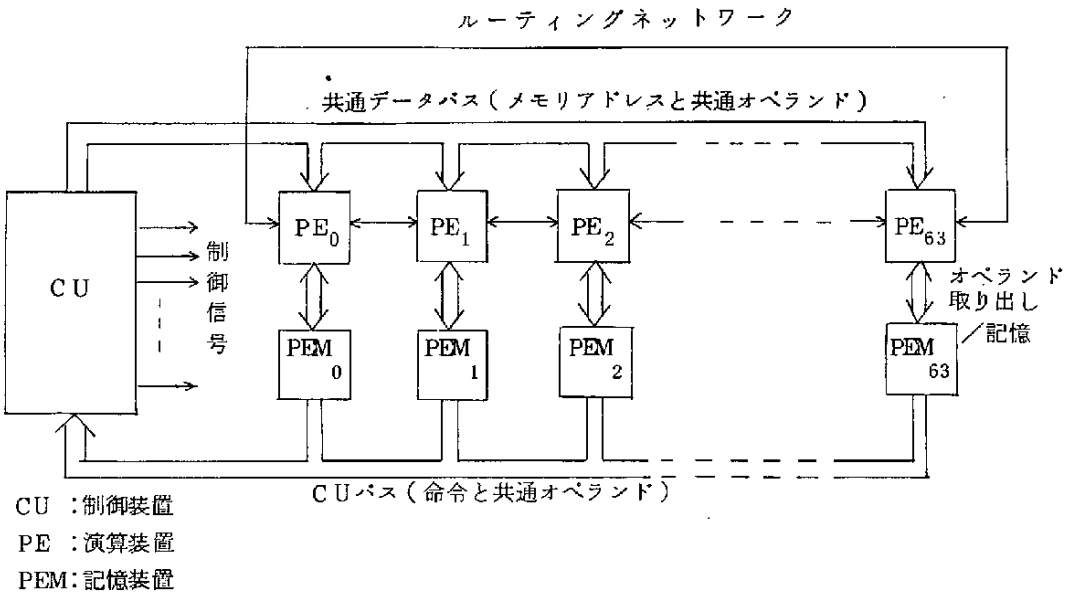


図 4.1.1　ＩＬＬＩＡＣⅣの概念図（1Ｑｕａｄｒａｎｔ）〔ＳＩＭＤ〕

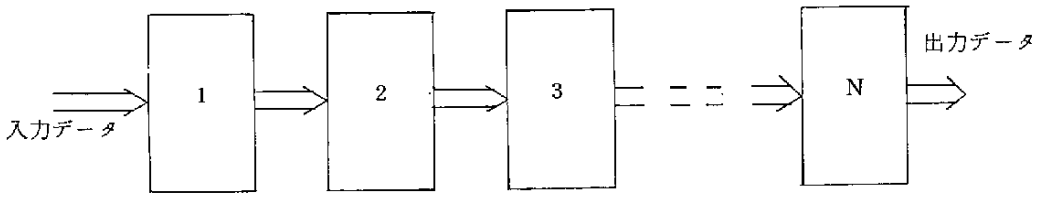


図 4.1.2　パイプラインプロセッサの概念図〔ＭＩＳＤ〕

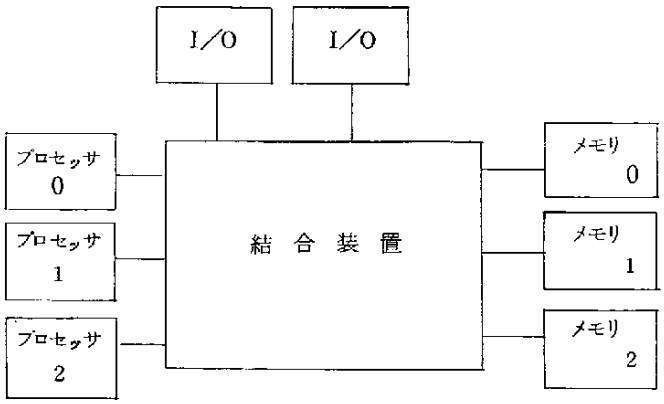


図 4.1.3　マルチプロセッサの概念図〔ＭＩＭＤ〕

プログラム中の命令またはステートメント・レベルで並列処理する（例：A＝B＊C＋D＊Eや，DO，WHILEステートメント・レベル）。

② タスク・レベル

プログラム中のサブルーチン（タスク）・レベルで並列処理する。

③ サブシステム・レベル

プログラム中のサブシステム・レベルで並列処理する（例：入出力，通信処理サブシステム・レベル）。

それぞれのレベルで分割されたプログラム単位（並列処理単位）がどのように実行されるかにより，図4.1.4に示す2方式に分類できる。

① 同時処理 ——→ 応答時間を短縮できる。

② パイプライン処理 ——→スループットを向上できる。

(3) アーキテクチャとソフトウェアの関係

プログラムの同時処理はＳＩＭＤ，ＭＩＭＤアーキテクチャ上で，パイプライン処理はＭＩＳＤ，ＭＩＭＤアーキテクチャ上で実現できる。

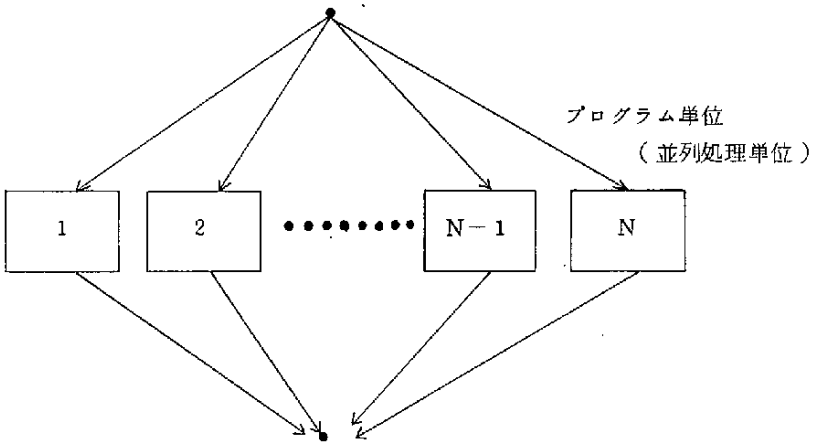
並列処理システムでは，プログラム（ソフトウェア）とアーキテクチャの並列構造が一致して，はじめて高度の並列性を発揮できる。プログラムの並列構造とアーキテクチャの並列構造間の写像を行う方法は，次のように分類できる。

① プログラマが，アーキテクチャの並列構造を直接意識し，写像用命令（同時命令等）を使用して写像を行う。

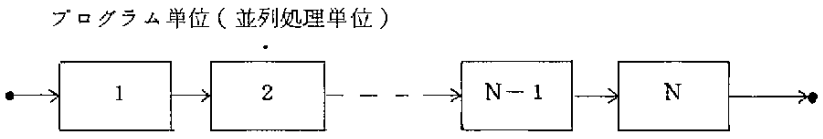
② プログラマが，アーキテクチャの並列構造を意識する必要はなく，コンパイラ等のソフトウェアが，自動的にプログラム中の並列構造を検出し，写像を行う。

③ プログラマが，アーキテクチャの並列構造を意識する必要はなく，ハードウェアが自動的にプログラム中の並列構造を検出し，写像を行う。

③が理想的であるが，まだ実現されていない。しかし，関数型プログラミングにより記述されたプログラムの並列性をハードウェア・アーキテクチャ



①同時処理



②　パイプライン処理

図 4.1.4　プログラムの実行の流れから見た並列処理の分類

が自動検出し実行するデータフロー計算機のアプローチがこれに近い。このアプローチでは，実現性はもちろん，関数型プログラミングにより，どの分野のどの範囲までの処理を記述できるのかを明確にしていく必要がある。

②では，ステートメント・レベルの並列性を検出するＦＯＲＴＲＡＮコンパイラが実用化されているが，まだ十分とは言えず，並列処理アルゴリズム，並列検出アルゴリズムの研究をより進めていく必要がある。*

したがって，現在は①が主体であり，プログラマは，ＦＯＲＫ，ＪＯＩＮ，セマフォなどの命令を用いて，並列処理単位の指定，実行順序，並列処理単位間の同期や，データ授受方法等をプログラム中に明示する必要がある。このためのプログラマの負担は大きい。よってこれまで以上に記述性がよく，Readability のよい並列処理記述用言語や図形表現によるプログラム作成技術の開発が重要である。

(4) 並列処理システムで解決すべき事項

並列処理システムを実現するためには，以下に述べる問題点を解決する必要がある。

① 処理すべき問題の並列処理可能構造への分解
② 並列処理記述言語の開発
③ 並列性自動検出アルゴリズム
④ 複数のプロセッサ（装置）へのプログラムおよびデータの割当て方法（入出力方法も含む）
⑤ プロセッサ間の通信と同期方法，割り込み処理方法
⑥ 資源の共用と保護方法
⑦ ローカルメモリと共用メモリに格納する情報の割当て方法
⑧ 故障の検出と，再構成方法
⑨ 負荷の配分，ボトルネックの検出と解消方法
⑩ プロセッサの結合方法やシステムの実装方法
⑪ 並列処理システム全体を管理するオペレーティングシステム

など。

＊ 並列処理アルゴリズム，並列性検出アルゴリズム等については，基礎理論研究分科会で調査，研究が進められている。

特に①をはじめとするソフトウェアサイドからの研究成果が，今後の新しい並列処理システム開発のカギとなるものである。

4.1.3 並列処理の適用と効果

現在までの並列処理の適用と代表的なシステムを表4.1.2に示す。

(1) ＳＩＭＤ型並列処理

現在は，汎用アレイプロセッサとして実現されている。処理速度向上効果は大であるが，アーキテクチャの並列動作を最大限に発揮するようにプログラムを記述するのが難しい。アレイプロセッサは，次のようなシステムで利用されている。

① 図形／画像／信号処理システム

② 資源発見／地震予知システム

③ 広域気象予測，熱力学，原子力，ミサイル，船舶，航空機等のシミュレーションシステム

④ 医療情報処理システム

⑤ 構造解析，統計解析システム

⑥ 防衛システム

など。

また，①～⑥のシステム用のＳＩＭＤ，ＭＩＭＤ型専用プロセッサも実用化されており，今後，プロセッサは多様化すると考えられる。

一方，連想プロセッサは，現在，航空管制システムに利用されている。今後も，図形／画像処理システム等をはじめ，次のようなシステムに利用されよう。

① ディスク等の補助記憶装置と結合されたデータベース処理システム（ロジック・イン・デバイス）

② 主記憶と結合されたソートやデータ検索システム（ロジック・イン・メモリ）

表4.1.2　現在の並列処理の適用と代表的なシステム

分類	説明		並列処理レベル	代表的なシステム	適用分野
MISD	CPU内部の並列処理（パイプライン）		命令またはステートメント・レベル	（大部分のシステムに採用されている）	（ソフトウェアからは，アーキテクチャの並列構造は見えない）
	パイプライン方式を用いたアレイプロセッサ			CRAY－1 STAR－100 ASC　など	・図形／画像／信号処理
SIMD	プロセッサ・アレイ型の並列プロセッサ（アレイプロセッサ）			ILLIAC－IV BSP DAP　など	・大規模科学技術計算処理
	連想プロセッサ			STARAN PEPE ECAM ALAP　など	・データベース処理
MIMD	データフロー計算機			LAU System／1 AMPS DDM1　など	（実験段階）
	複数台の計算機を結合したマルチプロセッサやボリプロセッサ，マルチマイクロプロセッサ	密結合 共通バス	タスクまたはサブシステム・レベル	BTI8000 SMS201　など	・汎用マルチプロセッサによる汎用処理
		密結合 マトリックスイッチ		C.mmp B7800　など	・多数プロセッサによるシミュレーション，天気予報などの特殊分野の処理
		密結合 マルチポート		UNIVAC 1100／80 IBM370／168　など	
		密結合 階層構成等		Cm* など	
		疎結合		HXDP　など	（並列処理の色彩はうすくなる）

処理速度向上効果は大であるが，連想処理できるデータ量を増加させる必要がある。

(2) ＭＩＳＤ型並列処理

現在は，ＣＰＵ内部における命令の解釈実行処理，演算装置内部における算術演算処理に，パイプライン制御が導入されている。このレベルでは，プログラマは，アーキテクチャの並列構造を特に意識する必要はない。パイプライン制御により同時処理される命令数は5命令以上になってきており，制御技術はほぼ完成された。今後は，汎用コンピュータにおけるパイプライン処理による飛躍的な性能向上は期待できない。

一方，パイプライン制御を導入した汎用アレイプロセッサは，(1)のアレイプロセッサと同じように使用されている。ただし，高速処理を達成するためには，高速の素子を利用する必要があり，安価でそれほど高速でないＬＳＩプロセッサを有効に利用できない。

(3) ＭＩＭＤ型並列処理

汎用プロセッサとして，①負荷分散プロセッサ，②機能分散プロセッサ，が実用化されている。これらのプロセッサで並列処理されるレベルは，タスクまたはサブシステム・レベルである。処理速度向上という観点からとらえると，汎用的な問題では，①プログラムを多数の並列処理単位に分割することが困難であり，②論理資源（制御表やファイル）の競合がプロセッサ台数を増加させると無視できなくなるため，１０台以上プロセッサ台数を増加させても効果がないと言えよう。

一方，マイクロプロセッサを多数結合したマルチマイクロプロセッサも研究実用化が進められてきているが，汎用プロセッサを狙っても性能向上効果は小さい。むしろ，タスク・レベル以下での並列処理を行う専用プロセッサとして実現する必要がある。ただし，多数のプロセッサを管理するＯＳをどうするか，障害の検出とシステムの再構成をどうするか等の問題点の解決と，多数のプロセッサの結合方式技術を開発する必要がある。なお，データフロー計算機については，4.3節で述べる。

4.1.4 今後の課題

並列処理システムは，以前から研究が進められているが，並列処理システムをサポートするハードウェア技術が不十分だったために，実際に実用化されたものは少なかった。しかし，ＬＳＩ技術の進歩により，特にくり返し構造を持つ並列処理システムは，ハードウェアの面からも実現が可能になり，すでに並列処理アルゴリズムが明らかにされている分野では，着実に並列処理システムが実用化されるものと考えられる。

一方，新しい並列処理システム実現のためには，

① 処理すべき問題の定義

② その中で，並列処理できる部分の発見，または，並列処理が必要な部分の抽出

③ 並列処理アルゴリズムの確立

という手順を踏む必要がある。アーキテクチャ（ハードウェア）のみを先に開発し，その後，その上で処理する問題も発見するアプローチは，混乱を招く恐れもあると考えられる。

なお，並列処理を適用した具体的なプロセッサについては，4.2節以降で述べる。

参 考 文 献

1) 松崎稔，田中善一郎：「プロセッサ・ハードウェアの発展経緯と今後」，日経エレクトロニクス，１９８０年1月7日号，pp. １５９−１７２

2) P. H. Enslow：“Multiprocessor Organization − A Survey”，ＡＣＭ Computing Survey，Vol. 9，№1，pp. １０３−１２９（１９７７）

3) M. Flynn：“Some Computer Organizations and their effectiveness,” ＩＥＥＥ Trans. on Computers, Vol. C-21, №9, pp. ９４８−９６０（１９７２）

4) 村岡洋一：「並列処理」，情報処理，Vol. ２０，№4，pp. ２８９−２９４（１９７９）

4.2 マルチプロセッサ

4.2.1 概　説

(1) マルチプロセッサの定義

マルチプロセッサを，2台以上のプロセッサが論理的，物理的に結合され，1つの目的に向けて有機的に統合，制御されるシステムと定義すると，ほとんど大半のシステムがこの中に含まれてしまう。マルチプロセッサは，図4.2.1の観点から分類できる。したがって，ここで言うマルチプロセッサは，表中の点線で囲んだもの，すなわち，MIMD方式の密結合負荷分散マルチプロセッサシステムとする。

(2) マルチプロセッサシステムの狙い

① 信頼性の向上

システムを複数のプロセッサで構成することにより，1つのプロセッサが故障してもフェイル・ソフトな運転を可能とし，システムの可用性を向上させることができる。

② システム再構成能力の向上

プロセッサの追加，削除ができるようにシステムを構成することにより，システムの拡張性，柔軟性を向上させることができる。

③ 資源の共用利用

④ 処理能力の向上

複数のプロセッサを並列に動作させることにより，単一プロセッサでは得られない処理能力を得ることができる。

⑤ 価格性能比の向上

安価ではあるがそれほど高速でないプロセッサを複数個使用して並列処理を行わせることにより，価格性能比を向上できる。

なお，マルチプロセッサシステムの最初の実用化や研究の目的は，①にあ

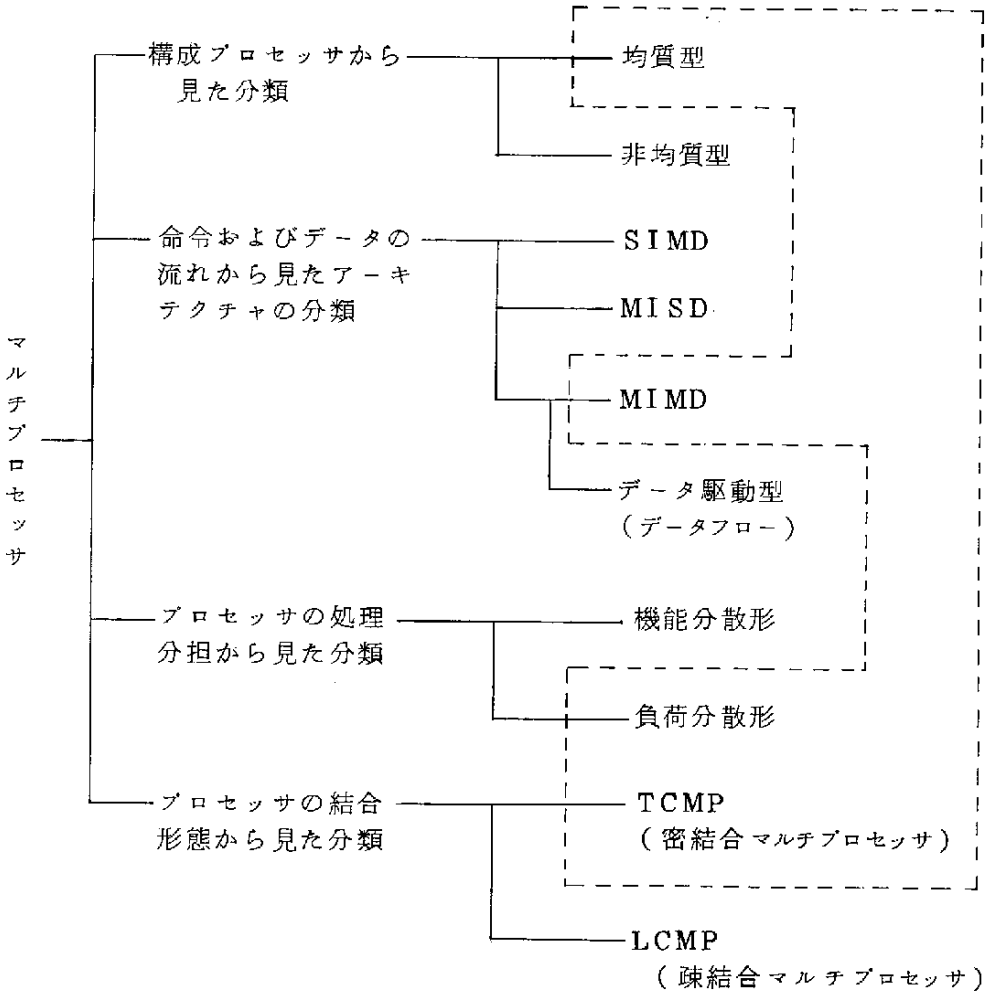


図4.2.1　マルチプロセッサの分類

ったことを付け加えておく。

(3) マルチプロセッサシステムの歴史

歴史を表4.2.1に示す。マルチプロセッサシステム開発の歴史をさかのぼると，マルチコンピュータシステムに行きつく。表4.2.1には，(1)で定義したもの以外についても参考として示している。

4.2.2 主な技術と問題点

マルチプロセッサシステムを実現する上での主な技術と問題点を述べる。

(1) プロセッサの結合方式[1), 2)]

プロセッサの結合方式は，次のように分類できる。

① 単一バス方式

② 多重バス方式

③ 専用バス方式

④ 環状バス方式

⑤ 階層バス結合方式

⑥ マトリックス・スイッチ方式

⑦ マルチポート方式

⑧ 中継論理装置（プロセッサ）結合方式

各方式の構成図と定性的な評価，問題点を表4.2.2に示す。

単一バス方式は，構成法が簡単であり，結合のためのハードウェアを安価にできるため，主に小規模計算機システムに用いられており，マルチポート方式は，プロセッサ間の転送能力を大きくすることができるため，大規模計算機システムに用いられている。

バス方式ではバスの使用権の制御が必要であり，これは，①バスの使用要求を共通の制御線を用いて制御するディジーチェイン方式，②ポーリング方式，③個別に要求信号と使用許可信号をやりとりする個別要求方式（非同期方式）等に分類される。

表 4.2.1　マルチプロセッサの歴史〔参考文献 1)等による〕

時期	主なマルチプロセッサシステム
1958	• National Bureau of Standards（総合運転できる3台の独立した
59	• IBM AN/FSQ-31, 32（マルチプロセッサではなく，コンピュータ）デュプレックスシステム）
1960	• Burroughs D-825（同一プロセッサを組合わせた最初のモジュラー・システム）
61	• UNIVAC LARC（CPU1台，入出力プロセッサ1台，本格的マルチプロセッサではない）
62	
63	• Burroghs B-5000 • IBM 704X/709X（直接結合システム）
64	• IBM MSC（宇宙センタ用特別設計のマルチプロセッサ） • Burroughs B−5500　• CDC 6600（不平等マルチプロセッサ）
1965	• GE645　• UNIVAC 1108
66	• IBM S/360 モデル67　• CDC 6500（特別設計のTSSシステム））
67	• CDC 6700　• XDS Sigma 5
68	
69	• CDC 7600
1970	• Burroughs B5700 • CII IRIS 80
71	• Honeywell 6050, 6060, 6080　• XDS SIGMA8, 9 • DEC システム 10/1055, 10/1077　• UNIVAC 1100
72	• CDC CYBER 72, 73, 74, 76　• TOSBAC5600　• HITAC8800, 8700
73	• PLURIBUS（特に信頼性向上をねらったマルチミニコンピュータ） • FACOM230-75
74	• IBM S/370 モデル158MP, 168MP　• FACOM230-58 • CDC CYBER175　• C.mmp（16台のマルチミニコンピュータ）
1975	• Tandem T16　• ACOS77/600, 700　• UNIVAC1100/20, 40 • UNIVAC 1100/20, 40
76	• IBM S/370 モデル158AP, 168AP　• M−190 • DEC システム 10/1088　• ACOS77/800
77	• Burroughs B−6800　• MELCOM-COSMO900 • M−180　• ACOS77/800
78	• ACOS77/900
79	• Burroughs B-7800　• M−200　• IBM3031, 3033MP, AP • BTI8000　AP

表 4. 2. 2　マルチプロセッサの結合方式(1)

（P：プロセッサ，M：メモリ，K：周辺装置）

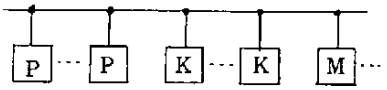
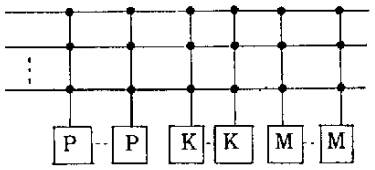
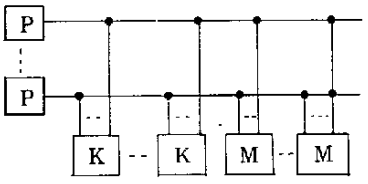
分類		結合方式の概略図	定性的評価
1	単一バス方式		• 経済的な方式であり，拡張性，融通性に富んでいる。 • バス使用における競合が問題である。 • バス要求間隔が，バスの転送時間に比べて比較的長い場合に有効である。 • バスの障害がシステムのダウンに直接つながる。 • 小規模システムで採用されている。
2	多重バス方式		• 単一バスに比較して，バス使用における競合が減少し信頼性も向上する。
3	専用バス方式		• 多重バス方式で，バス数がプロセッサ数と一致した時に，各プロセッサごとにバスを固定的に割当てた方式である。 • システムの拡張性に問題がある。 • バス障害は，それを使用しているプロセッサが使えなくなることを意味する。 • プロセッサ同志のバスの競合はなく，競合は，プロセッサ間での共用資源の取り合いのみによって起る。

表 4.2.2　マルチプロセッサの結合方式(2)

（P：プロセッサ，M：メモリ，K：周辺装置）

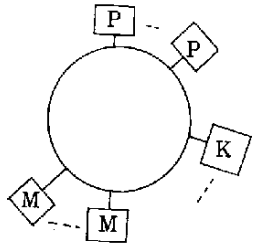
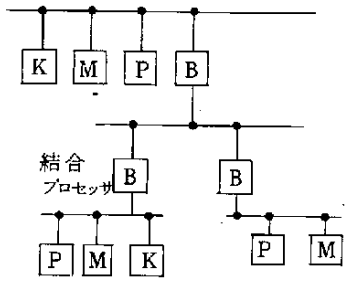
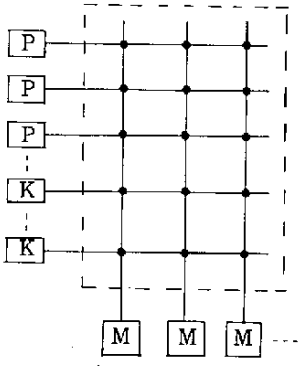
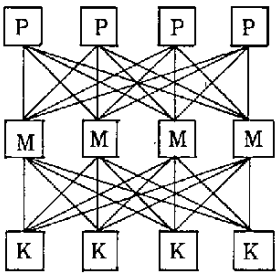
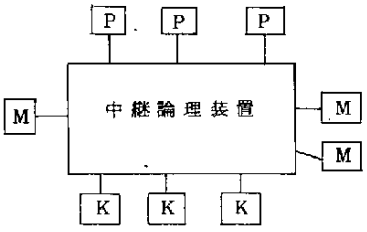
分類		結合方式の概略図	定性的評価
4	環状バス方式		• 環状バスは一方向性である。双方向性バスよりも実現しやすく，比較的距離が離れている装置間の結合に適している。
5	階層バス結合方式		• 大規模構成が可能。 • ローカルでないメモリの参照にかなりの時間がかかる。
6	マトリックススイッチ方式		• 平面上に配置されたスイッチにより任意の資源間を結合するものであり，m個とn個の資源をすべて接続するためには，m×n個のスイッチと（m＋n）本のケーブルが必要となる。 • スイッチのマトリックスの大きさがシステム規模拡張の限界となる。 • マルチポート方式同様性能的には優れた方式である。 • スイッチの信頼性がシステムの信頼性ともなる。

表 4.2.2　マルチプロセッサの結合方式(3)

（P：プロセッサ，M：メモリ，K：周辺装置）

分類		結合方式の概略図	定性的評価
7	マルチポート方式		• 専用バス方式に類似している。 • 結合すべき資源の数が限られていて，拡張性があまり要求されない場合には実用的である。 • 高速転送ができる。 • 大型機の少数マルチプロセッサシステムではよく使用される。 • m台のプロセッサとn台のメモリを結合するために，m×n本のケーブルと，各プロセッサにn個，各メモリにm個のポートが必要であり，m，nが大きくなると高価になる。
8	中継論理装置（プロセッサ）結合方式		• 転送速度は問題となるが，機能的には融通性がある。 • 結合路を動的に変更することができる。

プロセッサ間の通信は，①共用メモリ，②割込み，③直接通信，④相手プロセッサ内部レジスタへの直接転送／直接読み出し，等で実現される。

なお，マルチプロセッサでは，同時に走行するプロセッサ間の同期や通信処理を行うため，少なくともロック制御と呼ばれる基本命令が必要であり，これはハードウェアのTest and Set 命令やCompare and Swap 命令として実現されている。これらの命令は，データ参照から書換えまでが非可分操作として実現されている点が他の一般命令と異なる。

(2) ＯＳ構成方法[1),3)]

マルチプロセッサシステム用ＯＳと，マルチプログラミングシステム用ＯＳとの間には，概念的には大きな差はない。しかし，その複雑さは大幅に異っている。マルチプロセッサ用ＯＳとして必要な主な機能は次の通りである。

① 資源の割当てと管理

② 制御表およびデータセットの保護

③ システムデッドロックの防止

④ 異常の防止

⑤ 入出力負荷の均衡化

⑥ プロセッサ負荷の均衡化

⑦ システムの再構成

このうち，⑤，⑥，⑦がマルチプロセッサ用ＯＳとして固有なものである。マルチプロセッサ用ＯＳの設計には，表4.2.3に示す3つの基本構成がある。マスタ・スレーブ方式は，マスタプロセッサがダウンすると，直ちにシステム全体がダウンしてしまうという欠点はあるが，ＯＳは，シングルプロセッサ用ＯＳを拡張して比較的簡単に構成することができる。平等方式は，graceful degradation ができる，資源の効率的な使い方ができる等の利点はあるが，ＯＳ構成は最も難しい。

(3) 性　能

表4.2.3　マルチプロセッサ用ＯＳの代表的な構成法

マスタ・スレーブ	プロセッサごとのOS	平　等　形
・ＯＳは常に同一プロセッサ上で走行する。	・プロセッサは自分自身の必要な処理を行う。	・マスタの機能は，特定のプロセッサにくくりつけられておらず，プロセッサからプロセッサへと移る。
・ＯＳはリエントラントでなくてもよい。	・ＯＳはリエントラントであるか，またはプロセッサごとにコピーが必要である。	・複数のプロセッサが同一のプログラムを実行するので，ＯＳはリエントラントでなければならない。
・ＯＳでは制御表アクセス競合やロックアウトは生じない。（１プロセッサのみがＯＳを実行するため）	・各プロセッサごとにそれぞれが使う制御表がある。	・サービス要求の競合は優先度によって解決される。
・障害によって致命的な影響を受けやすい。	・１つの障害でシステム全体が致命的な影響をこうむることはない。しかし障害にあったプロセッサを再起動するのは非常に困難である。	
・柔軟性に乏しい。 ・マスタプロセッサが十分に高速でないと，スレーブ・プロセッサのアイドル時間が多くなる。 ・特殊目的の非均一システムに最適。 ・ソフトウェア，ハードウェアともに比較的簡単である。	・各プロセッサが各自のＩ／Ｏ装置を持つ。 ・Ｉ／Ｏ割込みは，これに対応するＩ／Ｏ処理要求を発したプロセッサが受け付ける。	・負荷のバランスがよい。 ・複数のプロセッサが同時にＯＳを実行するため，アクセス競合が大きい。 ・故障の影響を小さくできる。 ・資源を最も有効に使う。
・B－5500などのＯＳ	・Tandem 16などのOS	・C．mmp などのＯＳ

マルチプロセッサシステムの性能は，プロセッサ台数に比例しない。マルチプロセッサシステムにおける性能低下の主な要因は，次の通りである。

① 制御表，ファイルなど逐次的再使用可能な論理的な資源（ＳＲＲ）に対するアクセス競合 ——→ 制御表等を複数のプロセッサで共用使用するためアクセス競合が生じ，その待ち合わせによるロスが生じる。

② ロック処理ステップ数の増大 ——→ 制御表等のＳＲＲの排他的制御のためのステップ数が増加する。

③ メモリアクセス競合 ——→ 同一メモリバンク内に存在するプログラムやデータに対するアクセスの競合により，メモリアクセスに待ち合せが生じ平均命令実行時間が増大する。

④ ローカル（キャッシュ）メモリ内容の一致処理（ローカルメモリを持つプロセッサの場合）——→ 主記憶上のデータが更新されると，全プロセッサのローカルメモリ上の該当データと主記憶上データを一致させる必要があり，このためのオーバヘッドを生じる。一般には，ローカルメモリ上データの無効化を行うため，ローカルメモリの利用効率が低下し，平均命令実行時間が増大する。

これ以外にも，⑤入出力に伴うチャネルアクセス競合による性能の低下，⑥バス結合方式では，バス競合による性能低下，等がある。

①は，制御表などを細分化し，排他制御の単位を小さくし，競合する確率を低くすることにより減少させることができる。しかし，このようにしてきめ細かな排他制御を行うと逆に②のためのステップ数が増加する。したがって，排他制御の単位の大きさには，一般に最適値が存在する。③は，(i) メモリインタリーブ数を増加させる他，(ii) プロセッサごとに私用メモリを置き，この中にリエントラントなプログラムおよびリード・オンリのデータを格納する，(iii) ＯＳの中核部をファームウェア化し，プロセッサ個有の制御メモリへ格納する，等の方法により減少させることができる。更に (iv) ローカルメモリ容量を増大する方法も考えられるが，逆

