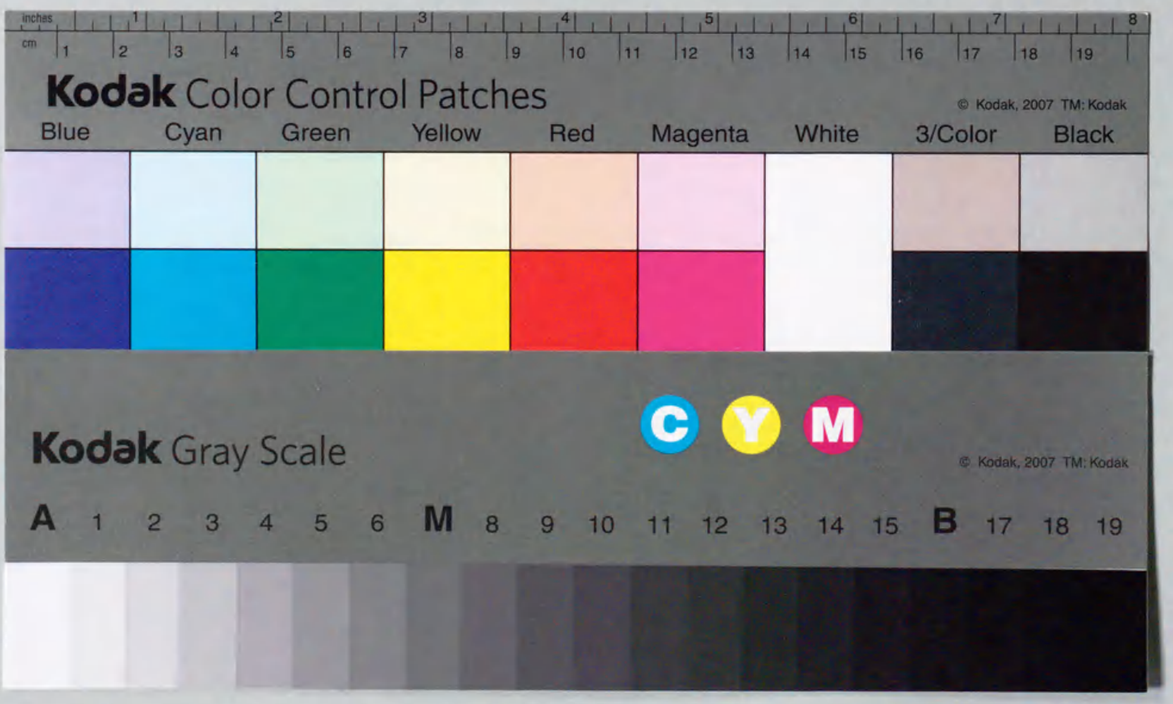
セルラ配列型ＳＩＭＤプロセッサの
構成法と応用に関する研究

近藤　利夫



セルラ配列型ＳＩＭＤプロセッサの構成法と応用に関する研究

近藤　利夫

第１章 序論

１．１ 研究の背景と目的

逐次型の汎用プロセッサは、誕生以来長足の進歩を遂げ、この２０年間で２桁から３桁の小型化、高速化が達成されている。それにもかかわらず、より高性能な並列プロセッサに対して、今日、大きな期待が寄せられている。これは、常に逐次プロセッサの処理能力を何倍も上回る超高速処理の要求が存在するばかりか、性能向上が進めば進むほど応用範囲、適用分野が広がり、その結果、性能向上あるいは小型化、経済化の要求がますます強まり、従来からの逐次プロセッサの改良だけで対応できる見通しが立たなくなって来ているからである。

この逐次プロセッサの性能限界の克服をねらった並列プロセッサの研究は、種々のレベルで古くから行われてきた。中でも、多数のプロセッサを用いるプロセッサ配列型並列プロセッサは、原理的に非常に高い並列度が達成できるため、超高速処理の実現が可能な技術として位置付けられてきた[1,2,3,4]。しかし、並列度が上がる分、プログラム開発が困難になるばかりか、ハードウェア規模も著しく大きくなるため、なかなか商用レベルのシステムとして普及するには至らなかった。

一方、プロセッサのハードウェアを支えるＬＳＩ製造技術の進歩は著しく、２年で２倍（論理ＬＳＩ）ないし3倍（メモリＬＳＩ）のペースで着実に集積規模が増加している。その結果、現在では論理ＬＳＩで１０００万トラジスタを越えるもの[5]まで実現されるに到っている。しかし、ＣＡＤ技術は、一時この進歩に追い付けず、１９７０年代から１９８０年代にかけて、論理ＬＳＩの分野で設計ネックが顕在化し、製造技術の進歩を活かし切れない傾向にあった。この現状を打開すべく、設計容易な論理ＬＳＩの構成法が求められて、１９８０年代には構成の単純化を追究したＲＩＳＣ型逐次プロセッサ[6]と単純な演算ユニットの規則的配列からなるシストリック配列[7]の研究が期を同じくして活発化した。

このような状況のもとで、本研究は、ＬＳＩの製造技術の進歩を活かしきれる並列プロセッサの実現をめざし、シストリック配列に匹敵する単純性と規則性を備えたセルラ配列型のＳＩＭＤプロセッサを取上げ、その構成法、設計法、処理アルゴリズムを追究する。そして、特定の応用分野で、従来不可能であったような超高速処理と経済性の両立に道を開くとともに、処理対象に適合させる専用ハードウェア並みの性能／価格を備える並列プロセッサシステムの実現を目指す。

１．２　セルラ配列型ＳＩＭＤプロセッサの位置付けと問題点

１．２．１　セルラ配列型ＳＩＭＤプロセッサの位置付け

ＳＩＭＤ（Single instruction stream, multiple data-stream）プロセッサは、並列演算部が制御部から配られる単一の命令で一括制御される同一構成のプロセッシングエレメント（ＰＥ：Processing Element）配列で構成されることから、最も単純な構成の並列プロセッサの１つといえる。その中でも、処理語長の短いＰＥを隣接結合ネットワークで結ぶタイプのＳＩＭＤプロセッサは、並列演算部が極めて単純で規則的な構成になることから、ＰＥが細胞にたとえられて、セルラロジック配列、セルラ配列プロセッサ等と呼ばれてきた[8,9,10]。

この単純性、規則性ゆえに、その原形は３０年以上も前に提案され[1]、１ビット構成のＰＥを用いるタイプを中心にいくつものシステムが開発されてきた[2,3,11,12,13,14,15]。１ビット構成のＰＥが好まれたのは、構成が最も単純であるために大規模なプロセッサ配列を比較的容易に実現できるばかりか、ビット演算を語長分繰り返すこと（ビット直列演算）によって短語長から長語長の演算まで柔軟に対応できるからである。それでも、大規模ＬＳＩの利用なくしては、コストに見合った性能が得られなかったのか、広く普及するには至らなかった。ところが、近年のＬＳＩ技術の進歩が状況を一変させた。１チップに多数のＰＥを搭載したＶＬＳＩを活用した大規模なシステムが実現され、科学技術計算からＡＩやニューラルネットの応用にまで際だった性能が発揮されることが示された[16,17]。その結果、現在ではセルラ配列型のＳＩＭＤプロセッサがスーパーコンピュータの一角を占める代表的なＳＩＭＤプロセッサとして位置付けられるに至っている。

１．２．２　ＭＩＭＤプロセッサとの対比[18]

ＳＩＭＤプロセッサを議論する上で常に対比されるのが同じくプロセッサ配列型の並列プロセッサであるＭＩＭＤ（Multiple instruction stream, multiple data-stream）プロセッサである。

ＭＩＭＤプロセッサは、ＰＥ毎に個別に制御部を有しているためＳＩＭＤプロセッサに比べると並列処理の融通性が高く、応用の範囲が広いという利点を有している。また、１チップマイクロプロセッサを組込むことで、ＰＥが比較的容易に構成される利点もある。このため、最近では汎用性が強く要求されるスーパーコンピュータの分野で、これまで主流であったCRAYで代表されるベクトルプロセッサを凌ぐ勢いにある[19]。とはいえ、本研究の目的である専用ハードウェアを越えるような高性能／コストの実現、

あるいは既存の小型スーパーコンピュータ並みのコストでの超並列処理の実現は、現状では困難である。これは、ＭＩＭＤプロセッサでは、各ＰＥが個別にシーケンス制御部を必要とする分ハードウェア規模が大きく、１チップに複数のＰＥを規則的に並べるＬＳＩ向き構成により、並列度をかせぐことが難しいからである。従って、同一ハードウェア規模当たりの並列度は、ＳＩＭＤプロセッサより低くなってしまう。その上、各ＰＥを独立あるいは非同期に動作させるためにプログラミングやデバッグが複雑化し、これを少しでも緩和するために、本格的なプログラミング支援ツールを開発しなければならなくなる。

このＭＩＭＤプロセッサに比較すると、ＳＩＭＤプロセッサは、同じプロセッサ配列型の並列プロセッサでありながら、相反する性質を備えている。すなわち、ＭＩＭＤプロセッサとは異なり、個別にシーケンス制御部を設ける必要がないために、ＰＥ構成は逐次型マイクロプロセッサに比べるとかなり単純化される利点がある。特に、セルラ配列型のＳＩＭＤプロセッサではネットワークが隣接結合をベースとした単純な構成で済むために、レジスタ付きＡＬＵにＰＥ間転送パスを付加する程度の極めて単純な構成でＰＥが実現される。従って、シストリック配列同様、同一構成のＰＥを複数、規則的に配列するチップ構成が可能となり、最近のＬＳＩの集積能力をほぼ完全に活かすことが可能になる。また、ＳＩＭＤプロセッサは並列演算部全体が同期して動作する１つの演算器とみなすことができるので、プログラミングやデバッグについても、従来の逐次プロセッサに近い感覚で行うことができる利点がある[20]。

