解　説

機能メモリのアーキテクチャとその並列計算への応用

4. 機能メモリによる超並列処理†

安　浦　寛　人††

1. まえがき

集積回路技術の急速な進歩は，高い並列性をもつハードウェアの実現を可能にしたが，いまだにこの可能性を十分に生かしきるような並列計算機アーキテクチャは実現されていない．すなわち，3年で4倍と言われる集積度の向上の成果を，直接的に並列度の向上に反映できるようなアーキテクチャはまだ確立したとは言いがたい．

半導体メモリは，記憶機能をもったセルを2次元格子状に配置し，それらを，縦方向（語方向）および横方向（ビット方向）のバスで接続した構成をしている．各セルは，2次元アドレス（語アドレスおよびビットアドレス）で指定される．各セルには，縦横数本の通信線（バス）が接続しているだけで，全体の構造が2次元配列であり，現在の集積回路技術に最も適した回路構造となっている．

メモリのもつこの高集積性を生かしたまま，演算機能を付加したものが機能メモリである．従来の機能メモリは，記憶機能を一義的に考え，演算機能は付加的なものとする考え方が強かった．しかし，演算機能を重視すると，機能メモリはきわめて高い並列度をもつ並列計算機構と考えることもできる[1)~3)]．近年の集積回路技術の成果を踏まえて，機能メモリをベースにきわめて高い並列度をもった超並列計算機を構成しようとする試みが始まっている[4)~8)]．本解説では，機能メモリを超並列計算機アーキテクチャとして捉え，数百万プロセッサの超並列計算機を実現する一つの有力な手法となることを示す．

† Massively Parallel Parocessing by Functional Memories by Hiroto YASUURA (Department of Information Systems, Interdisciplinary Graduate School of Engineering Sciences, Kyushu University).

†† 九州大学大学院総合理工学研究科情報システム学専攻

2. で，超並列計算機構として捉えた機能メモリの性質をまとめる．3. では，機能メモリの並列計算モデルに関する議論を紹介し，その計算能力を示す．4. では基本回路構成として CAM (Content Addressable Memory) を用いた機能メモリ型並列プロセッサアーキテクチャ FMPP (Functional Memory type Parallel Processor Architecture) の例を示し，FMPP 上の超並列アルゴリズムを紹介する．

2. 並列計算機構としての機能メモリ

2.1 機能メモリにおける並列計算

本来，メモリの記憶セルや語における記憶機能は，ビットごとや語ごとに並列に行われている．メモリのビット数や語数は，10^6～10^9 程度を考えることができ，これらがすべて並列に動作すると考えると潜在的にはきわめて大きな並列計算能力をもつ並列計算機構であると考えることができる．機能メモリは，半導体メモリの記憶セル，語，語の集合，アドレス選択系などに演算機能を付加したものであり，これは単に記憶機能のみならず演算機能も備えた強力な並列計算機構となる可能性を秘めている．

たとえば，メモリの各語が単純なプロセッサ機能をもてば，メモリの語数（10^6～10^9 程度）の並列計算機となる．CAM などで実現されている一致検索や並列書き込みは，各語を1ビットの ALU をもったプロセッサとみなすに十分な機能を与えている．基本的には，メモリのもつ2次元格子とバスによる通信という集積度を最大限稼ぐことのできる構造を維持しつつ，並列計算機としての能力を付加することで，きわめて並列度の高い並列計算機構を実現する現実的なアーキテクチャができる．

機能メモリを並列計算機構として利用する場

合，各語ごとに制御系をもつことは難しいので，必然的にSIMD (Single Instruction Stream/Multiple Data Stream) 型[23]の並列計算機となる．基本的に，ベクトル計算機やSIMD型の並列計算機に適したデータ並列型のアルゴリズム[23]に向いた計算機構となる．ただし，語間の並列通信機構は一般に強化するのが難しいので，プロセッサ間でのデータの移動が激しいアルゴリズムには適していない．

このような並列計算機構のシステムへの組み込みを考えると，それ自身メモリであるので，ホスト計算機の主記憶空間の一部に割り当てることができる（図-1参照）．すなわち，主記憶の中に演算機能が分散されたことになる．通常は，主記憶の一部として働き，データが各語にセットされた段階で，並列計算機としての動作を行う．計算結果は，ホスト計算機からみれば，主記憶の一部に入っているのであるから，通常のメモリアクセスで利用することができる．データを主記憶内部で処理することで，CPU と主記憶の間のいわゆるフォンノイマンボトルネックを解消でき，効率のよい処理が期待できる．記憶空間と同程度の規模のプロセッサ数をもつバックエンド並列計算機をもったシステムが，メモリボード1枚をワークステーションに付加するだけで実現できることになる．