に，④のために性能低下が大きくなる可能性がある。

性能上の問題と対策例を，図4.2.2，図4.2.3に示す。

なお，システムの信頼性向上の技術については，来年度検討する必要がある。

4.2.3 応用と効果

(1) 大型機における負荷分散少数マルチプロセッサシステム

2～4台から構成されるマルチプロセッサシステムは，汎用システムとしてすでに商用化されている。システムの性能は，2台で1.6～1.8倍，4台で約3.2倍程度向上している。

(2) マイクロプロセッサ，ミニコンなどによる負荷分散多数マルチプロセッサシステム

16台のミニコン（PDP－11）をマトリックス・スイッチ方式で結合したC．mmp が代表システムである。その他，いくつかの負荷分散多数マルチプロセッサが開発され実験が行われているが，現在次のような問題点が指摘されており，[4] 必ずしも成功していない。

① 大型機と同等の性能を実現するためには，マイクロプロセッサ，ミニコンが持つ機能が不十分である場合が多い。

② 多数のプロセッサと共用メモリを結合する部分での遅延時間や，共用メモリに対するアクセス競合による性能低下が大きい。

③ 一般のユーザプログラムを，多数の並列処理単位に分割し性能向上を図ることが困難である。

その後，プロセッサごとに個別メモリ（私用メモリ）を持ち，階層バス結合方式を採用したマルチマイクロプロセッサシステムCm* が開発されたが，Cm* により次の点が明らかになった。

① 共用メモリに対するアクセス競合は，個別メモリを持つことにより，かなり解決できる。

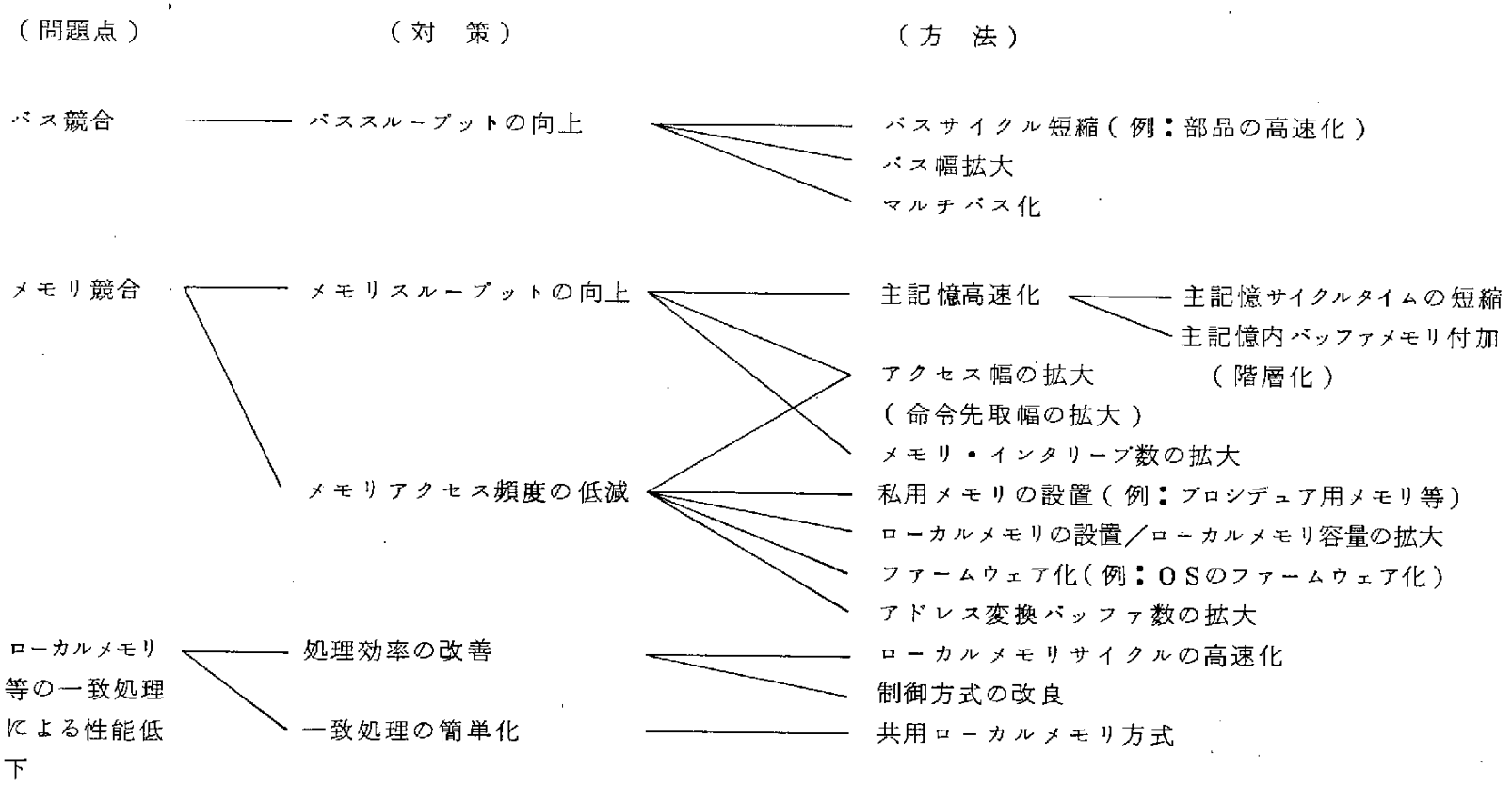


図 4.2.2　マルチプロセッサシステムの性能上の問題点と対策例（ハードウェア関係）

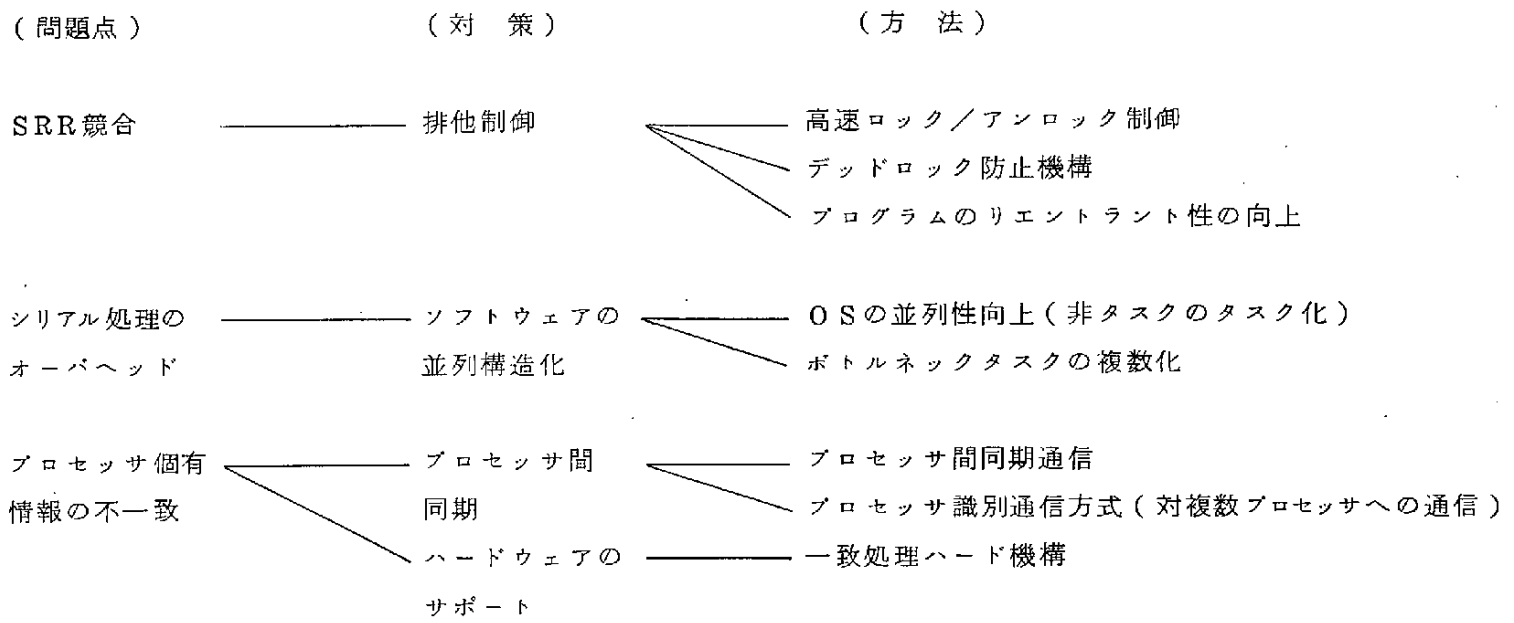


図4.2.3　マルチプロセッサシステムの性能上の問題点と対策例（ソフトウェア関係）

② しかし，多数のプロセッサを使用して性能を向上させるためには，プログラムが並列分解できること，同一メモリ上の参照が少なくなるようにプログラムおよびデータを分散配置することがきわめて重要である。

4.2.4 今後の課題

(1) 負荷分散汎用マルチプロセッサシステムは，①1つのＪＯＢ（トランザクション）の処理を並列処理することによる処理時間の短縮，②複数のＪＯＢ（トランザクション）の処理に対するスループットの向上が期待できる。しかし，現在は，主として論理資源のアクセス競合等によるソフトウェア上の要因により，プロセッサ台数は１０台程度が限度である。このため，ＯＳの構成法として全く新しい概念のものを検討する必要があるが，汎用システムについては，当分この状態が続くと考えられる。

(2) マルチプロセッサシステムの目的の一つは，信頼性の向上にある。このためには，エラー検出，エラー装置の識別と切離し，再構成，再開処理を自動化する必要があるが，現在のハードウェア，ソフトウェアは十分にこれに対処しているとは言えず，この面の検討が重要である。

(3) １０台以上のプロセッサを結合した負荷分散多数マルチプロセッサシステムは，今後，汎用システムを狙わず，特定分野の専用システムとして実現する必要がある。ＬＳＩ技術の進歩に伴ない，プロセッサ，メモリ価格は相対的に安価になるため，将来は，プロセッサ，メモリをぜい沢に使用できる環境になる。このような環境では，ソフトウェアのオーバヘッドを除去するため，プロセッサがidle状態になってもかまわないという割り切った思想が重要である。

現在，数十～数千個のマイクロプロセッサを結合した専用システムが検討されているが，

① 多数のプロセッサ，メモリ，入出力装置等の結合方法

② 多数のプロセッサ，入出力装置からの割り込み処理方法

③ 障害の検出と識別方法

④ プロセッサ機能の拡充

などが問題となっている。ソフトウェアの面では，このような多数のプロセッサを管理するＯＳ，処理すべき問題のアーキテクチャへの写像方法等が問題になるのは言うまでもない。また逆に言えば，専用システムは，処理すべき問題にｔｕｎｅした構造をとる必要があるため，プロセッサ，メモリ，入出力装置の結合状態を変更できる結合方式技術の開発も重要である。

(4) 今後は，文献5)でも指摘されているように，ＬＳＩ技術を中心とするハードウェア・テクノロジーの進歩が，アーキテクチャやソフトウェアの技術進歩を追い越してしまう可能性が強く，ＬＳＩ（ＶＬＳＩ）プロセッサの持つ機能を十分に引き出せない状況も想定される。したがって，マルチプロセッサシステムの実用化においても，ソフトウェア・サイドからの研究を強化し，最適なハードウェア／ソフトウェアのトレードオフを検討するアプローチが重要となる。

参 考 文 献

1) P. H. Enslow : "Multiprocessor Organization — A Survey," ACM Computing Survey, Vol. 9, №1, pp. 103—129 (1977)

2) 「マイクロコンピュータとその応用」，電子通信学会

3) P. H. Enslow : "Multiprocessor and Parallel Processing," John Wiley & Sons Inc. (1974)

4) 関野陽「分散処理技術」，情報処理，Vol. 20, №4, pp. 275—283 (1979)

5) D. P. Siewiorek , D. E. Thomas et al : "The Use of LSI modules in Computer Structures : Trend Limitations," Computer , July , 1978 , pp. 16—25

4.3　データフロー計算機

4.3.1　概　　説

データフロー計算機は，つぎのような計算機アーキテクチャへの課題に対する解答の一つとして提案された計算機モデルである。

1）並列処理性の追求

2）LSI技術の効果的利用

3）プログラミング容易性の追求

高性能の計算機を合理的なコストで実現するためには，何らかの形で単一プログラムの並列処理を行う必要のあることは，既存の高速科学計算機の例によっても明らかである。特に，同じものを繰返し的に量産する場合に，その経済性が最適化されるというLSI技術の特性を考えると，同種のプロセッサを多数組合せた並列処理計算機に大きな期待が寄せられることになる。しかし，こうした並列処理計算機に対するプログラミング言語の枠内では極めて困難である。ソフトウェア危機ということばが示しているように，通常のプログラムでも，その規模が大きくなるとわれわれの開発や保守の能力を超えたものになるという状況が，現実に発生している。並列処理機能を組込むことによって，プログラムの構造が一層複雑化し，その意味を把握するのがなお困難になるというのでは，問題外である。求められているのは，ソフトウェア危機の解消に役立つようなアーキテクチャである。

並列処理性の追求という観点からみると，現在の計算機の基礎となっているフォン・ノイマン・モデルには，ふたつの大きな難点が内在している。

1）逐次制御

2）“記憶語（メモリ・セル）”の概念

逐次制御は，いうまでもなく，一定の順序で並んだ命令を，“一時に一命令”の原則に従って逐次処理していくという，プログラム実行制御の基本則である。

この基本則の根底にあるのは，ひとつのプログラムを処理するのは1台のプロセッサだけという前提である。したがって，この基本則に基づいて組立てられたプログラムを，多数のプロセッサに細かく割振って，それぞれを非同期的に実行するためには，プログラム論理の精密な分析が必要になり，とうてい実行時はもちろんコンパイル時の自動処理にかなうようなものではない。また，このための指定をプログラマに負担させるやり方も，プログラミングをできるだけ容易にしようという，時代の要求に抵触することになる。

“記憶語”の概念は，プログラミング言語での変数が，主記憶のアドレスのもつ性質を引継いで，単に値を代表するものとしてだけではなく，値を記憶する機能をもった実体としての性質をもつものと見なされている事実をさすものである。このような変数の性質は，プログラムのある部分での実行結果を他の部分に反映させたい場合に，共通変数の値を書直すというような形で活用され，強力なプログラミング機構の実現に役立っている。しかし，副作用などのように値の書換えが暗黙裡に行なわれたり，あるいは多重の書直しがある場合には，並列処理性の追求が極めて困難になる。しかも，これらを禁止することは，記憶語の概念を事実上放棄することに通じている。

データフロー計算機は，逐次制御の原則と記憶語の概念を排除し，代わりに，

1）データ駆動の原則

2）関数型プログラミングの概念

を用いる計算機モデルである。

どのような形式のプログラムであっても，その実行は，与えられたオペレータを与えられたオペランドに作用させるという動作の累積として実現される。逐次制御の場合には，この動作の実行順序が各オペレータを相手にして制御され，オペランドはオペレータに付属するアドレスによって引出されてくる。これに対して，データ駆動の場合には，実行制御の対象となるのはオペランドの方であり，対応するオペレータはオペランドに付属するアドレスを通して得られる。外部からの入力あるいは他の演算動作の結果としてオペランドの内容が

整うと，対応するオペレータと組にしてプロセッサに送って作用を実行するという形で，実行制御が行なわれる。多数のオペランドが用意され，多数のプロセッサが存在していれば，実行は並列的に進行して，再び多数の新しいオペランドが用意されることになる。このようにして，データ駆動の原則によって，並列処理のための実行制御が自然な形で実現される。

関数型プログラミングというのは，簡単にいえば，プログラム各部分の作用が，他の部分から与えられた値に変換を加えて新しい値を導き，それを別の部分に明示的な形で引渡すということだけに，厳密に限定されているようにするプログラムの作り方のことである。これによれば，プログラムは純粋な意味での関数の集まりとして構成され，プログラムの各部分間には明示的な値の受渡しがあるだけで，記憶語を媒介にした作用の伝播はなくなる。したがって，副作用の問題も解消する。関数型プログラミングがどのように具体化されるかについては，次項で触れる。

フォン・ノイマン・モデルに代わるものとしては，データフロー計算機以外にも可能性をもつものが考えられている。それらは，いづれも，関数型プログラミングの概念を基本にしているという点でデータフロー計算機と共通しているのが特徴的であるが，利用する実行制御の原則に本質的な相違がみられる。すなわち，この第３のモデルでは実行制御の対象が，オペレータとオペランドの一方だけに片寄ることはなく，これらの組を対象とするようになっている。バークリングのリダクション・マシンは，この実現例のひとつである。

プログラムをその中でのデータの流れに主眼を置いて把えるという考え方は，１９６０年代にその例がみられるが，これをアーキテクチャに結びつけ，データフロー計算機として定式化したのは，１９７１年のデニス論文が最初である。以来，データフロー計算機ならびに言語を具体化するためのプロジェクトが多数編成され，活発な研究活動が続けられている。１９７９年末現在での主要なプロジェクトを表４．３．１と表４．３．２に示す。これにみられる通り，すでに３種類の試作機が稼動状態に入っている。

表 4. 3. 1　Data flow projects

		Language
1. Toulouse, France	operational	Ex−Fortran
2. Texas Instrument, U S A	operatienal	Fortran
3. Manchester, England	planned	machine
4. M. I. T, USA (fem 1 machine) Dennis	planned	Data Flow graphs (VAL)
5. New Castle, England	exploratory	machine
6. Irvine, California, USA	exploratory	Id
7. M. I. T, USA (U− interpreter chip) Arvind	exploratory	Id
8. Utah, U S A (one processor) Davis	operational	DDM−machine language
9. Utah, U S A (Keller, Lindstrom)	exploratory	Ex−Lisp
10. M. I. T, U S A (general) Dennis+Weng	exploratory	VAL

表 4. 3. 2.　主なデータフロー言語の特徴

言語名	開発機関名	年度	言語の主な特徴
DDF	MIT Dennis アメリカ	1974	通常の算術計算に加えて，手続き呼出し，条件判断，データ構造操作を図式表現した最初のデータフロー言語．決定性並列処理モデルに対する低レベルの表現
DDN	Utah 大学 Davis アメリカ	1975	同期，演算，ゲート，手続き呼出し，分配，選択，調整の7種類のセルによって低レベルの並列処理モデルを図式的に表現
TDFL	MIT Weng アメリカ	1975	データフロースキーマへの翻訳を目指した最初の高級言語．シンタックスは，Algol 60 から，GOTO,WHILE. 全域変数を除いたもの．初めてストリームに対する演算を導入
LAU	Toulouse 大学 Comte, Syre 外 フランス	1976	割付け文，EXPAND 文，CASE 文，LOOP 文からなる高級言語．各文のセマンティクスを DPS という概念によって定義し，実際の機械語命令との対応を与えている．共有変数等，単一割当規則で記述できないものに対し，履歴式を導入
VAL	MIT Ackerman, Dennis アメリカ	1978	厳密なデータ形定義とチェック，例外処理機構，並列実行文
DFPL	IBM Kosinski アメリカ	1973	OS の記述を目的としたグラフ表現の言語．LOOP ノードの導入による，繰返しプロセス，メモリセル機能の実現．PRESENCE/DONE によるノード間の非同期交信，非決定性表現の試み
ID	California 大学 (Irvine) Arvind, アメリカ	1977	ストリームによる演算 (history-sensitive なモデルに対する試み)．ユーザ定義のデータ形，非決定性表現のためのデータフローモニタによって，OS の記述，スケジューリング方策の容易な変更が可能
CAJOLE	London 大学 Hankin 外 イギリス	1978	条件式による判断機能，ブートストラッピング機能によって，新たな構文をユーザが定義できる．名前の局所化機能
GPL	Utah 大学 Boekelheide アメリカ	1979	図式表現による高級言語．CALL, ITERATE, SPAWN, CASE, SHARE の五つのブロック構造が，DDN と等価なデータフロー図式に変換される

4。3。2　技　法

(1)　基本的技法

データフロー計算機の機械語は，データフロー・グラフであり，これがシステム・アーキテクチャとユーザ向プログラミング言語のインタフェースを形成している。

データフロー・グラフは，オペレータを表わすノードを，オペランド値の通ってくる路である孤線で結んだものである（図4.3.1，図4.3.2参照）。オペランド値は孤線上をトークンと呼ばれる乗物で運ばれてくる。オペレータ・ノードは，それへの入力孤線のすべてにトークンがあり，出力孤線上にトークンがない状態で動作を開始し，入力トークンを消して出力トークンを送出した状態で動作を終了する。

データフロー・グラフは，オペレータをデータ依存性（どのオペレータからの中間結果がどのオペレータで使われるか）の関係で結合したものである。複数のオペレータが並列処理可能かどうかは，それらの間のデータ依存関係の有無によって決まるので，データフロー・グラフは与えられたプログラムの並列処理性を最大限に表現しうるものということができる。

データフロー・グラフで許されるオペレータ・ノードならびにオペランド値の種類，すなわち，データフロー計算機の命令体系のあり方については，現在様々な考え方が並立している状態にあって，簡単に総括するのは難しい。

データフロー・グラフを計算機のメモリ上に表現するためには，アクティビティ・テンプレートという形式が使われる。これはオペレータの種類を表わす語，オペランドの入る語，結果の宛先番地を示す語が組になったものである。これらの語は，それぞれオペレータ・ノード，入力トークン，出力孤線に対応するものである。

データフロー計算機の基本的な構造を，モデル化して示すと図4．3．3のようになる。

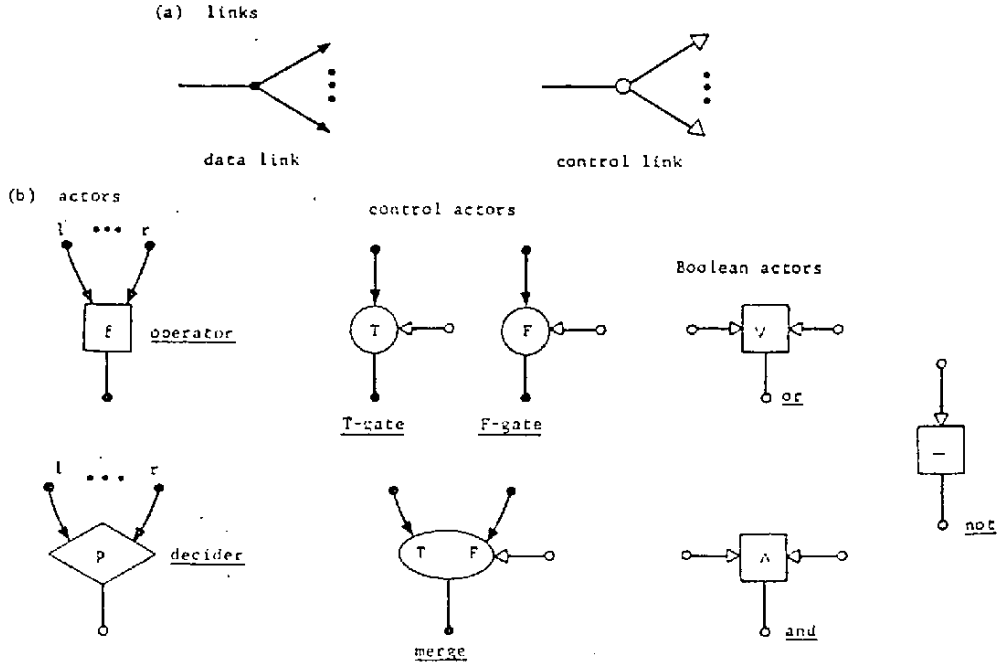


図４．３．１　基本的なノードの一例

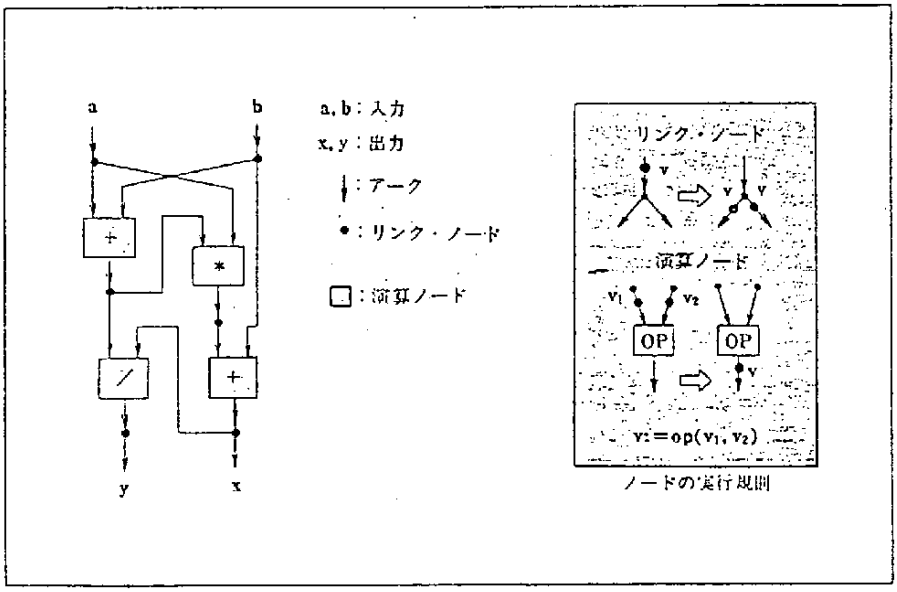


図４．３．２　データフロー・グラフの一例

“ y＝(a＋b)／x,　　x＝(a＊(a＋b))＋b”

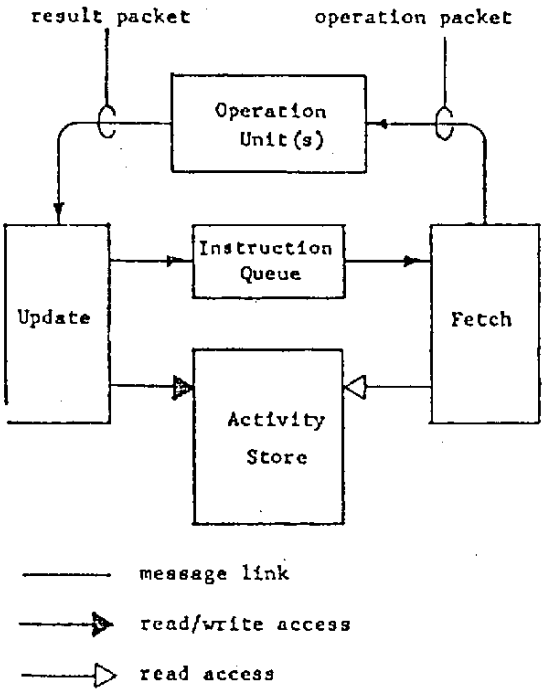


図４．３．３　データフロー計算機のモデル

データフロー・グラフはアクティビティ・テンプレートの形式で，アクティビティ・ストアに入っている。各テンプレートには番地がふられていて，オペランドが揃って実行可能の状態になったものの番地がインストラクション・キューに入る。フェッチ部はキューの先頭にある番地によって，アクティビティ・ストアからテンプレートを読出し，それをオペレーションパケットとしてオペレーション・ユニットに送る。オペレーション・ユニットはパケットのオペレータとオペランドによって演算を実行し，その結果と宛先番地をリザルト・パケットにまとめアップデート部に送る。アップデート部は宛先番地に結果を書込むと同時に，それによってそのテンプレートで必要とされるオペランドがすべて揃うことになるかどうかがチェックされ，そうであればテンプレートの番地がインストラクション・キューに入れられる。

(2)　プログラミング言語

データフロー・グラフの形式は，通常のプログラミングにおける思考の表現形式とかけ離れているため，これを直接ユーザ向のプログラミング言語として用いることは問題が多い。ユーザ・プログラミング言語としては，FORTRAN

などと同じような変数を用いたステートメント形式が望ましい。また，各種の便利な言語機能もできるだけ残しておきたい。

一方，単一プログラムの並列処理を目指すデータフロー言語としては，プログラムの並列処理性が最大限に表現されるようなものであることが要請される。すなわちデータフロー言語の条件としては，つぎのようなものが基本的である。

1）プログラム・オペレーション間のデータ依存関係を最少限に抑えうるものであること。

2）実行順序を決める制約条件となるものが，うえのデータ依存関係だけになっていること。

これらの条件を満足するための目安となる言語的特徴としてあげられるのは，

1）副作用のないこと

2）実行効果の局所性

3）単一割当て規則

4）関数型

などである。副作用と関数型については，すでに触れてある。実行効果の局所性は，各ステートメント間のデータ依存関係があまり広汎にならず，それぞれプログラムのある与えられた範囲内だけに限定できるようにすることである。このためには，何らかの形のブロック構造化，広域変数の禁止，GOTO文などの制御ステートメントに対する制限などの手段が考えられる。単一割当規則は，プログラム中の変数が“記憶語”としての機能で用いられることを防止するために，割当文に課せられる制限で，“各変数は，その有効範囲内で，割当文の左辺に2度以上現われることがあってはならない”とするものである。

このように，データフロー言語に対しては様々な制約が課されることになり，通常の言語にみられる便利な機構が無条件には利用できなくなる。例えば，繰返しなどはそのままの形では残すことができなくなる。しかし，これらの制約は，データフロー言語の立場とは独立に，ソフトウェア工学の観点からも必要な条件として要請されているものばかりであり，その意味では自然な制約とも

いえる。しかも，こうした制約から，逆に通常の言語では実現が困難であった機構 ― 例えば非決定性プログラミング ― の組込みが可能になるなどの新しい可能性も指摘されている。

(3) 技法上の課題

これまで，データフロー計算機の問題点として指摘されているもののうち，主なものを列挙するとつぎのようになる。

言語関係：

1）しかるべき高水準言語がない。

2）提案されているような言語では一般のプログラマに不向き。

3）FORTRANやCOBOLが使えない。

性能関係：

1）具体的な性能データが得られていない。

2）内部コミュニケーションのためのオーバヘッドが大きく，期待されるような性能が得られるかどうか疑問。

3）ある種の数値計算や多くの記号・数式処理上の問題に対しては，論理深度（データ依存関係の深さ）が非常に深いアルゴリズムしか知られていないので，複合計算機による並列化の効果はほとんど期待できない。

4）関数型プログラミングでは，行列やデータ構造など集団データの処理が，構造を単位とした段階的な取扱いになるため，却って従来の計算機より不利になることがある。

ヒストリィ・センシティビティ（まえのプログラムの残した情報によってあとのプログラムの行動が左右されるという性質）関係：

1）計算モデルの中にはデータベースの更新などのように状態の保持を本質的に必要とするものがある。また，本質的とはいわないまでも，それが可能であれば好都合なことが多い。フォン・ノイマン・モデルではそれを"記憶語"の概念によって実現しているが，データフロー・モデルではそれが利用できない。

2）ヒストリィ・センシティビティを追放した言語をサポートするためのアーキテクチャは，実現が難しく，できたとしても効率が大変に悪いものになるのが通例。

汎用性関係：

1）並列処理の要求される分野でデータ駆動型コンピータが実用化されるのは，そんなに遠い先のことではなさそうである。しかしながら，汎用的なアーキテクチャの実現という意味においては，現在提案されている各種の関数型マシンのそれぞれについて，まだまだ研究の余地が残されていそうに思われる。

現在進行中のデータフロー・プロジェクトでは，これらの問題に対して様々な解答が用意されつつある。しかも，データフロー計算機そのものが非常に新しい考え方であるために，可能性の評価が定まっていない分野が多く，したがって，アプローチも極めて多様である。例えば，プログラミング言語でのアプローチについてみると，

1）既成言語を直接利用するもの。

2）既成のプログラミング・スタイルに近い新言語。

3）革新的なプログラミング・スタイルを狙うもの。

4）機械語の水準にとどまるもの。

という具合で，あらゆる可能性が試されている状況にある。

また，性能関係では内部コミュニケーションの方式が最も重要な研究課題であり，処理要素間の近接関係を利用して極限の高速性を追求するネットワーク構造をとるか（この場合にはこの特性を活かすためにプログラムのアルゴリズムに特別の工夫が必要），速度は多少犠牲にしてもプログラミングの容易性を重視して，均質的な接続関係をもった構造にするかといった選択が迫られている。

さらに，ヒストリィ・センシティビティについては，ストリームの概念が有力視され，その実現方式が模索されている。

汎用性については，これを特に指向したアーキテクチャが意識され始めた段階である。

4．3．3　応用と効果

理論的な可能性としていえば，データフロー・モデルの適用性は非常に広く，フォン・ノイマン・モデルによる合理的な対応が困難な領域は当然として，一応の対応に成功している領域についても，コスト・パフォーマンスの一層の向上など新しい可能性を開くことが期待されている。

前者の意味での応用領域として特に注目されているのは，つぎのふたつである。

1）超高性能計算機アーキテクチャ

2）関数型言語サポート・アーキテクチャ

超高性能計算機に対するデータフロー・モデルからのアプローチは，いうまでもなく並列処理性の徹底的な追求にあり，その点では従来の科学計算並列処理マシンのそれと共通である。しかし，データフローの場合には，アルゴリズムに内在する並列性の活用がより深化できること，問題ならびにアルゴリズムの種類に対する適合性が高く，プログラミングに個別的な特殊な工夫が要求されない，あるいは要求されることが少ないこと，といった利点がある。ただ，こうした利点の効果を定量的に評価することは，これまで極く少数の散発的な結果が知られているのみで，むしろ本格的には今後の研究課題である。