ＳＩＭＤプロセッサの欠点は、先に述べたようにＭＩＭＤプロセッサほど並列処理に対する融通性が高くなく、このために高速化可能な処理が定型的な配列演算に限られる傾向にあることである。この欠点は、並列演算部全体を単一命令で一括制御するＳＩＭＤ型の制御方式そのものに起因しており、ＳＩＭＤプロセッサの利点とは裏返しの関係にある。従って、この欠点を根本的に解消することは不可能といえる。それでも、それを少しでも改善しようと、多少の並列度低下を伴っても各ＰＥにＭＩＭＤ的な自律制御機能を付加することで汎用性を高める努力が、これまで種々なされてきた[20]。今後も、この汎用性向上と並列度向上との両立の問題は、ＳＩＭＤプロセッサにおけるアーキテクチャ設計上の最重要課題の一つであり続けるものと考えられる。

ＳＩＭＤプロセッサとＭＩＭＤプロセッサとの間で比較すべき、もう一つのアーキテクチャ上の重要なポイントに、ＰＥ間ネットワーク構成があげられる。これは、大規模化と遠隔ＰＥ間の通信性能の確保を両立しようとすると、構成次第ではプロセッサ配列自体を上回ってしまうほどＰＥ間ネットワークのハードウェア規模が大きくなるからで

ある。このため、ＰＥ間ネットワークの構成法は、ＰＥ間の高速転送とハードウェア規模低減の両立をめざして、階層バス、クロスバ、２次元隣接結合、ツリー、ハイパーキューブ、多段結合網等の多くのネットワークが、古くから研究されてきた[21]。

ＭＩＭＤプロセッサとＳＩＭＤプロセッサのネットワークの違いは、転送を分散的に制御する必要性の有無にある。すなわち、ＭＩＭＤプロセッサではランダムに発生する任意のＰＥ間の転送をサポートするために分散的な制御機構が必要なのに対し、ＳＩＭＤプロセッサでは転送をプロセッサ配列全体で同期的に行えるので、分散的な制御機構を必要としない。従って、その分ＳＩＭＤプロセッサのネットワークの方がハードウェア規模が小さくなるといえる。しかし、このようにいえるのはあくまでプロセッサ配列のサイズを同一と仮定した場合のことである。ネットワークのハードウェア規模がＰＥあるいはネットワークを構成する中継ノード間の結線の総数に強く依存することからすると、ＭＩＭＤプロセッサ以上の並列度をねらうＳＩＭＤプロセッサの方が、ＰＥ間ネットワーク増大の問題は、より深刻と言える。

なお、ここでは、もう一つの代表的な配列型の並列演算器であるシストリック配列との比較は行っていない。この理由は、シストリック配列が、一般に汎用性がほとんどなく、プロセッサというよりも専用の高速演算器であり、専用ハードウェアの構成要素にはなっても、並列プロセッサそのものではないと判断したからである。一部には、かなりの汎用性を備えたシストリック配列も実現されてはいるけれども、それらはＳＩＭＤプロセッサ、あるいはＭＩＭＤプロセッサの一種ともみなせ、ＳＩＭＤプロセッサとＭＩＭＤプロセッサの比較のみで十分と考えたからである。

１．２．３　セルラ配列型ＳＩＭＤプロセッサの問題点

セルラ配列型ＳＩＭＤプロセッサは、当然ＳＩＭＤプロセッサの一種であり、処理の柔軟性が低い、並列度が大きい分ネットワークのハードウェア規模低減の制約が強い等、抱えている問題自体が異なるわけでない。違いは、他のＳＩＭＤプロセッサに比べ、これらの欠点を補うための対策がとりにくいことにある。この理由は、問題解決に効果的だからといって、ＰＥ単体あるいはＰＥ間ネットワークの複雑化をまねくような構成を採れば、単純、規則的でかつ高並列なセルラ配列構成の利点そのものが損われてしまうことにある。ただし、どこまでの複雑さが許されるかについては、その時点でのＬＳＩの製造技術、設計技術、さらには目的とするシステムの位置付けによっても変わるので注意が必要である。結局、このような周囲の状況をみながら、いかにＳＩＭＤプロセッサの問題点を克服し、性能／コストの高い、あるいは並列度の大きいシステムを実現するかが、アーキテクチャ構成上のポイントとなる。

１．３　対象とするセルラ配列型ＳＩＭＤプロセッサの基本アーキテクチャ

１．３．１　代表的なＳＩＭＤプロセッサの概観

前節で述べたように、セルラ配列型ＳＩＭＤプロセッサの歴史は古く、その発展形としてＩＬＬＩＡＣ　Ⅳ[3]をはじめとする各種のＳＩＭＤプロセッサが開発されてきた。その結果、本研究の開始時点には、すでに近年のＳＩＭＤプロセッサの代表的なアーキテクチャが出そろっていた。本節では、これらのうち本研究を進めるに当たって特に参考としたプロセッサであるＩＬＬＩＡＣ　Ⅳ、グッドイヤＳＴＡＲＡＮ[11,14,15]、ＩＣＬ　ＤＡＰ[12]、ＣＬＩＰ－４[13]を概観し、セルラ配列構成を前提としたアーキテクチャの利害得失を明確にする。

ＩＬＬＩＡＣ　Ⅳは、現在のＳＩＭＤプロセッサのルーツとも呼ぶべきＳＩＭＤプロセッサであり、図1.1に示す単一段の２次元隣接結合ネットワーク、ＰＥ毎に個別のアドレス生成を可能とするレジスタ間接アドレシングによるデータアクセス、ＰＥ毎の個別の制御レジスタによるＰＥの活性制御等の現在のＳＩＭＤプロセッサを構成する上での基本技術のほとんどが組込まれている。しかし、ＰＥ自体は６４／３２ビットの浮動小数点演算、４８／２４／８ビットの固定小数点演算等をサポートする大掛かりなもので、本研究の目標とする複数ＰＥ搭載のチップを並べるだけのセルラ配列構成を実現するには程遠い複雑な構成をとっている。

ＩＣＬ　ＤＡＰは、商用化された数少ない汎用のＳＩＭＤプロセッサの一つである。ＩＬＬＩＡＣ　Ⅳとは単一段の２次元隣接結合ネットワークでＰＥ間を結合するＰＥ配列構成を採る点では共通ながら、４,０９６個の１ビット構成ＰＥからなる典型的なセルラ配列構成を採る点では対極的である。ＰＥの構成は極めて単純で、１チップに多数のＰＥを搭載することも比較的容易で、ＬＳＩ化によりセルラ配列構成の利点を引出すのに十分な構成と言える。

ＣＬＩＰ－４は、先駆的なセルラ配列型ＳＩＭＤプロセッサであり、実際に４ＰＥを搭載したＬＳＩで９６ｘ９６のＰＥ配列を構成している。ＰＥの構成は単純でＰＥ対応の局所メモリの容量も小さいが、セルラ配列型ＳＩＭＤプロセッサにおける伝搬演算機能の有用性を示した最初のシステムである。

グッドイヤのＳＴＡＲＡＮも商用化された数少ないＳＩＭＤプロセッサの一つである。極めて単純な構成の１ビットプロセッサ２５６個でＰＥ配列を構成する点ではＩＣＬ　ＤＡＰ、ＣＬＩＰ－４と共通である。違いは、２次元の隣接結合ネットワークの代わりに、一種の多段結合のネットワークであるフリップネットワークを装備している点と局所メモリの配列をＭＤＡ（Multi-Dimensional Access memory）構成としている点

にある。これらのフリップネットワークとＭＤＡとを用いることで、ＭＤＡ内の２次元配列データに対し行単位、列単位の両方向のデータ列に対するアクセス機能（２次元アクセス機能）を実現している。また、フリップネットワークは単体で置換やシフトも実現できる。このようにＳＴＡＲＡＮの構成は高い機能を提供できるものの、ＭＤＡの構成にはメモリのビット毎にアドレスを修飾できる必要がある、フリップネットワークの構成には多数のスイッチ、多数の遠隔ＰＥ間の結線が必要になる等、単一段の２次元隣接結合ネットワークに比べるとハードウェア規模はかなり大きくなってしまう。

１．３．２　研究の対象とする基本アーキテクチャ

（１）２次元セルラ配列構成

ＳＴＡＲＡＮで採用された多段の結合ネットワークは、機能の高さからは極めて魅力的である。実際、ＳＴＡＲＡＮが実現している２次元アクセス機能は、１次元モードで動作させるＰＥ配列で２次元の配列データを処理する場合に必須とも言える機能である。しかし、ＳＴＡＲＡＮの採用している多段の結合ネットワークは、ハードウェアに対する負担が大きく、大規模化あるいは小型経済化を第一義的な目標とするセルラ配列構成の考え方とは相容れない。

そこで、ＩＬＬＩＡＣ Ⅳ、ＩＣＬ　ＤＡＰ等にならい、Ｍ個からなるＰＥ配列に対してＰＥ間最大距離（ＰＥ間距離）が$O(\sqrt{M})$と比較的小さく、かつＬＳＩチップの平面構成に整合する点でＬＳＩ化に適している図１.１の２次元隣接結合型をベースとしたネットワークを採用する。