このような並列計算機構上でのアルゴリズムは，通信能力は低いが計算空間が広いという性質を考慮して考えなければならない．すなわち，通信コストがきわめて高く計算／記憶コストが比較的低い環境での並列アルゴリズムが必要となる．ベクトル計算機が得意とするデータ間の依存関係が低い配列計算などは機能メモリ上の超並列計算に向いている．このほか，本来メモリであることを利用して，各種のデータ構造とその基本演算を直接実現できる．プライオリティキューや辞書などの直接実現が考えられる[12]．さらに，単純な虱潰し探索や動的計画法のようなアルゴリズムもこのアーキテクチャ上で効率的に並列化できる[4),5),12]．一般的に，データの移動をともなわずに大量のデータに一斉に同様の演算を施すようなアルゴリズムに適している．

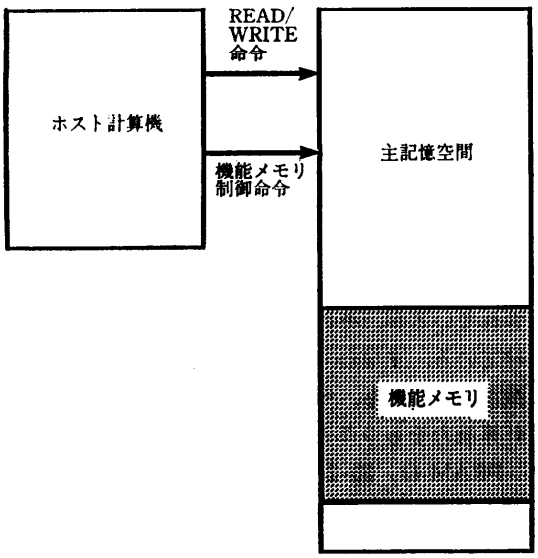


図-1 超並列計算機構としての機能メモリ

ソフトウェア開発の立場からみると，バッタエンド型の SIMD マシンなので OS やプログラム開発環境の大幅な変更は必要ない．ホスト計算機の主記憶自身が並列計算機となっているので，プログラマは，自分のプログラムの並列化したい部分だけを並列計算機用のプログラムとして書き直し，データを機能メモリの所定の位置に配置し，その並列プログラムを呼ぶだけでメモリの内容が並列に書き変わって，結果が得られる．並列化できる部分だけを順次並列プログラムに書き換えていくだけで，並列処理が行えるので，既存のプログラムの変更の手間も少なくてすみ，ソフトウェアやアルゴリズム資産の継承の意味からも有利である．

2.2 VLSI 技術と機能メモリによる超並列計算

VLSI 技術を用いた超並列計算機のアーキテクチャは，種々の VLSI の特性に起因する制約を考慮して設計されなければならない．ここでは，機能メモリの超並列計算機構としての優越性を VLSI による実現の立場から整理する．

システムの大規模化にともなう設計コストの上昇は，並列計算機構の大規模化にとって重大な問題である．VLSI では，設計コストと生産されるチップの数に最終的なチップのコストが大きく影響されるので，できるだけ少ない種類のチップを大量に使って大きなシステムを組み上げることが有利となる．チップのレベルでも，チップ内の機能ブロックレベルでも，同じ構造の回路を規則的

に接続することが設計コストの大幅な低減につながるので，この点を考慮したアーキテクチャ設計が重要となる．機能メモリは，同じ単純なメモリセルの規則的な配列であり，また，アドレスバスとデータバスによる単純なチップ間結合で，同一のチップを大量にならべて複数のチップで大規模なシステムが簡単に構成できるという性質をもっており，大規模化にともなうコスト上昇を最小限に抑えることができる．

これまでにも，多くの並列計算機アーキテクチャが提案されてきたが，その多くは比較的短い期間で陳腐化している．その原因は，アーキテクチャの提案時に予測した以上に集積度が増大したことにより，集積度の向上がもたらす質的な変化が当初のアーキテクチャ設計に考慮されていなかったことと，集積度の向上により，逐次型のアーキテクチャが十分に性能を向上させたことにある．この中で，半導体メモリは，過去 20 年間の集積度の向上に対して，アーキテクチャの変化を必要としなかったほとんど唯一の例といってもよい．これは，メモリのもつ1次元の語の配列という単純な構成に起因している．将来の並列計算機アーキテクチャの設計においては，今後もしばらくは続くと思われる微細化により，集積度が大幅に向上したときにも，設計変更が少なくてすむアーキテクチャを考えるべきである．機能メモリは，従来の RAM や ROM と同じように集積回路化のメリットを最大限享受でき，プロセス技術の進歩をそのまま並列度の増加に反映させることができるアーキテクチャとして，今後の集積度の向上に対しても陳腐化しない構造の一つであるといえる．