超高性能計算機を必要とするような問題には様々な内容のものがありうるが，それらのすべてに対してデータフロー計算機が卓越しているという訳ではない。データフロー・モデルに適するには，処理すべきデータ量が多い場合，中間結果間のデータ依存関係が局所的である場合などである。このようなものとしての典型例は，気象，原子炉，地震などの偏微分方程式を利用したシミュレーションである。

ソフトウェア危機の解消，人工知能関連の新処理分野などで果たす関数型言語の重要性については，ここで詳しく議論するだけの余裕はないが，関数型言語が

必要になった場合それをサポートするためのアーキテクチャとしては，フォン・ノイマン・モデルに難点が多くてデータフロー・モデルに期待される面の多いことは確かである。しかし，ひとくちに関数型言語といっても，その中には性質を異にするものが包括されており，そのすべてに対してデータフロー・モデルで十分という保証はない。例えばＬＩＳＰ言語については，さらに第三のモデルが必要であるというような見解もみられる。

つぎに，すでにフォン・ノイマン・モデルの守備範囲となっている領域におけるデータフロー・モデルの可能性についてみると，高度な並列処理性の追求に基づくプロセッサ利用効率の改善，超ＬＳＩ技術への適合性などからコスト・パフォーマンス面での優位性といったことが期待できる。既存の規模での並列処理科学計算，イメージ処理，信号処理，データベースのためのトランザクション処理などへの応用がこの例として挙げられるものである。ここに挙げなかった一般的な事務処理分野，科学技術計算分野に対するデータフロー・モデルの適用性についても，それをアプリオリに排除すべき理由は見当らない。

データフロー・モデルの当初の発想は，10^2 ～10^4 台という規模のプロセッサ集団から成る単一システムを実現することにあった。並列処理システムの基本性能を測す尺度としては，そこに含まれるプロセッサ台数と得られるパフォーマンスの関係が重要である。データフロー・モデルの場合にこの関係がどうなるかについては，現在までに若干のシミュレーション結果が得られているが，それらをもとにして一般的な議論をすることは難しいというのが現状である。

データフロー・モデルに対しては，その適用性が超高性能（科学）計算機に限られるのか，それとも汎用計算機に及ぶのかを巡って議論が多く，汎用性に対して疑問を呈する専門家も少なくないのが現状である。この議論に対するデータフロー計算機の創始者であるＭＩＴデニス教授の見解はつぎの通りである。

“一方においては，われわれのデータフロー・マシンは，それが言語ベースのものであり，したがってマシンに対して設定されている言語で書かれたもので

あれば，任意のプログラムを実行しうるという意味で，汎用である。他方，それがマシンの水準ごとにサポートしうる言語が限定されているという点では，専用といえる。さらに，マシンをその極限まで追求するような大規模のアプリケーションに対しては，マシンはその問題に適するように専用化されざるを得ない。しかし，このことはあらゆる方式の計算機にもいえることなのである。”

4．3．4 第5世代における可能性

データフロー計算機の実現可能性を考える場合には，技術的な実現可能性と社会的な実現可能性を区別して考えるのが便利である。データフロー計算機はその存立の前提としてプログラミング言語の革新を要請することになる可能性があり，そうなると広い意味での互換性に支障を生じ，その克服には多くの社会的な手続が必要になることが予想されるためである。

第1項でみたように，データフロー・モデルの試作機にはすでに作動を開始したものがいくつかあるが，これが直ちに実用化モデルに結びつくかどうかには保留が必要であり，むしろ現状ではプロセッサ間通信網の構成とオーバヘッド評価，ヒストリィ・センシティビティの具体化などの基礎的問題に明確な見通しを与えるような研究が先行すべきであると考えられる。この意味からいうと，科学計算用にせよ汎用にせよ，本格的なデータフロー・モデルの実現を予測することは大きなリスクを冒すことになる。

さらに，ハードウェアとしての範囲では相当なものが実現できることになったとしても，それが実際に利用されるためには，少なくとも安定したプログラミング言語が確立されることが必要である。言語の確立は技術的な問題としてよりも，社会的な手続としての側面に多くの曲折を伴なうものであり，それがどのような経過をたどることになるかは予想が困難である。

こうした点を考慮すると，データフロー・モデルの実用化は，専用性が高くて限られた専門家だけがプログラムすればよい領域でまず開始されることになる可能性が高い。このような領域としては，データベース・マシン，通信プ

ロセッサ，信号プロセッサなどであり，その時期は'８０年代のうちと予測される。こうした領域では，データフロー・モデルが第５世代の基礎技術となることも大いに考えられる。

参考文献

１）松崎「コンピュータ構造を変革するデータフロー・コンピュータの動向」（上）：（中）：（下），日経エレクトロニクス，１９７９年５月２８日号；６月１１日号；６月２５日号．

２）樋口，山口「データフロー言語の研究動向」，電子通信学会誌，Vol．63，№１（１９８０年１月），pp．８３～８７

4.4 科 学 計 算

4.4.1 概　　説

(1) 歴史的経緯

計算機の開発は，科学計算を高速に行なうことを目的として始められた。したがって，当初の計算機の歴史は，科学技術計算プロセッサの歴史でもある。

ＩＢＭ３６０システムの開発とその成功により，計算機システムは，汎用性に重点が置かれて開発されるようになった。また，その間に，事務処理，オンライン処理などの分野への計算機の利用が急速に発展した。

科学技術計算の分野では，ＩＢＭ７０９０，ＣＤＣ６６００，ＩＢＭ３６０／９１，ＣＤＣ７６００，ＩＢＭ３７０／１９５などの大型機が開発され，大型化する科学技術計算に対応してきた。

しかし，科学技術の進歩に伴ない，これらの汎用性をもつ大型機ではなく，超高速計算を行なう科学技術計算専用プロセッサ開発の試みが行なわれるようになった。

１９７０年代前半に稼動したものとしてＳＴＡＲ－１００，ＩＬＬＩＡＣ Ⅳ，ＡＳＣ（Advanced Scientific Computer）などがある。これらは，一般にスーパーコンピュータとも言われている。

この時期のスーパーコンピュータは，初期の半導体技術，コアメモリなどを使用しており，物理的にも大きく，台数としては，多くは製作されなかった。

その後の半導体技術の進歩を基盤としたハードウェア技術の進歩により，１９７０年代後半に，さらにコンパクトで高速な科学技術計算専用機が開発された。これらには，ＣＲＡＹ－１，ＢＳＰ（Burroughs Scientific Processor），ＤＡＰ（Distributed Array Processor），ＣＹＢＥＲ－２０３などがある。

なかでもＣＲＡＹ－１は，米国，欧州を中心に１０数台設置され，気象計算，

原子物理計算などに使用されており，この種の計算機がなければ理論分野の研究が進められないと言われる状況になりつつある。

科学技術計算専用プロセッサを含む専用プロセッサの動向を図４．４。１に示す。汎用プロセッサの軸の右側にある技術計算プロセッサ，Navier－Stokes Solver，信号プロセッサ，画像プロセッサなどが，広い意味での科学技術計算プロセッサである。

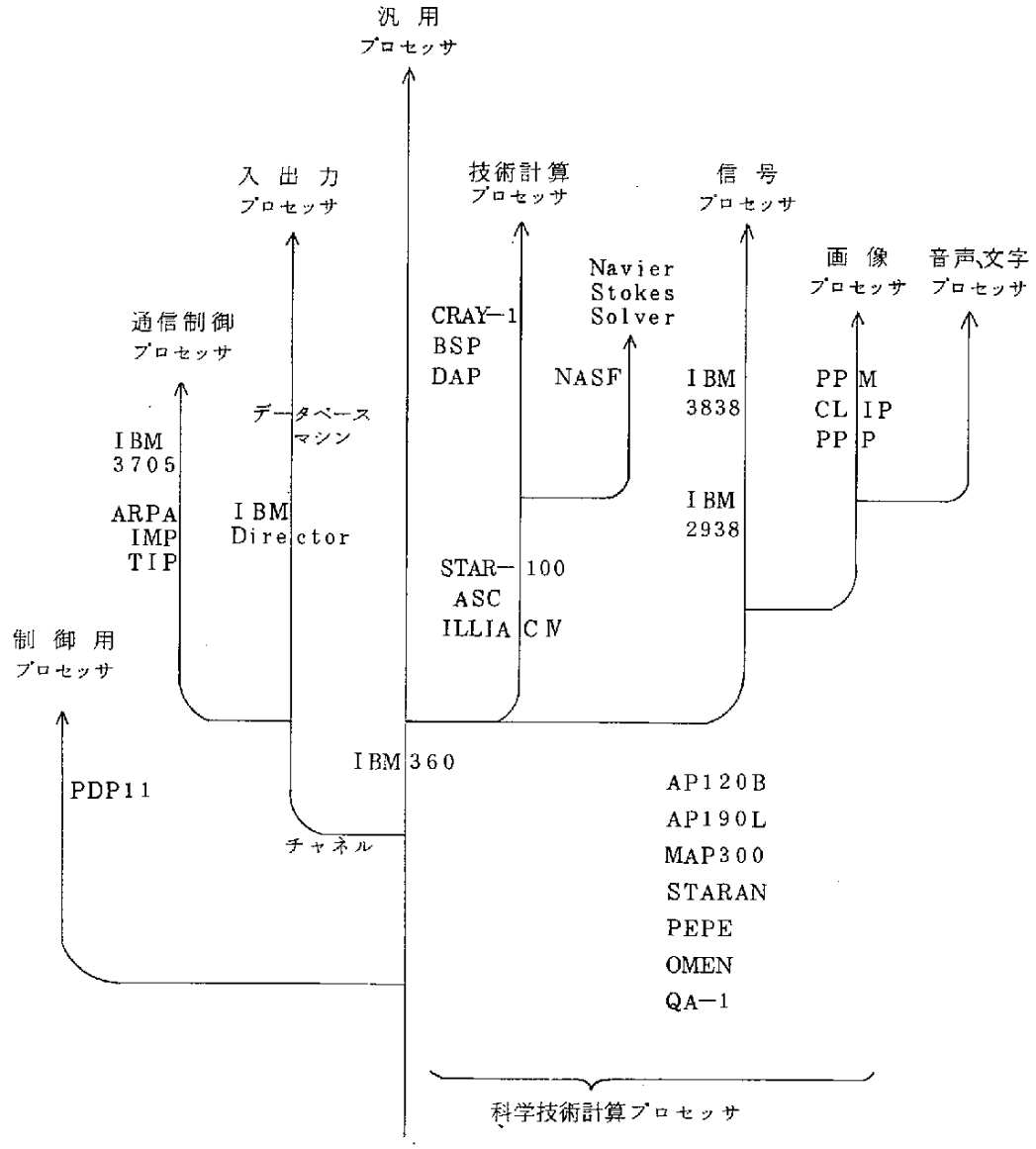


図４．４．１　専用プロセッサの動向

(2) 目　　的

科学技術計算プロセッサの目的は，科学技術上の問題を数値計算により解くことである。

近年，対象となる問題は，自然科学，工学の分野だけでなく，経済などの分野の問題も含め，多様化しつつある。

工学の問題は，多くは古典物理学の範囲で連続な媒体中の運動量保存の方程式，連続の方程式，拡散の方程式，熱エネルギー保存の方程式，状態方程式などが中心となる。

物理学の問題も偏微分方程式が中心であるが，この他に，粒子に着目した問題があり，この場合には，粒子ごとの状態量を求める問題となることがあり，必ずしも偏微分方程式の形とならない。

また，資源探査のように大量均質のデータの中から必要な情報を取出すフィルタリングの問題が中心となる分野もある。

これらの問題のすべてに適合する処理方式の科学技術計算専用プロセッサを開発することが望ましいが，むずかしい問題である。

スーパーコンピュータと言われる科学技術計算専用プロセッサは，ベクトル処理の高速化を図っていることから，一般には，連立偏微分方程式の数値計算を超高速に行なうことを対象としている。これは，汎用の科学技術計算プロセッサとも言えるが，必ずしも，多様化して行く問題のすべてに十分な適応性をもっているとはいえない。

一方，信号プロセッサや画像プロセッサなどの専用プロセッサは，それぞれ汎用のプロセッサではできない問題を処理したり，大幅に価格性能比を向上することが目的である。これらは，大量均質のデータを高速処理する目的のものが多く，特殊目的の科学技術計算プロセッサと言える。

(3) 他の技術との相関関係

汎用計算機の処理能力の向上は，ハードウェア技術の進歩と，論理方式上の工夫とによる。これまでの汎用大型機の処理能力の向上はマシンサイクルの向上と，パイプライン方式などの論理方式上の工夫により達成されたものである

が，マシンサイクルの向上による寄与分が重要な部分を占めている。

このことは，ハードウェア技術の進歩が本質的な役割を果して来たことを意味しており，これからもハードウェア技術が重要な役割を果すと考えられる。

科学技術計算の分野でも，同様に，ハードウェア技術の進歩が重要な役割を占めると考えられる。したがって，第3章のデバイス技術に述べられている技術動向の影響が大きい。

また4.1の「並列処理」，4.2の「マルチプロセッサ」も，今後の科学技術計算の高速処理のために重要な技術である。

今後の科学技術計算の高速処理は，ハードウェア技術の進歩をベースに，パイプライン処理，並列処理，連想処理などの工夫により進められて行くと考えられる。以下では，4．1，4．2に述べている技術的な立場からの動向とは別の立場で，具体的なニーズとプロセッサの具体例を中心に述べる。

4．4．2　科学技術計算の技法

(1)　処理方式

超高速計算を行なう処理方式としては，大きく分けてパイプライン方式と並列処理方式の2つの方式がある。

並列処理方式には，演算ユニットを複数個用意した形のものと，処理エレメントをアレイ状に多数配列した形のものとがある。これらを別種のものと考えることもできるので，ここでは，処理方式を3つに分類することにし，パイプライン方式，並列処理方式，プロセッサアレイ方式と呼ぶことにする。

これらの方式の特長を，浮動小数点加算の処理方式を例として汎用計算機と比較し，図4.4.2に示す。

汎用計算機では，図4．4．2に示すように，命令解読，オペランド読出しなどは，演算とオーバラップ処理されるが，演算そのものはオーバラップ処理されない。

パイプライン方式の科学技術計算専用プロセッサでは，演算そのものもオーバラップさせた演算パイプライン処理がおこなわれる。並列処理方式およびプ

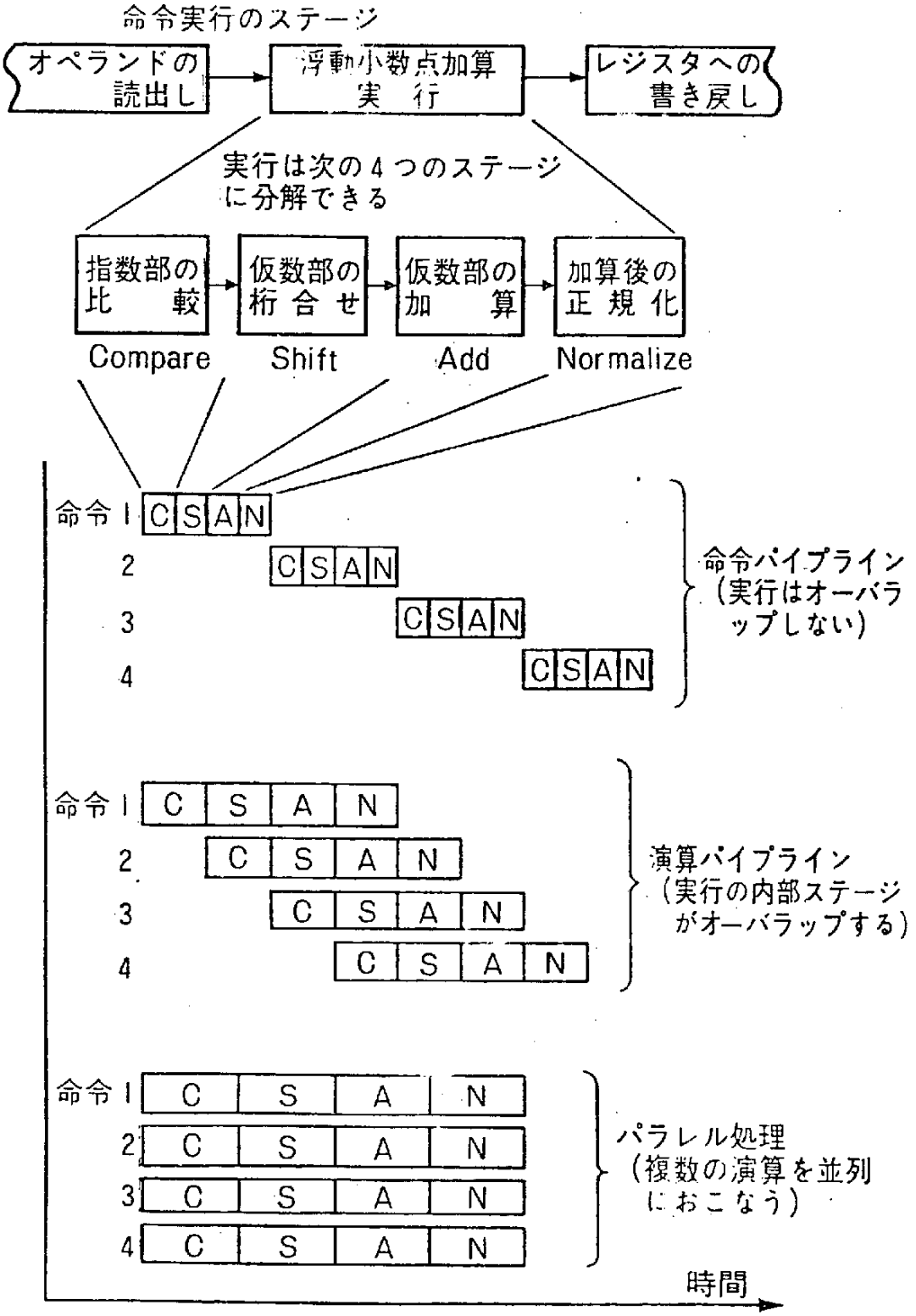


図 4．4．2　処理方式の比較

ロセッサアレイ方式のものでは複数の演算機構により，並列に演算が行なわれる。

パイプライン方式の場合には，パイプラインの時間刻みを小さくし，その刻みごとに1個の演算結果を得られるようにし，実効的な処理能力を高めることができる。現在，この刻みの最も小さいものは，ＣＲＡＹ－１の１２.５ nsである。

並列処理方式の場合には，適当な数の演算ユニットを置き，それぞれの演算ユニットを高速にすることにより処理能力を高めることができる。

プロセッサアレイ方式の場合には，演算エレメントの数を増すことにより処理能力を高めることができる。

以上は，一般的な科学技術計算の高速処理の技法であるが，以下に，特殊目的の科学技術計算プロセッサの処理方式の特長について簡単に述べる。特殊目的のものとしては，Navier－Stokes Equation Solver，信号プロセッサ，画像プロセッサなどを挙げることができる。

Navier－Stokes Equationは，基本的には超高速の科学技術計算プロセッサと考えることができるが，Navier－Stokes Equationの数値計算アルゴリズムが，並列処理が可能であることから，大規模な並列処理を行なう方式が提案されている。

信号プロセッサは，地震波の解析，レーダ信号の処理あるいは各種の実験データの処理を行なうものであるが，その目的によってコンボリューション積分の高速処理などに適した構造のものや，連想記憶をもち，リアルタイム処理や並列処理に適した構造のものがある。

画像プロセッサは，人工衛星からの写真の解析を行なうものや，画像の識別を行なうものなど目的が多様化しており，プロセッサの構造も多様であるが，基本的には画素データに対する四則演算，比較，大小判定などの機能をもつ。

(2)　代表的システム

現在までに稼動ないし発表されている超高速計算を目的とした科学技術計算

専用プロセッサを表４．４．１に示す。

（ａ） CRAY－１ [1)]

パイプライン処理方式を追求して設計されたものの代表として，CRAY－１を挙げることができる。

CRAY－１の主な特長としては，フレオン冷却によるコンパクトな実装，高速ＩＣメモリのベクトルレジスタ，ベクトルのチェイニング処理などを挙げることができる。

主なハードウェア仕様は，次のようである。

- マシンサイクル　　１２.５ ns
- 演算ユニット数　　１２個（浮動小数点３個）
- データ幅　　６４ビット
- レジスタ構成　　８×６４語
- 主記憶　　最大１M語，１６バンク
  サイクル　５０ ns

最近，新たにＣＲＡＹ－１Ｓが発表され，主記憶容量は最大４M語に拡張されている。

このパイプライン方式では，１２.５ ns刻みで１個の浮動小数点演算結果を得ることができるから，ピーク処理能力は８０ＭＦＬＯＰＳ（Mega Floating Operations/Sec.）である。

しかし，ベクトルチェイニング処理機能により，さらにピーク処理能力を高めることができるように設計されている。

例えば，次のような

Ｚ（Ｉ）＝Ａ（Ｉ）＊（Ｂ（Ｉ）＋Ｃ（Ｉ））

を演算する場合，Ｂ（Ｉ）とＣ（Ｉ）の加算の結果を直ちに乗算ユニットへ送り，Ａ（Ｉ）との乗算を行なうことにより，加算と乗算とが並列に処理されるような形にできる。この場合には，１２.５nsの刻みごとに，加算と乗算の２つの浮動小数点演算結果を得ることができるから，この場合のピーク処

表4．4．1　科学技術計算専用コンピュータ

名　　称	メ　ー　カ	処理方式	稼動または発表時期
STAR−100	Control Data Corp.	パイプライン	1973
ASC	Texas Instruments	パイプライン	1972
ILLIAC Ⅳ	Burroughs	並列処理（プロセッサ・アレイ）	1973
CRAY−1	Cray Research	パイプライン	1976
BSP	Burroughs	並列処理	1977発表
DAP	International Computer Ltd.	並列処理（プロセッサ・アレイ）	1979
CYBER−203	Control Data Corp.	パイプライン	1979発表
FACOM 230／75APU	富士通	パイプライン	1976
HITAC M−200H IAP	日　立	パイプライン	1979

理能力は160MFLOPSとなる。ただし，実際にはオペランド供給が間に合わず，これだけの処理能力を出すことはむずかしい。

（b）　BSP[1)]

並列処理方式の代表的なものとしてBSPを挙げることができる。BSPは，複数の演算ユニットを並列に動作させ，ベクトル処理の高速化を図っている。

主な特長として，１６個の演算ユニット（ＡＥ：Arithmetic Element）による並列処理，４～６４Ｍ語のＣＣＤメモリの２次記憶，コンフリクトフリーメモリ，ＤＯループの自動最適化などを挙げることができる。

主なハードウェア仕様は，次のようである。

- マシンサイクル　　８０ns（ＡＥは１６０ns）
- 演算ユニット数　　１６個
- 浮動小数点加算時間　　３２０ns
- 浮動小数点乗算時間　　３２０ns
- 主記憶　　最大８Ｍ語，１７バンク
  サイクル１６０ns

この並列処理方式では，１６個のＡＥがすべて並列に動作したとき，３２０nsごとに浮動小数点演算結果を１６個得ることができるから，２０ns刻みに１個の浮動小数点演算結果を得るパイプライン方式と等価であり，ピーク処理能力は５０ＭＦＬＯＰＳである。

１６個のＡＥを効率よく動作させるため，行列データを主記憶（パラレル記憶と呼んでいる）上に格納したとき，どの行や列についても１６個の要素が一度にアクセス可能なように素数である１７バンクの主記憶インタリーブを行ない，コンフリクトフリーメモリと呼んでいる。

同様に，１６個のＡＥを効率よく動作させるためには，ベクトルの要素数が１６以上であることが望ましい。例えば，二重のＤＯループ

```
      DO  10  I＝1，70
      DO  10  J＝1，5
      A（I，J）＝B（I，J）＋C（I，J）
10    CONTINUE
```

の処理では，内側が５回のループなので，要素の数５のベクトル演算を７０回繰返すことになる。このＤＯループでは，ＩとＪの演算順序を変更しても結果は同じなので，内側をＩについて７０回のループにすれば，要素の数

７０のベクトル演算を５回繰返すことになり，この方が効率よく処理される。このようなとき，自動的にＩとＪの演算順序を入換える機能をもたせ，処理効率の向上を図っている。

（Ｃ）　ＤＡＰ[1)]

プロセッサアレイ方式の代表的なものとして，ＤＡＰを挙げることができる。ＤＡＰは，並列処理をさらに推し進めたもので，１ビット幅の演算器をもつプロセッサエレメント（ＰＥ）を３２×３２（１０２４）個，６４×６４（４０９６）個などのように配列し，ベクトル処理の高速化を図っている。

各々のＰＥは，４Ｋビットの記憶をもち，６４×６４配列した場合には，記憶容量は合せて２ＭＢとなる。この２ＭＢの記憶部は，ＩＣＬ２９００処理装置の主記憶の一部として接続される。言い換えれば，ＩＣＬ２９００処理装置の主記憶の一部にＰＥが組込まれている。

ＩＣＬ２９００処理装置がＰＥの組込まれている主記憶上にデータを用意し，ＰＥに演算を指令する。ＰＥが演算をしている間，ＩＣＬ２９００処理装置は通常の主記憶を使用して，通常の演算処理を行なうことができる。ＰＥの演算が終了すると，ＩＣＬ２９００処理装置に割込みをかける。

現時点での主な仕様は，次のようである。

- マシンサイクル　　２００ns
- ＰＥの個数　　４０９６（６４×６４）
- 演算データ幅　　１ビット
- 浮動小数点加算時間　　１３５ns（３２ビット）
- 浮動小数点乗算時間　　２５０ns（３２ビット）
- 主記憶　　最大２ＭＢ（６４×６４×４Ｋビット）

このような方式は，今後の半導体技術の進歩により，ＰＥを１チップで構成し，マシンサイクルの向上やアレイの個数の増加が可能であり，性能向上ポテンシャルが大きい。

次に，特殊目的の科学技術計算プロセッサの具体例について述べる。

（ｄ） Navier－Stokes Equation Solver[5)]

ＮＡＳＡが，ＮＡＳＦ（Numerical Aerodynamic Simulation Facility）プロジェクトで計画しているNavier－Stokes Equation Solver の目標性能は，１００×１００×１００の格子点でのNavier－Stokes Equationの解が１０分以内に求まることであり，具体的には１０００ＭＦＬＯＰＳである。

プロセッサの構成のガイドラインとして，ＦＭＰ（Flow Model Processor ）とＳＰＳ（Support Processor System ）とからなり，ＦＭＰは数値計算専用プロセッサであり，ＳＰＳはコンパイル，ジョブスケジュール，ファイル処理，会話型処理などを行なうプロセッサである。

このＮＡＳＦプロジェクトには，バローズ社とＣＤＣ社からの提案があり，いずれも大規模な並列処理を採用したプロセッサである。

バローズ社の提案は，３ＭＦＬＯＰＳのＡＥ（Arithmetic Element）を５１２個並列に接続して，Ａ（Ｉ）＊Ｂ（Ｉ）の処理で３×５１２≒１５００ＭＦＬＯＰＳのピーク処理能力をもつような構成である。この構成は，バローズ社のＢＳＰの１６個のＡＥによる並列処理を大幅に拡張したものと考えることができる。

ＣＤＣ社の提案は，ＳＴＡＲ－１００をベースにしたパイプラインプロセッサ複数台による並列処理の構成である。すなわち，８台のパイプラインプロセッサを並列に接続したもので，Ａ（Ｉ）＊Ｂ（Ｉ）で５００ＭＦＬＯＰＳ，Ａ（Ｉ）＊Ｂ（Ｉ）＋Ｃ（Ｉ）＊Ｄ（Ｉ） の複素数計算で１５００ＭＦＬＯＰＳの処理能力をもつような構成である。

このようなＮＡＳＡのプロジェクト以外にも並列処理によるNavier－Stokes Equation Solver についての提案は多い。その代表的なものとしてRand Corporationの提案がある。これは１００×１００の10000個のマトリクスプロセッサである。1個のＰＥ（Processing Element）

は1個の半導体チップで構成されるものとし，これを2次元アレイ状に配列したものである。

この場合に問題となるのは，1個のＰＥは前後左右のＰＥとのみデータ交換が可能である点である。Navier－Stokes Equationがこのような条件で解くことができれば，この方式は将来のハードウェア技術の進歩に沿って高速にすることができる。

（ｅ）　その他の科学技術計算専用プロセッサ[3]）

ＰＥをアレイ状に配列する方式のもので，ＰＥとして汎用のマイクロプロセッサを使用しているものがある。

これらの例として，シーメンス社中央研究所で，気象予報計算を対象に開発しているSMS 201がある。これはPEとしてIntel 8080 を使用している。

国内でも，京大原子力研究所で，原子炉内のシミュレーションを対象に開発しているＰＣＡＳがある。これはＰＥとしてMotorola M 6800を使用している。

（ｆ）　信号プロセッサ[2]）

信号プロセッサには，ＩＢＭ社の３８３８アレイプロセッサをはじめ，Floating Point 社のＡＰ－１２０Ｂ／１９０Ｌ，Computer Signal Processor 社のＭＡＰ－３００など商用機として広い範囲に使用されているものがある。

代表的な利用分野としては，資源探査，特に石油探査の分野がある。

これらのプロセッサは，雑音の中に埋れた信号を取出すということで，フィルタリング処理，コンボリューション積分，フーリエ変換などを高速処理することが特長である。

また，信号プロセッサの特殊なものとして，弾道ミサイル防衛のために開発されたＰＥＰＥ（Parallel Element Processing Ensemble）や，航空管制のリアルタイム処理や画像処理のために開発されたＳＴＡＲＡＮなどがある。これらのプロセッサでは，並列連想処理による高速処理を行なっている。

（ｇ） 画像プロセッサ[4)]

画像処理には，信号プロセッサを利用することが多く，先に挙げたＡＰ－１２０ＢやＭＡＰなどが使用されることがある。またＳＴＡＲＡＮも同様である。

このようなプロセッサでは，画像の微分，平滑化，強調，画質の改善などの空間フィルタリング処理などを行なうことができる。

また，画像処理に必要なものとして，データの拡大，反転，シフト，座標軸の変換などの処理があるが，このような処理を効率よく行なうために専用の画像プロセッサが開発されている。

画像プロセッサとしては，研究開発段階のものが多く，スエーデンのＰＰＭ（Parallel Picture Processing Machine），ロンドン大学のＣＬＩＰ（Cellular Logic Image Processor）など，日本では東芝総研のＰＰＰ（Parallel Pattern Processor）などがある。

(3) 処理方式上の課題

今後の課題の最大のものは，さらに高速処理を実現するには，どうすればよいかということである。

パイプライン方式では，パイプラインの時間刻みを小さくする必要があるが，ＣＲＡＹ－１はすでに１２.５nsであり，１０倍高速にするためには，刻みを１ns 近くにしなければならず，半導体の技術ではむずかしい問題であろう。

このために，今後，処理能力を向上させるためには，並列処理化せざるを得ない。並列化を極端に進めていくと，ＤＡＰのようなＰＥをアレイ状に配列する方式になる。この方式では，ＰＥの数を増加させることにより，性能向上を図ることができる。また，半導体技術上も１チップでＰＥを構成し，これを高速化させることができる。

しかし，これまでの数値計算のプログラムは，一般的に直列処理を前提に作られているものが多い。したがって並列処理プロセッサを効率良く動作させるためには，プログラムの変更やアルゴリズムの開発が必要になる。

今後の課題としては，次のような事項を挙げることができる。

（ａ） パイプライン方式の場合

- 新しい高速論理素子の開発
- 複数パイプラインによる並列処理

（ｂ） 並列処理およびプロセッサアレイ方式の場合

- 並列処理アーキテクチャの開発
- 数値計算アルゴリズムの開発
- 高級言語の開発

また，これらと異なる次元の問題として，計算精度の向上と，計算数値範囲の拡大の問題がある。特に，数値範囲の制約は，数値計算アルゴリズムにも影響が大きい。このためオーバフローの生じない数表現などに対しても十分留意する必要がある。

(2) 第５世代における利用形態の可能性

今後，超高速計算のニーズは増大して行く見通しであるが，応用分野が多様なことやそれぞれの分野で計算したい問題が異なることなどのために，利用形態としても多様にならざるを得ない。

現在，多くの科学技術計算専用プロセッサは，科学技術研究機関において，計算センタ的な形態で使用されている。

このことは，今後の科学技術計算専用プロセッサのアーキテクチャに重要な影響を与えるであろう。すなわち，科学技術計算専用プロセッサも汎用機を中心とした計算機システムの中に組込めるものでなければならない。

したがって，一般的には，汎用機をホストプロセッサとして，それに接続される形で演算実行のみを超高速処理する専用機として使用されるであろう。この場合には，処理方式が，パイプライン方式であっても，並列処理方式であっても問題はない。