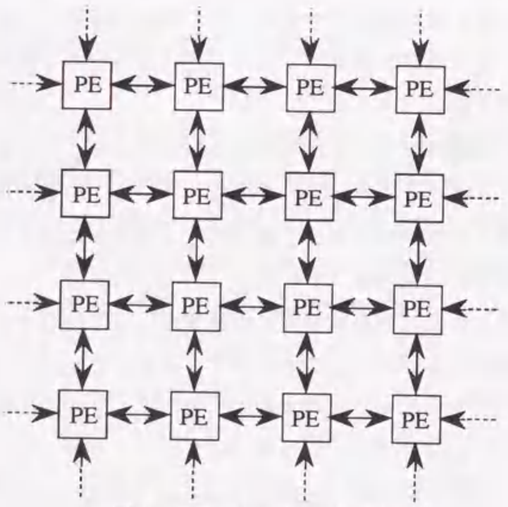


図１．１　ＰＥ配列の構成

ただし、$O(\sqrt{M})$のＰＥ間距離は大規模なＰＥ配列において遠隔ＰＥ間の転送を行うのに十分高速とはいえず、なんらかの補強手段を考える必要がある。ＰＥ配列を１次元モードで動作させる場合の２次元アクセス機能についても、多段の結合ネットワークを用いない単純な方法を検討する必要がある。

（２）修飾型ＳＩＭＤ制御

本研究で対象とするＳＩＭＤプロセッサの全体構成を図１．２に示す。図１．１を機能分担が明確になるように書き直し、制御部を付加した構成であり、上部の制御部で発生した信号により、その下に配置する２次元の隣接結合ネットワーク、ＰＥ配列、メモリ配列等からなる並列演算部全体を制御する典型的なＳＩＭＤ制御方式を採用する。また、この図に示されるように、メモリユニットは局所メモリとしてＰＥと１対１で接続する標準的な構成（各ＰＥが１個の局所メモリを個別にアクセスする構成）を採る。異なるメモリユニットとの通信は、ＰＥ配列の上側に示される隣接結合ネットワークを介して行う構成とする。

ＰＥ単位の自律的な制御手段としては、ＩＬＬＩＡＣ Ⅳ以来、各種のＳＩＭＤプロセッサに採用されてきた活性制御機構を採用する。この制御機構は、演算結果をデスティネーションのレジスタへ書込むか否かをＰＥ毎に設ける１ビットのレジスタ（活性

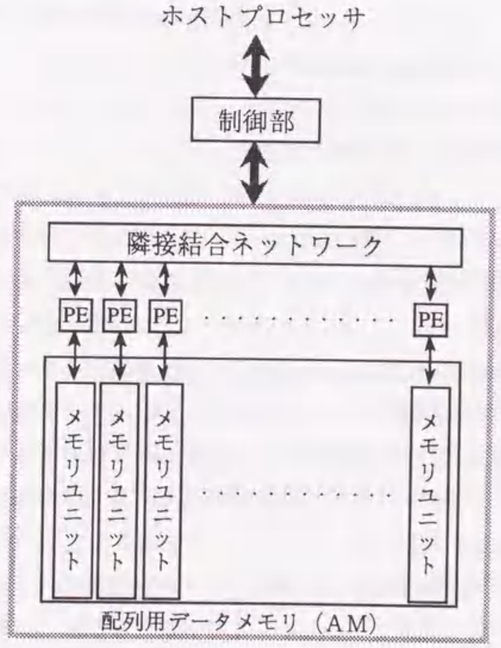


図１．２　セルラ配列型ＳＩＭＤプロセッサの構造

制御レジスタ）で制御することにより等価的にＰＥの活性制御を行うものである。さらに、制御部からの全ＰＥ共通の命令を、ＰＥ単位で修飾するための制御レジスタを個別に設け、ＰＥ配列の機能の向上をはかる。ただし、この制御レジスタによる命令修飾機能の一つであり、ＩＬＬＩＡＣⅣで採用されたレジスタ間接アドレシングによりＰＥ単位で独立のアドレスを生成する機能ついては、そのままの構成では搭載しない。ＰＥ単位に個別にアドレス端子が必要で、ＬＳＩやボードに対する負担が大きいからである。

１．４ 研究の課題とそれに対する解決方針

１．４．１ 目的達成上の課題

本研究では、１．１で述べた目的を達成するために、最近のＬＳＩの集積能力を完全に引出せるセルラ配列構成との両立が可能な以下の３つのＰＥ配列部構成法、

①効率的な遠隔ＰＥ間転送が可能なＰＥ間ネットワーク構成法

②２次元アクセス、ＰＥ単位の自律的なアドレシング機能等の多様なアクセスモードをサポートできる局所メモリアクセス機構構成法

③高速性と処理データの構造変化に適宜対応可能な適応的なプロセッサ配列構成法

を示すこと、さらにはこれらに基づいてＰＥ配列部を実現する場合の

④ＰＥ配列部用ＬＳＩおよびＳＩＭＤプロセッサシステムの構成法、設計法

⑤実際の応用でのＰＥ単体あるいはＰＥ配列部全体の演算法、処理法

を明らかにすることを主要な課題とする。

１．４．２ 課題解決のための基本的な方針

課題解決のために、２つの基本方針を採る。第１の方針は、ハードウェア化が性能向上に寄与しない部分については極力ソフトウェアで実現することである。これは、性能向上と開発工数低減の両立をはかるためである。この方針は、デバッグが困難でかつ設計・開発の一巡の期間（ＴＡＴ）の長いハードウェアに比べると、同一機能の実現ならば、ソフトウェアの方が開発工数が数段小さいことに基づく。ただし、多数のアプリケーションプログラムを開発しようとする場合、ハードウェアの機能が低いとその分、個々のプログラムの記述負担が増加して、全体の開発工数が増加する可能性については、コンパイラやマクロライブラリを必要に応じて構築することで回避できるものとして考えない。

第２の方針は、応用を特定の分野に限定し、その中で専用ハードウェア並みの高い性能／コストの確保を目指すことである。これが実現できれば、並列プロセッサの持つ処理アルゴリズムや処理方式の可変性により専用ハードウェアに対し十分優位にたてるからである。また、これは、逆に不特定多数の分野にまで適用できるだけの汎用性の確保までは目指さないことを意味している。セルラ配列構成との両立が難しいだけでなく、多くのアプリケーションの開発と、そのための汎用コンピュータ並みのプログラム開発環境の提供が必要となり、全体の開発負担が大きくなりすぎるからである。

特定の分野としては、文字認識とＬＳＩ－ＣＡＤの２つを選ぶ。身近な応用の中では、２次元の配列データを直接並列に演算することで処理が進められる割合が大きく、ＳＩＭＤプロセッサ向きの応用と考えられるからである。また、一方の文字認識処理が、ＯＡ機器に組込むために小型経済性を要求される分野、もう一方が大規模なＬＳＩを効率良く設計するために超高速処理が要求される分野と、セルラ配列型ＳＩＭＤプロセッサの能力をはかる最適な応用と考えられるからである。

１．４．３ ＰＥ配列部構成上の課題に対する具体的な取り組み方針

本研究の最重要課題であるＰＥ配列部構成法に対しては、セルラ配列構成の利点を享受するため、単純性、規則性を優先することとし、以下の３つの方針のもとに取り組む。

①直接サポートする算術演算機能は、短語長の固定小数点演算に限定する。

②ＰＥ間ネットワークの構成は２次元の隣接結合にＣＬＩＰ－４で用いられたＰＥ間の伝搬演算機構を組合わせたものとする。

③制御レジスタによる各ＰＥの個別の命令修飾機能、またはこれを利用した複数ＰＥの協調動作により対応できる範囲で、可変のＰＥ配列構造を実現する。

ここで、①の方針は、１．４．２で選定した応用に対しては短語長の固定小数点データ主体の処理で十分であるとの観点に基づき、ＰＥ構成の単純化を指向したものである。従って、低速なソフトウェアエミュレーションで実行することになる長語長の固定小数点演算、浮動小数点演算等の頻度の高い応用では、十分な高速性能が得られなくなる点に留意する必要がある。

②の方針は、ハードウェア規模低減をはかるために、遠隔ＰＥ間の転送パスの負担の小さいネットワークとすることを目指したものである。伝搬演算機構は、演算結果をＰＥ間で順次伝搬させることで結果の得られる演算（走査演算［scan］とも呼ばれる）を対象とした高速化機構であり、効率の良い遠隔ＰＥ間の転送演算を実現するセルラ配列構成向きの転送手段の１つと考えられる。しかし、高速化可能な転送・演算が限られるわけで、処理アルゴリズム自体をこれに適合させることによって実行性能の向上を図る必要が生じる。

③の方針は、ＰＥの構成をすべて同一とし、ＰＥ配列の規則性を確保するための方針

である。従って、ＰＥの構成上の工夫により、いかに柔軟な可変構造を実現するかがポイントになる。

１．５ 本論文の構成

本論文は１０章からなる。以下、各章の概要を示す。

第２章では、本研究で前提とした隣接結合ネットワークに対し、遠隔ＰＥ間の演算あるいは転送を補強する伝搬演算機構の構成方式として、バイパス方式とセレクティブ方式を提案し、高速化のためにツリー型の階層構成を導くとともに、そのハードウェア規模、速度性能を評価する。