並列計算機アーキテクチャの設計においては，プロセッサ間の通信とプロセッサ内の計算の双方を同時に考慮することが重要である．単位面積あたりの素子数は，3年で4倍というような驚異的な速さで増加しているが，チップと外部を結ぶピンの数は，高々数百ピンしか利用できない．すなわち，VLSI は，一般に計算空間が広がっても入出力バンド幅は制限されるという性質をもっている．このため，チップの入出力バンド幅の制限を十分に考慮したプロセッサ間通信網の設計が重要となる．とくに，集積度が向上して，1チップ上に集積される回路の規模が大きくなっても，ピンの数を増やさないでよいアーキテクチャは重要である．半導体メモリは，回路規模が2倍になっても入力としてのアドレス情報が1ビットしか増えない構造をもっており，ピン数制限への対応としても有利である．機能メモリの設計においてもこの点には十分な配慮が必要である．

超並列計算機アーキテクチャの設計において，現在の VLSI 技術が2次元平面上のレイアウトに縛られていることは，大きな制約である．超並列計算機アーキテクチャの結線構造として，種々の構造が提案されているが，1次元あるいは2次元の配列のように，2次元レイアウトとの整合性が良いものはきわめて限定される．3次元配列やハイパキューブ構造は，ピン数制限とレイアウトの両面から，VLSI 技術との整合が難しくなる構造である．一方，1次元あるいは2次元の配列の問題点は，遠距離への通信(大局的な通信)に時間がかかる点である．大局的な通信にはバスを利用するのが一つの有効な手段である．機能メモリは，2次元格子状のセル配列をワード線，ビット線という互いに直交する2種類のバスで結合しており，レイアウト的にも理想的な構造をしている．

このように，機能メモリは，VLSI 技術との整合性がきわめてよく，機能メモリを基本とした並列計算機アーキテクチャは，今後の数百万プロセッサ規模の超並列計算機の構成法として有力な候補であるといえる．

3. 機能メモリの並列計算能力

3.1 FRAM

高木らは，機能メモリを形式化した計算モデルを定義し，その並列計算能力に関する理論的な研究を行っている[9]．FRAM (Random Access Machine with a Functional Memory) は，通常の逐次型計算機のモデルである RAM (Random Access Machine) に機能メモリを付加したものである．FRAM では，機能メモリの機能は最小限に限定されているが，その容量は無制限という仮定を設けている．すなわち，最小限の機能しかもたない機能メモリでも，その容量が十分であれば大きな計算能力をもつことが示される．

FRAM は，**図-2** のように，通常の RAM (1本の入力テープ，無限個のレジスタ，プログラムカウンタ，有限長のプログラムからなる) と機能メモリおよび検索結果レジスタで構成される．機能

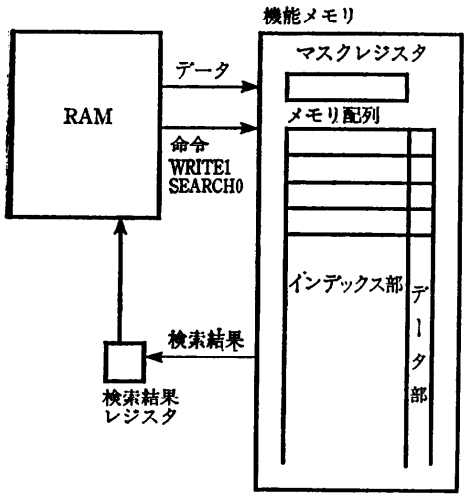


図-2 FRAM

メモリはマスクレジスタとメモリ配列からなる．メモリ配列の各語は，アドレスに対応するインデックス部と1ビットの書き込み可能なデータ部からなる．機能メモリに与えられるデータに対し，マスクレジスタでマスクされたデータが検索データとなり，各語のインデックス部と一致検索が行われる．この機能メモリに対する命令には二つある．一致検索を行って一致した語のデータ部にのみ1を並列に書き込むWRITE 1命令と，一致した語の中にデータ部が0のものが存在するか否かを調べるSEARCH 0命令である．SEARCH 0命令の結果は検索結果レジスタ（1 bit）に入り，RAMから参照することができる．

FRAMでは機能メモリをオラクル（神託）チューリング機械のオラクルのように使って，クラスNPやco-NPの問題を多項式時間で解くことができることが示される．基本的には，機能メモリで論理式の充足可能性判定問題が直接解けることを利用している．一般に，クラスNPに属する問題に対するオラクルをもつオラクルチューリング機械で多項式時間でとける問題のクラスΔ_2^pと，FRAMで多項式時間でとける問題のクラスとが一致する[9]．すなわち，FRAMは非決定的な計算によって並列化できる問題を現実的に並列に計算するモデルであるといえる．