次に，対象とする科学技術の分野により，並列処理が適していれば，並列処理方式の専用機を接続し，プロセッサアレイ方式が適していれば，それを接続

するという形で，各方式の専用機が汎用機システムに接続されて使用されるであろう。

4．4．3　超高速計算技術の応用と効果

科学技術計算専用機の応用分野は多様であるが，これまでに使用されてきた分野としては，次のような分野がある。

- 資源探査（石油探査など）
- 気象予報
- 航空工学（飛翔体の空力設計など）
- 原子核物理学（原子炉設計など）

以上の分野は代表的なものであり，今後の科学技術の進歩を考えると，さらに計算が大型化して行く分野として，

- 核融合
- 画像処理
- 分子科学
- 構造計算
- 宇宙航空工学

などがある。

特に，今後は，シミュレーションの問題が増加すると考えられる。例えば，航空工学の風洞実験のシミュレーションがある。風洞実験を行なうのと同程度の時間で，精度の良い結果が得られるようになれば，大部分を計算機でシミュレーションし，確認のために風洞実験を行なうということができるであろう。ＣＲＡＹ－１の１０～１００倍程度の処理能力があれば，このようなことがある程度可能なようである。

また，核融合のシミュレーションについても同様である。

Navier－Stokes Equation Solver が風洞のシミュレーションを目的とし，ＳＭＳ２０１が気象計算を目的とし，ＰＣＡＳが原子炉のシミュレーシ

ョンを目的としているように，今後は，問題のニーズに対応してプロセッサが開発される可能性があり，それらの特長を十分に発揮することができれば，科学技術の進歩への寄与も大きい。

4．4．4　現実的可能性

当面の課題をＣＲＡＹ－１の１０～１００倍の超高速計算という目標を置いた場合に，次のような問題を解決しなければならない。

パイプライン方式で実現するとすれば，マシンサイクルを１ns近くにしなければならないので，半導体技術ではなくジョセフソン素子などの技術を使用可能にしなければならない。

複数パイプラインによる並列処理，あるいは，プロセッサアレイ方式で実現するとすれば，複数の演算ユニットを効率よく使用するための，プログラムの変更，数値計算アルゴリズムの開発，並列処理言語の開発などが必要である。また，ハードウェアの構成としては，主記憶と多数の演算ユニットとの間のデータ転送方式の解決が課題となる。

例えば，１個の浮動小数点演算を５００nsで処理するＬＳＩチップを100×１００個配列すれば２０,０００ＭＦＬＯＰＳとなりＣＲＡＹ－１を８０ＭＦＬＯＰＳとしたときの２５０倍になる。このようなものを１０年以内に作ることは，多分，可能であろう。

しかし，このような計算機を使いこなすには，システムへの接続方式の開発，並列処理言語の開発が必要であり，今後の課題となろう。

１９９０年代には，ハードウェア技術の面では，超大型並列プロセッサを作ることができると思われるので，信号処理，画像処理の分野も含めて，これまで処理できなかった問題が短時間に処理できる可能性がある。

参考文献

1）中沢，小高「最近の超高速コンピュータの動向」，日本物理学会誌，３４巻，７号，　pp. ５８１～５８９，１９７９

2）内田「信号処理プロセッサ」，情報処理，Vol. １８，№４，　pp. ４０２～４０９，１９７７

3）星野「偏微分方程式解析のためのマイクロプロセッサ複合体」，情報処理，Vol. ２０，№１１，　pp. ９７４～９８２，１９７９

4）木戸出，篠田「ディジタル画像高速処理装置の流れを追う」，日経エレクトロニクス，１９７８．７．２４，　pp. １１０～１４０

5）「Future Computer Requirements For Computational Aerodynamics」，NASA Conference Publication ２０３２，Ames Research Center, Feb. 1979

6）村上，西川「コンピュータアーキテクチャの最近の進歩」，電子通信学会誌，Vol. ６０，№６，　pp. ６６９～６８０，１９７７

7）「High Speed Computing and Algorithm Organization」，Academic Press，１９７７

4.5 機能分散型計算機

4.5.1 概　説

(1) はじめに

機能分散型計算機（Distributed Function Computer System）とは，計算機システムの中核となる中央処理装置の上で，ソフトウェアを含めて実現されている機能の一部を，論理的にも物理的にも明確に区別できる機能ユニットとして独立させ，それを計算機本体と有機的に結合して動作させる形態の計算機システムを示すと云える。この機能分散型計算機システムのとらえ方としては，次の２種に大別される。

- 機能分散型計算機システム：計算機そのものを，機能分散により構築する。
- 機能分散型データ処理システム：データ処理をいくつかの段階に分割し，それぞれを階層的，あるいは平面的に配置した複数台の計算機に分担させるシステムである。

後者の機能分散型データ処理システムでは，その配置される計算機システムの地理的条件から，さらに次の２種に分けられる。

- ローカル・ネットワーク型：同一室内，同一フロア，同一建屋，あるいは同一構内等の割合い限られた領域内に分散配置される計算機群で構成され，計算機間の接続は，高速バス，高速ループ，チャネル等，大量情報交換を可能とする方法で行なわれるもの。
- グローバル・ネットワーク型：一般には，地理的に離れて分散配置され，計算機間の接続は，通信回線等で行ない，限られた情報の交換しか行なえないタイプのもの。

以下では，機能分散型計算機システムおよびローカル・ネットワークによる機能分散型データ処理システムについて述べることとし，グローバル・ネットワ

ークによる機能分散型データ処理システム，および，複数個の同一計算機を何らかの手法で結合し，その各々で機能的にはまったく同じ処理を行なう負荷分散型データ処理システムについては扱わない。

(2) 機能分散化の歴史的経緯

機能分散型計算機システムは，１９５８年に米国標準局ＮＢＳ（National Bureau of Standards)で開発されたＰＩＬＯＴが端緒と云える。ここでは入出力処理，システム制御，および演算処理の３台の処理装置がある。本格的に実用化されたのは，１９６３年に発表されたＣＤＣ－６６００システムであり，ここでは，中央処理装置の他に１０台のペリフェラル・プロセッサで入出力処理およびスケジューリングが行なわれている。この後，通信制御処理，アレー処理，シンビオント処理，など種々の機能ユニットの分散化が図られている。最近では，ＣＰＵにファームウェア機能を導入することにより，特定言語の処理の実行，ＯＳ機能の高速化などの機能分化も図られつつある。

一方，機能分散型データ処理システムとしては，１９６７年に発表されたＩＢＭのＡＳＰ（Attached Support Processor）を皮切りに，疎結合マルチプロセッサの考え方が登場し，さらに最近では，ミニコンピュータを複数台結合し，大型機に代るシステムを構築する動きも出ている。

以上述べた通り，機能分散の考え方は，徐々に広まりつつあり，その分散の内容も単純なレベルから，複雑なレベルへと発展しつつあるように思われる。以上の歴史的経緯のうち，代表的システムを表４．５．１に示す。

(3) 機能分散化の狙い

機能分散化の主な狙いには次のものがある。

- 専用化による個別機能の処理性能の向上を図る。
- 要求性能に応じモジュラにシステムを発展させうること。
- システム信頼性の向上を図る。
- ハードウェア／ソフトウェアの開発をモジュラに行なうことにより効率化を図る。

表４．５．１　機能分散型計算機の歴史

年　代	機能分散型計算機システム	機能分散型データ処理システム
１９６０	ＰＩＬＯＴ DATANET CDC－６６００	
１９６５	２９３８AP	ＡＳＰ
１９７０	LOGICON2+2 3705CP　SYMBOL U-1100　CP/SP 370/125 BCC－５００	ＪＥＳ３ C.mmp DCS
１９７５	NCR－COBOL　VM APL Assist　PPS－R MVS/SE	Bank of America PLURIBUS TANDEM－１６
１９８０		

(4)　機能分散化の前提条件

機能分散化が，その当初の狙い通りの効果を発揮するために，システム構築に当り考慮しておかなければならない条件に次のものがある。

・トータル・システムとして従来手法より性能／価格比のすぐれていること：機能分割が中途半端なため，分割されたモジュール相互間のコミュニケーション・オーバーヘッドが増えたり，専用プロセッサ間で負荷の不均衡が生じ，全体としては採算のよいシステムとはならない恐れがある

・システムの保守性が良いこと：ハードウェア／ソフトウェアの分割により，

保守が複雑とならないこと。

- システムの運転性が良いこと：分散により，システムの運転が複雑とならないこと。
- 技術（ハードウェア／ソフトウェア）の発展に大幅なる変更なく追従できること：その時々で得られる最新の技術で，最善の性能／価格比を得るのは望ましいが，次の世代でアーキテクチャの根本的変更を伴うのでは，種々の面からみて問題が多い。

4.5.2 機能分散化の技法

(1) 機能分散化のレベル

現在までに提案されている，あるいは，試作／実用化されている機能分散化の手法のレベル分けをしてみると，次の3種に大別される。

- 組込みレベル：計算機システムを構成している個々のコンポーネントに，特定の機能を付加し，その目的に使うときのみ，専用コンポーネントとみられるようにする方式。ここでは，ＣＰＵ，メモリ，ディスク等の個々のコンポーネントへの各種機能付加が考えられる。
- サブシステム・レベル：計算機システムの備えるべき部分的機能，たとえば，入出力処理，データ管理，通信管理などの機能を専用のプロセッサを設け，分散させる形態である。
- システム・レベル：既に独立した計算機システムとして存在しうるものを，何らかの手法により複数台接続し，それら各々に，専用化した機能を割当てるもの。分散化されているコンポーネント・システムは，各々同じ機能を発揮できる素質は備えているが，そこに接続されている入出力装置およびソフトウェアにより専用化されている形態である。

(2) 機能分散化の方式

以下では，上記3種のレベルに応じた機能分散化の方式を，具体的例に基ずいて列挙する。

（ａ） 組込みレベル

組込みレベルとしては，メモリ，ＣＰＵ，ディスクなどのコンポーネント毎に，各々，各種方式がある。

（Ⅰ） メモリにおける組込み機能

この組込みレベルとして具体化されている例は余りないが，将来はかなり伸びると予想される機能分散方式である。例えば，連想メモリ方式，ディスクリプタまたはケィパビリティ情報をメモリ側に付加しておき，ＣＰＵ／チャネル等からのアクセスに対し，メモリ側でチェックする方式などが考えられる。

（Ⅱ） ＣＰＵにおける組込み機能

この組込みレベルとしては既に数多くの具体例があり，今後とも，その発展は大いに期待される機能分散方式である。例えば，技術計算の高速化のために，浮動小数点演算専用のプロセッサを付加するＦＰＰ方式，アレー専用の機能を付加するＩＡＰ方式，各種関数サブルーチンをファームウェアで高速に行なう方式などがある。また，ＯＳの高性能化を図るため，一部機能のファームウェア化を行なっているものにＩＢＭのＭＶＳ／ＳＥ，ＶＭ Assist などの例がある。以上は部分的専用化の例であるが，さらに範囲を広げて，特定の言語処理そのものを専用ファームウェアで処理してしまうＩＢＭの**APL Assist**，ＮＣＲのＣＯＢＯＬバーチャル・マシン，ＦＯＲＴＲＡＮバーチャル・マシンなどの例もある。

（Ⅲ） ディスクにおける組込み機能

ディスクそのものに機能を組込んだ例としては，ＩＢＭのディスクにおけるContent Search 機能がある。これは，プログラムの与えたキーと，ディスク上に記憶されているキーとの簡単な論理条件チェックにより，所定のレコードを引出す一種の連想読み出し方式である。実験レベルではリレーショナル・データベースの実現方式として，デ

ィスクのヘッド毎に特定の論理機能を組み込むＲＡＰなどの方式がある。

（ｂ）　サブシステム・レベル

サブシステム・レベルでの機能分散は，現実のシステムでも数多く見られるものであり，次の６種のものが，何らかの形で実用化されている。

- 入出力処理プロセッサ
- 通信処理プロセッサ
- データベース処理プロセッサ
- アレー処理プロセッサ
- 保守・診断プロセッサ
- 階層メモリ制御プロセッサ

（Ⅰ）　入出力処理プロセッサ

この機能分散型計算機としてはＣＤＣ－６６００がある。ここでは，主プロセッサの他に１０台のペリフェラル・プロセッサがあり，入出力処理および主プロセッサのタスク・スケジュールを行なっている。これ以外にもいくつかの実用化例がある。

（Ⅱ）　通信処理プロセッサ

ＧＥのＤＡＴＡＮＥＴシリーズ以来数多くの例があるが，ＩＢＭシステム／３７０と共に登場した３７０４／３７０５コミュニケーション・プロセッサが最も典型的である。ここでは，通信制御処理の一部をホスト計算機からオフロードし，性能およびソフトウェア保守などの隘路を解消することを狙っており，ネットワーク・アーキテクチャの発展と共に重要性を増してきていると云えよう。

（Ⅲ）　データベース処理プロセッサ

汎用機でこの専用プロセッサ適用例はまだ出現していないが，オフィス・コンピュータではＤＡＴＡＰＯＩＮＴ－１００などの如く実用化例が出はじめている。実験的システムでは，数多くの事例があり，計算機システムにおける処理時間比重がきわめて高いことと相俟って，近い将

来汎用機においても導入が期待されている。

（Ⅳ） アレー処理プロセッサ

ＩＢＭ３８３８，ＣＤＣ－ＳＴＡＲ，バローズＢＳＰ，ＣＲＡＹ－１など数多くの実用化例があり，ミニコンピュータに接続するアレー処理専用プロセッサも数多く存在している。データフロー・マシン技術などの進展により，今後の発展の期待される機能分散化の方向である。

（Ⅴ） 保守・診断プロセッサ

最近の汎用機では，ほとんどの機種がこの種の専用プロセッサを装備している。システム運転操作の簡単化，自動化の方向とも合体して，保守・診断プロセッサの役割は，さらに強化されることとなろう。

（Ⅵ） 階層メモリ制御プロセッサ

異るメモリ階層間の情報交換を専用機能により行なうのは，キャシュ・メモリが最初であり，続いてＩＢＭ３８５０ＭＳＳの如く，ディスクと磁気テープの間での階層記憶制御の実用化へ進み，最近では，主メモリとディスクの間へ，新たなる階層を持ち込み，ディスクとの間の情報交換を自動的に制御するディスク・キャシュの実用化が，一部で導入されつつある。

サブシステム・レベルの機能分散を行なうシステムでの個々の機能分散ユニットの結合方式の面では，次の２種の方法が使われている。

- チャネル接続：機能分散ユニットをチャネルの先に一種の入出力制御装置として接続する。
- 内部バス接続：ＣＰＵと特定のバスで接続し，場合によっては，主メモリをも共有する。

今のところ，前者が多いが，今後は後者のアプローチも重視されてこよう。

（ｃ） システム・レベル

システム・レベルの機能分散とは，既成の計算機システムを複数台接続し，各々に専用化した機能を持たせる方式である。これの代表的例として，以下

に２種を挙げる。

第一の例はＩＢＭ－ＪＥＳ３である。ここではＭＶＳの下で制御される１台のグローバル・プロセッサの下に，最大７台迄のやはりＭＶＳで制御されるローカル・プロセッサをチャネル間接続し，一体のシステムとするものである。グローバル・プロセッサは，システム全体の管理，ジョブの前処理，およびジョブの後処理を行ない，ローカル・プロセッサは，ジョブの実行のみを行なう。

第二の例は，ＴＡＮＤＥＭ－１６である。ここでは，最大１６台迄のＣＰＵを２重化された高速バスで接続すること，および２台のＣＰＵ間で，ディスクの共有を図ることにより，システムを構成している。このシステムでは，通信制御を主体とした前処理と，ファイル処理を主体とした後処理を各々複数台のＣＰＵで行なうことにより，高信頼性の実現と，モジュラに処理能力を拡充することを目的としている。

システム・レベルの機能分散を行なうシステムでの個々の計算機を接続する方式には，次の３種がある。

- チャネル接続：チャネル間結合装置を介して２台のシステムを接続する。
- ディスクの共有：複数計算機間でディスクの共有アクセスを行なう。
- 共通バス接続：複数の計算機間を高速バスで接続する。

4．5．3　機能分散化の効果

既に機能分散化の狙いの項で述べた通り，機能分散化は，性能の向上，モジュラな展開，信頼性の向上，および，ハードウェア／ソフトウェアの開発の効率化などを目的としている。これらの狙いが，現時点でどのように効果をあげているかを次に述べてみる。

(1)　性能の向上

組込みレベル，サブシステム・レベルで述べた例は，何らかの形で性能向上に繋がっており，この面で，機能分散の効果は大いに出ていると云え

る。たとえば，組込みレベルでのＦＰＰ，ＩＡＰなどでは数倍以上の性能向上が一般に期待されており，ＯＳのファームウェア化，高位言語の直接実行においても同様の効果があると云われている。また，サブシステム・レベルの機能分散では，一桁以上の性能向上を期待されているデータベース処理，アレー処理などの例もある。しかしながら，本格的にどの汎用機でも受け入れられている共通的機能分散の方式と云うものはない点に注目したい。

(2) モジュラな展開

システム・レベルでの機能分散は，まさにこの例であり，サブシステム・レベルでの機能分散の中でも，同一サブシステムを複数個接続することにより，要求に応じたモジュラな展開を果しているものもある。この面でも機能分散は認められつつあるが，広く受け入れられているとは云い難い状況である。

(3) 信頼性の向上

サブシステム・レベルでの分散例としての保守・診断プロセッサ，システム・レベルの分散例で述べたＪＥＳ３，ＴＡＮＤＥＭ－１６などの例も，信頼性向上のための機能分散化と云える。保守・診断プロセッサは，殆んど全ての汎用機にとりこまれて，その効果を発揮しつつあると云えるが，システム・レベルの機能分散化は，今後の推移をみないと広く普及するか否かは予想し難い。

(4) 開発の効率化

特定機能を切離すことにより，ＣＰＵ，ＯＳなどの肥大化を防ぐ試みは，サブシステム・レベルでの機能分散の主要な狙いとなっており，この面では，大いに効果を上げていると考えられる。今後は，ＯＳの巨大化を防ぐために，ＯＳ自体をモジュラに構成し，それを各々専用のプロセッサで実行する方向は増々強まるであろう。それにつれて，多重プロセッサ制御，プロセス間通信，セキュリティ／プロテクション，イン

ターロック，システムの動的再構成およびスケジューリングなどの各方面において，より優れた方式の開発が必要となろう。

4．5．4　機能分散化の将来性

機能分散型計算機システムが受け入れられて行くには，既に述べた各種観点での長所が認められるだけでは不十分であり，機能分散化の前提条件の項で述べた通り，保守性，運転性の面で従来手法に比べ遜色のないこと。現行システムとの互換性の維持の図れること。一時的総花とならず，ある程度連続的に発展できる方式であること等が必要条件となろう。この機能分散化の将来性および実現の上での技術面，アーキテクチャ面での問題点などについて以下で述べる。

(1)　適 用 分 野

機能分散型計算機システムの今後の主たる適用分野としては，次の３種のものが考えられる。

- オフィス・オートメーション
- 超高性能を期待する分野
- 新しい機能を求める分野

オフィス・オートメーションの分野では，文書処理，イメージ処理，ファイル処理，メッセージ／イメージ／音声などの複合交換処理，各種機器の制御などの種々雑多な処理機能の統合化が求められており，一方では，一般事務器並みの低価格化も要求されている。このような分野では，ＬＳＩ技術，マイクロコンピュータ技術を駆使して，個々の処理要素を専用化してコンパクトかつ低廉に作り上げ，必要に応じてモジュラに増設しうる『機能分散型システム』は，最も適合したものとなると予想される。

超高性能を期待する分野とは，技術計算処理，データベース処理を中心とする大規模オンライン処理などを示す。ここでは，処理時間の短縮，あるいは，スループットの向上を図ることにより，従来では実現できなかった新たなシステム作りが可能となってこよう。このためには，単なる汎用

機だけでなく，目的別に専用化した機能を持ち，かつ，高速・大容量の処理を行なえる『専用プロセッサ』の出現が望まれる。

また，新しい機能を求める分野としては，従来型計算機では不得手なイメージ処理，あいまいさを許すパターン認識処理，何らかの意志決定を行なうための高速・大容量のシミュレーション処理などを使用する適用分野が考えられる。この様な処理が可能となれば，社会，経済，経営，人事管理等の各種意志決定上の複雑な問題での計算機の果す役割は，一段と深まるであろう。

(2) 実現の可能性と問題点

（ａ） オフィス・オートメーション

６０～７０年代が，ブルーカラーを中心とする単純作業の自動化，省力化の時代であったとすれば，８０～９０年代は，ホワイト・カラーを中心とする各種日常業務の生産性向上を図る時代と云える。ここでは，ＬＳＩ技術，マイクロコンピュータ技術，伝送技術，ストレージ技術などの基礎技術の発達により５年程度でかなりの目標は達成されると考えられる。しかしながら，高度の判断を必要とされる経営問題，社会，経済に係る問題の解決のためのサポートは，このオフィス・オートメーションの流れでは解決されないであろう。

（ｂ） 超高性能を期待する分野

超高速技術計算，高速大容量データベース処理などの実現には，ＬＳＩ技術を中心とするハードウェア技術の進歩のみでは不十分であり，データフロー・マシン，連想記憶技術などのアーキテクチャ技術，各種演算アルゴリズム，データベース・モデル，分散データベース管理モデルなどの理論を含む多方面にわたる発達が必要とされよう。これらの実現は５年～１０年程度でかなり解決され，１９９０年を迎える頃には，現状の計算機技術と同じくらい成熟したものとなるであろう。

（ｃ） 新しい機能を求める分野

既に述べた通り，この分野の発展のためには，イメージ処理，パターン認識処理，シミュレーション処理などの理論面での研究，すなわち，アルゴリズムの確立と，それを実現する上での新しいアーキテクチャの導入の両方が必要とされる。これらの実現には，従来のVon－Neumann 型でない新しい計算機アーキテクチャの導入が必須であり，単に力に頼るのみでは，本格的発展はないと予想される。いずれにしろ，機能分散化のためには，上記に述べた個々の機能モジュールの構築方法の研究の他に，個々のモジュールの結合方式，モジュール間の情報交換方式，モジュールの構成制御，および，ＬＳＩ／実装などのハードウェア技術などの多方面にわたる関連技術の研究も必要とされる。

以上．機能分散型計算機システムの将来動向を述べてきたが，ハードウェアのシステム価格に占める比重の大幅低下の予想される現在，問題解決に必要とされる機能の定義と，機能を駆使するための標準的言語，および，それを効率的に実現する専用プロセッサの開発の方向は，８０～９０年代を通しての計算機技術の基調となると予想される。

参考文献

1）相磯秀夫；機能分散型計算機システム，情報処理，Vol. １８，№４，pp. ３２５～３３３（１９７７）

2）関野　陽；分散処理技術，情報処理，Vol. 20，№４，pp. ２７５～２８３（１９７９）

3）David J. Farber ; The Distributeb Computing System, COMPCON １９７３, pp.31～34

4）Earl C. Joseph ; Distributed Function Computer Systems : Innovative Trends, COMPCON, Feb. １９７４ pp.７１～７４

4.6　ユニバーサル・ホスト・プロセッサ

4.6.1　概　　説

(1)　マイクロプログラミング技術における位置づけ

ユニバーサルホストプロセッサ（以下，UHPと略す）は，各種の機械語命令セット（以下，ＩＳＰ）アーキテクチャを実現できるという意味での汎用性を具えたマイクロプログラマブルプロセッサである。したがってマイクロプログラミング技術との関連が重要になるが，その詳細を述べる紙面はないので，ここではＵＨＰの位置づけのみを簡単に述べておく。

マイクロプログラミング制御方式の基本思想は図4.6.1に示すように，計算機アーキテクチャに２つのレベルを設定し，応用側に接する上側レベルの命令（ＩＳＰ）をハードウェアに直結した下側レベルの命令のサブルーチンとして実現するという階層構造性にあるが，この階層性の利用方法には２つある。

第１は上側のレベルを固定にしておき，下側レベルの位置（したがって，ハードウェアとファームウェアの比率）を可変にして同じく機械語命令（ＩＳＰ）レベルアーキテクチャを持つ各種の性能のマシンを実現することである。これは通常ファミリーマシンと呼ばれ，３～４世代の殆んどすべての商用機で採用されている。

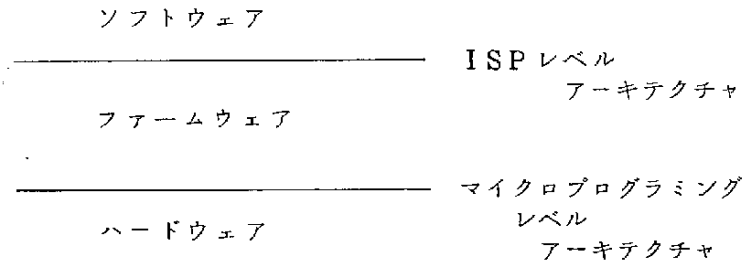


図4.6.1　計算機アーキテクチャのレベル

第2は下側レベルを固定して，マイクロプログラミングによって各種の応用に適した上側レベルアーキテクチャを実現するという考え方であり，このような目的に特別に設計されたプロセッサがＵＨＰである。

(2) 発展史

図 4.6.2 にＵＨＰ関連技術の歴史的経緯を示す。概括的に述べると，ＵＨＰの考え方の最初の萌芽は第2世代計算機の第3世代計算機によるエミュレーショ

	関連概念技術	ユニバーサルホストプロセッサ	その他の計算機
1951	マイクロプログラム制御の提案（Wilkes）		
1961	高級言語マシン		B5000（スタックマシン）
1964	マイクロプログラムによるファミリーマシン（第2世代機の）エミュレータ		IBM 360
	ダイナミックマイクロプログラミング		
1967	ファームウェアの概念（Opler）		
1969	内部計算機の概念（Rakoczi）		
	ランゲージボード	MLP-900	IBM 370
		QM-1	SYMBOL
	LSIマイクロプロセッサ（Intel社）	META-4	（高級言語マシン）
		B1700	
1972	ナノプログラミング ビットアドレシング	FCPU	C.mmp
1974		MATHILDA EMMY	
			IBM 5100 （パーソナルコンピュータ）
1977	4KスタティックRAM	CDC 480 クライテリオン 8500	

図 4.6.2　ユニバーサルホストプロセッサの発展

ン技法の開発に見られる。即ち，1960年代半ばに前者に対する既存のプログラムを後者上で実行するためのシミュレータの実行効率を高めるために，シミュレートしにくい命令の一部をマイクロプログラムでサポートすることが行なわれた。これをエミュレータと呼び，ＩＢＭ360上で走るＩＢＭ7090，1401等に対するものがかなり長期的に使用されていた。

このエミュレーションの考え方を一般化して，純粋にハードウェアで実現されるプロセッサ（図4.6.1の下側のレベル）とこの上にマイクロプログラムで実現されるプロセッサを明確に分離するという考え方を打出したのが，L.L. Rakoczi であって1969年のことであった。Rakoczi は両者を内側計算機（Inner Computer）と外側計算機（Outer Computer）と呼んだが，現在では前者をＵＨＰと呼ぶのが一般的である。

このＵＨＰの考え方は，書き換え可能な高速記憶素子の発展にともなってFlynn 等によって提唱されたマイクロプログラムの書き換えできるプロセッサ（ダイナミックマイクロプロセッサ）方式と結びついて発展し，1970年代には各種の研究開発が進められることとなった。

その結果，後に述べるようにエミュレーションの効率向上のための各種ハードウェア技法が開発され，多数の試作機と数種の商用機が実用に供されるに到っている。

(3) 他技術との関連

表4.6.1にＵＨＰと本章に述べられる他のアーキテクチャ技術との関連を示しておく。特に，機能分散型計算機，高級言語マシン，適応計算機ではＵＨＰに対する依存度が大きい。

表 4.6.1　ＵＨＰと他のアーキテクチャ技術の関連

節の名称	UHPへの影響	UHPからの影響
並列処理	並列非同期動作 UHP	
機能分散型計算機		各機能プロセッサをＵＨＰによって機能に適するように構成する。
コンピュータネットワーク		
科学計算		
高級言語マシン		間接型高級言語マシンのホストプロセッサとしての利用
データベースマシン		
適応計算機		動的ＩＳＰアーキテクチャの適応マシンのホストとして利用
データフロー計算機		プロセッサエレメントに利用
ソフトウェア工学	マイクロプログラム作成の効率化 ＩＳＰ命令の設定	ＩＳＰの高レベル化，適応化によるソフトウェアの減少，作成の容易化

4.6.2　技法とシステム

本節では，これまでに開発されたＵＨＰの主な技法と，代表的システムについて簡単に述べる。

(1)　基本的技法

ＵＨＰでは，エミュレーション対象のＩＳＰアーキテクチャをマイクロプログラムによって，ＵＨＰのハードウェアに効果的に写像できるかどうかがその有効性の重要なポイントとなる。特に，ＩＳＰ命令のデコードのように頻繁に実行される操作に多くの時間を要すると，せっかく命令の機能の実行をハードウェアに近いレベルで処理しても，その効果が生れない。そこで，これまでのＵＨＰ技法は，主にこの写像の効率化のサポートとエミュレーション用のマイクロプログラムの動的変更の容易さの強化という２つの方向を指向して開発されてきた。以下に，主な技法の要点をまとめておく。

①　ＩＳＰ命令デコードの効率化

デコード論理を結線論理として数種用意し，マイクロプログラムで選択利用

するランゲージボード（可変論理セット）方式，高速シフタとマスクレジスタの組合せ方式，ビットアドレシング方式，ＲＯＭによるジャンプテーブル法等が開発されている。

② 写像の効率化

ＩＳＰアーキテクチャをＵＨＰアーキテクチャに効率的に写像するために，個々のシミュレーション対象ＩＳＰアーキテクチャにおいては固定的である制御情報をレジスタ中にセットしておき，ハードウェアが直接利用する技法がある。これをレジデュアル制御と呼び，データ路の接続パターンの設定，演算モード，データ語長の設定等に使用されている。

③ 多層化

ＵＨＰはマイクロプログラムレベルとＩＳＰレベルの２層性を利用する技術であるが，この階層を３層以上に増加させ，制御性とプログラミングの容易さを柔軟に選択できるようにした技法にマルチレベルマイクロプログラミング方式がある。

④ 制御記憶空間の拡大と内容の動的変更の効率化

次のような方法がある。

a. 制御記憶ＲＯＭのオフライン交換

b. 入出力チャネルからの書き込み，専用の特殊命令

c. 主記憶空間と制御記憶空間の共用

ⅰ）主記憶空間を制御記憶用にそのまま使用

ⅱ）高速バッファの併用（低位アドレス，バウンダリレジスタによる指定，マイクロキャッシュ）

ⅲ）部分的かさね合せ

(2) 代表的ＵＨＰ

表 4.6.2 にこれまでに開発されたＵＨＰの内，代表的なシステムについてその特徴をまとめておく。

表 4.6.2　主なＵＨＰの特徴

システム名	開発組織	開発年	特　　徴	現　　状
MLP-900	Standard Computer Corp.	1970	• Rakoczi の会社で開発された最初の大型ＵＨＰ • IBM 7000シリーズを主対象とする 36ビットマシン • 写像の効率化のためにランゲジボード，マスクレジスタ，高速シフタを使用。	１台のみ製作され，南カリフォルニア大にPRIMシステムとして存在。 ARPA網に接続され，研究用に使用されている。
QM-1	Nanodata	1972～	• ニューヨーク州立大（バッファロー）の研究をもとに商品化されたＵＨＰ。 • マルチレベルマイクロプログラミング，データ路接続パターンのレジスタ設定方式の導入。 • ターゲットは主に IBM 360/370 であるが，ＵＨＰとして汎用性を生かした利用をねらっている。	1974年より軍関係と大学関係を中心に約20セットが出荷されている。
B1700／1800	Burroughs	1972	• ビットアドレシング機構を導入し，語長適応性を与えている。 • Cobol，Fortran 等の高級言語マシンの形で提供されている。	商用機として一応成功し，相当数が出荷されている。
FCPU	Data Saab	1972	• MLP-900 と同じ Lawson の設計。 • 演算，制御，メモリアクセスの３ユニットに分割し，非同期並列動作をさせている。 • ランゲージボード類似の可変論理セット機構を備える。 • メモリアクセスユニットは各種語長のアクセス等，論理記憶操作機能を持つ。 • 同社のD32をターゲットとする。	スエーデンの商用機であるが，利用状況は不明。
Cons	MIT	1974	• ジャンプテーブル用ディスパッチメモの使用。 • Lisp マシンのホストとして使用。	
CDC-5600	CDC		• データ長 8～32ビット（４ビットおき）可変 • ＡＬＵモード設定機能 • トランスフォームモジュール（１種のランゲージボード）	

この他，実験機として，Emmy（スタンフォード大），Mathilda（Aarhus大），ＡＣＥプロセッサモジュール（ＥＴＬ），PPS-1プロセッサモジュール（東大，富士通），PDP11-40／E（CMU），QA-1（京大）等がある。また，マイクロプログラムの変換をオフラインで行なうものに Meta-4 がある。