第３章では、ＰＥ配列の一層の稼働率向上を目指し、全ＰＥ共通の命令を、ＰＥ毎に付加する制御レジスタで個別に修飾することによって実現されるＰＥ配列の各種の可変構造的な機能について述べる。さらに、前章で述べた伝搬演算と組合わせることによって、汎用化に有効なＰＥ配列内の矩形の局所領域毎の総和演算やＬＳＩ－ＣＡＤの論理シミュレーションの中核部分等が効率的に実行できることを示す。

第４章では、ＬＳＩやボードの構成に負担の小さいメモリアクセス機構として、基本のＳＩＭＤプロセッサにわずかな改造を加えるだけで実現される転送回転型メモリアクセス機構を提案し、その各種のアクセス法を明らにするとともに、従来型の２次元アクセス機構と比較して、端子数、アクセス速度等を評価する。

第５章では、セルラ配列構成の利点をすべて引出すことを狙って開発した１ビット構成の大規模セルラ配列型ＳＩＭＤプロセッサＡＡＰ（Adaptive Array Processor）、１ボードＳＩＭＤプロセッサＬＩＳＣＡＲ（LIne Scannable Cellular ARray processor）、およびこれらのＰＥ配列部の構成要素であるセルラ配列型ＬＳＩの設計、構成について述べる。また、それらの基本性能も評価する。

第６章では、小型経済化と高性能化の両立をはかるために、第４章までの成果を積極的に取入れるとともに、積和演算能力を強化した８ビット構成の１ボードＳＩＭＤプロセッサＬＩＳＣＡＲ２の設計、構成、基本性能評価等について述べる。

第７章では、身近な応用であるＬＳＩ－ＣＡＤの中から、特にセルラ配列型ＳＩＭＤプロセッサ向きと考えられるマスクパターンレイアウト設計の自動配線と、汎用コンピュータでは処理時間が大きくなりすぎる論理シミュレーションを取上げ、ＡＡＰ向きの処理アルゴリズムを提案するとともに、性能を評価する。

第８章では、従来の文字認識処理ハードウェアでは実現困難であった小型高速性と搭載アルゴリズムの可変性の両立を目指し、複数の文字認識処理の搭載を試みた結果について述べる。具体的には、手書き英数かな文字認識処理、印刷漢字認識処理、手書き漢字認識処理のＬＩＳＣＡＲ１上での処理方法を示すとともに、処理性能を評価する。さらに、高速化手法を明らかにし、速度性能改善をはかるとともにＬＩＳＣＡＲ１の問題点を示す。

第９章では、ＬＩＳＣＡＲ１の処理速度が特に問題となった手書き漢字認識処理に対して、さらに高度化したアルゴリズムをＬＩＳＣＡＲ２で高速に処理することを試みる。具体的には、高速化のポイントになる特徴抽出処理、次元圧縮処理、大分類処理、識別処理の処理方法を示すとともに、処理性能を評価する。

第１０章では、本論分の研究成果を総括するとともに、今後の研究の方向と課題を展望する。

第２章　２次元隣接結合ネットワークを補強する伝搬演算機構

２．１　はじめに

１．３．２の基本アーキテクチャの項で説明した２次元隣接結合型のネットワークは、比較的小さなハードウェア負担で、頻度の高い近傍ＰＥ間通信を効率良く実行できる点で優れている。しかし、規則性の高い定型的な処理に限っても総和演算のように一定の割合で出現する遠隔ＰＥ間の通信を高速に実行することはできない。かといって、他の高度のネットワークのように複雑な構成をとることはセルラ配列構成では許されない。そこで、遠隔ＰＥ間転送については、汎用性は劣るもののハードウェア負担の小さい伝搬演算機構を、方針の項でも述べたように隣接結合ネットワークの中に組込むことにする。本節ではこの伝搬演算機構の最適な構成法を追究し、その性能、ハードウェア規模を評価する。

さて、伝搬演算とは、データ配列$a_1, a_2, a_3, \ldots a_M$に対し$a_1(a_1@a_2),(a_1@a_2@a_3),\ldots,(a_1@a_2@a_3@a_M)$を求める演算であり、配列要素がＰＥ配列に分散配置されている場合に、途中の演算結果をＰＥ間で順次伝搬させることによって求められる。ここで，@は、＋，Ｖ，Λ，ｍｉｎ［最小値を求める演算］，ｍａｘ［最大値を求める演算］等の演算子である。この演算は、放送、総和、部分和、部分コピー等の比較的頻度の高い遠隔ＰＥ間の転送、演算を実現できるため、セルラ配列型ＳＩＭＤプロセッサの中核的な演算機能の１つとして位置付けられている[13,22]。また、ソート・マージ処理、グラフ・幾何学問題等の基本処理を高速に実行できることも、示されている[23,24,25]。

このような有用性から、セルラ配列型ＳＩＭＤプロセッサでは、伝搬演算が、古くから、ＰＥ間の伝搬（縦続的に接続されるＰＥ間で、途中、クロックで同期をとることなくデータを転送すること）を利用することで実現されていた[13,22]。しかし、この方式ではプロセッサの並列度に見合った伝搬演算性能は得られない。演算にかかわるＰＥが、伝搬の先頭の波面に位置するＰＥに限られ、ＰＥ配列の並列性を生かせないからである。

並列度を上げる比較的単純な方法の一つに、縦続的に連なるＰＥ列の適当な間隔Ｎごとにバイパスを設ける方式がある。この方式によれば、長さＭの配列データに対する伝搬演算時間を、$O(\sqrt{M})$程度にまで短縮できる。それでも、Ｍが大きくなると十分な演算速度は得られなくなって来る。

これに対して、最近では２進ツリー構成の演算機構により、伝搬演算時間を$O(log_2M)$程度まで短縮する手法が提案されている[23]。ハードウェア量も、ハイパーキューブ型のＰＥ間転送系で、同一の２進伝搬演算アルゴリズムをエミュレートする[26]のに比べると、かなり小さい。それでも、単純なＰＥ配列のハードウェア規模に比べ無視できる規模ではない。ハードウェア量低減には２進からＮ進への拡張が有効と考えられるにもかかわらず、速度性能の低下をきらってか、十分な検討はなされていない。

そこで、ここでは高速性を損なわないＮ進構成の実現をめざし、バイパス型伝搬および加算器の高速化によく用いられるセレクティブ伝搬のツリー状の階層化について検討する。そして、多段の階層化がＮ進ツリー構成の伝搬演算機構として一般化されること、Ｎを４に選ぶことで、従来の２進ツリー構成に比べ、１／２程度のハードウェア規模で２倍の高速性の得られるツリー構成の伝搬演算機構が実現できること等を示す。

本章では、伝搬演算の概要、基本的な伝搬演算方式による伝搬演算機構の概要、バイパス型・セレクティブ伝搬の多段の階層化によるＮ進ツリー構成伝搬演算機構の導出、各方式の性能・ハードウェア規模の評価、ＬＳＩのチップレイアウトの基本的な検討結果等について述べる。

２．２　伝搬演算の概要

はじめに、１次元配列（ベクトル）Ａに対する代表的な伝搬演算の例を図２．１に示す[23]。ここで、Segment-Flagsとは、配列Ａが複数の部分配列に分割される場合の部分配列の始点か否かを示す１次元配列、＋-propagation(A)は、Ａに対する加算伝搬結果、max-propagation(A)は、Ａに対する最大値伝搬結果である。first-propagation(A)は、始点データの部分配列内での放送伝搬の結果である。

first-propagationでは、部分配列の始点に送信ＰＥを、その部分配列のいずれかに受信ＰＥを、それぞれ対応させることで、遠隔ＰＥ間のデータ転送を行なうことができる。

A	＝	[7 1 3 9 4 2 5 0 6]
Segment-Flags	＝	[1 0 0 1 0 1 0 0 0]
＋-propagation(A)	＝	[7 8 11 9 13 2 7 7 13]
max-propagation(A)	＝	[7 7 7 9 9 2 5 5 6]
first-propagation(A)	＝	[7 7 7 9 9 2 2 2 2]

図２．１　代表的な走査演算の例

この転送機能は、転送区間が重複しなければ、それぞれに対応する部分配列内で並列にデータ転送が行なえる点で優れている。これに対し、一般によく用いられる共通バスでは、同時にバス全体で１組のデータしか転送できないので、転送区間が重複していなくとも、データの組数分だけの転送を繰り返す必要がある。従って、転送速度が同等ならば、所要時間は繰り返しの回数分だけ長くなってしまう。

伝搬演算の有用性の一端を、文字認識処理で特徴抽出に用いられる黒点連結の長さ（ランレングス）の抽出処理を例に示す。この場合の黒点連結の長さの抽出処理は、２値データの１次元配列Ｂに対して１の連なりの長さを求める処理であり、加算型の伝搬演算として図２．２のように簡単に実現される。ここで、Segment-Flagsの配列データは、配列Ｂについて１の連なりの左端、０の連なりの左端をそれぞれ検出することで生成することができる。

なお、以上の説明では、理解し易くするためにワード単位の演算例を示しているけれども、ビット単位の伝搬演算機構を利用する場合には、このワード単位の演算をビット直列処理により行うことになる。

B	＝［００１１１１０００１１０］
Segment-Flags	＝［００１０００００１００１０１］
＋-propagation(A)	＝［００１２３４５０００１２０］