この結果は，CAMを初めとする機能メモリの多くが備えている並列書き込みと一致検索だけで，並列計算能力が大きく向上することを示している（本特集1.および2.参照）．さらに，機能メモリに対する命令を強化して現実の種々の機能メモリの変形を考えても，本質的にはその計算能力は変わらないことも導かれる[10]．

3.2 機能メモリを利用した並列アルゴリズム

機能メモリを利用した並列アルゴリズムの設計手法には大きく分けて二つの方法がある．

一つは，逐次処理のアルゴリズムの中で用いられる種々のデータ構造の中で，計算コストがかかるデータ構造を機能メモリで直接実現する方法である．この方法では，機能メモリは，データ構造に対する各種の操作を並列に実行する専用計算機と考えられる．このデータ構造を直接実現する方法は，古くから機能メモリの研究のモーティベーションの一つにもなっており[1]，プライオリティキューや辞書，UNION-FIND問題用のデータ構造などの効率的な実現が知られている[8),11),12]．たとえば，プライオリティキューは，一致検索を利用した最大値検索によってデータ挿入がO(1)，最大要素の削除がO(k)（kは語のビット長）で行える．機能メモリを用いたソーティングやデータベース処理[13]，自然言語処理[14]，図形処理[15]，文字列処理[16]，論理型プログラム処理[11),17]などへの応用もデータ構造の直接実現とみなすことができる．この場合，このデータ構造を利用する上位のアルゴリズムからは，機能メモリの利用はまったく透明である．すなわち，これまで逐次処理で実現していたデータ構造をそのまま機能メモリで置き換えるだけで，上位のアルゴリズムはほとんど変更なしに高速化ができる利点がある．

もう一つの方法は，直接機能メモリ上の並列アルゴリズムを設計する手法である．すなわち，機能メモリをSIMD型の並列計算機構とみて，これに適した並列アルゴリズムを開発する方法である．この方法では，メモリ内の語（プロセッサ）間の通信や外部との通信のバンド幅がきわめて狭いことに十分な配慮が必要となる．できるだけ相互のデータ依存度が少ないアルゴリズムやデータを動かさないアルゴリズムの開発が必要である．4.で述べる組合せ最適化問題に対する並列虱潰しアルゴリズム[4),5)]やテキストを動かさないパターンマッチ[12]は，機能メモリに適した並列アルゴリズムの例である．これらのアルゴリズムは基本的には，データ並列アルゴリズム[23]のうちで特に通信に強い制限を加えたものと考えられ，ベクトル

計算機に適したアルゴリズムと共通点をもっている．各種算術演算[6),7),18)]，動的計画法[12)]，論理シミュレーション[19)]，故障シミュレーション[20)]，神経回路網のシミュレーション[21)]，画像処理[22)]などのアルゴリズムも知られている．

4. 機能メモリ型並列プロセッサアーキテクチャ FMPP

4.1 FMPP の構成[4),5)]

CAM (Content Addressable Memory) は，従来から広く研究されている代表的な機能メモリであり，記憶内容によるアドレス指定が可能であるという特徴をもつ．その提案は 1950 年代にまで遡るが，現実に大きな容量をもつ CAM が開発されたのは，ここ数年のことである．ここでは，CAM を基本回路として用いた機能メモリ型並列プロセッサアーキテクチャ FMPP (Functional Memory type Parallel Processor) について紹介する．

図-3 に CAM 型 FMPP のブロック図を示す．FMPP は，アドレスバス，データバス，命令入力，選択語検出線の 4 種の入出力ポートをもっている．アドレスバスは，n ビットの入力出力兼用のバスで，ホスト計算機のアドレスバスに接続される．アドレスバスの内容はアドレスレジスタにラッチされる．データバスは，m ビットの入力出力兼用のバスで，ホスト計算機のデータバスに接続される．データバスの内容はデータレジスタにラッチされる．アドレスレジスタとデータレジスタの内容は，マスクレジスタの内容でマスクされてメモリアレイ部に送られる．メモリアレイ部は最大 2^n 語からなる．各語は，n ビットの ROM 部と m ビットの RAM 部，論理演算部およびフラグ部をもつ．ROM 部は，アドレスデコーダを兼ねており，各語のアドレスを記憶している．RAM 部は，書き込みが可能なメモリセルで，各語をプロセッサとみたときの内部レジスタに相当する．論理演算部は 1 ビットの ALU に相当し，ここでは単純な論理演算だけを仮定する．フラグ部は，論理演算部の演算結果を格納するレジスタに当たる．この回路構成は，ほとんど既存の CAM（本特集 1., 2. および 3. 参照）と変わらない．