(3) 技法上の課題

デコードの効率化，制御記憶のアドレシング等のアーキテクチャ的技法は既に10年以上の歴史があり，アイデアはかなり出つくした感がないでもない。しかし，ＵＨＰの実用化は始まったばかりであり，今後もＬＳＩ化，並列非同期処理方式の導入等の新しい要因に対応したＵＨＰアーキテクチャの開発が必要である。

また，ＵＨＰを利用するための技術はまだ初歩段階であり，ＵＨＰの効果を生かすために研究開発すべき事項は少くない。これについては後に 4.6.4 でまとめる。

4.6.3 応用と効果

(1) ＵＨＰの応用

ＵＨＰはその２層構造という特徴を生かして，一般に次のような目的に利用することができる。

① １種のＵＨＰによる多種のＩＳＰアーキテクチャの実現

ＵＨＰの汎用性が高ければ，同一のＵＨＰに異なるマイクロプログラムを与えて，既存の（とは限らないが）各種ＩＳＰアーキテクチャを実現することができ，経済的であるし，少種多量生産というＬＳＩの特性にも適合している。この利用法ではＩＳＰアーキテクチャの動的変更が要請されない場合も多いと考えられ，その時は一部の機能を専用ハードウェア化してエミュレーション効率の向上を図ることもできる。System Ａにおいては，４世代機までのＩＳＰアーキテクチャのエミュレーション用に重要である。

② 応用に指向したＩＳＰアーキテクチャの実現

第５世代においては高級言語をベースとしたいわゆる高級言語マシンが機能マシンとして重要な役割を果たすと予想されるから，それを実現する母体としてＵＨＰは極めて重要である。即ち，ＩＳＰとして個々の高級言語に指向したものを設定し，そのインタプリタをファームウェアによってＵＨＰ上に実現

するわけである。この方法は間接型高級言語マシンと呼ばれ，ファームウェアの変更で動的に各種高級言語マシンを利用したい場合には，効率のよいUHPは必須であるといえよう。System Aでは各ユーザーのパーソナルコンピュータがこの形で用いられると予想される他，中央サイトの高級言語マシンでもＵＨＰが構成要素として利用されることが考えられる。

③　ＩＳＰの動的適応の実現

第2章に述べられているように，応用適応性は第５世代アーキテクチャを特徴づけるものの一つである。特にＩＳＰを応用にあわせて動的に変更していくには，ＵＨＰを用いて測定データに応じ，マイクロプログラムインタプリタを再構成する以外に適当な方法はない。

(2)　ＵＨＰの効果

①　実行速度の向上

ＩＳＰを高水準化し，マイクロプログラム部分を増加することによる実行速度の向上度は，一般に，３～10倍程度であるとのデータが発表されている。当然のことながらＩＳＰが応用に指向した高機能のものであればあるほどその効果は大きい。一方，対象がこれまでの一般計算機のエミュレーションである場合は命令機能に対し，デコードの比率が高くなるので，マイクロプログラムのみでデコードするとあまりよい性能は得られない場合が多いので，新しい工夫がない限りハードウェアのサポートによる半固定方式が不可欠となる。

②　ソフトウェアへの効果

ＩＳＰが高水準であるから，コンパイラ等のソフトウェアは簡単化される。

③　経済性

同一機種のハードウェアで多種のＩＳＰの実現ができれば，開発費の大きな減少が得られ好都合である。しかし，これによる効果の具体的数値は不明である。

④　柔軟性

ＩＳＰのレベルとマイクロプロセッサレベルのインタフェースが完全に分離

されているので，それぞれの変更に対する処置がファームウェアでのみ行なうことができ，将来の拡張，改良，修正が容易である。但し，これはＵＨＰ独特というよりマイクロプログラム制御プロセッサの一般的特徴といった方がよいであろう。

4.6.4　現状と将来性

ＵＨＰは，その基本的技術はかなり完成に近くなっているが，実用という点になるとまだその緒についたばかりであり，これが普及するにはなお若干の時間が必要である。ＵＨＰに対しては研究用としてはおもしろいものの，商用の効果は疑わしいとする意見もあるが，最近は，Burroughs, NCR等の大手メーカーでも採用を始めているのでおそらく，80年代後半から90年代初頭にはかなりの普及を見るものと考えられる。ここでそのために今後解決すべき問題点をあげると以下のようになる。

- ○　ＬＳＩ向きのＵＨＰアーキテクチャ
- ○　書きやすいマイクロプログラム用言語
- ○　マルチエミュレータを管理するＯＳ
- ○　ＩＳＰの動的適応化技術
- ○　ロジックレベルやレジスタトランスファレベルでの可変技術
- ○　ＰＭＳ（プロセッサーメモリースイッチ），レベルアーキテクチャ，即ち並列処理構造の可変技術

参考文献

1) 飯塚　肇：'マイクロプログラミングとミニコンピュータ'，情報処理，Vol.14，№6，p.429（1973）

2) A.B. Salisbury：'Microprogrammable computer architectures'，Elsevier（1976）

3) 飯塚　肇："ユニバーサルホストプロセッサ"，情報処理ハンドブック18編6章（1980）

4) L.L. Rakoczi：'The computer-within-a-computer，A fourth generation concept'，IEEE Computer group news，Vol.13，№1，p.22（1967）

5) S.H. Fuller et al：'The effects of emerging technology and emulation requirements on microprogramming'，IEEE Trans. Comput.，Vol. C－25，№10，p. 1000（1976）

4.7 高級言語マシン

4.7.1 概　説

現在の一般的ノイマン型コンピュータがハードウェアレベルで実行する命令（機械語命令）の機能とユーザがプログラムを作るときに使用するプログラム言語の機能との間には大きな差があり，これをSemantic gapなどとよんでいる。このギャップをうめるためにコンパイラがプログラム言語で書かれたプログラムを膨大な処理により機械語命令のプログラムに変換している。

この大きなギャップが存在するために，

- 理論的にはコンピュータ・システムで検出可能なプログラム誤りが検出できずソフトウェアの信頼性を低下させている。
- 単純な機能の命令を多数用いてプログラム言語の機能を実現しなければならないので性能が低下し，またプログラムの占めるメモリ量が増大する。
- コンパイラの機械語命令発生部が非常に複雑になる。
- 変換効率をよくするために機械語命令のもつ制限事項をプログラム言語に反映させることがあり，プログラム言語を歪めている。

など様々な弊害がもたらされており，これらの問題を解決することはソフトウェア開発の生産性を高める上でも重要な課題といわれている。

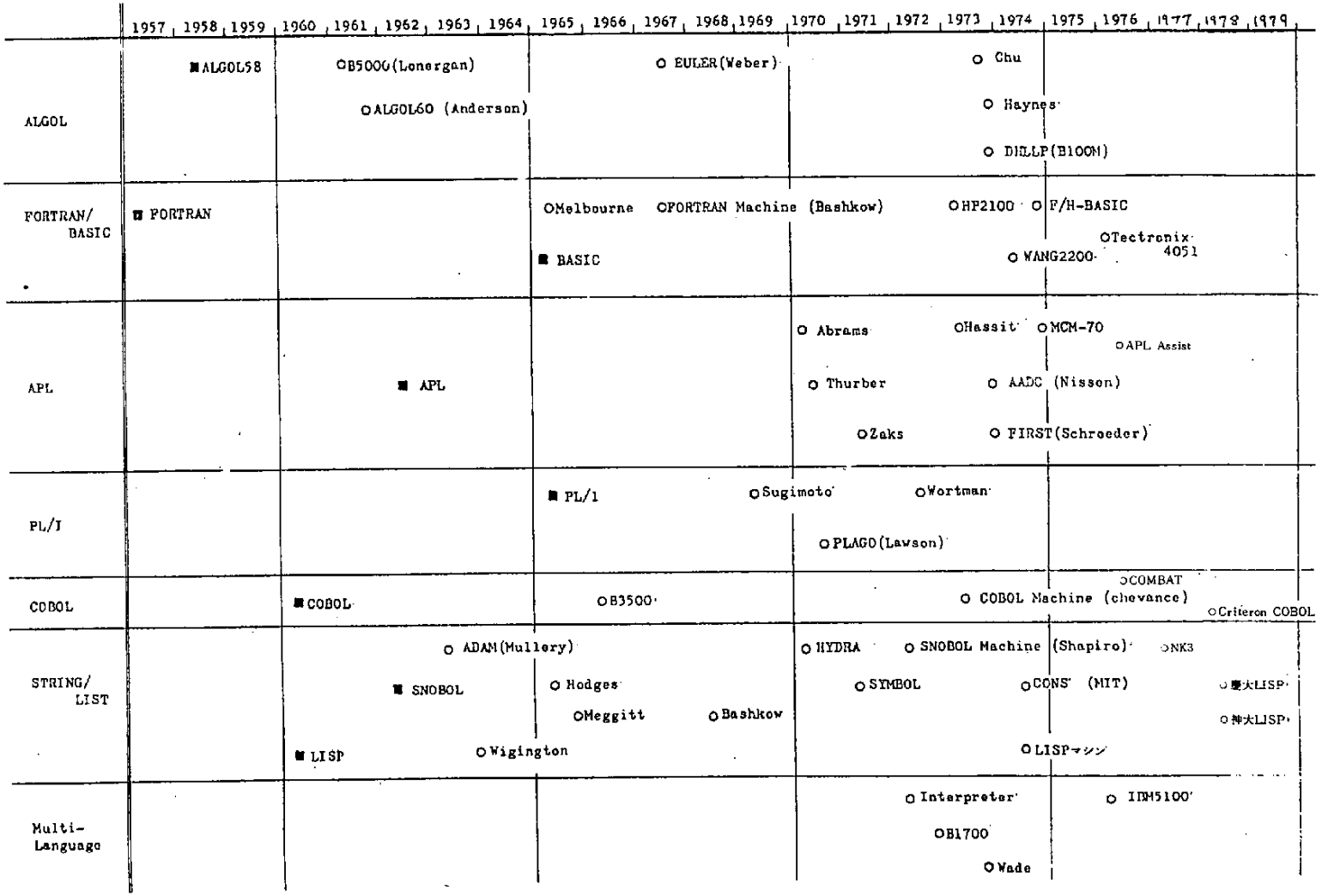
LSI/VLSI技術の進歩によって今後ハードウェアは高性能になりコストも低下すると予想されているが，単に従来のアーキテクチャのままでハードウェアを高速化，低コスト化しても前記の問題が解決される訳ではなくアーキテクチャの質点変化が必要である。

このため，機械語命令の機能を高めてプログラム言語に近づける，あるいはプログラム言語を直接実行する等，伝統的なノイマン型アーキテクチャを越えた新しいアーキテクチャが研究されており，表4.7.1，表4.7.2に示す如く，古くはB 5000(バロース社，1961年)などが有名である。このようは高級言語に指

表 4.7.1　高級言語マシンの発展経過

	1961 第1世代	1967 第2世代	1972 第3世代	198x(?) 第4世代
	ペーパーマシン	実験機段階	商用機の出現	本格的商用機の活躍
契機	B 5000／ALGOLマシン	EULERマシン	B 1700	(?)
特徴	• コンパイラでは効率の悪い言語（ALGOL,ストリング／リスト処理）を効果的に実現する方式の提案	• μ－P計算機上での部分的，特定機能の実験 • 専用ハードウェアマシンの提案と一部の試み • 専用言語マシン	• 商用機の出現 • 専用機，スタンドアロン • ミニコン，ポータブル計算機 • マルチプロセッサ構成 • Multi-Language • μプロセッサの使用	• 超LSIの駆使 • マルチ環境 • 中／大型機 • VHLL指向 • 本格的Multi-Language • 汎用エミュレータ • 機能分散，複合体システム
実例	B 5000 ALGOLマシン ADAM	EULER, HASSITT FLYNN SYMBOL	B 1700 WANG, HP 2100 IBM 5100 FIRST, DHLLP APL Assist Criterion COBOL	(?)

表 4.7.2　高級言語マシンの開発年表



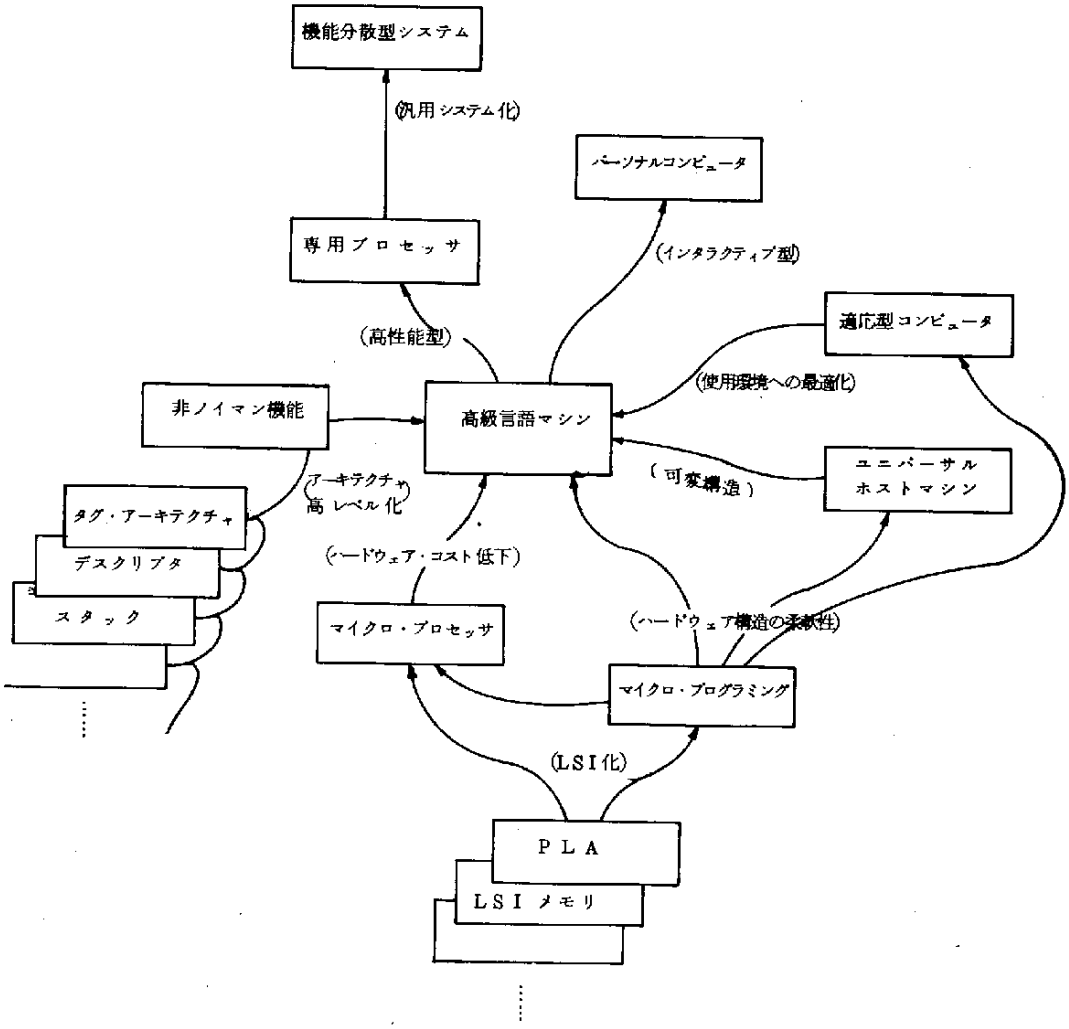


図 4.7.1　高級言語マシンと他の技術との関連

向したアーキテクチャをもったコンピュータを高級言語マシンとよんでいる。

高級言語マシンと他の技術との関連について，図 4.7.1 に示した。

4.7.2 技　　法

(1) 実現方法からみた分類

a. 言語処理方式

高級言語マシンにおける言語処理方式は次の3つに分類することができる。

• 中間言語方式

高級言語をコンパイラによって中間言語に変換しこれを機械語命令としてハードウェアが実行する。

機械語命令の機能として，ポリッシュ・ストリングの処理，タグ付データやデスクリプタによるデータの記述，ワンレベル・ストアなどが用いられる。従来のアーキテクチャに比較してコンパイラは簡単になり，実行時にも種々のチェックが出来るなど利点があるが，実行時の機械語命令当りの処理速度は遅くなる。

• ソースステートメント間接実行方式

上記と同じであるが，コンパイラがソフトウェアでなくハードウェアで作られている点が異なる。即ちハードウェアが一度中間言語のオブジェクト・コードを作り出し，それを実行するので，外から見るとソース・ステートメントをハードウェアが直接実行したように見える。

• 直接実行方式

ソース・ステートメントを直接インタプリートする方式でコンパイラは存在しない。実行速度は遅くなる。

b. ハードウェアの構成方法

• マイクロ・プログラムを主体とした方式

ハードウェア機能は出来るだけ単純にしてマイクロ・プログラムで複雑な機能を実現する。実験的に作られた多くの高級言語マシンやB1700(バロース社)がこの方式をとっており，複数言語に対する可変性を実現するのに有利などの利点がある反面，実行速度の点では不利である。

• ハードウェアによる方式

ハードウェア論理によって機能を実現するのでマイクロ・プログラム方式と相反する特徴があり，速度は有利だが可変性がなく，またハードウェアも非常に複雑になる。対象する言語が一つに定まっている場合，あるいはサポートする機能レベルが比較的低い場合（即ちノイマン型に近い機能しかない場合）でないと実現がむずかしいと思われる。例としては，SYMBOL（フェアチャイルド社，1971年）がある。

• マイクロ・プログラムとハードウェア機能の組合せ方式

上記の両方の良い点をとり入れることを狙い，ハードウェアもある程度機能レベルを高め，さらにマイクロ・プログラムによって複雑な機能を実現する。対象言語は一般に一つあるいは似たもの複数になる。例としては，COMBAT（日電，1976年）などがある。

以上の高級言語マシンをさらにシステムで見た場合には全体を1台のプロセッサで処理しているか，機能別の専用プロセッサの集合で構成しているかによって分類できる。

たとえば，SYMBOLでは全部で8種類のプロセッサがあり，その中の1つのプロセッサ（TRANSLATOR）がコンパイルし，セントラル・プロセッサがそれを実行する。セントラル・プロセッサはさらに4種のプロセッサ（Instruction Sequencer, Reference Processor, Arithmetic Processor, Format Processor）から構成されている。

c. 対象言語と狙い

処理する言語の種類でみると，1種しか処理できないもの，2～3種処理できるもの，および多種処理できるものがある。

1種しか処理しないマシンは専用機で，FORTRANマシン，ALGOLマシン，LISPマシンなどがあり，実行速度の点で有利である。しかし商用機としては2～3種の処理は必要で，B1700(COBOL／RPG, FORTRAN／BASIC)やIBM5100(APL, BASIC)にその例を見ることができる。

多種類の言語を扱う場合はハードウェア構成上も特別の配慮（たとえば，MLP 900 の Language Board）がなされる（4.6のユニバーサル・ホストマシン参照）。

COBOL, FORTRANなどを対象とする時は実行速度よりもデバッギングや移行性が重視され，APLやLISPでは実行速度の向上を狙ったものが多いようである。

d. マシン・アーキテクチャ

前述したSemantic gapを小さくするアーキテクチャを考える上で問題になることは，特定の言語に対するギャップを小さくすると他の言語とのギャップが大きくなることで特定言語の専用マシンではよいが，複数言語を用いる汎用コンピュータの場合には望ましくない。

これを解決するには2つの方向がある。1つは各言語に共通な範囲内でできる限りアーキテクチャのレベルを高めること，他は言語毎に最適化したアーキテクチャを設定し実行時に動的にアーキテクチャを切換えることである。

前者の方式をとる場合のアーキテクチャとしては次のようなものがある。

①多種類のオペランド形式の処理：ビットアドレスや，構造をもったデータの処理など。

②メモリ上での演算機能：現在の一般的コンピュータではレジスタ上の演算処理に重点がおかれているのに対し，3アドレス演算などメモリ上の演算機能を強化あるいはレジスタをなくしてしまうなど。

③ data driven 構造：タグやデスクリプタを用い，異なる方式のオペランド間の形式変換を自動化する。

④スタック構造：サブルーチン・リンケージのハードウェア・サポートなど。

⑤各種保護機能：セグメンテーションや capability based addressing など。

⑥ワンレベル・ストレージ

後者の方式を実現するには1つのマシンで複数のアーキテクチャを動的に切換える必要があり，マイクロ・プログラムの切換え／入替えで対応するのが一般的である。

(2) 高級言語マシンの例

a. B1700

B1700の特徴は種々あるがここでは特にSコードについてとりあげる。前述したようにマシン・アーキテクチャを特定言語に近づけようとすると他の言語とのギャップが広がり汎用性を失うので，B1700は言語毎にそれに最適化したアーキテクチャ（Sマシーン）を定義している。Sマシーンの実行する機械語はSコード（あるいはS言語）とよばれSDL−Sコード，COBOL Sコード，FORTRAN Sコードの3種がある。ハードウェアはマイクロプログラムを用いてSコードを解釈実行する。

図4.7.2に従来のコンピュータとSマシーンの各種言語や応用分野に対する適合性を示す。また表4.7.3に各プログラムに対する言語，オブジェクト・コードおよび実行時のインタプリタを示す。

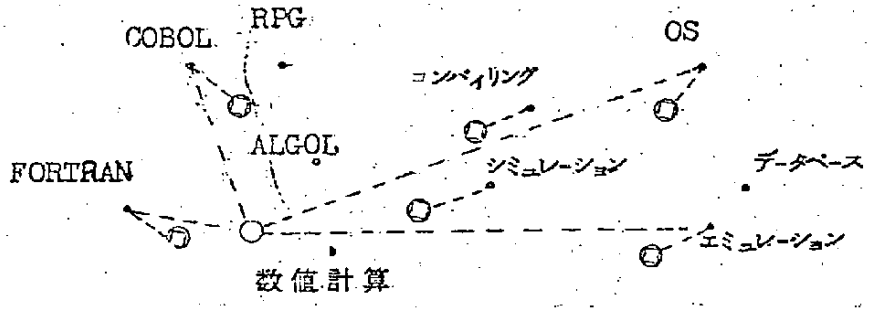


図4.7.2　従来のコンピュータ（○），Sマシーン（◎）と応用分野（・）間の適合性

表4.7.3　プログラム言語とS言語とインタプリタの関連

	ソフトウェア及びプログラム	使用した言語	オブジェクト・コード	実行時のインタプリタ
システム・ソフトウェア	マイクロ・コーデッド MCP	*MIL	マイクロ・インストラクション	な し
	MCP II	*SDL	SDL S−コード	SDL/INTERP
	COBOL & RPG コンパイラ	SDL	SDL S−コード	SDL/INTERP
	FORTRAN & BASIC コンパイラ	SDL	SDL S−コード	SDL/INTERP
ユーザー・プログラム	ユーザー・プログラム I	COBOL & RPG	COBOL S−コード	COBOL/INTERP
	ユーザー・プログラム II	FORTRAN & BASIC	FORTRAN S−コード	FORTRAN/INTERP

＊インタプリタはMIL言語を使用して作成される。

b. APL Assist Feature

APL Assist Featureは IBM 370/145 以上で利用可能なAPLのマイクロ・プログラムによるインタプリータである。APLシステムの主要なコンポーネントは以下の通りである。

- スーパーバイザ：端末との通信，割込み処理など。
- トランスレータ：APLステートメントを内部表現へ変換する。この変換はアセンブルと同様に一対一の変換で，結果はwork spaceに置かれる。ユーザが定義したnameのテーブルを保有し，内部表現から外部表現へ変換を容易にしている。
- インタプリタ：上記内部表現をインタプリティブに解釈，実行する。
- 補助プロセッサ：OS，ファイル，I/O，他のプロセッサなどと通信する。

APL Assist Featrueを用いた場合，ユーザがAPLのステートメントを入力すると，システムがこれを内部表現に変換してwork spaceにおき，APLEC(APL Emulator Call)命令を実行する。

この命令が実行されると制御はインタプリタのマイクロプログラムに移り，入力されたAPLのステートメントを実行する。実行が終了するとIBM 370のコンディション・コードなどをセットしてIBM 370のマイクロプログラムに制御を戻す。これによって前記APLEC命令の次のIBM 370命令が実行される。APLインタプリタに用いられたマイクロプログラムの量は約20Kバイトである。

本featureの効果については19本のテストプログラムを370/145上で走らせた結果，APL/360を370/145上で走らせる場合に比し2倍から20倍高速になったと報告されている。

c. CONS (MIT)

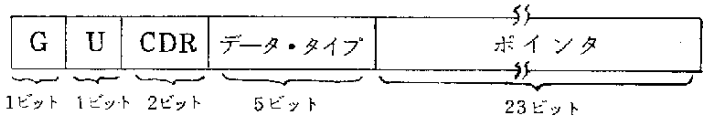
CONSはMITで開発されたLISPマシンである。LISP言語は人工知能分野などで重要な秀れた言語であるが実行時の処理効率の悪さ，メモリの使用効率の悪さなどから限られた使用にとどまっている。これについては言語

自身の改良と共にLISP言語向きのコンピュータを作る研究も行なわれている。ここではその例として1974年にMITから発表され1977年にプロトタイプが完成したLISPマシンのプロセッサ"CONS"の概要を述べる。

CONSプロセッサはマイクロプログラム制御のプロセッサで内部データ幅32ビット，マイクロ命令用に16kW×48ビットの制御記憶をもっている。CONSプロセッサはLISPプログラムの実行に適した機械語命令(図4.7.3)を解釈実行する。使用しているデータの形式は図4.7.4に示すようであり，5ビットのタグを用いて各種データ・タイプを表示している点に特徴がある。このプロトタイプはミニコン規模ながらLISPプログラムの実行においては従来の大型機(PDP-10)と同等あるいは数倍速いといわれている。

OP ：主オペレーションの指定
DES ：デスティネーション
REG ：レジスタ（インデックス）
DISP ：ディスプレイスメント

図4.7.3　CONSプロセッサの機械語



G ：ガーベージコレクション用ビット
U ：ユーザビット
CDR：CDRコード指定(NORMAL, NIL, CDR NEXT, ERROR)
データ・タイプ：TRAP, LIST, アレイ・ヘッダ, 記号アトム, 整数型数値アトム, INVOKE, μCOD-ENTRYなどがある。

図4.7.4　CONSプロセッサの基本データ形式

d. COMBATマシン

COMBATはCOBOL向きの高級言語マシンで，そのアーキテクチャの特徴は下記の通りである。

- 内部データ・タイプを豊富に持つ：数値データ（8種），文字データ（3種），指標データ（2種），編集制御データ（1種）。
- データ・デスクリプタの採用：これによって，データ・タイプ／コード変換，配列処理，Scaling，ゼロ／ブランクづめ，四捨五入処理，編集処理などをハードウェア・アーキテクチャ・レベルで実現した。
- 各種オペランド形式：指標／添字による配列データやCOBOLのデータ項目へのアクセスをサポート。
- 約50種の高レベル命令：1ソース・ステートメントは大部分1COMBAT命令に対応する。演算，移送ステートメント用に2,3及び多数オペランド形式もサポート。

COMBATマシンのハードウェア構成上の特徴としては，

- マイクロ・プロセッサ等を用いたマルチプロセッサ構成。
- 機能モジュールの採用，即ちプロセッサ間データ授受用のFIFOのメモリ，制御用命令の制御データ保存用LIFOメモリ，PLAによるマイクロプログラム制御回路等を使用。

などをあげることができよう。

(3) 課　題

高級言語マシンの技術上の問題点，課題として次のようなものがある。

a. 中間言語のレベル設定

対象言語，処理方式などによって最適な中間言語のレベル（機能レベル）があると考えられる。一般に中間言語レベルを高くするとコンパイラは作成も容易になりコンパイル時間も短くなるが，実行時間は長くなる（但し，どこまでハードウェア化するかによって実行時間は変わり得る）。

b. 複数言語の処理

高性能を実現するには言語の種類を絞り専用マシン化することが有利だが，商用機としては複数言語の処理が必要であり，どの方式がよいか（即ち前述のB1700型か，汎用アーキテクチャの高度化か）研究が必要である。

c．処理方式

直接実行か，コンパイル／アセンブル後実行か，言語，使い方によって最適な方式が異なる。たとえば同じ言語でもプログラム開発／デバッグ中と実業務処理のときでは処理方式は異なるべきであろう。この要求にうまく対応できるアーキテクチャ上の工夫が必要である。

d．ハードウェア構成法

LSI／VLSI技術の進展によって高性能のマイクロ・プロセッサ，PLA，各種高速メモリ等新しい素子が利用可能になるので，それらを活かしたマシンのハードウェア構造，マイクロ・プログラムとハードウェア論理とのトレードオフなどを研究する必要がある。

e．ユーザの使用状況に応じた処理の最適化

ステートメント，データ・タイプ等の静的および動的使用頻度はユーザ毎に，また同じユーザでも日が変われば異なる。したがってこの種の変動に対してマシン自身のアーキテクチャを変化させられることが望ましい。アーキテクチャのレベルが高くなるほどこの種の最適化の必要性も高くなると思われる。

4.7.3 応用と効果

(1) 効　果

高級言語マシンには次のような効果が期待される（直接的な効果）。

• 処理速度の向上：翻訳フェーズについては特にコード作成部が高速化される。実行フェーズの高速化については報告されている例を表4.7.4に示す。

• オブジェクトのメモリ・サイズの減少：ソース・ステートメントとほぼ1対1に対応する中間言語を用いることによって大幅に減少することが期待され

る。例を図 4. 7. 5 に示す。

表 4. 7. 4　高級言語マシンの性能（その 1）

(1)　APL　Assist Feature

STATEMENT	*RATIO* ＊注
X ← *IS*+*IS*	12.56
X ← *IV*100+*IV*100	3.07
X ← *IA*5_20+*IA*5_20	3.23
X ← *IV*1000+*IV*1000	2.30
X ← *RS*+*RS*	8.42
X ← *RV*100+*RV*100	1.76
X ← *RS*+*RS*	4.80
X ← *RV*100×*RV*100	1.05
X ← *RA*10_10⌹*RA*10_10	1.06
X ← *IV*200[*IS*]	14.54
X ← *IV*200[*IV*100]	3.38
X ← *IV*200[⍳100]	6.54
X ← ,⍉(10⍴2)⍴*IV*1024	2.67

＊注：APL Assist Featureを用いた時と用いないときの性能比

(2)　Hassitt の APL マシン

例 1.　APL ステートメント　　A←B+C

	APL マシン	IBM 360/25
Scalar	682	298
N= 5	2385	1976
=10	3985	3911
=20	7185	7781
=40	13585	15521
=n	785+320n	41+387n (μsec)

例 2.　APL ステートメント　　J←A[I]

APL マシン	IBM 360/25
599+73.8n	414+243.9n

表 4.7.4　高級言語マシンの性能（その 2）

(3)　B1700

	〔B1700〕 (Sコードの平均実行時間 35 μs)	〔IBM System 3−10〕 (平均命令実行時間 6 μs)
RPG Ⅱ プログラムの一例 ：	25 秒	208 秒

(4)　COMBAT マシン

同規模の汎用中型機の 3〜5 倍

(5)　LISP マシン

HP 2100 のマイクロプログラムとアセンブラを用いて同じ仕様のインタプリタを記述したときの比較

命　令	機械語のステップ数	マイクロコードのステップ数	速度比
CAR	2	4	3
CDR	3	5	4
GET	24	26	4
NUMBERP	30	28	4〜6
SLOAD	29	34	6〜7
PUSHFT	19	28	5
POPFT	16	22	4.4

注：ステップ数はインタプリタ内の対応する処理ルーチンの大きさを表わしている。

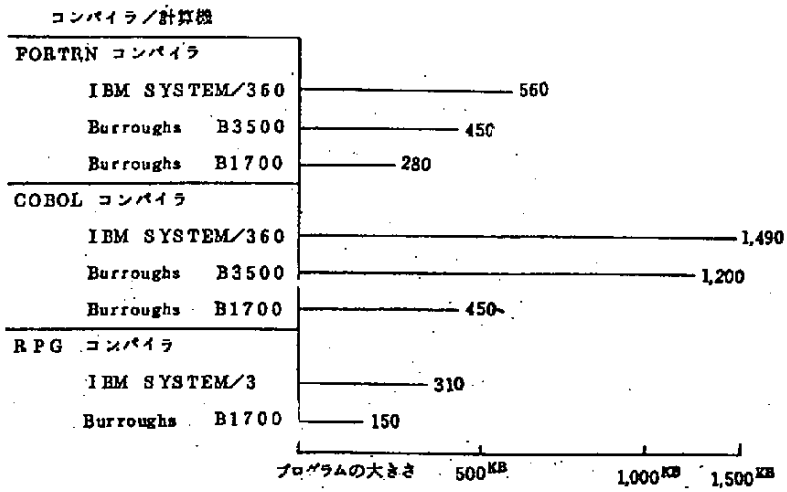


図 4.7.5　コンパイラのオブジェクト・コードのメモリ・サイズ比較（B1700 の S コード導入によるメモリ低減の例）

• コンパイラ／トランスレータのメモリ・サイズ：コード作成部が簡単になるなどの要因でコンパイラやトランスレータのメモリサイズ減少が期待される。

• デバッグ，診断機能の向上：ソーステートメントとほぼ1対1に対応する中間言語の場合ソースレベルのデバッグ，診断情報が得やすい。中間言語のインタプリタによる実行ではステートメント内の詳細情報を得ることが可能である。さらに，デスクプリタ等を用いるアーキテクチャではデバッグ指示制御情報によるトレースモニタリングが可能になるなど。