図２．２ ＋-propagationによる黒点連結の長さの抽出

２．３ 基本的な伝搬演算方式

２．３．１ 逐次伝搬方式

ＰＥ間の隣接結合をそのまま利用する最も基本的な伝搬演算方式である。図２．３に示すように、送信ＰＥからの送出データを、途中、各転送ＰＥでいちいち同期をとるこ

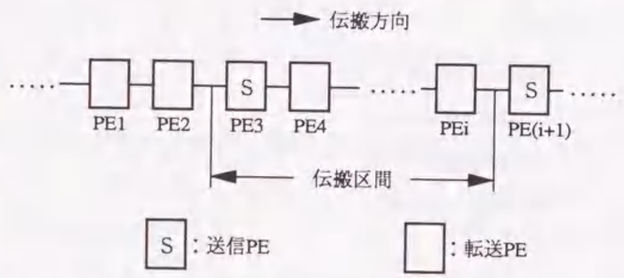
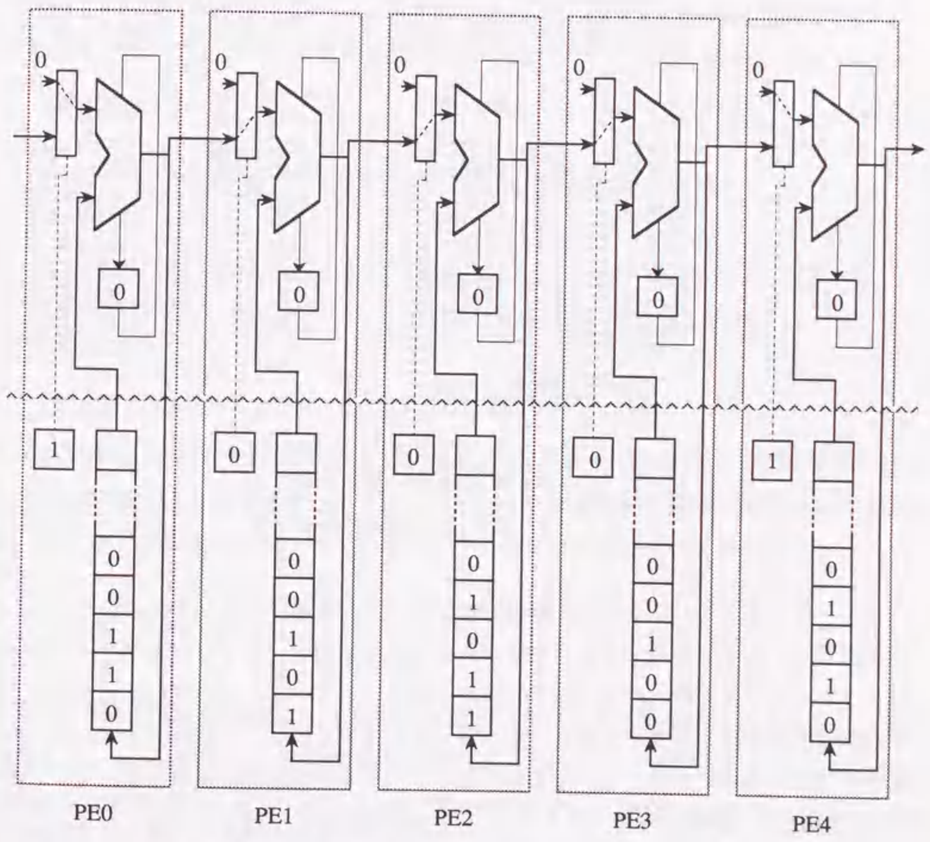


図２．３ ＰＥ間の隣接結合をそのまま利用する逐次伝搬演算方式

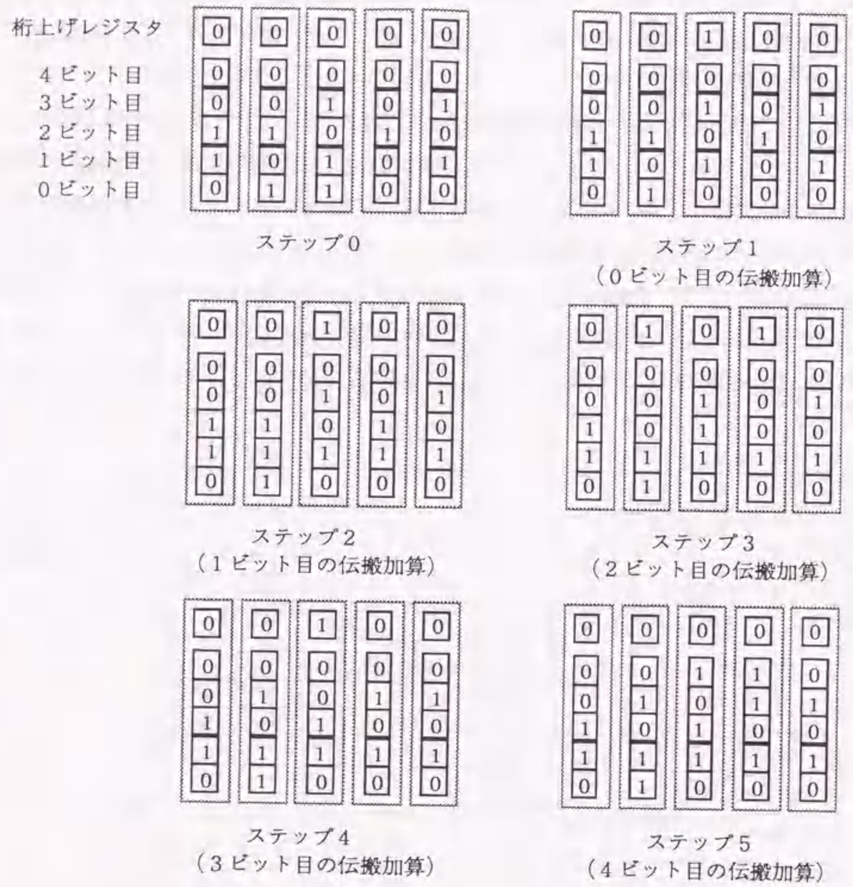
となく、所定の演算を施しながら順次転送すること（伝搬演算）によって部分配列ごとの伝搬演算を実現する。この動作から明らかなように、部分配列の区切りは送信ＰＥの指定により自由に設定できる。

図２．４は、伝搬演算の中でも、セルラ配列型のＳＩＭＤプロセッサにおいて最も基本的でかつ有用なビット直列型の伝搬加算を例に、逐次伝搬方式の伝搬演算を具体的に説明した図である。図の（ａ）は、ビット直列型の伝搬演算を行うための必要最小限のハードウェアである１ビット幅のＡＬＵ（枠内右上）、レジスタファイル（枠内右下）、入力セレクタ（枠内左上）、桁上げレジスタ（枠内右ＡＬＵ直下）、セグメントフラグレジスタ（枠内左中央）からなるＰＥの１次元の配列を示している。図の（ｂ）は、最下位ビットから１ビットずつ順次伝搬加算を実行し、それぞれのＰＥのレジスタ



（ａ）伝搬演算に必要最小限のハードウェアからなるＰＥ配列

図２．４ ビット直列型の伝搬加算



（b）伝搬加算の演算手順

図２．４　ビット直列型の伝搬加算

ファイルの０から４ビット目に伝搬演算結果が得られる様子を示している。各ビットの伝搬演算は、各ＰＥが入力セレクタから入力される伝搬データとレジスタファイルの指定ビットのデータと桁上げレジスタに保持されているデータとの間で加算を実行し、和を右隣のＰＥに出力するとともに、桁上がりを桁上げレジスタに格納することで、左から右方向に向かって進行する。この例では１ビットずつ伝搬演算を繰り返すビット直列型の伝搬加算を説明したが、２ビット以上を単位とする伝搬演算も可能である。この演算幅の拡張によれば、繰り返しの回数が低減されるので、伝搬演算の高速化が可能である。また、本方式は後述する他方式に比べるとこの演算幅の拡張が容易であるという特

徴を持っている。その理由は、ＰＥ間の結線を、隣接ＰＥ間にのみ演算のビット幅分だけ設ければよく、他方式のように遠隔ＰＥ間に多重の結線を設ける必要がないからである。

伝搬演算に直接関わる波線より上の部分を、伝搬演算エレメント（ＰＯＥ）の配列として切りだすことにより構成した伝搬演算機構のブロック図を図２．５に示す。図２．４とは異なり、伝搬演算を実行する上で意味のない左端のＰＯＥは省略している。各ＰＯＥ内のＰ1_1は伝搬演算器であり、入力セレクタ、ＡＬＵ、桁上げレジスタ等で構成され、左隣から太実線の経路を介して入力される伝搬データとＰＥから太実線の経路を介して供給される演算対象配列の要素データとの間で所定の演算を実行し、結果を右隣へやはり太実線の経路を介して出力する。太破線は演算結果をＰＥへ戻すための経路である。細破線はＰＥのセグメントフラグレジスタから入力セレクタに対する制御信号を受け取る経路であり、この値が１の場合に、Ｐ1_1は始点の伝搬演算器として、ＰＥからの要素データを直接右隣のＰＥに出力するようにしている。