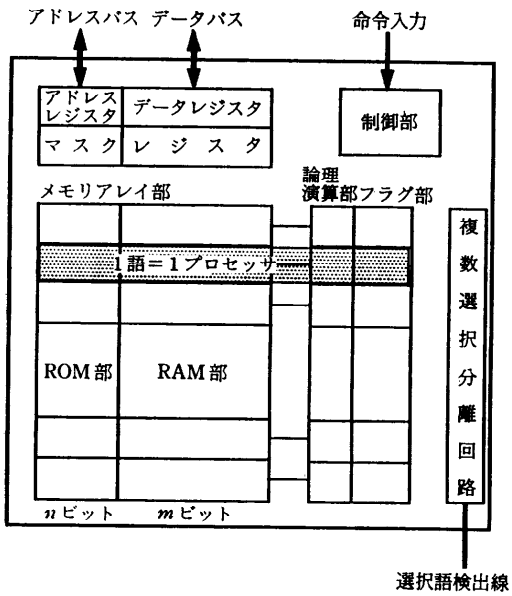


図-3 CAM 型 FMPP のアーキテクチャ

FMPP の命令の主なものを以下に示す．

REF <function, address, data>：入力されたアドレスとデータにマスクレジスタの内容でマスクをかけ，各語に放送し，一致検索を行って，完全に一致した語は論理値 1 を，一致しない語は論理値 0 を論理演算部に送り，フラグの内容と function で指定した演算を行って，フラグに結果を格納する．

READC：フラグの内容が 1 である語を検索し，そのアドレスと RAM 部の内容を出力する．複数の語が条件を満たす場合は，複数選択分離回路によってその中の 1 語が選ばれる．

WRITEC <data>：フラグの内容が 1 である語に対し，並列に書き込みを行う．書き込みデータはマスクレジスタの内容でマスクされる．

READA <address>：通常のアドレス指定による読み出し．

WRITEA <address, data>：通常のアドレス指定による書き込み．

MASKSET <data>：マスクレジスタへの値のセット．

SHIFT <direction>：フラグの内容の 1 語シフト．

通常のメモリに対しての読み出し／書き込みのほかに，一致検索 (REF)，検索読み出し (READC)，並列書き込み (WRITEC)，隣接シフト (SHIFT)，などが加わっている．特に，一致検索と並列書き込み機能が，FMPP の SIMD 型並列計算機としての基本操作となる．これらの機能は既存の CAM でもすでに実現されている．このアーキテクチャ

では，各語が1ビットの演算器をもつプロセッサであり，それが語数分並んでいるものと考えられる．複数のチップによる構成も，通常のメモリと同じように実現できる．現在の集積回路技術で，1チップで64kビットから256kビットが実現できると考えられ，複数チップで数百万プロセッサも実現できる可能性がある．

これまでに，NTTで試作された実験機（32ビット4096語）[17]や電子技術総合研究所の意味ネットワークマシンIXM2[18]などの上で上記のFMPPとほぼ同じ機能をもつ並列計算機の実験が行われている[12),19)]．

4.2 FMPP上での並列アルゴリズム[12)]

ここでは，FMPP上での並列アルゴリズムとその性能評価の例を示す．

まず，基本的な諸演算に関して，主な演算とその実行時間（命令ステップ数）を表-1に示した．外部の値を各プロセッサ内のデータ(data 1)に一斉に演算する場合と，各プロセッサ内のデータ同士(data 1とdata 2)を演算する場合とがそれぞれ示されている．外部からのデータの放送は，一斉に行えるが，1語内のデータの移動や演算および語間の通信はすべてビット直列となる．したがって，ほとんどの演算がビット直列となり，演算時間はデータ長に依存する．しかし，各語は並列に動作するため，プロセッサ数(データ数)には依存しない．図-4に，CAM型FMPP上での並列加算（各語ごとに内部にもつ3ビットの2数を加算する）のプログラム例を示す．ビット直列演算であるが，プロセッサ数が大きく加算すべきデータが多いと全体の性能は高くなる．

文字列のパターンマッチングは，FMPPに最も適した問題の一つである．逐次型の処理では，テキストをスキャンしなければならないので，テキスト長に比例した時間がかかる．FMPPでは，テキストをFMPP中に格納し，パターンに対応するオートマトンを各語において，それらがテキスト内を一斉にシフトしながらスキャンしていくアルゴリズムが使える．図-5の例は，パターン 'abc' を見つける例で，(6)でフラグに1が立っている語がマッチングした部分の最後尾を表している．