• 移行性の向上：ソースプログラムを即実行することによりオブジェクトコードが不要になり移行性の向上に役立つ。

(2) 応用分野と可能性

高級言語マシンの応用分野はその処理方式によって分けて考える必要があろう。

中間言語を用いる方式は汎用コンピュータの高級言語適応性を高める方法として有望と考えられ，特に各種言語の共通性を考えて機械語レベルを高める方法は小型から大型コンピュータまで適用可能であり，現在のコンピュータ・アーキテクチャからの移行性の点でも比較的実現しやすいと思われる。言語毎にマシン・アーキテクチャを変化させる方式はマイクロプログラム制御が深いハードウェア構造でないと実現がむずかしく中小型機にむいていると考えられる。しかし今後のハードウェア技術の進歩で将来はかなり高速のコンピュータもマイクロ・プログラム制御化されるであろうから将来性はある。

ソーステートメントを直接実行する方式ではインタラクティブなものとそうでないものが考えられ，前者はパーソナル・コンピュータやプログラムのデバッグ用のマシンなど処理の高速性が強く要求されない応用に向いている。後者は前述の中間言語方式に比較してハードウェア／マイクロプログラムが複雑になり，処理速度上もかならずしも有利とはいえない。複数言語に対する処理など課題も多く一つの理想型ではあるが実用化の点では最後になろう。

システム構成から見た場合は，中間言語方式は単一プロセッサのシステムへの適用が可能であるが，直接実行型では専用プロセッサとなるので機能分散型システム構成が前提になると思われる。

(3) 第５世代における利用形態

• パーソナル・コンピュータ：インタプリティブに高級言語を実行する。言語の種類や機能の高度化などが考えられる。

• 汎用プロセッサ（１台で各種のジョブを処理するコンピュータの場合）：高レベルの中間言語によって処理速度を改善したプロダクション・ランのモードとインタプリティブに処理するプログラム開発／デバッグ用のモードををもつプロセッサなどが考えられる。

• 専用マシン：各種言語に対する専用の直接実行型マシンを接続した機能分散型システムである。プログラム開発／デバッグ用のマシンとプロダクション・ラン用のマシンは別になる可能性もある。

• 各種の問題向き言語用マシン：汎用のプログラム言語だけでなく各種問題向き言語用の専用マシンも考えられる。

4.7.4 実現の可能性

最後に高級言語マシンを実現する上での現実的問題について考察する。

• 言語仕様のあいまいさ：プログラム言語の仕様にあいまいな点が多く，ハードウェア化する上で implementation 毎に解釈が異なる可能性がある。

（例）データの重なりの処理：AをBへ移行する際，AとBの領域に重なりがあるとハードウェアのメモリ幅や移送方式（work area を介するか否か）によって結果が異なる。

• 異常／例外の処理：正常な動作に対して効率のよい方式でも異常／例外の処理まで考慮するとアーキテクチャが複雑になりすぎる。

（例）プロセデュアのネスティングのためにスタックを用いるが，実行中に異常がおこると何処迄 recovery 処理すればよいかなどがわかりにく

い。このためスタックの内容に対してポインタを設けるなどを考える必要もあり，アーキテクチャが更に複雑になる。

• 互換性：現在の単純な機械語命令をもったコンピュータでも完全な互換性を保障するのは困難であり，高級言語マシンでは上述のような問題もあって，完全な互換性を保障するのは非常にむずかしい。

• 従来のコンピュータとの互換性：アーキテクチャを変える事に伴う一般的問題として，既存のソフトウェアをどう継承してシステムを移行させるかが大きな問題である。

• ノイマン型アーキテクチャのコンピュータとの競争力：現在使用されている言語はほとんどCOBOLとFORTRANであり，ノイマン型アーキテクチャでも比較的効率よく実行できる。これに打ち勝って高級言語マシンが普及するにはアーキテクチャやコスト／パフォーマンスの良さだけでなく他の要因，即ち，使用言語の多様化，ソフトウェア開発効率の改善への要求などコンピュータの使用される環境の変化も重要である。

• ミニデルファイ調査結果では，高級言語マシンの問題点として既存コンピュータとの互換性や従来のアーキテクチャに比較して飛躍的な性能改善は期待できない等が指摘されている。

以上の様に実現上種々の問題はあるが，すでにパーソナル・コンピュータでは実現されており，今後この基盤が拡大していくに伴って徐々に小型，中型の商用コンピュータへ影響が出てくると思われる。したがって80年代は高級言語マシンあるいは高級言語マシン的要素をとり入れたコンピュータの普及が徐々に拡大していき，90年代には本格的に普及するものと予想する。

参 考 文 献

1) G. J. Myers,"Advances in Computer Architecture," John Wiley & Sons, Inc., 1978

2) 島田俊夫, 坂村 健, 山口喜教, "高級言語マシン," 情報処理, Vol. 18 №4, April 1977, pp. 386～393

3) "LISPマシン開発の現状を探る," 日経エレクトロニクス, 1979. 3. 19 pp. 62～79

4.8 データベース・マシーン

4.8.1 概　説

(1) データベース・マシーンの背景

大量のオンライン・ユーザに対して，共有化したデータの並列的アクセスを許す集中統合したデータベース・システムは，今後ますます高速化，大容量化が要求される。一つの予測によれば，近い将来データベース・システムはピークロード時に1,000,000要求／秒の処理能力が要求されるようになるという。この値は現在の高パーフォーマンス・データベース・システムの処理能力である10～100要求／秒をはるかに越えるものである。また扱わねばならないデータの数も現在のそれ(現在でも10^{12}バイトを越す)よりはるかに大きいものになることが当然予測される。(一説では10^{18}バイトのストレージも，コスト・パーフォーマンスの面で実現性がある。) しかしながらこのような大規模，高速なデータベース・システムを，現在の計算機アーキテクチャにより実現することは，パーフォーマンスの面で非常に困難であるのみならず，信頼性の面でも大きな問題が生ずる。この解決の方向として現在注目を浴びているのが，データベース管理システム(DBMS)の基本機能を支援する専用ハードウェア，即ちデータベース・マシーンの開発である。

(2) ソフトウェア・データベース・システムの問題点

a) 複雑なデータ構造

現在のデータベース・システムの複雑さの最大の原因は，問合せで与えられるシンボリック名をメモリ番地へマップするために複雑なデータ構造（例えばB－木等）が必要であり，挿入，削除等の基本操作に対し，この複雑なデータ構造を正しく保持しなければならない点に起因する。

b) 機能の多様性

現在のデータベース・システムは多くの機能的に異なる部分（例えば，問合

せのパージング，ディレクトリのアクセス，ディレクトリの処理，データ検索，追加・削除，セキュリティの管理）から成っている。これらを同一のハードウェア上で実現すると，多くの部分にボトルネックが生じバランスの良いシステムを構成することが困難となる。特に現在の von Neumann 型計算機は，数値計算や単純なデータ処理を主目的として最適化されており，a)で述べた複雑なデータ構造の処理，二次メモリとの I/O 等の目的には必ずしも適したものとは言えない。

c) 大規模ソフトウェア

データベース管理システムは，今後ますます大規模・複雑化する事は明らかである。また多様な応用の発生，あるいは使用パターンの変化に伴う再構成の問題を避けて通ることはできない。このような状況下では，大規模ソフトウェアの設計技術の進歩と共に，DBMS の基本機能のハードウェア化によるソフトウェアの単純化が，信頼性の向上のために重要である。

(3) データベース・マシーンの基本概念

前述のデータベース・システムの問題点を解決するために，現在の von Neumann 型計算機と本質的に異なったシステム・アーキテクチャが必要となる。データベース・マシーンの基本構成図は図 4.8.1 に示されている。その基本概念を以下に述べる。

a) フロント・バックの構成

von Neumann 型計算機をフロント・エンドに，データベース・マシーンをバック・エンドに置く構成である。データベース・マシーンに DBMS のどの機能まで持たすかは，各システムにより異なる。

b) コンテント・アドレッシングと並行処理

シンボリック名から直接二次メモリへアクセス可能なコンテント・アドレッシングの機構が，データベース・マシーン・アーキテクチャの基本となる。この結果，当然(2)－a)で述べた複雑なデータ構造が必要でなくなる。コンテント・アドレッシングでは，探索単位に従い二次メモリを適当に分割することにより，

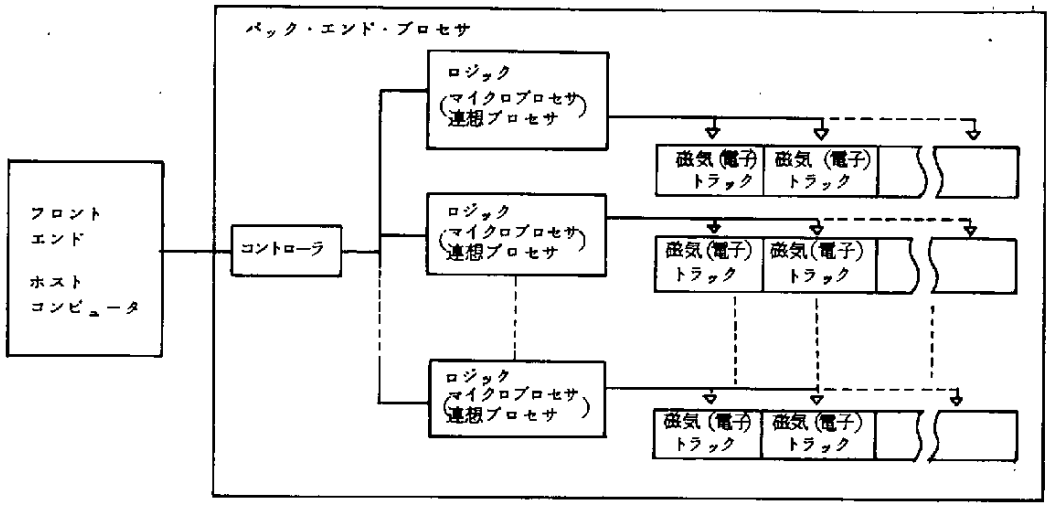


図 4.8.1　データベース・マシーンの基本構成

並行的な探索が可能となり，パーフォーマンスが飛躍的に向上し得る。

c)　DBMSの高次機能のプリミティブ化

DBMSの高次機能をデータベース・マシーンのプリミティブとすることは，パーフォーマンスの面と，信頼性の面で重要である。例えば関係データベースにおける，セレクション，プロジェクション，ジョイン等をプリミティブとしてハードウェア機能に持たす試みは多い。

4.8.2　技　　法

(1)　セルラ・ロジック・アプローチ

固定ヘッド・ディスクの読み・書き機構にロジック（演算装置）を附随させ，ロジック相互間に通信方式を与えることにより，比較的容量の大きい(10^8バイト程度)，しかしやや速度の遅いコンテント・アドレッシング装置が設計される。この方式は，トラックの数だけロジックが必要となるためコストが高くなり，大規模データベースの実現の目的には向かないが，中・小規模データベース・マシーンあるいは大規模データベースのディレクトリ・メモリとして有望であ

る。セルラ・ロジックに於ては，近年技術革新が目覚ましく，実用段階に入ったCCD，磁気バブルメモリが固定ヘッドディスクに代る傾向にある。

1） CASSM（Content Addressable Segment Sequential Memory）

CASSMは，セルラ・ロジックをデータベースに適用した最初の試みであり，スタンド・アローンのデータベース・マシーンとして，固定ヘッド・ディスクスクを用いて実現された。

特　徴

a） ネットワークモデル，階層モデル，関係モデルのいずれをもサポート可能である。

b） セグメント順次メモリ装置は，セルの集まりから成る。各セルは固定長セグメントを含み，ディスク・トラックとそれに附随するロジックから構成される。データはディスク・トラックに沿ってビット順次，語順次型に貯えられている。データ構成は可変長ブロック型であり，各ファイルを階層木として表現する。

c） 各ロジックはその前後のロジックと直接情報交換を行う。

d） 各ロジックは補助的メモリとしてRAMビットアレイを持っている。このRAMビットアレイは，探索中間結果のマーク，出力用のマークのために用いられる。

問題点

a） RAMビット・アレイの使用により，Boolean演算操作に対する能力を強めているが，各レコードあるいはフィールドの出現順とRAMビットアレイの番地を対応させているために，柔軟性を失っている。例えばこの結果として関係モデルに於るプロジェクション操作が殆ど不可能になっている。

b） 距離の遠いデータ間の通信が困難である。

2） RAP（Relational Associative Processor）

RAPプロジェクトはTronto大学で1975年に開始され，1976年にRAP.1と呼ばれるプロトタイプが作製された。その後大幅な設計変更が行われ，1977

年にRAP.2システムが作られた。

(a) RAP.1

特　徴

a) 関係データベースをサポートする。

b) セルはディスク1トラックと1024ビットのCCDバッファ及びそれに附随するロジックから成る。データはビット順次・語順次型にしまわれる。タプルに対しては固定長表現が用いられ，一つのトラックには一つの関係だけが含まれる。

c) ロジック間の結合方式は，一つの関係を表わしている多くのセルが連続的位置を取らなくてもよいので，ロジック間の全ての対を結合するネットワーク構造が用いられる。

問題点

a) ロジック間の結合方式が複雑となり，一つのロジックがその他の全てのロジックを駆動させるためのドライバーのコストが高くなり，通信線の長さが大きくなり，信頼性と並列処理能力で限界が生ずる等の欠点がある。

b) データベース全体に比し，セルの大きさが小さいため製作コストが大きい。

(b) RAP.2

上記a), b)の問題点の解決と，RAP.1で専用ハードウェアで実現したコントローラにより柔軟性を持たす目的で，RAP.2の作製が行われた。

RAP.1とRAP.2の相違点

a) RAP.2ではコントローラがマイクロコンピュータ（PDP 11/10）で実現された。

b) RAP.2ではセルメモリとして10^6ビットのCCDトラックを用いた。

c) RAP.2では，各ロジック間の結合を完全に無くし，各ロジックが独立に稼動する方法へ変った。各セル毎に1K語のRAMバッファをI/O用に用い，コントローラ（PDP 11/10）の高速ポーリング方式で探索結果を読み込み，同期もPDP 11/10でとることにしている。

3) CAFS

ICL (International Computers Limited)で開発用に作製された関係データベース用のデータベース・マシーンである。特に関係代数に於るセレクション，プロジェクション，ジョイン等の高速化を意図したシステムである。

特　徴

a) ロジック・パー・トラック方式のCASSM, RAPとは異なり，12ディスク・チャネルまでのマルチプレクス出力を並列的に処理する。ロジック・パー・トラック方式が並列度は高いが一走査当りの探索能力が低かったのに対し，ロジックの能力を強力にすることが主眼である。

b) CASSMと同様，補助的メモリとしてRAMビットアレイを用いているが，データの値によりRAMのアドレスを示しているので，CASSMよりはるかに柔軟性があり，プロジェクション操作等が簡単である。

問題点

a) プロジェクション用のインデックス，ジョイン用のカップリング・インデックスを事前に（プリコンパイル時に）用意しなければならない。

b) 各操作に対するホスト・コンピュータの負担が大きい。

c) 基本的探索中心で適用範囲が狭い。

4) その他

前述のセルラーロジック装置の他にも，幾つかの興味ある研究がなされている。

(a) RARES (Rotating Associative Relational Store)

関係問合せ言語の最適化の観点から，問合せを並列的プロセサに分配して実現させることが主要な点である。

(b) Intelligent Memory と Chang マシーン

Intelligent MemoryはCCD，バブルメモリを想定したスタンド・アローン・マシーンである。Changマシーンはバブルメモリのmajor/minorループを採用している。

上記(a),(b)いずれも設計だけで終っている。

(2) 連想プロセサ・アプローチ

コンテント・アドレッシングと並行処理の観点から，連想プロセサ上でのデータベース・システムの実現は，1970年頃から注目されていた。しかしながらモノリシックの連想アレイは，そのコストと容量に対する限界から，この上に全データベース・システムを構成することは悲観的な状態である。現在では大容量のディスクに対するステージァとして連想アレイを用いる方式が有望視されている。この場合探索に比し，連想アレイへの入出力の負荷が大きくなるため，入出力ボトルネックを解消するための研究が重要である。

1) STARAN型

ビット順次・語並行（ビット・スライス）型連想プロセサによりコンテント・アドレッシング，並行処理を行う。

(a) IFAM (Information System for Associative Memories)

IFAMは1970年にRorce Air Development CentreでGoodycar Aerospace Cooperationの連想メモリを用いて作製された。このシステムでは連想アレイ用のデータ構造ANF (Associative Normal Form)が用いられた。このシステムは全データベースが連想メモリに入ることが必要である。

(b) SETF (Staran Evaluation and Training Facility)

STARAN連想アレイプロセサを用いて開発された研究目的のデータベース・システムである。このシステムで特に注目されるのはPHTD (Parallel Head per Track Disk) と呼ばれるディスクで，各トラックの1セクタが並列的にSTARANにロードされ探索される。全部のデータベースが一枚のディスク上にあるとすると，約2回のディスクの回転（約80msec）で探索が完了する。

(c) APCS

Linde等による仮想的なバイト順次・語平行型の連想プロセサシステムである。このシステムは，二つの連想プロセサ・ユニットと順次プロセサ，平行I/Oチャネル，1.6×10^9バイト／秒で連想メモリとデータ転送を行うランダム・アク

セス・メモリ等から構成されている。Berraの試算によると，APCSとIBM 370/145で同じことを行う場合，APCSが探索速度で32～100倍速く，更新速度も15～210倍程度になる。

入出力の問題

連想アレイプロセサを用いたデータベースシステムの最大の問題点はI/Oの速度とコストである。現在のところ，アレイにデータをロードする時間は，アレイ中のデータの探索時間より大きい。この問題の解決のためには，高速・大容量メモリーとアレイを結ぶバンド幅の大きいインタフェイスを開発するか，あるいは通常のディスクの様な二次メモリから高速・小容量のステージャへのデータ転送に対する効果的なスキームを開発する必要がある。高速・大容量ストーレッジ方式としては，Farnsworth等がRome Air Development SystemのSTARANに適用するギガバイト・ストーレッジを報告している。これは，STARANと高速ファイルの間を3×10^9ビット/secでつないでいる。RADCのSTARANは4個のアレイモジュール(256×256×4)を持ち，マスメモリとSTARAN間に1024の平行ラインがある。アレイに完全にデータをロードするためには，256ビットが順次に転送される必要があるが，転送速度は300～450m sec/ビット・スライスであるので256ビットの順次転送にかかる時間は115μsec以下である。Berraの試算では，この様な構成の下では，128Mバイトのデータベースに対し，その4%のタプルを検索するのに必要な時間は約2秒となる。

2) DIRECT

DIRECTはWisconsin大学で開発中の関係データベースをサポートするスタンド・アローンのデータベース・マシーンである。

特　徴

a) 連想メモリとして8個のCCDメモリ・モジュールを用いている。データベースはマスメモリ中にストアされ，16Kバイト単位のページ・フレームに分割され，1ページは1つの関係しか含まない。ページ中の関係のタプルは固定長語で表現される。CCD中のデータの表現は，ビット順次・語平行型で

ある。

b) 問合せは各問合せプロセサに分配されて実行される。各問合せプロセサ毎に異なる処理を行っており，全体としてはMIMD型のアーキテクチャである。

c) 問合せプロセサ群とＣＣＤメモリ・モジュール群の間は，cross-pointスイッチで結合される。

問題点

この場合も最大の問題点は，マスメモリとＣＣＤメモリ・モジュール間の転送方式を如何に高速化できるかにある。

(3) ＤＢＣ（Data Base Computer）

大規模データベース・システムを考えた場合，データベースの蓄積装置として，コストの低い可動ヘッド・ディスクが，今後長期間利用され続けると思われる。そこで，セルラ・ロジックのセルのサイズを飛躍的に増大し，セル間の通信負荷を軽減するため，可動ヘッド・ディスクの１シリンダの全てのトラックを並列的に読み出す構成法の研究がある。この方式では，"tracks-in-parallel"読み出し中にコンテント・アドレッシングを行うことにより，可動ヘッド・ディスクのシリンダ単位のコンテント・アドレッシングが実現可能である。可動ヘッド・ディスクの場合，データベースの構造的情報（ディレクトリ情報）を利用することにより，シリンダに対する遅いアクセス（アームの動き）を最小化することが必要になる。この考え方は，データベースへのアクセス，ディレクトリ情報へのアクセス，ディレクトリ情報の処理等，ＤＢＭＳの各機能をバランス良く実現し，ボトルネックを排除しようというトータル・データベース・システム設計へつながってゆく。

ＤＢＣはOhio州立大学で研究の進んでいるデータベース・マシーンである。ＤＢＣはデータベース・マシーンとしてのトータル設計を目指した初めての試みであり，極めて野心的なものである。特徴を以下に列挙する。

a) マスメモリと情報メモリ

全体のデータベースをマスメモリと構造メモリ（ディレクトリ・メモリ）で実現

する。

b) PCAM (Partitioned Content Addressable Memory)

大容量ファイルをセグメントに分割することにより，速度は遅いが，ある程度のコンテント・アドレッサビリティが得られる。例えば，可動ヘッド・ディスクに於て，シリンダ単位でコンテント・アドレッシングを行うように設計すると，ギガバイト程度までのコンテント・アドレッシング装置が得られる。

c) 領域ポインタの使用

インデックスとコンテント・アドレッシングを併用する方法である。即ち，問合せに対するディレクトリの処理の結果，目的データを含むマスメモリのセグメントを示す領域ポインタが得られる。このセグメント内では，コンテント・アドレッシングによって目的データが検索される。

d) 機能単位の専用ハードウェア化

DBCの機能は次の様な機能単位に分割される。

PES：外界(ホスト・コンピュータ)

KXU：キーワードの内部表現への変換機能

SM ：ディレクトリの検索・更新・削除機能

SMIP：SMから得られる情報の演算を行う集合演算処理装置

IXU：SMIPの演算結果である論理インデックスを物理アドレスへ変換する。

DBCCP：全体のコントロール部で，ホストとのインタフェイス，コマンド・スケジューリング，セキュリティ情報の構成等を行う。

MM ：データベースのストーレッジ

SFP：DBCCPのセキュリティ情報により，セキュリティ・チェックを行う。

DBCは，システムの各種のボトルネックを避けるため，各モジュール毎に最適なハードウェア設計を目指している。その一例として，DBCCP，IXU，KXU，SFPは通常の計算機上での専用マイクロプログラムにより，MMは可動ヘッド・ディスクのシリンダ単位でのコンテント・アドレッシングにより，

SMは固定ヘッド・ディスク（あるいはCCD，バブルメモリ）を用いたセルラ・ロジックで，SMIPはRAMとマイクロプロセサで実現することが考えられる。

e）セキュリティ管理

データベースのクリエータがセキュリティに関する情報を指定することにより，データベースをセキュリティ・アトムに分割する。それに基づき各ユーザの特権アクセス・リストがDBCCP上に作られSFPでセキュリティ・チェックが行われる。

DBCの場合，各専用ハードウェアとして最適な方式が何であるかは，今後漸進的に決定されていくと思われる。例えば，Hsiao等は可動ヘッド・ディスクのコンテント・アドレッシング機構としてSCMD（Single Content addressability and Multiple Data Stream）を考えている。この方式ではロジックの能力が弱く一走査当りの探索能力が低く，複雑な探索には数多くのディスクの回転が必要になるが，遅い可動ヘッド・ディスクのアクセスを考えた場合，充分なパーフォーマンスが得られるか問題である。また，アームの動きを最小化するための，データの局所化等の問題等は，全体のパーフォーマンスに重大な影響を与えるため，単にハードウェア・デバイスの開発だけではなく，データベース・システム理論の発展が同時に望まれる。

4.8.3 実現へのアプローチ

a）データベース・マシーンの研究は今後盛んになるであろうし，将来商業用のものも現われるであろう。

b）現在では未だ多くのボトルネックが存在しており，また，各機能に応じたデバイスが得られているとは言えない。そのためには今後データベース・システムのアブストラクション，機能分析等の研究が進むことが，ハードウェア・デバイス技術の進歩と共に望まれる。

c）データベース・マシーンで用いられるデバイスの技術革新はめざましいので，現在提案されているデータベース・マシン自身の各機能が更に，新しい

ハードウェア・デバイスに置き換ってゆくことが予想される。その意味で，データベース・マシーンの発展は革命的であるというよりも，漸進的であろう。

d) データベース・マシーンは多くの専用プロセサの複合されたバックエンドプロセサとして実現されるものと思われる。

参考文献

1) G.F.Coulouris, J.M.Evans and R.W.Mitchell, "Towards content-addressing in data bases," The Computer Journal, Vol.15, №2, pp.95-98, 1972

2) L.D.Healy, G.J.Lipovski and K.L.Doty, "The Architecture of a context addressed segment-sequential storage," AFIPS Fall Joint Computer Conference, pp.691-701, 1972

3) G.P.Copeland, Jr, G.J.Lipovski and S.Y.W.Su, "The Architecture of CASSM : A cellular system for non-numeric processing, "Proc. The 1st Annual Symposium on Computer Architecture, pp.121-128, 1973

4) C.R.DeFiore and P.B.Berra, "A data management system utilizing an associative memory," Proc. NCC, pp. 181-185, 1973

5) R.Moulder, "An implementation of a data management system on an associative processor," Proc. NCC, pp. 171-176, 1973

6) R.R.Linde, R.Gates and T.F.Peng, "Associative processor applications to real-time data management," Proc. NCC, pp.187-195, 1973

7) C.R.DeFiore and P.B.Berra, "A Quantitative Analysis of the Utilization of Associative Memories in Data Management," IEEE Trans.on Computers, Vol.C-23, №2, pp.121-133, 1974

8) P.B.Berra, "Some problem in associative processor applications to data base management," Proc. NCC, pp. 1-5, 1974

9) R.H. Canaday, R.D. Harrison, E.L. Ivie, J.L. Ryder and L.A. Wehr, "A Back-end Computer for Data Base Management," CACM, Vol.17, №10, pp. 575-582, 1974

10) T.M. Marill and D. Stern, "The data computer- A network data utility," Proc. NCC, pp. 389-395, 1975

11) S.E. Madnik, "INFOPLEX-Hierarchical decomposition of a large information management system using a microprocessor complex," Proc. NCC, pp. 581-586, 1976

12) E.A. Ozkarahan, S.A. Schuster and K.C. Smith, "RAP-An associative processor for data base management," Proc. NCC, pp. 379-387, 1975

13) C.S. Lin, D.C.P. Smith and J.M. Smith, "The Design of a Rotating Associative Memory for Relational Database Applications," ACM Trans. on Database Systems, Vol. 1, №1, pp. 53-65, 1976

14) M. Edelberg and L.R. Schissler, "Intelligent memory," Proc. NCC, pp. 393-400, 1976

15) R.I. Baum and D.K. Hsiao, "Database Computers-A Step Towards Data Utilities," IEEE Trans. on Computers, Vol. C-25, №12, pp. 1254-1259, 1976

16) S.S. Yau and H.S. Fung, "Associative Processor Architecture- A Survey," Computing Surveys, Vol. 9, №1, pp. 3-27, 1977

17) D.K. Hsiao and S.E. Madnick, "Database Machine Architecture in the Context of Information Technology Evolution," Proc. 3 rd Very Large Data Bases, pp. 63-84, 1977

18) K. Kannan, "The Design of a Mass Memory for a Database Computer," Proc. 5th Annual Symposium on Computer Architecture, pp. 44-50, 1978

19) D.J. Dewitt, "DIRECT-A Multiprocessor Organization for Supporting Relational Data Base Management Systems," Proc. 5 th Annual Symposium on Computer Architecture, pp. 182-189, 1978

20) H. Chang, "On Bubble Memories and Relational Data Base," Proc. 4 th Very Large Data Bases, pp. 207-229, 1978

21) J.Banerjee and D.K.Hsiao, "Performance Study of a Database Machine in Supporting Relational Databases," Proc. 4 th Very Large Data Bases, pp.319－329, 1978

22) J.Banerjee, D.K.Hsiao and R.I.Baum, "Concepts and Capabilities of a Database Computer," ACM Trans.on Database Systems, Vol.3, №4, pp.347－384, 1978

23) E.Babb, "Implementing a Relational Database by Means of Specialized Hardware," ACM Trans.on Database Systems, Vol.4, №1, pp.1－29 1979

24) S.Y.W.Su, "Cellular-Logic Devices: Concepts and Applications," IEEE Computer, Vol.12, №3, pp.11－25, 1979

25) D.C.P.Smith and J.M.Smith, "Relational Data Bata Base Machines," ibid, pp.28－38

26) P.B.Berra and E.Oliver, "The Role of Associative Array Processors in Data Base Machine," ibid, pp.53－61

27) D.S.Kerr, "Data Base Machines with Large Content-Addressable Blocks and Structual Information Processors," ibid, pp.64－79

28) G.A.Champine, "Current Trends in Data Base Systems," IEEE Computer, Vol.12, №5, pp.27－41, 1979

4.9 コンピュータ・ネットワーク

4.9.1 概　説

複数のコンピュータを結合したシステムは一般に，主記憶を複数のプロセッサで共有するような形での密結合マルチプロセッサシステムから，通信回線等によって結合したシステム迄様々なものがあるが，ここでは後者のようなものをコンピュータネットワークとして取扱う。従って，各コンピュータは独立した制御プログラムを持ち，自立して動くことができるものとする。この代表的な形態には次のようなものがある。

① 中央コンピュータと複数のサブコンピュータよりなるトリー状のシステム

② 複数のコンピュータを一本の同軸ケーブルやリングバスによって接続したローカルシステム

③ 各地のコンピュータをデータ通信網によって対等に接続したシステム

①は企業内で組織間結合のシステムとしてよく用いられ，②はパーソナルコンピュータやプロセス・コントローラ等の結合システム等として用いられ，そして③はリソース共有を目的とした広域システムに多い。

コンピュータネットワークの目的はシステムによって異なるが，一般に次のようなことがあげられる。

① 分散処理の利点追求

② リソース共有

③ 高級な情報システム

①としては，各企業組織の構成に合わせて分散システムを構築することにより，各部分組織のニーズに合わせたシステムが出来ること，ローカルな処理を情報発生点近くでおこなうことによって，集中システムに比して回線コストが減少すること，部分システムの障害が全体に及び難いことにより危険分離が可能となること，システムの拡張性が高いこと等があげられる。②は，データベ

ースを共用したり，特殊なデバイスを共用することによって経済化をはかったり，あるシステムが混んでいたり障害中の時は他を用いること等を意味する。③としては，電子郵便が使えること，情報検索など様々なオンライン情報サービスの基礎になること，システム間の迅速且つ高級な情報授受が可能になること等が考えられる。

システムの発展というのは，まず単純なシステムがそれぞれの目的に合わせて個別に作られることより出発し，次にそれらが互に有機的に結合してゆく段階をむかえるという形が自然であるから，個別のコンピュータシステムからコンピュータネットワークが構成されてゆくのは自然であるとも言えよう。

コンピュータネットワークが本格的に研究され始めたのは1969年のARPA-NET 以来であるが，その基礎技術としてのTSSは1960年中頃に発達し，高級なデータ交換方式であるパケット交換がほぼ同時期のミニコンピュータ出現によって可能となり，道具だては揃っていたと言える。その後，各所で研究が行なわれ幾つかの実験網が作られた。一方，コンピュータを多くの端末から使うという形態はその頃既に流布し始めていたが，その通信系を整理・統合する必要から1974年IBM社よりSNA(Systems Network Architecture)が発表された。SNAはその後，複数のホストを含むように拡張され，分散処理機能を拡充する為の分散処理用プロセッサが次々と発表されている。一方，コンピュータネットワークが各所に作られるとそれらの相互乗入れが問題となり，公衆データ交換網の発達とともに国際標準の設定が必要になって，CCITT SGⅦ，ISO TC97 SC6／SC16等において活発に検討が進められている。

4.9.2 技　　法

(1) 基本的な技術

コンピュータネットワークはデータ通信技術と計算機利用技術の総合技術であり，それには様々なものが含まれるが基本的なものをあげれば次のようになろう。

① データ通信技術：ディジタル伝送，伝送制御手順，データ交換，網構成，新データ網，網間接続等の技術

② ネットワークアーキテクチャ技術：階層構成，諸プロトコル設定，網仮想端末，網仮想ファイル，プロトコル記述言語，プロトコルの検証等の技術；

③ 分散処理用に仕事，データ等を分割する手法：通信量を減らす。

④ 分散リソースの管理技術：分散ジョブ管理，分割リソースの割当て，デッドロック防止／検出・回復等の技術

⑤ 網内リソースのアクセス技術：TIPのような端末収容装置技術，Network Access Machine，REXのようなユーザアクセス援助システム，網内リソース情報等の技術