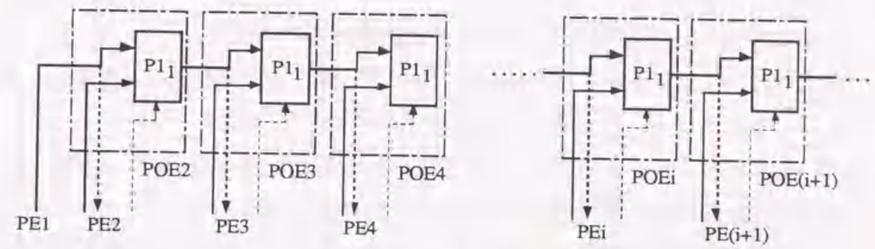


図２．５　逐次型伝搬演算機構

ここで伝搬演算時間T_{OP}を評価する。伝搬演算時間T_{OP}は、途中のＰＯＥをすべて経由する必要があることを考慮すると、個々のＰＯＥの伝搬時間が積算されるため、最悪条件（部分配列が配列全体に一致する場合）で、次式のように表される。

$$T_{OP}=(M-1)\cdot t_{pd} \qquad (2.1)$$

ここで、Ｍは伝搬演算対象の１次元データ配列全体に１対１で対応する１次元ＰＥ配列のサイズ、t_{pd}は各Ｐ1_1の伝搬遅延時間である。この（２．１）式から明らかなように伝搬演算時間が経由するＰＥ数に比例して増加するため、遠隔ＰＥ間の転送には全く対応できないように見える。しかし、１チップに複数ＰＥを搭載するセルラ配列構成では、伝搬時間t_{pd}が、ＰＯＥの構成を最適化することでクロックのサイクルタイムの数

分の１以下にまで低減できることから、伝搬距離をＰＥ通過段数で十数段以下に限定できるならばこの単純な構成でも転送サイクル数の多さが問題になることはない。なお、ハードウェア規模の指標であるＰＯＥ数N_Pは、図２．５から明らかなように、Ｍ－１である。

２．３．２ バイパス伝搬方式

２．３．１の単純な構成では、配列サイズMが大きく、ＰＥ通過段数で数十段以上の距離の転送が必要になる場合には、当然伝搬時間が長くなりすぎる。これを改善する最も直感的な方法は、バイパスを付加することである。図２．６は、そのバイパスを一定の間隔ごとに付加するバイパス伝搬方式の概念図である。図２．７は、バイパスの付加単位であるＰＥ配列（伝搬演算ユニット：ＰＯＵ）の構成を示している。ＰＯＵは、伝搬演算器$P1_1$，$P2_1$，始点検出器ＤＴからなるＮ－１個のＰＯＥおよびユニット内の集約用伝搬演算器$P1_2$からなり、次の３つの演算を並列に行うことで、伝搬演算を実行する。

①ＰＯＵ内の$P1_1$の並びで行なう局所的な伝搬演算、

②局所的な伝搬演算結果をＰＯＵ間でバイパスと演算器$P1_2$を介して累積し、各ＰＯＵのオフセット値を求める伝搬演算、

③この累積によって得られたオフセット値と先の局所的な伝搬結果から全域的な伝搬演算結果を得る$P2_1$での演算。

ここで、ＰＯＵのオフセット値とは、そのＰＯＵがカバーする部分伝搬領域の直前までの伝搬演算結果である。また、図２．７で、実線、破線等によって示される経路の役割は、図２．５の場合と同じである。

ＰＯＵ内に始点のＰＥがある場合、演算器$P1_2$は、バイパス側の入力は無視し、直前のＰＯＥＮからの入力をそのまま出力する。また、ユニット内の始点以降の演算器$P2_1$は前段のユニットからの累積結果を無視し、$P1_1$で得られた結果を最終的な伝搬演

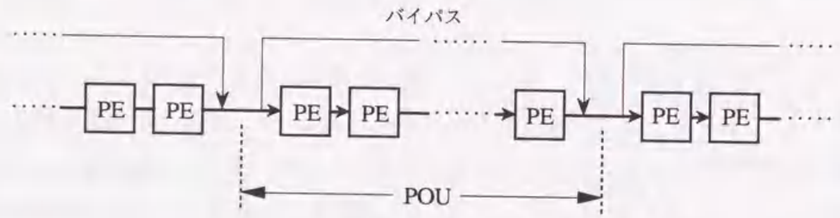


図２．６ バイパス型伝搬方式の概略

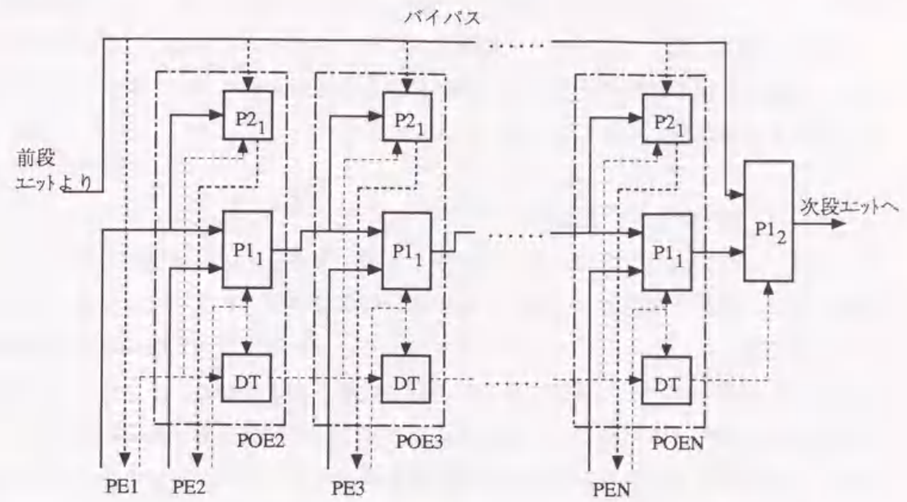


図２．７ バイパス付き伝搬演算ユニットＰＯＵの構成

$P1_1$: 局所伝搬用演算器，$P1_2$: 累算用演算器，

$P2_1$: 広域的な演算結果生成用演算器，

DT:送信ＰＥ検出器，POEi: 伝搬演算エレメント

算結果とする。これらの始点の有無に関係する制御は、始点検出器ＤＴの連なりから生成される信号によって行なう。なお、$P1_1$，$P1_2$は同一構成の伝搬演算器である。

ＰＯＵの構成・動作から明らかなように、ＰＯＵ内の伝搬は、ＰＯＵごとに並列に実行され、ＰＯＵの個数分だけの並列度が得られる。広域的な伝搬演算結果を得るための$P2_1$の演算についてはＰＥ間で独立に実行され、ＰＥ数分だけの並列度が得られる。

以上より伝搬演算の伝搬遅延時間T_{OP}は、

$$T_{OP}=(N-1)\,t_{pd}+\{(M/N)-1\}\,t_{pd}+t_{pd} \qquad (2.2)$$

となる。ここで、第１項はＰＯＵ内の伝搬遅延時間、第２項はＰＯＵ間の累積のための伝搬遅延時間、第３項は最終的な結果を得るための$P2_1$の演算遅延時間である。Ｎは各々のＰＯＵが分担する部分配列サイズである。$P1_1$，$P1_2$，$P2_1$の演算機能が同一であることから、伝搬遅延時間はいずれもt_{pd}としている。また、ＰＯＵが、Ｎ－１個のＰＯＥと１個の$P1_2$の計Ｎ個と、伝搬演算対象の部分配列の要素数と同数で構成

されることから、全体のＰＯＥとＰ１$_2$の個数N_PはＭとなる。なお、ビット幅の拡張は、ビット幅に比例して増加する伝搬演算系のハードウェア規模については問題ないものの、遠隔ＰＥ間の結線に相当するために、長くなりがちなバイパスがビット幅分必要になる点で、逐次型より難しくなる。

２．３．３ セレクティブ伝搬方式

以上述べてきた各方式では、その構成から明らかなように、高速化が必要ならば、演算器、ＰＯＥ間の結線のビット幅、すなわち伝搬演算の演算単位を大きくすることによって対応できる。しかし、バイパス伝搬方式では、長くなりがちなバイパス用結線の増加が、ＬＳＩやボードの端子数ネックの原因になる可能性がある。このため、小型化・経済化を優先する場合には、演算単位拡大による高速化はできれば避けたい。

逆に、演算単位をビット直列型の伝搬演算を前提に１ビットに限定するならば、加算器の高速桁上げ方式の一つであるセレクティブ伝搬方式[27]によって、一層の伝搬演算の高速化が可能である。

この方式は、前段ユニットからの入力される可能性のあるデータ（演算単位を１ビットに限定する場合には０，１の２通りしかない）の両方に対し、ユニットごとに専用に設けた伝搬演算系で、先行して伝搬を行なっておき、その結果を実際の入力で選択することによってユニット間の伝搬の高速化をはかる。

図２．８に演算単位が１ビットのセレクティブ構成のＰＯＵの構成を示す。各ＰＯＥの演算器$P1_A$と$P1_B$は演算器の幅が１ビットに限定されることを除きバイパス伝搬方式の$P1_1$と同一機能の演算器である。それぞれはユニット内で伝搬系を形成し、ユニットに対する入力に先行して、０入力と１入力の２通りの伝搬演算を行なう。各ＰＯＥの下側のセレクタSELは、ＰＯＵへの実際の入力に応じて、２通りの先行演算結果を選択し、最終的な結果を得るためのものである。また、ここでは、理解の容易化のため図示していないが、加算や減算の伝搬演算を１ビット単位で逐次的に実行する際に必要となる桁上げに関しても、同様に先行生成結果を二者択一する構成によって求められる。