表-1 FMPP上の基本演算の実行時間

基本演算	ステップ数
データ転送 (外部 → *data 1*)	1
データ転送 (*data 1* → *data 2*)	$3k+4$
データ転送 (*word i* → *word i+j*)	$(j+1)k$
加算 (*data 1*+外部 → *data 1*)	$5k$
加算 (*data 1*+*data 2* → *data 2*)	$9k-4$
インクリメント (*data 1*+1 → *data 1*)	$3k$
比較 (Compare *data 1* with 外部)	$2k+1$
比較 (Compare *data 1* with *data 2*)	$5k+3$
最大値検索	$3k+1$
論理演算	
k-入力 AND, NAND, OR, NOR	7
k-入力 EXOR, EXNOR	8

k: データの語長

```
Add: data 2←data 1+data 2.
MASKSET <1,001,001,10>  /*addition of the LSB*/
REF thru <1,--1,--1,0->
WRITEC   <1,--1,--0,1->
REF thru <1,--1,--0,0->
WRITEC   <1,--1,--1,0->
MASKSET <1,010,010,10>  /*addition of the second LSB*/
REF thru <1,-0-,-0-,1->
WRITEC   <1,-0-,-1-,0->
REF thru <1,-0-,-1-,1->
WRITEC   <1,-0-,-0-,1->
REF thru <1,-1-,-1-,0->
WRITEC   <1,-1-,-0-,1->
REF thru <1,-1-,-0-,0->
WRITEC   <1,-1-,-1-,0->
MASKSET <1,100,100,10>  /*addition of the MSB*/
REF thru <1,0--,0--,1->
WRITEC   <1,0--,1--,0->
REF thru <1,0--,1--,1->
WRITEC   <1,0--,0--,1->
REF thru <1,1--,1--,0->
WRITEC   <1,1--,0--,1->
REF thru <1,1--,0--,0->
WRITEC   <1,1--,1--,0->
```

図-4 FMPP上の並列加算プログラム

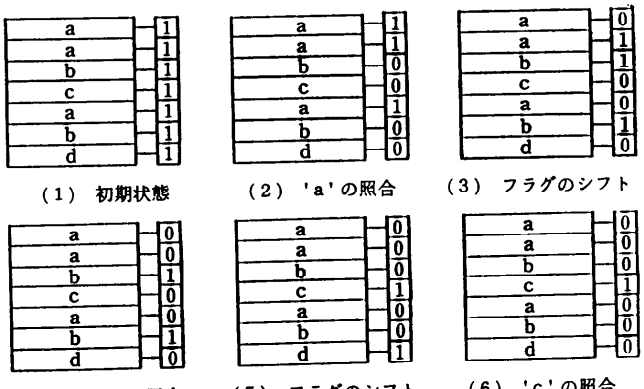


図-5 FMPP上のパターンマッチング

このため，完全一致検索の場合，テキスト長には無関係にパターン長のみに比例した時間で処理が終了する．一般にテキスト長はパターン長よりも十分に大きいのでこの効果はきわめて大きい．**表-2** に，逐次アルゴリズムと FMPP（理想的なアーキテクチャと NTT の実験機）における処理時間を比較している．

NP 困難なクラスに属する組合せ最適化問題も FMPP の処理に適した問題である．例として，ナップサック問題への応用を示す．基本的なアルゴリズムは，虱潰し法の並列化 (PES: Parallel Exhaustive Search) である．**図-6** にアルゴリズムとその動きを示す．アドレスの各ビットをそれぞれの荷物に対応させ，各語を荷物の部分集合の一つに対応させる．荷物の重さ (w_i) を順次放送し，その荷物を含む語で並列加算を行う (Step 1)．総重量制限 (C) を放送し，並列比較を行い，重量制限を越える語を無効化する (Step 2)．荷物の価格（p_i）を順次放送し，その荷物を含む語で並列加算を行う (Step 3)．無効化されていない語の中で最大値をもつ語を最大値検索で求めると最適解が得られる (Step 4)．2^n 語を用いて n 個の荷物に対するナップサック問題が解ける．現在の集積回路技術では，数十メガビットのシステムまでが限界である．ナップサック問題でも荷物の数が 16 から 20 程度までしか扱えないと考えられる．そこで，逐次型のアルゴリズムと組み合わせて用いる方法が考えられる．**図-7** に，分枝限定法 (B&B)，逐次型虱潰し法 (SES)，およびそれらと PES（$n=16$ までが取り扱える）を組み合わせたときの計算時間を示す．高速化の効果が現れているのが確認できる．