②にはユーザプログラムから網を使う時のインタフェース（cf. VTAM）設定手法が含まれる。③は企業組織に合わせてシステムを作るような場合に重要で，仕事，データを区分して中央に集中するべきものと各所に分散設置するべきものを見極める手法である。⑤はリソース共有網で重要な事柄で，ユーザが容易に網にアクセスできること，どこにどのようなリソースがあるかをユーザに知らせてそれを容易に利用できるようにすること等である。

(2) システム例

コンピュータネットワーク構成に関係深いシステム要素として，伝送媒体，交換方式，公衆データ網，分散処理プロセッサ，ネットワークアーキテクチャを選び表にしたのが表4.9.1である。表中，年代はその発表時期を示す。

また，様々なコンピュータネットワークをその用途別に分類したのが表4.9.2である。年代は特に説明がない限りそれが稼動し始めた時期を示している。

表4.9.1　システム要素の例

システム要素	例　（国名，発表年等）
伝送媒体	ペアケーブル，同軸ケーブル，光ケーブル，通信衛星，Radio（UHF）
交換方式	パケット交換，回線（時分割）交換，Aloha方式，Ethernet，SATNET
公衆データ網	Telenet（米，1974），Datapac（カナダ，1978），Transpac（仏，1978），DDX（日，1979），PSS（英，1979）
分散処理	IBM 8100（1978），IBM System/38（1978），DP/6（東芝），N4700（NEC），V77DIP（Univac），DS-7（パナファコム，1979）
ネットワークアーキテクチャ	SNA（IBM，1974），DNA（DEC，1975），国産各メーカ（1976，77），DCNA（電電公社，1977）

表4.9.2　諸コンピュータ・ネットワークの例

種別	システム例
企業内網	チェーンストアにおけるオーダ・エントリ・システム，総合生産管理（製鉄業），ACT—Ⅱ（旭化成，1976）
計装制御	CENTUM（横河，1975）
企業グループ間網	全銀ネット（1973），CASHシステム（1975），SITA，SWIFT
研究用ローカル・ネットワーク	SPIDER（ベル研，1972），DCS（UCI，1971），Ethernet（Xerox，1973），EPICS（電総研，1974）
リソース共有網（広域）	ARPANET（米，1969），CYCLADES（仏，1974），JIPNET（情開協会，1974），EIN（欧州，1976），N1（国立7大学，1977），Euronet（欧州，1978）
国際データ網	IPSS（英，1978），VENUS（KDD，1979），UDTS（ITT）
分散データベース実験	SDD—1（CCA，1977プロジェクト開始），Polyphem（仏，1976プロジェクト開始），CONIT（米，1974プロジェクト開始），SIRIUS（仏，1975プロジェクト開始）

(3) 課　　題

基本的な技術はかなり固まって来たが，より高度な利用を目指したり，融通性のある利用を試みる時には，未だ未だ多くの問題が残されている。コンピュータネットワークの持つ可能性という面から見れば，現在は未だほんの入口であると見ることもできよう。

データ通信技術に関しては，光ケーブル，衛星等を用いてより高速・安価な回線を作ること，通信制御プロセッサに機密保護機能を組込みデータ伝送の機密を護ること，各国のデータ網を相互結合すること等がある。高速・高品質・安価なデータ通信網の開発は広域にわたるコンピュータネットワークに不可欠の要素であり，従来の電話網にとらわれないデータ通信のニーズに合ったデータ網サービスが望まれる。

ネットワークアーキテクチャに関しては，まず上位層迄に至るその国際標準を定めることがあり，中でも網管理機構の組込み，網間接続の考慮，諸応用に向いた様々なプロトコルを定めることがある。我国に於いては，多くのコンピュータメーカがそれぞれ独自のネットワークアーキテクチャを発表し，電電公社もメーカ4社と協同してDCNAを発表している。これらの間の非両立性は，網が発達してゆくと問題になる為国際標準の早期勧告が望まれる。また，プロトコルの記述言語，実際的規模で使える検証手法，ユーザのプログラミング言語インタフェース等に関しても未解決である。特に，言語については全くといって良い程で，幾つか研究はあるが，実用的な分散処理記述用言語は作られていない。プロセス間通信に基づいたプログラミング言語が望まれる。

分散リソース管理については，網規模が大きくなり，複数の網リソースを単一ジョブが使用する環境下では深刻な問題である。利用形態の制限によるデッドロックの回避，能率の良いデッドロック検出方式等が望まれる。これが特に問題になるのは分散データベースである。一般に複数ケ所のデータベースを同時使用する時にこの問題が出てくる他，信頼性を確保する為に複数の

コピーを置くとその内容の一貫性問題が派生する。後者に関してはタイムスタンプや論理時計を用いる方式が考えられているが，様々な障害対策とともにより実用的な手法が望まれる。また，諸データモデル間の変換，複数データベース管理システムの統合に関しては今後の課題である。

一方，企業や研究所等の組織に合わせた分散処理システムをより広く普及させる為には，広がった多くの分散処理プロセッサに対する応用プログラムの開発道具を充実し，非常に簡単なユーザ用言語（コマンド）のサポート，コンピュータの自動運転など，コンピュータ要員を特に必要としない工夫が必要である。

更に，ファクシミリ網の発達，CAPTAINシステム等のオンライン情報サービスの普及，公衆データ網の発達，従来の電話網のディジタル化等は現在，個々独立に進行中であるが，これらを統合して情報ユーティリティを実現してゆき，国際間の諸サービス統合をおこなってゆくのも今後の課題であろう。

4.9.3 応用とその効果

(1) 応用範囲と可能性

現在，流通業，製造業，金融業等で用いられているシステムは，本店に中央システムを置き，各事業所・支店等にサブシステムを置いて各支店でのサマリーを本店へ送る程度の応用であるが，次には中央システムをデータベース化し各支店とより論理的に結合してデータベースにアクセスをする形態が考えられる。更に進めば，総合情報システムとして発展し，網形のシステムが形成されリソース共有が進展することが考えられる。又，全国的規模の企業グループ網が形成され，グループとしての収益性を求めるようになろう。

住民登録，戸籍等を扱う地方自治体の公共システムがオンライン化され，迅速且つ正確で省力化をはかることが出来よう。但し，個人データに関してはプライバシーの問題から，規制が明確になる迄遅れることも考えられる。

公害監視システム，道路交通監視制御システム，医療情報システムなど，公共システムに関しては地域に密着した関係上，ネットワークに適したシステムである。

高度な情報システムとしての発展では，電子郵便，図形，音声などの実時間信号が自由に人々の間で授受可能となり，複数の情報検索システムの統合，各所にある様々な情報を自由にアクセスして，情報処理システムで任意の形に加工すること等が考えられる。

又，各所で開発されたソフトウェアは完全なインタフェースの記述とともにシステム内に登録され，他所からの利用要求に対して「関数型」で応答したり，プロセス間通信で他のプログラムと結合して共同作業をすることが考えられる。

データの共用に関しては，個々のデータベースの量的・質的発達とともに，複数のデータベースを統合して利用者には仮想的な巨大データベースとして見せるシステムが発展してゆくものと思われる。信頼性，応答性等の面からデータベースはコピーを幾つか持つようになるが，それはシステム内部の構造であってユーザからは見えない。データの内容の無矛盾性や機密保護はかなり細かく護られよう。

超高速科学計算マシン，大規模偏微分方程式解析マシン，大容量高速メモリなど非常に専用化されたシステムは今後ますます必要になるものと思われるが，これらはネットワークを通しての共同利用によりコスト／パーフォーマンス良く利用できよう。

(2) 第5世代での利用形態

まず研究用網について考えると，パーソナルコンピュータが発達し，それを結合したローカル網が各研究機関で作られる。高速のデータ回線で相互に結んだリソース共有網は基幹システムとして実用になっており，第5世代ではローカル網と基幹網の結合がおこなわれているであろう。利用者は通常，自分のパーソナルコンピュータを使うことにより用が足りるが，同研究所内

のプリンタ，ファイル等を必要とする時はそのローカル網内でアクセスをする。更に，他所のデータなりプログラムなり，又超高速計算なりを必要とする時は，ローカル網のゲートウェイを通して基幹網にアクセスをし必要なリソースを獲得する。基幹網は，他の国の基幹網とも接続されていよう。ローカル網のデータ転送速度は数10 Mbit／sec 程度のものが使われ，ファイル転送も余り時間を要せずに可能であろう。基幹網に関しては衛星，光ケーブルの利用により速度は数 Mbit／sec 程度が使われよう。

CAPTAIN等の情報システム，ファクシミリ伝送網は発達してかなりの量になっており，ディジタル伝送技術の全面導入によってデータ網との統合が開始されようとしているが，未だ融合は進んでいないであろう。しかし，通信制御プロセッサの発達，回線速度の向上によって，パケット交換での伝達遅延は回線交換と同程度となり，音声等の実時間信号を高品質で伝送できる能力を持ってくるであろう。

データベースに関しては，その利用が広く流布しているであろうが，複数のデータベース管理システムの統合化は同種システム間について幾つか行なわれている程度で，異種モデルシステム間の統合化は結合が浅いレベルに止まり，深いレベル迄の結合は実験段階であろう。

4.9.4 現実的可能性

(1) 諸機能の実現時期

企業組織に合わせた分散処理システムは，現在実用化が始まっており，ここ数年の内にかなり普及するであろう。集中ファイルのデータベース化は余り急速には進展しないであろうが，着実に少しずつ浸透してゆくものと思われる。データベースシステム相互の統合化については，80年代は実験・研究がおこなわれ，比較的容易な形の結合が実用されるのは80年代の前半にも始まりそうであるが，仮想化の進んだシステムについては90年代に入ってからのことであろう。

研究用システムに関して，基幹コンピュータネットワークの建設は80年代前半に始まり，1990年にはかなりの規模に達していると思われる。パーソナルコンピュータについてはVLSIの普及とともに増えて，80年代後半には我国に於いてもかなりの数となるものと思われる。ローカル網の本格的な普及はそれと歩調を合わせて進むであろう。

公共システム各種のオンライン化は急には進まないであろうが，避けられない方向であろう。

ファクシミリはSTOC網の発達とともに広くビジネスに普及し，90年にはよく用いられるであろう。CAPTAINについてはコストがらみであるが，80年代前半に商用化が始まり90年にはかなり利用されているものと思われる。データとともにこれらが統合化されるのは，90年代半ばになるであろう。

ISO等で，ネットワークアーキテクチャの国際標準が決まるのはここ2～3年のことであり，改善，拡張が為されて80年代後半には普及をみているものと思われる。

(2) 解決すべき問題

前に述べた課題をまとめると次のようになる。

- 高速・安価なデータ回線
- ネットワークに機密保護機構の導入
- 能率の良いネットワークアーキテクチャ標準
- プロトコルの記述・作成技術
- 分散処理用プログラミング言語
- 仕事・データの分散化手法
- 分散リソース管理技術
- 分散データベース管理技術
- 分散システムの自動運転
- 諸情報システムの統合化
- コンピュータの利用容易化

参考文献

1) 分散処理特集号，情報処理，Vol．20，№4，1979

2) 小特集「コンピュータネットワーク」，情報処理，Vol.16，№7，1975

4.10 適応計算機

4.10.1 概　説

(1) 適応計算機の必要性

電子計算機の発展はその誕生以来驚異的であり，応用分野は，当初考えられた特殊科学技術分野に留まらず，産業コントロール，産業オートメーション，企業管理，事務機械化などの非常に多くの分野に広がっている。このように計算機の応用分野が広がった理由はいくつか考えられるが，アーキテクチャ側から見ると，それが第三世代の計算機と言われるＩＢＭ Ｓ／360のアーキテクチャに代表される，いわゆる統一された汎用計算機の概念に集約されると考えるのは妥当であり，それをきっかけとして，応用分野が爆発的に拡大したと思える。

ここで，汎用化の概念をまとめるならば，シングル・プロダクト・ライン・コンセプト（命令セットレベルで完全な互換性をもつが，処理速度，記憶容量などの性能の異なるモデル構成からなる製品群）に基づくことであり，その結果，単一のソフトウェアで小規模から大規模までのシステム構成を可能とし，ソフトウェア開発が最少限に押えられ，しかも問題の規模が拡大しても，大型機への移行は比較的容易となった。ハードウェアの開発，製造，保守などのコストが下がり，大量生産を可能とした。そしてこのような大量生産が応用分野の拡大をさらに助長した。しかし，その反面，応用分野の拡大は，当初考えられていなかった従来のそれとは全く異なる新しい分野を生み出し，またすでに適用されていた分野でも，使用経験が更に高度な使用法の要求につながり，それが逆に，汎用機の概念では効率よく処理できないという問題，また汎用機ではシステムの開発が困難になるという新しい問題を提起することとなった。すなわち，汎用機の概念は確かに，与えられた応用問題を広い範囲に渡り“平均的に”効率よく処理することができる。しかし，

応用問題によっては，従来の汎用機のような平均的適応度を得るのではなく，特定の問題，あるいは問題群に対して高い適応性，つまり高い処理効率を上げたい場合がある。

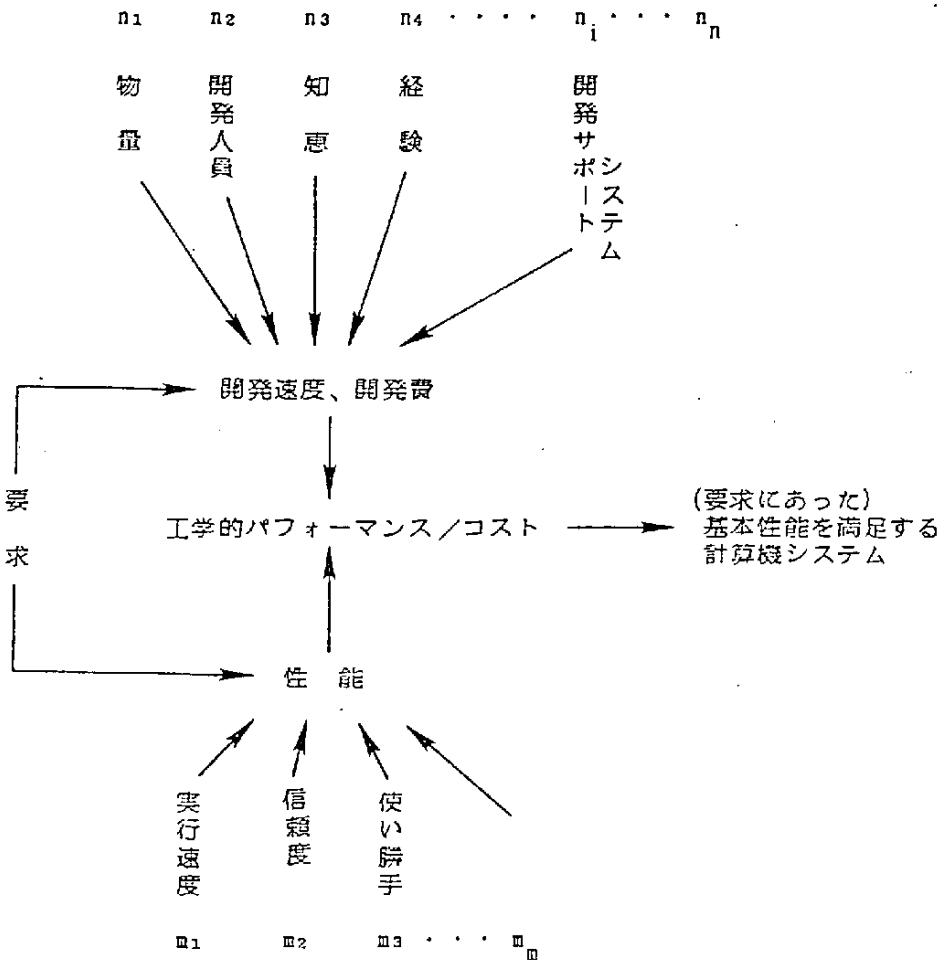
このような問題に対する一つの解決法は，その汎用化の概念を変えずに，命令セットレベルアーキテクチャの変更は行なわず，構成素子の改良を行うことであった。確かに，構成素子の進歩には著しいものがあり，真空管，トランジスタ，集積回路技術と移るにつれ，着実に効率の改善は行なわれている。しかし，そのような方法には自ら限界があり，今やこのような力づくな方法だけでは解決できない所まで来てしまったように思える。

このように汎用化という概念は，計算機の発展にとって重要であったのであるが，現在の社会的要請に基づき，徐々にそれが障害になりつつあるという技術的背景のもとで，従来の汎用機の概念では得ることのできない，アーキテクチャ的により応用問題に適した計算機——適応計算機——の必要性が生じた。

(2) 計算機における「適応」とは

計算機における「適応」を正確に定義することは難しいが，一般的には，「適応」とはある基準をどれ程満しているかということを表わす指標と考えられる。図 4. 10. 1 に示すのは，計算機における適応の因子であるが，計算機システムを開発に関する因子に分け，それぞれを細分化して考え，その因子の中で要求水準を満足するか否かで適応しているか否かを判断することは，割と容易であろう。そこでこのように適応を細分化し一つ一つを独立に考えることを，「狭義の適応性」の考察ということとする。

計算機の誕生以来の研究を振りかえると，その研究のほとんどは，狭義の適応性を満足する方向で進められてきた。そして，適応性の因子を細分化して独立に考察すると，確かに着実に進歩しているのである。しかし，計算機をより高度な問題解決のために使おうとすると，これらの因子を独立に考えることでは不十分になってくる。すなわち，各因子はそれぞれ相互関係を持



図でn、m一つ一つを独立に考えることは狭義の適応性の考察であり、
それらの関係まで考える時は広義の適応性を考察していることになる。

図 4. 10.－1　計算機における「狭義の適応性」と「広義の適応性」

ち，お互いに影響を及ぼし合う。したがって，これら因子が有機的に結合されたシステムという形で，「適応性」を考える必要が生じてくる。このように細分化された因子の集合，一つの計算機システムというまとまった形で適応を考えることを「広義の適応性」の考察という。適応性の尺度を正確に定義するのは難しいが，因子の適応のバランスが平衡している状態，すなわち工学的価格性能比（パフォーマンス／コスト）が優れていることとしておく

と，幅広い問題に対して広義の適応性を満足する計算機システムの実現が，現在かなり困難になってきていることに気づく。その理由はいくつか考えられるが，一つには 4. 10. 1. (1) でも指摘したように，応用分野の拡大により，計算機誕生時には考えられてもいなかったようなもの，例えば，人工知能，パターン認識の問題などを解決する要求が高まってきたのであるが，現在の固定した概念に基づく計算機構造では対処しきれなくなってきているからであり，またもう一つの重要な理由として，計算機に対する要求の著しい上昇にあると考えられる。人間の果てしない欲望，願望が計算機に対する要求を上昇させるわけであり，計算機システムが人間生活と密着したものである以上，これは仕方のないことなのであろうが，要求の上昇は適応のための基準変更を意味し，一昔前なら適応していたシステムを時代とともに現在では適応しないシステムに変えてしまっている。

このような理由からも，これからの計算機システムには，拡大する応用問題ならびに要求が変化する問題に対してアーキテクチャが柔軟に対処できるような従来のシステム，特に現在の商用計算機には見られない柔構造アーキテクチャ，問題，要求に応じて自由にその構造が変えられるアーキテクチャ，可変構造性があるアーキテクチャを持った「適応計算機」が必要とされるのである。

(3) 構築のための課題と位置づけ

適応計算機を構築する場合，明確にされなければならない事は，

① 解くべき問題に，「どのように」計算機の構造を適応させるかの適応化の方法理論

② 具体的に構造を可変にする（構造を適応させる）方法

の二点である。

ここで，①は解くべき問題と計算機間に存在するセマンティック・ギャップ(注) をいかに小さくするかを明確にする事を目的としており，具体的には，適応の目的・ゴールの設定，目的設定値達成のための制御の方法からなる

（適応化理論）。また②はアーキテクチャに柔軟性をどのように持たせるかを明確にすることを目的としており，要求されるアーキテクチャレベルで容易にシステムが可変に出来なければならない（可変適応技術）。

なお，他分野では，ユニバーサルホストマシン，ダイナミックマイクロプログラミング，モニタリング技術，高級言語マシン，高信頼性技術などと関係が深い。

(注) セマンティック・ギャップ

セマンティック・ギャップとは，Gagliardi によって初めて使われた。（U. D. Gagliardi, "Report of Workshop4 — Software — Related Advances in Computer Hardware," Proccedings of a Symposium on the High Cost of Software, Menlo Park, Calif.: Stanford Research Irstitute, 1973）
当初の定義では，「高級言語とISPアーキテクチャの相異」ということであった。セマンティック・ギャップの存在により，ソフトウェアの不信頼性，パフォーマンスの悪さ，プログラム・サイズの巨大化，コンパイラの複雑化などの障害が現われる。最近ではセマンティック・ギャップの意味は拡大され，「解くべき応用問題（のアルゴリズム）とアーキテクチャの相違」と解釈されることが多い。

4.10.2 可変適応技術と適応化理論

(1) 可変適応技術

図 4.10.2 は，計算機システムにおいて，システムを問題に適応させるレベルをいくつかの段階に分けたもの（これを"適応レベル"という）と，可変適応技術との関係を表わしたものである。

・適応レベル

計算機の構造をいくつかの階層にわけたものである。図 4.10.2 において，上の段階に行く程，計算機の構造が応用問題に近づき人間にとっては考えやすくなる。

・可変適応技術

計算機の構造を変化させることのできる技術のことで，ソフトウェア，ファームウェア，ハードウェアの三つが考えられる。ここで，ソフトウェア技

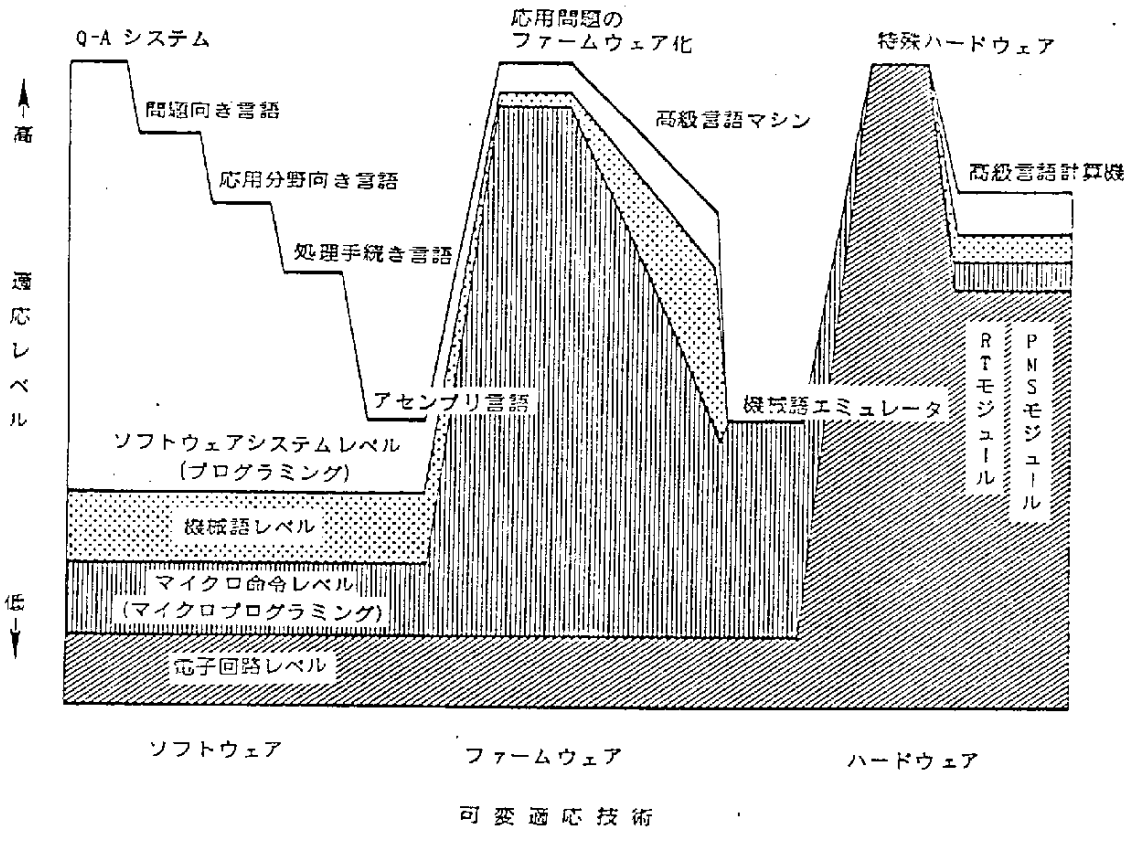


図 4. 10. 2　　計算機システムにおける適応レベルと可変適応技術

術では，機械語レベルは固定で，その上のソフトウェアシステムレベル（プログラミングレベル）で可変で適応させることができ，ファームウェア技術では，電子回路は固定で，マイクロ命令レベル（マイクロプログラミングレベル）で可変，ハードウェア技術では，電子回路レベルで可変にさせることができる。また，ここでは変化ということを必ずしも自動的にということに限定しない。

これらの技術を使い，適応レベルのそれぞれの段階に応じて計算機の構造を応用問題に適応させていくことが可能である。一般的には，可変適応技術としてハードウェア技術を用いることは，実行効率の因子の適応度が上げられるが，コストの因子の適応度は下る。ソフトウェア技術はその逆であり，

ファームウェア技術は中間に位置すると言える。また普通，これらの技術は組み合せて使われる。

以下，それぞれにつき具体的な例を挙げ，どのように適応させるかを論じる。

① ソフトウェア

ソフトウェアによる可変性は，計算機誕生とともに知られ，また研究もすすめられてきた。ソフトウェアによる可変性は，命令セット（Instruction Set Processor，略してＩＳＰ）レベルは固定で，その上での可変性を得ることができる。可変性を利用して適応を持たせる例として，FORTRAN，COBOLなどの処理手続言語（Procedure-oriented languages）があり，更に，問題の記述のしやすさ（表現の仕方）という面では，適応度の高い応用分野向き言語（Application - oriented language）：LISP（リスト処理向き），SNOBOL（記号処理向き），GPSS（シミュレーション向き）などがあり，また問題向き言語（Problem-oriented language）として，RPG（レポート作成用），STRUDL（構造解析用）などがある。

このように，ソフトウェアには，問題の表現方法についての適応度を上げることができる。しかも，適応を得るための可変性に対してのコスト上昇は機構上ない。しかし，ソフトウェアで記述された問題は，コンパイラやアセンブラで固定された命令セットに変換され，実行されるわけであるから，言語や記述されたものと，命令セットに著しく相違がある場合は，処理速度という点で適応しなくなる。すなわちＩＳＰレベルが固定であるとソフトウェアの適応性が生かせなくなる場合がある。

② ハードウェア

ハードウェアによって計算機に適応を求める場合は，処理速度の因子では最も優れている。ハードウェアには，ソフトウェアに見られたような可変性の自由度は少ない。ハードウェアで応用問題に計算機の構造を適応さ

せる場合，最も望ましいのは，回路レベルから問題に適応したものを作り上げることであろう。しかし，これは一般に以下に述べる理由により，非常に難しい。それは，先ず第一に，応用問題と回路の間に概念上の開きがありすぎて対応がつけにくいこと，もしできたとしても，適応範囲が著しく狭くなり，コスト高になるからである。

そこで歴史的に見ると，先ず経済的に採算のとれる方法，すなわち少しでも可変性をだし，回路の標準的機能をインタフェースの統一されたモジュールとして用意しておき，それを用いて適応構造を構築する方法が考察された。モジュールの機能レベルならびに種類は，その時点での半導体技術，応用からの要求により決めるが，技術的には，低機能のものの方が作りやすいので，回路レベルより一段階上のレジスタとそれに関係するデータの処理機能を単位としたレジスタ転送レベル（Register Transfer level 略してRTレベル）のような機能的には低いものから作られた。例えば，1967年にはWashington大学でマクロモジュールが開発され，1971年にはDEC社からRTMというモジュールセットが売り出された。当時の技術レベルからして当然のことながら，物理的な大きさや接続ケーブルの多さなどから，これを用いて応用問題に計算機構造を適応させた場合，ハードウェアオーバーヘッドを大きくしてしまうという結果を招いたこともあったが，うまくいく場合もあった。当時のデータによると，DEC社のRTMを使い，特殊目的，例えばFFT専用機を作った場合，当時のCDC 6600などの大型機と比べて，同じくらいの速度が得られたと報告されている。これに対しPDP－8のような計算機を作ったときは，コスト2倍，速度は40％程度であると報告されている。

しかしながら，最近の驚異的な半導体技術の進歩は，これらマクロモジュールやRTMに代わる新しいRTレベルの素子を生み出した。一般にビットスライスのマイクロプロセッサと呼ばれるもので，1974年に発表されたIntel 3000シリーズに始まり，その後AMD 2900，Fairchildマクロ

ジックなど続々と発表された。これらは，ほとんどがバイポーラ系技術を使っており，高速，小型で，しかもＲＴレベルでの汎用性を兼ね備えているというようにＲＴＭやマクロモジュールの欠点を克服した。例えば，最近では，これらと後述するマイクロプログラムの技術を使って，ＩＢＭ370シリーズのような大型機を安く作る試みもある。またこれらの素子を使い，言語プロセッサを作ったり，特殊計算機を作ったり，よりユーザに近いレベルでの応用も当然考えられている。

次にＲＴレベルモジュールより一段上の機能をもつＰＭＳ（Processor Memory Switch）レベルモジュールについても簡単に触れておく。ＰＭＳとは，その名の示す通り，計算機の要素をモジュールの一単位と考え，これを組み合わすことによって応用に適したシステムを作り上げるものである。これは最近の半導体技術の進歩により，特にマイクロプロセッサの発展が，その実現の裏付けとなっているのであるが，現在実験されているものは可能性の追求という程度なので，マイクロプロセッサ化されていないものもある。

例えば，Lockhead 社は，1973年にＳＵＥというプロセッサモジュールを発表したが，これは，ミニコン程度の機能を持つ初めてのPMSモジュールで，実際にＢＢＮによって，ＡＲＰＡ網の計算機間通信制御用システム製作の際に使われた。

また，カーネギーメロン大学では，ＣＭ＊といって，ＤＥＣ社のマイクロコンピュータＬＳＩ－11を中核に使ったPMS モジュールを組み合せた応用指向計算機の，また我国の電子技術総合研究所でもＡＣＥというＰＭＳレベルモジュールを使った応用指向計算機の研究が進められている。これら二つは，特定の応用は考えずに，ある応用問題が与えられた時いかに適応した構造がとれるかを模索するための実験システムである。目下のところ，ＰＭＳレベルという性格上，各モジュール間の通信能力，通信方法における柔軟性，応用問題の並列構造性などが明らかにされなければなら

ないなど，研究段階にとどまっている。

最後に，最近の超ＬＳＩ技術を背景としたカスタムメイドＩＣについて述べる。これは，ユーザが応用問題に合わせて回路レベルから適応素子を作る技術である。この項の初めに指摘したように，応用問題に合わせ回路レベルで適応させることは二つの理由で難しいと述べたが，少なくともその一つ，すなわち経済的に採算がとれるかという点では，うまくいくようになってきた。しかし，もう一つの応用問題と回路との写像をどのようにとるかは，依然未解決である。したがって，現在技術的には可能であるというだけで，適応素子としては機械的に低いものしか作られておらず，高密度のものでは，現在の計算機そのものを一素子とするというようなことしか試みられていない。

以上のことをまとめておく。

全般的に言えることであるが，ハードウェアによる適応性を得ると言っても，前述したように，応用問題を回路レベルにすぐ写像するのは難しいので，回路レベルの機能を上げ，ＲＴレベルまで持ち上げ，このＲＴレベルのモジュールを使って応用問題に適応するような方法がとられたのである。ＲＴレベルでの適応性を得るということは，具体的には，問題の性質を反映するようなＩＳＰを考え，それをＲＴレベルのモジュールを使って作り上げるということを行うわけである。このような手法が広く行き渡るには，ＲＴレベルのモジュールとしてどのようなものを用意するかが明らかにされ，それを組み合せてシステムを作るための開発サポートシステムの充実を行うことが重要であると思われる。すなわち，ＲＴレベルで作り上げたシステムは一度できてしまうと，可変性がないため，応用が変わったりした場合困るが，次のシステムが直ちにできるならば問題はない。従って，以上論じた手法は問題が固定した小さなシステムを作る場合有効であると思われるが，複雑な問題になる程問題を固定することは難しくなるので，そのような場合には，次に述べるファームウェア技術の利用が望ま

れよう。

またPMSレベルでの適応性を得るには，その性質上，問題の性質が並列性を考慮してシステムに写像できることが先ず重要であるが，並列性が検出でき各PMSに並列分散させる場合は，各PMSは一つ一つ独立して考えることができ，その中ではISP が重要となる。したがって，その基礎となるのはやはりＩＳＰレベルでの適応性であると言えよう。