なお、この構成では、基本の伝搬方式、バイパス伝搬方式とは異なり、演算結果が伝搬方向にずれることがないので、後処理無しで最終的な伝搬演算結果が得られる。

伝搬演算時間T_{OP}は、以上の動作より、

$$T_{OP}=N\,t_{pd}+\{(M/N)-1\}\,t_{pdS}+t_{pdS} \qquad (2.3)$$

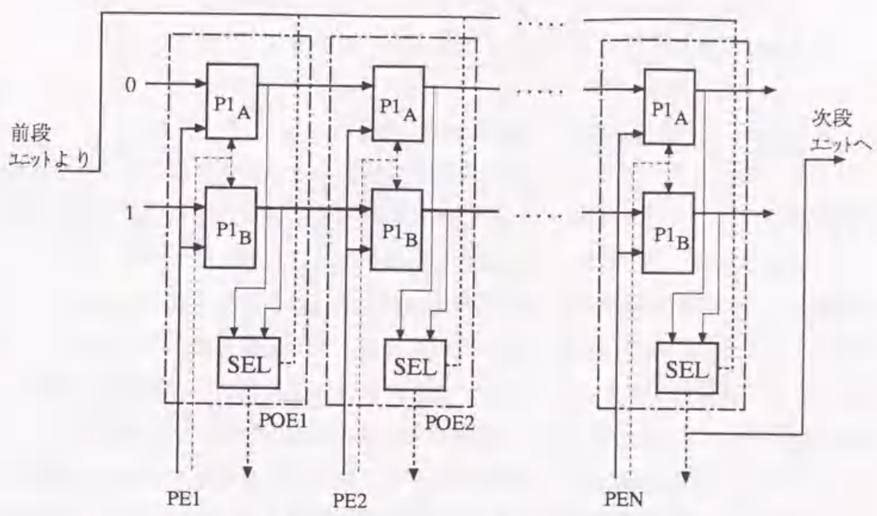


図２．８ セレクティブ伝搬ユニットＰＯＵの構成

$P1_A$,$P1_B$:演算器，SEL: 2-1 セレクタ

となる。ここで、第１項はＰＯＵの内部遅延時間、第２項はＰＯＵ間の遅延時間、第３項は、先行演算結果の二者択一のための遅延時間である。また、t_{pd}は$P1_A$あるいは$P1_B$に関わるＰＯＥの伝搬遅延時間、t_{dpS}はセレクタに関わる伝搬ユニットの伝搬遅延時間である。第１項の係数から－１がなくなっている点を除くと、（２．２）式と同じ形をしている。ただし、伝搬演算器$P1_2$がセレクタに置換されて遅延時間がt_{pdS}（t_{pd}の半分程度以下）になる分、演算時間は短縮される。また、ＰＯＥの全個数N_Pは、各ＰＯＵが伝搬演算対象の部分配列の要素数と同数のＮ個のＰＯＥで構成されることから、Ｍとなる。

なお、このセレクティブ伝搬方式でも、演算単位拡大による高速化は不可能ではない。しかし、拡大可能な演算単位は精々２ビット程度までである。伝搬ユニットごとで、入力される可能性のあるデータの種類（2^W通り。ここでWは、演算単位である。）だけ、伝搬演算系を設けねばならず、これ以上では、ハードウェア規模が大きくなり過ぎるからである。

２．４ Ｎ進ツリー構成の伝搬演算機構

本節では、バイパス方式とセレクティブ伝搬方式の両方に対して一層の高速化をはか

るために、多段の階層化について検討する。ただし、演算単位については、セレクティブ伝搬方式に合せてすべて１ビットとして議論を進める。

２．４．１ バイパスＮ進ツリー構成導出

バイパス伝搬方式では、ＰＯＥ間の伝搬遅延低減のために、ＰＯＥ間の並びに対して、ＰＯＵ内にバイパスを設けている。これと同様に、ＰＯＵの並びに対して同様にバイパスを設ければ、ＰＯＵ間の伝搬遅延も低減される。高速化のために、さらに、その上の階層にもバイパスを設けることが考えられる。このようなバイパス階層の多段化に対応できるように、階層２段に対応している図２．７の伝搬演算ユニットから、１階層分を切出した伝搬演算ユニットＰＯＵの構成を図２．９に示す。右端のＰ1_iの出力がこのＰＯＵのカバーする局所領域の伝搬演算結果、右端のＤT_iからの出力がこのＰＯＵのカバーする領域に伝搬演算の始点があるか否かの判定結果であり、かつ各Ｐ2_iへ向かう入力にこの局所領域までの伝搬演算結果を入力すれば、各ＰＯＥまでの伝搬演算結果が求まることから、結局、このＰＯＵをツリー状に階層的に並べることによって、図２．１０に示すＮ進ツリー構成の伝搬演算機構が構成される。

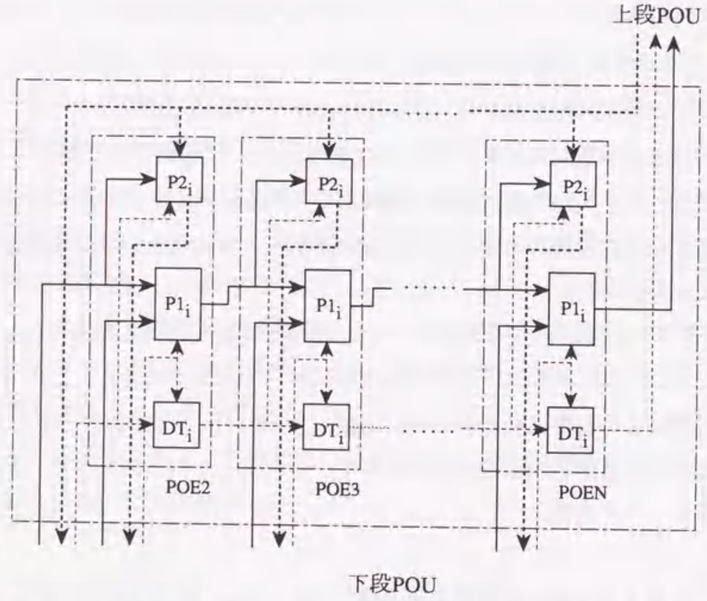


図２．９ Ｎ進ツリー構成用バイパス伝搬演算ユニットＰＯＵの構成

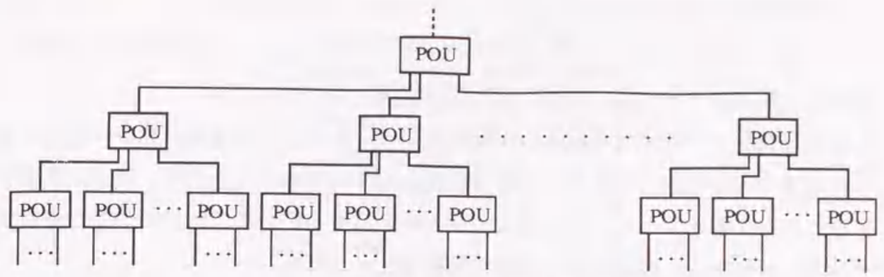


図２．１０ Ｎ進ツリー型走査演算機構の構成

このＮ進ツリーの動作は、

①各ＰＯＵの局所的な伝搬による伝搬演算結果を、ツリーの下段から上段に向かって累積する、

②累積結果を元に、上段から下段に向かって各ＰＯＵのオフセット値を順次求めるの２つからなる。この場合、最下段のＰＯＵからのＰＥに対する出力は、やはり、オフセット値であり、各ＰＥは、自身の保持データとの間で最終的な伝搬演算結果を得るための後処理を行なう必要がある。

ところで、図２．１０のツリー構成は以下のように、（２．４）、（２．５）の式の変形を再起的に繰り返すことによっても、導かれる。

$$\sum_{i=S}^{E} a_i = \left(\sum_{i=S}^{N[S/N]+N} a_i\right) @ \left(\sum_{k=[S/N]+1}^{[E/N]} \sum_{j=(k-1)N+1}^{kN} a_j\right) @ \left(\sum_{i=N[E/N]+1}^{E} a_i\right) \qquad (2.4)$$

$$= \left(\sum_{i=S}^{N[S/N]+N} a_i\right) @ \left(\sum_{k=[S/N]+1}^{[E/N]} b_k\right) @ \left(\sum_{i=N[E/N]+1}^{E} a_i\right) \qquad (2.5)$$

ここで、$a_1, a_2, a_3, \ldots, a_M$は演算対象のデータ配列を、$a_S, a_{S+1}, a_{S+2}, \ldots, a_{E-1}, a_E$は、その部分配列を、Σは伝搬演算の累積を、［］はガウス記号を表している。また、b_kは次式で表される。

$$b_k = \sum_{j=(k-1)N+1}^{kN} a_j$$

（２．４）式は部分配列全体に対する走査演算が、Ｎ項単位の伝搬演算にくくれることを意味している。（２．５）式は、Σb_kが上位階層の伝搬演算と考えることがで

き、伝搬演算が2段に階層化されることを意味している。また、（2．4）と同様の変形をΣb_kに対しても加えれば、さらに2段に階層化されることがわかる。結局、この階層化を順次繰り返せば、N進ツリー状の演算手順が導かれる。

このN進ツリーの演算手順から伝搬演算ユニットＰＯＵの機能が導かれる。すなわち、（2．5）式が、あるＰＯＵの下位出力の1つを表す式であり、第1，2項がそのＰＯＵに対するオフセット値、第3項がそのＰＯＵの出力までの局所的な走査演算結果であること等から、図2．9のＰＯＵ機能が求められる。

2．4．2 セレクティブN進ツリー構成導出

セレクティブ伝搬方式もバイパスと同様に階層の多段化が可能であり、図2．11に示すＰＯＵを積み重ねることによって、N進ツリー構成が実現される。このＰＯＵの構成とN進ツリー構成は、セレクティブ伝搬のための伝搬系の二重化を、ＰＯＥ間の伝搬経路から、バイパスに相当するセレクタ制御経路の二重化、さらにそれによって新たに必要となる上位のセレクタ制御経路と、順次拡張することから導かれる。