このほか，FMPP 上での，各種データ構造の実現[12]，動的計画法[12]，論理シミュレーションアルゴリズム[19]，故障シミュレーションアルゴリズム[20]，神経回路網のシミュレーション[21]などが報告されている（本特集 5. から 8. 参照）．

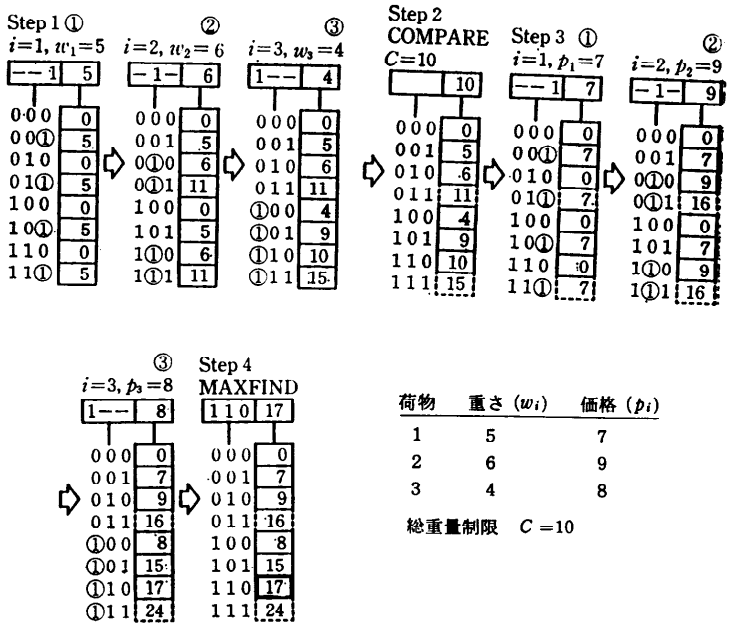


図-6 FMPP 上の並列虱潰し探索

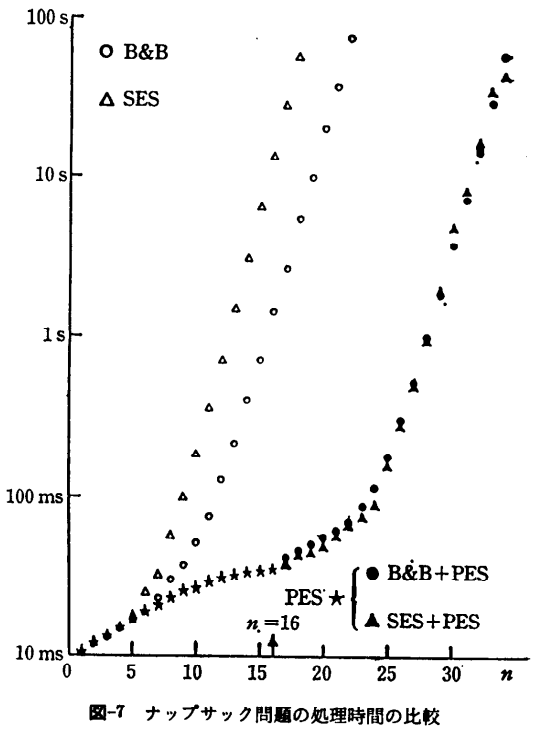


図-7 ナップサック問題の処理時間の比較

表-2 FMPP 上での文字列のパターンマッチの処理時間

テキスト長	逐次型アルゴリズム (25 MIPS CPU) (μs)	理想的な FMPP (μs)	実験機による実験結果 (μs)
256	76.8	4	19.8
1K	307.2	4	19.8
4K	1,228.8	4	19.8
16K	4,915.2	4	—
1M	314,572.8	4	—

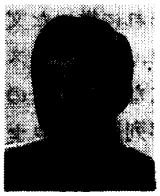
5. あとがき

機能メモリは，集積回路技術の進歩を直接的に生かして，プロセッサ数を無限と考えてよい並列計算機構を構成する最も現実的なアーキテクチャの一つである．現在の技術で，数百万プロセッサを構成できるほぼ唯一のアーキテクチャといえる．我が国の優れた半導体メモリ技術を生かして，機能メモリをベースとした本格的な超並列計算システムが実現されることを期待したい．今後の研究課題としては，機能メモリによる並列計算に向いた「データを動かさない並列アルゴリズム」の開発，可変長語への対応などがあげられる．