③　ファームウェア

前述したソフトウェアとハードウェアの中間に位置し，そのインタフェースの役割りを占めるのが，ファームウェア技術（またはマイクロプログラム技術といわれる＊）である。

マイクロプログラムによる可変性は，基本的にはＩＳＰレベルにおいてである。しかし，ＩＳＰレベルといっても，考え方によっては大変広く考えることができる。いわゆる計算機初期の段階では，ＩＳＰは，例えばIBM S／360の命令セットのような計算機の機械語と考えられていた。そこで，マイクロプログラムを使い他の計算機の機械語をシミュレーションする技法が生れた。これをエミュレーションという。これを使うと，旧型計算機のプログラムをマイクロプログラムサポートのもとで新型機の上で走らすことができるので，確かに有用であった。しかし前述したように，計算機の応用分野が広がるにつれ，機械語が唯一のものとは考えられなくなり，ＩＳＰとして考える範囲，機能，レベルも拡大解釈されるようになる。すなわち，エミュレーションという言葉が，より広範囲に使われるようになった。例えば，計算機ではなく，言語プロセッサのシミュレーションなどもエミュレーションと言われるようになった。FORTRANやCOBOLに向いたＩＳＰは，従来の機械語といわれるＩＳＰ よりはるかに機能的にレベルが高く，――そのため，FORTRAN用ＩＳＰとは言わずFORTRAN用

＊　ファームウェアとは，マイクロプログラムを使い，システムを問題に適応させる技術のことで，ソフトウェアに対して使われるようになった。

IML（Inter-Mediate Language：中間言語）と言われる場合が多い――，すなわち，応用問題により適応している。この考え方を押し進めると，さらに応用問題に近いレベルでのISPを設定できるわけで，これは，応用問題のファームウェア化，またはユーザマイクロプログラミングなどと言われているが，一応ソフトウェアと同等のレベルまで可変適応度はあると考えてよい。

また，ISPとして実行時における適応性を追求するものを用意するだけでなく，デバッグや保守のためのISPをも設定できる可能性があり，より広い範囲にわたり適応させることができる。

マイクロプログラム化した場合の効果については，先ず実行時間の改善ということが考えられ，いくつかの報告もあるが，それよりも，表現のしやすさなどシステムの構造を問題に適応させた場合の他の利点に注目すべきである。

このようにファームウェア技術の特徴としては，ISPを問題に合せて可変にできることである。すなわちISPを問題に適応させることは，ハードウェアの技術を使ってもできたが，この場合自由にその構成を変えないで変更することはできなかった。ファームウェアでは，変更に対しての自由度は大きく，WCS（Writable Control Store）を使うとソフトウェア並みにその変更が可能である。しかし，解決せねばならない問題点として，どのようにISPを問題に合わせていくのかが明らかにされなければならず，これは重要な問題である。すなわち，計算機の構造をISPレベルで問題に適応させる方法では，問題の環境をどのように検知し，ISPレベルで計算機の構造に写像するかが明確にされなければならない。

またファームウェアはソフトウェアとハードウェアの中間に存在する。先ずソフトウェアとのインタフェースを考えるならば，ソフトウェアでは，前述したように，解くべき応用問題を最も記述しやすいように適応させているので，結局そのソフトウェアが最も効率よく動くようにファームウェ

アを使いサポートしてやる。具体的には，高級言語がコンパイラなどを使ってＩＳＰに落される時，落しやすいような構造にしてやったり，またファームウェアで実現されているＩＳＰにソフトウェアからパラメータなどの情報を受け渡す方法が明確にされなければならない。またハードウェアとのインターフェースを考えると，先ず多くのＩＳＰを効率よく記述できるようなマイクロプログラムを備えたマシンをどのように作るかということがあげられる。このような計算機は，汎用マイクロプログラム制御計算機と呼ばれるが，これにどのような機能を持たせるか，どのようなマイクロプログラムの形式にするかが重要となる。

(2) 適応化理論

① 適応化方法の分類

計算機の構造を解くべき問題に適応させる方法としての手順をまとめると，

1. 問題の性質を知る（適応環境の理解）。
2. 問題の性質により，目的関数を設定する。
3. 構造を適応化する（適応化）。

のようになる。

ⓙ 問題の性質を知る（環境の理解）。

問題の性質を知る最も望ましい方法は，解くべき問題のアルゴリズムの把握から知ることである。これをここでは直接法，または静的環境情報による方法というが，この方法は現在難しくなりつつある。それは，前述したように，現在計算機上で解かれる問題は多種多分野に渡り，複雑，巨大化しており，それが個々の部分的な性質は明らかであっても，問題を全体でとらえた場合の性質を不明確にしているからである。

そこで，問題の性質を知る方法として，間接的な方法が考えられた。間接的に問題の性質を知る方法は，実行時における計算機の動作状態の偏りに注目するものであり，これはいくつかの実際のデータに基づいて

いる。例えば，KnuthらはFORTRANプログラムの動作状態のデータを解析し，プログラムの実行時にはプログラムの4％以下の部分で実行時間の50％以上が費やされているということを報告している。Darden, Hellerらも多くのプログラムを調べた結果，コードの3％の部分で実行時間の60％を費やすという事実を示している。また，SaalはISPの動作状態についての偏りについて報告している。このように，計算機の実行動作をもとにして間接的に問題の性質を知ることができるが，例えば，プログラムの偏りが知れると動作時におけるプログラムの状態がわかる。これは，間接的にではあるが，解くべき問題においてどの部分が重要であるかなどがわかることで，問題の性質を知ったことになる。また問題によっては動作させないとその性質が明らかにならないものもある。そこで以上の方法を動的環境情報による方法という。

(ii) 最適構造の決定

問題の性質が知れると最適化のための目的関数を設定し，適応化を行う。

目的関数とは，例えば，ISP構造でいえば，一つにはISPの種類（どのような動作を行うISPを用意するか）と，ISP間で変数受け渡しのためのオペランドの種類などの構成法である。

(iii) 構造適応化法

具体的な技術としては，4.10.2(1)で述べた3通りのものがあるが方法としては，静的な方法と動的な方法が考えられる。静的な方法とは解くべき問題を実行させる前に最適構造にする方法であり，動的な方法とは，実行途中に構造を入れ替える方法である。

(iv) 方法の分類

問題の性質を知る方法と構造適応化の方法にはそれぞれ二つの方法があった。そこでこれを組み合わせ，適応化の方法として合計四つの方法が考えられる。

すなわち

1. 静的環境情報による静的な方法
2. 静的環境情報による動的な方法
3. 動的環境情報による静的な方法
4. 動的環境情報による動的な方法

である。

② チューニングと学習

動的環境情報により環境を知る方法では，計算機を一度動作させ，その結果を解析することによって，問題の性質を知る方法である。従って，何らかの方法で一度構造を設定し，これをもとにして改良することにより，適応構造を作り出す。そのため，このような方法はチューニング(tuning)とも呼ばれる。

チューニングは人手で行ってもよいが，自動的に行う方法の開発が望まれる。また，チューニングにおける“学習”の概念は重要である。ここでの学習とは，基本的には普通使われている学習と差異はない。すなわち，一度経験した事を覚えておき，次回同じような状態に遭遇した時，その経験を生かすということであり，これをチューニングにあてはめるなら，一度チューニングフェーズで発見された適応構造を覚えておき，次回のチューニングで活用することなどが考えられる。これは一例であり，他にも場合によりいくつかの学習が考えられるが，この基礎となるのは，“記憶”と“判断”であり，チューニングの自動化と合わせ実現されることは極めて興味深い。なお，このような機構を備えたマシンを“Adaptive Machine”とか“Learning Machine”と呼ぶ(図4.10.3参照)。

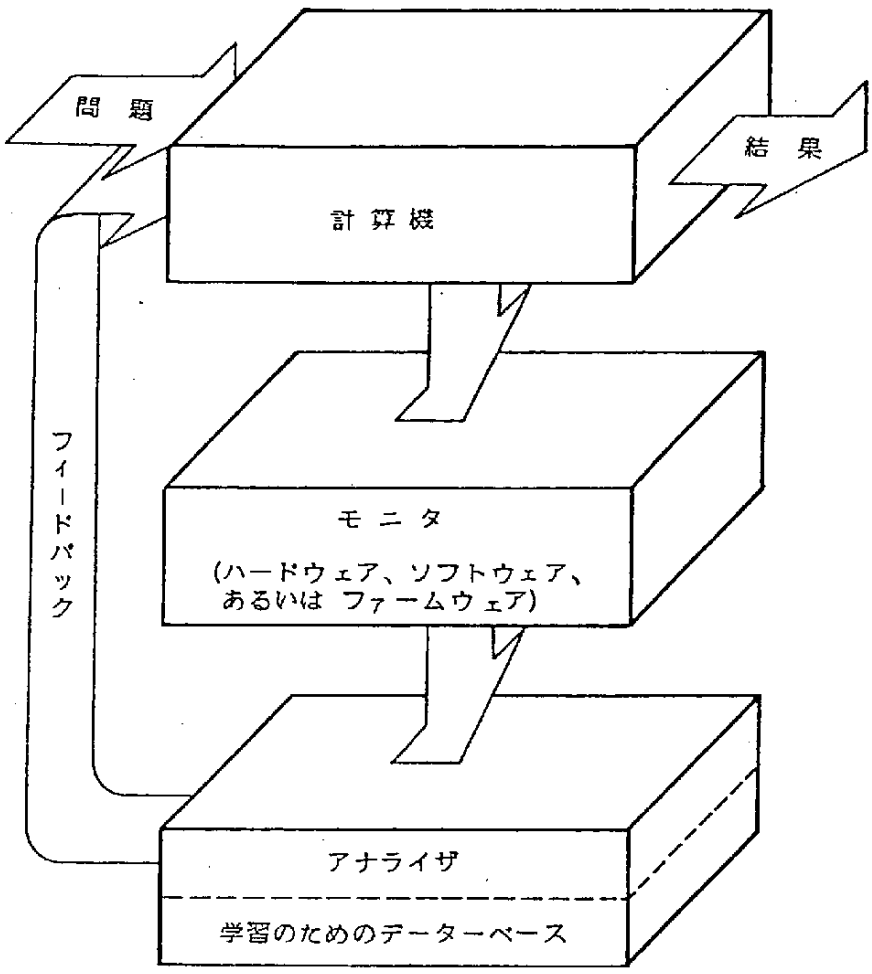


図 4.10.3　自動チューニング機構のモデル

4.10.3　応用と効果

(1)　静的環境情報における方法

（問題が最初から明らかな場合）

①　静的な方法

この方法には，

- （機械語）エミュレーション
- 高級言語マシン
- 応用問題のファームウェア化

などがあり*，期待される効果としては，

- 実行効率の向上

（およそ2桁の効果）

- ソフトウェア生産性向上

などがある。

② 動的な方法

デバッグを行う場合，計算機を動作させながら，その構造を動作中にダイナミックに変え，高度なデバッグを行うという例がある（Sakamura）。図4.10.4に示すのは，この原理図であり，エミュレータをデバッグの要求コマンドごとにダイナミックマイクロプログラムを使い，つくりかえている。

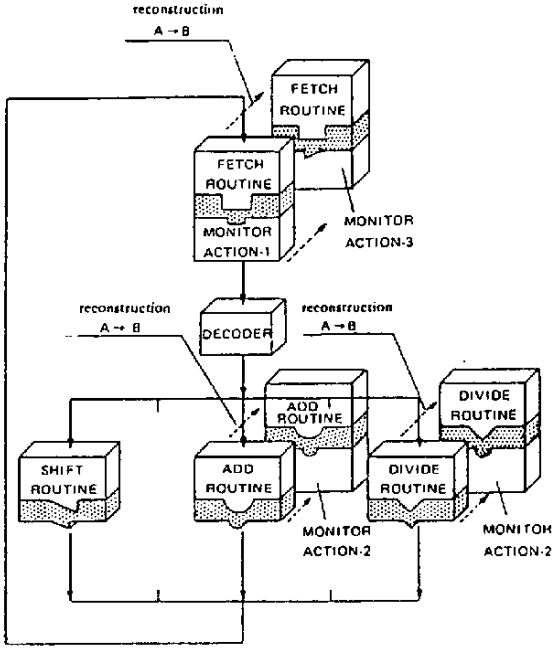


図4.10.4 デバッグ用エミューレータの原理図

効果としては，従来出来なかった高度なデバッグ機能の提供（デバッグ時におけるバックトラックや入出力デバッグ）の他，デバッグタイムの大幅な短縮の報告がある。

(2) 動的な環境情報における方法

（適応させる対象が定まっていない場合）

① 静的な方法**

これにはいくつかのレベルが考えられる。

(i) 言語レベル

＊ ユニバーサルホストマシン，高級言語マシンの項を参照されたい。

＊＊ これを一般的にチューニングと言う。

たとえば，G．J．Hansenによる Adaptive Compiler では，オプティマイズのレベルを自動的に変える事により（実行回数が増えるに従いオプティマイズを深く行う），「コンパイル時間＋実行時間」が従来のコンパイラに比べ，優れている事を示している。

(ii) オペレーティングシステムのチューニングとしては，問題の性質により動的な情報をもとに（たとえばワークロード），システムパラメータを可変とし，適応させる方法が考えられる。特に空間利用率（主記憶，2次記憶の有効利用）の向上，資源の有効利用などの効果があるといわれる。たとえば，P．R．Blevinsらによる Dynamically Adaptive Operating System では，普通のOSに Identification パート（モニタ）と，Decision パート（評価），Modification パート（最適化）が加わったものであり，(1) Identification パートで，デイスクリプタ（システムの事象，パラメータの推定量，モニタの結果わかる現在の統計モデルの形などからなる）と呼ばれるシステム動作の統計を動的に収集する。(2) Decision パートで，現在のシステムディスクリプタとシステムポリシーディスクリプタを比較して，その誤差（システムエラーセットと呼ばれる）を評価結果として出力する。(3)システムエラーセットをもとに最適化を行う，のような手順で動作する。シミュレーションによれば，リソース管理の最適化が行え，ラウンドロビン方式と比べると，十分満足のいくものであったとされている。

そして，ここでの理論的課題は，適応オペレーティングシステムのモデルをより詳しく行う事にあり，また技術的課題は，このようなオペレーティングシステムのチューニングをあらかじめ考慮してマシン設計時から導入することである。

(iii) （狭義の）アーキテクチャレベル

• ISPレベル

これは，特にアーキテクチャ・チューニングと呼ばれる。アーキテク

チャ・チューニングでは，初め「問題」をとりあえず与えられた，例えば汎用計算機アーキテクチャの上でインプリメントレ，その実行動作をモニタ解析することによって適応する．（セマンティック・ギャップのより小さな）ISPを自動的に作り上げる。原理的には前述した実行時局所性にあり，ポイントとしては，どうやって局所部分を発見するかにある。

以下，2つの例を述べる。

① Abd—Allaらの方法

実行時アドレス状態を問題を解く環境と考え注目する。

まず，アドレストレースによりプログラムのループを見つける。これとスタティックな方法と異なる点は，ループの動特性も合わせて検出することで，ループの回数，ループ中で参照されるメモリロケーションの回数をデータとして収集する。解析アルゴリズムは，ループ中で最もよく使われるメモリのリージョンを，マイクロプログラムだけでアクセスできるMGPレジスタ（マイクロプログム用ジェネラル・パーパスレジスタ）に置き換えてやることである（実際には，上位いくつかが置き換えられる。）

自己修正機能としては，ループに入る前にメモリーレジスタ転送を一度だけ行ない，ループ中のメモリ参照命令は，レジスタ参照命令に置き換わるようにしてやることであり（Preloadという），再びループを抜けでる時にレジスターメモリ転送を行ない，もとに戻す（Restoreという）。

実験結果を図4.10.5に示す。この方法で4倍ぐらいまで実行速度が改善されると報告されているが，この方法では，ループが少ない，または構造のきたないプログラムでは効果が少ない。またチューニングの結果に一般性はない。すなわち，チューニングは特定の問題のみに対して行われ，他のチューニングにその結果を役立たせることはで

きない。

㈰ Sakamuraの方法

この方法ではアドレスパターンではなく，インストラクションのパターンに注目する。

インストラクションパターンとは中間言語がそれぞれどのくらい実行され，また実行された後でどの中間言語に移るかのチャートをいう。ここで，遷移の大きなインストラクションパターンに注目する。

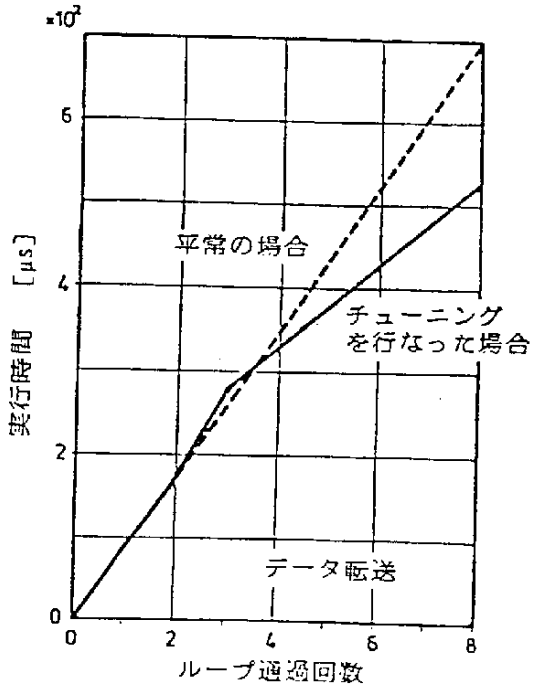


図4.10.5　チューニングの結果（HP-2100機械命令におけるデータ転送プログラム）

ある基準を設け（例えば頻度），一定基準以上のインストラクションパターンを一つの命令と考え，命令合成を行ない，例えば新たにマイクロプログラムでインタプリトルーチンを作り，新命令として登録する。以後再実行する場合には，新命令が使われる。アルゴリズムには，チューニングのためのオーバーヘッド（新命令を作った時に増えるWCS量の増加）を予測して，それが最少になるように抑える戦略が用意されている。効果としては図4.10.6に示すようなもので，実行効率は問題にもよるが，1.5～5倍程度である。

この方法では一度行なったチューニングの結果を使ってチューニングを行ない，チューニングのオーバーヘッドを減らすことができる。（チューニングにおける学習の概念）

- PMSレベル

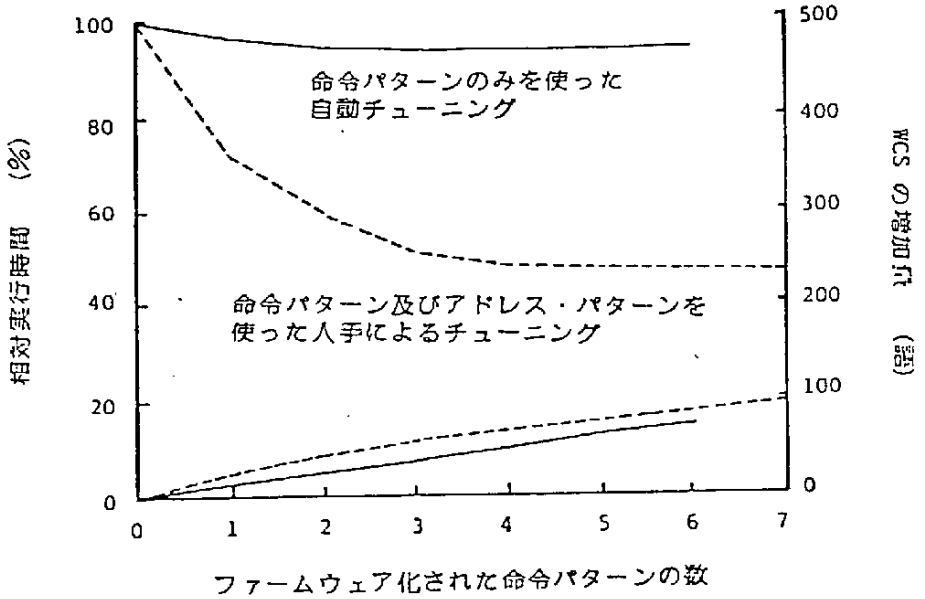


図 4.10.6　チューニングの結果（HP—2100 機械命令における
バブル・ソートの問題）

このレベルでは，実行特性を向上させるため，機能モジュールを動的な情報をもとに最適結合させたり故障時において故障モジュールを切り離し自動回復させるなどが考えられる。実際試みられているものとしては，4.10.2(1)—②でも述べたカーネギーメロン大学のCM*，電総研のＡＣＥなどに機能向上を目的としたＰＭＳレベルでの適応アーキテクチャの概念がみられる。また，高信頼性アーキテクチャは，多くの所で試みられているが，ＰＭＳレベルのものは，少なく，（ほとんどがＲＴレベルである。）従来の問題となっていたスイッチが，ＬＳＩ技術の発展により現実的な規模でつくれる可能性が出てきたことなどから，今後の研究に待つべき面が多い。

② 動的な方法

動的な環境情報に基づく動的な方法とは，モニタに基づき，それを実行中に解析し，オンラインでフィードバックをかけ，アーキテクチャを適応

させてしまうようなものを言う。この分類に入るものは必要性からして多くはないが重要であると思われる。例えば一例として，Sakamuraらによって示された数値計算適応計算機の例がある。この計算機では，数値計算実行中に精度落ちを常にモニタしている。精度落ちがモニタされると，プログラムをバックトラックし，演算桁数を増加させ再計算する。このような計算機では精度落ちの考慮がユーザーレベルではいらないのでプログラムの方法が従来の方法と異なり，大変解り易くなるという特徴をもつ。

4.10.4 第5世代における可能性

(1) 今後の課題

適応計算機実現にあたって解決しなければならない課題は以下の通りである。

(i) 理論面での課題（適応計算機のモデル化）

適応させるには，ゴールに相当する目的の設定と，そのための可変パラメータを明確にする必要がある。そしてこれは，適応計算機のモデル化に他ならない。現在のところ，適応計算機のモデル化は，個々の問題に対して個別に検討されているにすぎない。

モデルの統一化は必ずしも必要であるとは思われないが，広義の適応性の達成のためには個別にモデル化された物同士の関係を明確にする必要があり，これは今後の重要な課題である。

(ii) 技術上の課題

実現技術上の課題としては，まづ可変構造計算機をどのように構築するかがあるが，これはユニバーサルホストマシンと密接な関係があり，特にダイナミックマイクロプログラムを適応技術としてどのように使っていくかの研究が重要であろう。また，特に（環境）モニタリングのための機構は，あらかじめマシン構築時にアーキテクチャ上，備えられている事が望ましくこれをどのように装備するかも課題の1つである。その他適応のた

めのオーバーヘッドの見積もこれは実現コストとも関係してくるので，より正確に行う必要があろう。

(2) 第五世代以前

以上述べてきたように，適応化の理論，技術は，目的指向型アーキテクチャ構築のためには重要であり，したがって第五世代を待たずに一部実用計算機に，取入れられていくと思われる。また，すでに取入れられているものもある。これらは主として，4.10.2(2)の分類による，静的な情報による静的な方法や，動的な情報による静的な方法である。

たとえば，

- 実行時特性を重視したISP構造を持つアーキテクチャ
- チューニングを意識したオペレーティングシステムをもつアーキテクチャ

などである。また，アーキテクチャレベルでの構造の可変性も取り入れられつつある。

(3) 第五世代

第五世代ではこの傾向がますます強まろう。特に「動的」な情報にもとづく適応性が，システムアーキテクチャレベルで確実に保障されると思われる。特に，

- 動的環境情報をもとに動的に適応するアーキテクチャ，高信頼性アーキテクチャ
- 適応オペレーティングシステム
- 広範囲な適応性（システム全体レベル）
- 自動チューニング理論

などが実現レベルで保障出来る可能性が強い。そしてその効果は，

- 実行速度の向上
- システム資源の有効利用

などだけでなく，

・使用感の向上

・ソフトウェア開発生産性の向上

・高信頼性の実現

など広い範囲にわたり，大きく有効であると思われる。

4.11 ソフトウェア工学

ソフトウェア工学とはソフトウェアの開発技術に関する工学であり，ソフトウェアの生産性の向上のために以下の問題に取り組んでいる。[1),2)]

(1) ソフトウェアの開発の効率化

(2) ソフトウェアの信頼性の向上

(3) ソフトウェアの保守と運用の効率化

ここでは，現在のソフトウェアの生産の問題点とその長期的解決策を検討し，ソフトウェア工学の観点から1990年代のコンピュータが持つべき主な機能を検討する。

4.11.1 ソフトウェア生産の問題点

ソフトウェアという言葉が使い出された当初は，これは紙と鉛筆の仕事であり，頭脳を用いるので日本人に適した産業になり，日本の得意な産業になるであろうと言われていたが，それから大分たった現在，残念ながらソフトウェアは日本の得意な産業とは言い難い。

ソフトウェアの生産性が問題になっているのは，なにも日本だけの問題ではなく，欧米でも大きく取り上げられている。

1960年代の後半からいわゆる「ソフトウェアの危機」に対処すべくソフトウェア工学の研究が始まり，Dijkstra の構造的プログラミングを始めとするソフトウェア開発技術が利用され，効果を上げている。そして，ソフトウェアの生産の工学としてソフトウェア工学の研究が盛んになっている。

日本と欧米とを比較すると，日本の方がソフトウェアの生産により多くの問題があると言われている。これは日本語がコンピュータで処理しにくかったことが最大の原因であろう。[3)]

今後はオフィスオートメーションの時代と言われたり，マンマシンインターフェース向上のための人工知能の時代であるといわれたりしている。確かに，ハ

ードウェアの進歩によりその可能性は十分にあっても，ソフトウェアの生産にボトルネックがあれば実現はおぼつかない。是非とも，ソフトウェアの生産性を飛躍的に向上する必要がある。また，資源を輸入にたよっている日本は，頭脳産業であるソフトウェア産業で世界に貢献したいものである。

日本の高学歴社会という特長を活かして，ソフトウェアが日本の得意産業になるように努力しなくてはならない。

ソフトウェアの生産には，現在次のような問題点がある。

問題点の解決に当っては，欧米の技術を参考にするのは勿論であるが，日本独自の技術を開発する必要がある。

1）システムの巨大化

2）膨大なドキュメント

3）マンマシン・インターフェースの不自然さ

4）ソフトウェアの互換性のなさ

4.11.2 問題解決の方向

ここでは，コンピュータの技術の観点から，ソフトウェア生産性の問題点を解決する方向を検討する。

従来のソフトウェアの生産法は，何と言っても人間集約的であり，コンピュータのハードウェアの製造技術の進歩で，コンピュータが低価格になってもソフトウェアの低価格化にはつながらない。反対に人件費は年々，上がる一方であるので，ソフトウェアの価格も高くなってしまう。

そこで，コンピュータのハードウェア（プロセッサ，メモリ等）の能力を大量に使い，ソフトウェアの生産過程で人間は細かな作業をしなくてもよいように，省力化すべきである。例えば，超高級プログラミング言語を用いて，ソフトウェアの生産性を上げて行く必要がある。

ソフトウェア工学では，次の3項目が重要とされている。

(1) 機能の高レベル化

アセンブラ言語よりは高級プログラミング言語を使用した方が能率よくプログラミング出来るように，マンマシン・インターフェースを高レベル化し，人間は創造的な作業のみをすること。

(2) 仮想化

仮想記憶システムでは，プログラマがメモリの実装容量を意識しなくてもよいように，プログラマはハードウェアの性質にとらわれることなく，論理面のみに集中できるようにする。ワンレベル・ストーレッジの技術はディスク装置等の二次記憶装置を仮想化するものである。

(3) 標準化

高レベル化したインターフェースと仮想化した機能を標準化することで，ソフトウェアの生産性の向上と互換性が達成できる。

これらの項目を短期間で，一気に解決することは，ほぼ不可能である。コンピュータの性能，メモリの容量，外部記憶装置の容量等が，現在の10倍から100倍，更にそれ以上必要となると考えられるからである。

幸いなことに，超ＬＳＩの製造技術の進歩で，コンピュータの利用者は10年後には1桁以上も余分にハードウェアの能力を使用できる見通しがある。

更に，アーキテクチャの新技術によれば，適用分野にもよるが，その上に更に1桁以上も性能向上ができる見通しがある。

このように，10年後のコンピュータを考える時，現在から十分対策をたてておけば，ソフトウェアの諸問題を解決できることは確実である。

以下に問題解決の主要な具体策をあげる。

(1) 人工知能（自然言語処理，知識ベース）

ソフトウェアの設計・作成・保守用の日本語のドキュメントをコンピュータに登録し，特定のソフトウェアに関する情報をコンピュータに向って問い合わすと，所要のデータが端末に表示されるようなシステムがあれば，ソフトウェアの生産性の向上に大いに貢献することは確実である。

このような機能は，人工知能の研究の目標でもあり，自然言語処理の研究，

知識ベースの研究等が今後必要である。なお，当面はコンピュータ処理を考えた制約条件の強い日本語であっても，かなり役に立つと考える。

(2) 要求仕様技術

作成するソフトウェアが何をすべきかを，無予盾であいまいさなく，完全に定義するための技術である。ソフトウェアの要求定義でよく使われる基本概念を表現する用語をキーワードとして設定した要求定義言語の研究が盛んである。また，図形表現の要求定義技術も研究されている。

日本では，日本語を用いかつ形式的な要求定義言語と，人間のパターン認識能力を活用しソフトウェア設計者間のコミニュケーションを円滑にする図形表現とを両用する必要がある。

また，要求仕様を与えると自動的にプログラムが生成される（自動プログラミング）システムも今後の課題として重要である。

(3) 超高級言語

COBOL，PL／I，FORTRAN，PASCAL 等の高級プログラミング言語は，長年使用されてきておりこれらに対抗できるような言語は他にないように見えるが，実は1990年代のプログラミング言語としては，次の機能を強化する必要がある。

- 抽象データ構造のサポート
- データタイプの標準化
- 2進数とか10進数，ビット，文字等のデータ・タイプを意識しなくてもよいようにする。
- 集合やリストなどのデータ構造のサポート

(4) 人間の視覚を活用する図形表現の利用

ソフトウェアの設計に図形表現を用いる研究が盛んである（ＩＢＭ社のＮＳチャート，日立のＰＡＤ，日電のＳＰチャート，富士通のＳＤＤ等）。論理を考える時の視覚の重要さは心理学的にも指摘されており，今後，グラフィック・ディスプレイ端末の普及とともに実用化されるであろう。また，図

形表現を用いるソフトウェアは将来，データフロー・ダイヤグラムという点でデータフロー言語とも関係を持って来ると予想される。[4),5),6),7),8)]

(5) 機械翻訳機能

ソフトウェアの製品を海外に輸出する時には，日本語のドキュメントは機械翻訳の機能で各国語に翻訳する。これも人工知能技術の応用である。

なお，第五世代のコンピュータのＯＳの構造についての具体的な検討は来年度に行う計画である。

4.11.3 新アーキテクチャとの関係

以上のソフトウェア工学上の問題点を解決するために，新アーキテクチャの持つべき機能の一例を示す。

(1) 文字情報の高速処理

- データ形変換
- ストリング比較
- リスト処理

これらの機能の高速化は自然言語処理，知識ベース，（超）高級言語の処理に役立つ。具体的には，ロジック・イン・メモリとかアソシアティブ・メモリ，リスト処理プロセッサ等の技術が関係するであろう。

(2) ファイルの一元管理

知識ベースを始めとするファイルの一元管理をサポートするアーキテクチャが必要となる。これにはワンレベル記憶域管理方式が有力な候補である。

また，ファイル等のリソースのセキュリティの一元管理のためにはケーパビリティ・アドレッシング機能が必要となろう。

(3) 図形処理機能

図形表現によるソフトウェアのドキュメントをサポートするためにも，使い易い図形処理システムが利用出来なくてはならない。

参考文献

1) ソフトウェア・エンジニアリングの動向，日経エレクトロニクス，1－22，№204，1979

2) ソフトウェア技術の発展経緯と今後，日経エレクトロニクス，1－7，№209，1980

3) ソフトウェア工学における日本語の役割，神田，杉本，情報処理学会ソフトウェア工学研究会資料，1978.1

4) D.C.Smith, PYGMALION: A Creative Programming Environment, Report STAN-CS-75-499, Stanford Artificial Intelligence Laboratory, June 1975

5) R.Wirry, Dimensional Flowcharting, Software-Practice and Experimence, 1977

6) Y.Chu, Introducing Software Blueprint, Proc. COMSAC, 1977

7) Y.Kanda and M. Sugimoto, SDD: Software Diagram Description, 3rd UJCC, 1978

8) W.J.Tracz, Computer Programming and Humane Thought Process, Software-Practice and Experimence, Vol. 9, 1979

昭和55年3月発行

発行所　財団法人　日本情報処理開発協会
東京都港区芝公園3丁目5番8号
機械振興会館内
TEL（434）8211（代表）

印刷所　株式会社　昌文社
東京都港区芝5丁目26番30号
（全専売ビル）
TEL（452）4931

54-R015