参考文献

1) Kohonen, T.: *Content-Addressable Memories*, 2nd ed., Springer Series in Information Science, Springer-Verlag (1987).
2) Parahami, B.: *Associative Memories and Processors: An Overview and Selected Bibliography*, *Proc. IEEE*, Vol. 61, No. 6, pp. 722-730 (1973).
3) Yau, S. S. and Fung, H. S.: *Assosiative Processor Architecture—A Survey*, *Comput. Surv.*, Vol. 9, No. 1, pp. 3-27 (1977).
4) Yasuura, H., Tsujimoto, T. and Tamaru, K.: *Parallel Exhaustive Search for Several NP-Complete Problems Using Content Addressable Memory*, *Proc. of 1988 IEEE Int. Symp. on Circuits and Systems*, pp. 333-336 (1988).
5) 安浦，辻本，田丸：組み合わせ問題に対する機能メモリ形並列プロセッサアーキテクチャ，電子情報通信学会論文誌，Vol. J-72-A, No. 2, pp. 222-230 (1989).
6) Scherson, I. D. and Ruhman, S.: *Multi-Operand Arithmetic in a Partitioned Associative Architecture*, *J. Parallel and Distributed Computing*, Vol. 5, pp. 655-668 (1988).
7) Scherson, I. D. and Ilgen, S.: *A Reconfigurable Fully Parallel Associative Processor*, *J. Parallel and Distributed Computing*, Vol. 6, pp. 69-89 (1989).
8) Blair, G. M.: *Content Addressability: An Exercise in the Semantic Matching of Hardware and Software Design*, *IEE Proceedings*, Vol. 136, pt. E. No. 1, pp. 41-47 (1989).
9) 高木，武永，矢島：メモリ型並列計算モデルとその計算能力—スーパーコンピュータへの第3のアプローチ，情報処理学会論文誌，Vol. 31, No. 11, pp. 1565-1571 (1990).
10) 武永，高木，矢島：連想メモリによるメモリ型並列計算モデルとその能力，電子情報通信学会コンピュテーション研究会資料，COMP 89-118 (1990).
11) 大久保，安浦，高木，矢島：機能メモリを利用したハードウェア向き単一化アルゴリズム，情報処理学会論文誌，Vol. 28, No. 9, pp. 915-922 (1987).
12) 安浦，渡辺，左達，田丸：CAM を用いた機能メモリ型並列プロセッサ上での並列アルゴリズム，情報処理学会アルゴリズム研究会資料 21-2 (1991).
13) Falkoff, A. D.: *Algorithms for Parallel-Search Memories*, *J. ACM*, Vol. 9, No. 5, pp. 488-511 (1962).
14) Kitano, H. and Higuchi, T.: *Massively Parallel Memory-Based Parsing*, Proc. of IJCAI-91 (1991).
15) 鈴木，大附：連想メモリを用いた VLSI 設計用図形処理ハードウェア，電子情報通信学会論文誌，Vol. J 72-A, No. 3, pp. 550-560 (1989).
16) Takahashi, K. Yamada, H. and Hirata, M.: *A String Search Processor LSI*, *Journal of Information Processing*, Vol. 13, No. 2, pp. 185-189 (1990).
17) 長沼，小倉：連想メモリを用いた Prolog マシンの実現とその評価，電子情報通信学会論文誌，Vol. J 73-D-I, No. 11, pp. 856-863 (1990).
18) 国分，樋口，古谷：意味ネットワークマシン IXM における並列連想記憶，情報処理学会計算機アーキテクチャ研究会資料，80-9 (1990).
19) 安浦，渡辺，左達，田丸：機能メモリ型並列プロセッサ FMPP 上での論理シミュレーション，電子情報通信学会計算機システム研究会資料 CPSY 90-94 (1991).
20) 石浦，矢島：連想記憶を用いた線形時間故障シミュレーション，第4回回路とシステム軽井沢ワークショップ論文集，pp. 63-68 (1991).
21) 小野寺，竹下，田丸：Kohonen ネットワークのハードウェアアーキテクチャ，電子情報通信学会集積回路研究会資料 ICD 89-146 (1989).
22) Nakano, A., Yasuura, H. and Tamaru, K.: *Functional Memory Type Parallel Architecture for Image Processing*, *Proc. of VLSI 89*, pp. 329-338 (1989).
23) 斉藤，馬場編：「超並列マシンとその応用」特集，情報処理，Vol. 32, No. 4, pp. 347-413 (1991).

(平成3年7月10日受付)



安浦 寛人 (正会員)

1953年生．1976年京都大学工学部情報工学科卒業．1978年同大学院修士課程情報工学専攻修了．1980年より京都大学工学部助手．1986年より同電子工学科助教授．1991年より九州大学大学院総合理工学研究科教授．並列アルゴリズム，並列計算機アーキテクチャ，論理設計，VLSI 用 CAD の研究に従事．1987年度電子情報通信学会，1990年度本学会論文賞受賞．IEEE, ACM, 電子情報通信学会，ソフトウェア科学会，LA シンポジウム，EATCS 各会員．