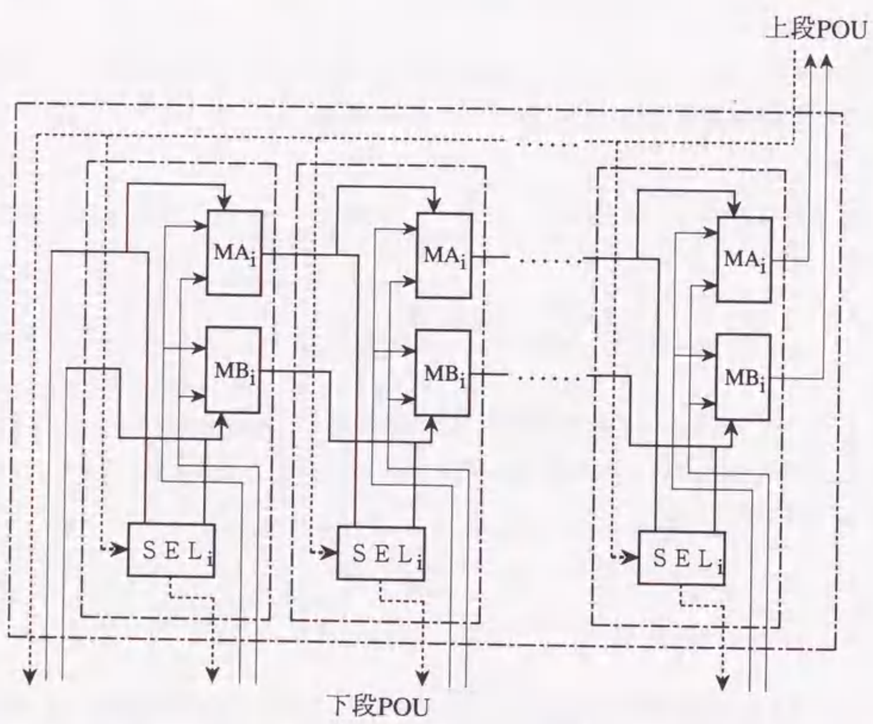


図2．11 N進ツリー構成用セレクティブ伝搬演算ユニットＰＯＵの構成

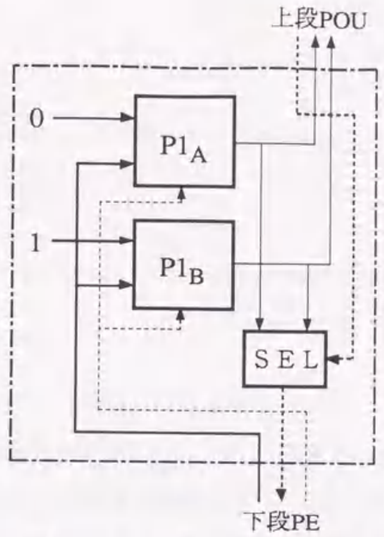


図2．12 インタフェースエレメントＩＥの構成

ここで、ＰＯＵ内の左右間の太実線は、二重化したセレクタ制御信号の伝搬経路である。また、各ＰＯＥに対する2本の細実線は下位のＰＯＵからの2通りの演算結果の入力経路であり、太破線は上位から下位に向かう2通りの演算結果のいずれを選択するかを制御するための信号用経路である。

最下位のＰＯＥと、ＰＥ間の接続は、直接は無理で、2通りの演算結果を生成するために、図2．8のＰＯＥと基本的に同一の構成の図2．12のインタフェースエレメントＩＥを介して行なう。これによって、演算器の遅延がすべてセレクタの遅延に置き換えられることになる。ここで、ＰＥに向かう太実線、太破線、細破線の経路の役割は2．4．1の伝搬方式のそれと同じである。

セレクティブN進ツリーで注目すべきは、ＰＯＥが、3つの2－1セレクタのみで構成される点である。伝搬演算のためにＰ1、Ｐ2の演算器を必要とする図2．9のバイパス構成のＰＯＥと比べると、はるかに単純である。

2．4．3 N進ツリー構成の演算時間とＰＯＥ数

N進ツリー構成の伝搬遅延時間は、下位から上位に向かう演算におけるＰＯＵ内の伝搬段数がN－1あるいはN段であること、上位から下位に向かう演算における伝搬段数がＰ2あるいはSELの系の1段のみであること等から、バイパスツリー構成では、

$$T_{OP} = (N-1)(\log_N M)\, t_{pd} + ((\log_N M)-1)\, t_{pd}$$
$$= (N\log_N M)\, t_{pd} - t_{pd} \qquad (2.6)$$

となる。セレクティブツリー構成では、さらにインタフェースエレメントＩＥによる遅延を考慮すると

$$T_{OP} = (N-1)(\log_N M)\, t_{pdS} + ((\log_N M)-1)\, t_{pdS} + (t_{pd} + t_{pdS})$$
$$= (N\log_N M)\, t_{pdS} + t_{pd} \qquad (2.7)$$

となる。ここで、Ｐ１，Ｐ２，ＤＴの遅延をいずれも t_{pd}、各セレクタの遅延をいずれも t_{pdS} としている。これらの式より、t_{pdS} は t_{pd} の半分程度以下にできることから、セレクティブツリー構成がバイパスツリー構成の２倍程度以上高速であること、Ｎ値が自然対数の底ｅで最も高速化されること等がわかる。

また、ＰＯＥ数は、いずれの方式もＰＯＵがＮ－１個のＰＯＥからなることから、各階層の総和より、

$$N_P = (N-1) + N^1(N-1) + \cdots\cdots\cdots + N^{(\log_N M)-1}(N-1)$$
$$= M-1 \qquad (2.8)$$

となり、Ｎによらない。

２．５　性能、ハードウェア規模の評価

２．５．１　速度性能

式（２．２）、（２．３）、（２．６）、（２．７）より、各方式における配列サイズＭと数式上の最適条件（階層２段の方式では、$N=\sqrt{M}$、ツリーの方式では、$N=e$）での等価伝搬段数 T_{OP}/t_{pd} との関係を求めた結果を図２．１３に示す。ただし、t_{pdS} は t_{pd} の１／２と仮定している。これより、セレクティブＮ進ツリー構成が、従来型の２進ツリー構成、バイパスＮ進ツリー構成と比べ２倍程度高速であることがわかる。

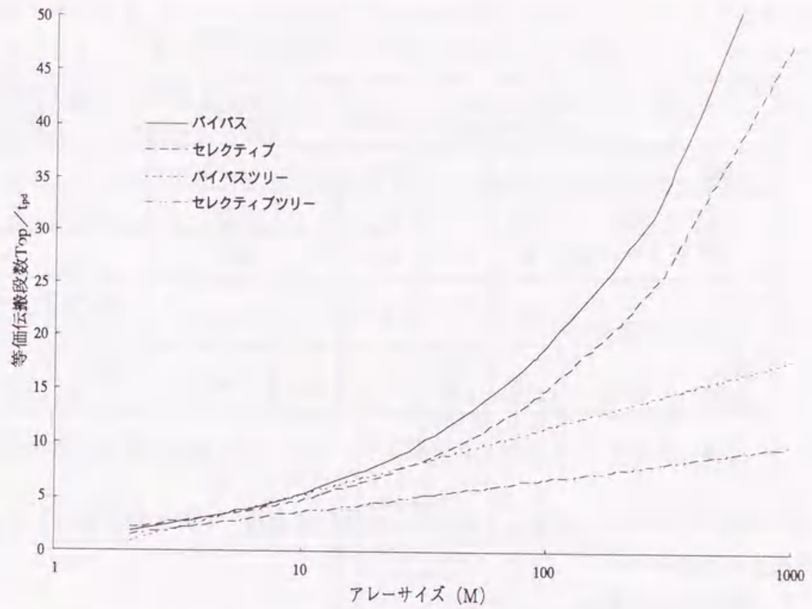


図２．１３　配列サイズＭと伝搬段数の関係

２．５．２　ハードウェア規模

ハードウェア規模は、その１つの指標であるＰＯＥ数からいえば、いずれの方式もＭ程度で差がない。しかし、実際には差が生じる。これは、①ＰＯＥ単体の規模、②規則性、局所性の低いＰＯＵ間の結線量、③ＰＥに一体化が容易な最下層のＰＯＥ数等が方式間でかなり異なることによる。これらについて後述する回路構成を前提に各方式を比較した結果を表２．１に示す。この表より、Ｎ進ツリー構成について、以下の２点が明らかとなる。

①ＰＥ配列に一体化困難でＰＥ配列の外に出るゲート数（伝搬演算機構の実効的なハードウェア規模とみなせる）とＰＯＵ間結線数は、ともにＮの増加にほぼ反比例して減少する。

②セレクティブ構成は、総ゲート数が、バイパス構成に比べ２５％程度大きい。しかし、ＰＥに一体化困難な部分のゲート数については、バイパス構成の１／４程度と小さいことから、チップレイアウト時の実効的なハードウェア規模では、バイパス型と同程度かそれ以下にできる。

表２．１　ハードウェア規模決定要因の比較

方式	POE規模＊	IE規模＊	POE数	IE数	演算機構総ゲート数	PE配列以外のゲート数	POU間結線数
伝搬	19	－	M-1	－	19 (M-1)	0	－
バイパス伝搬	40	－	M	－	40M	19M/N	－
セレクティブ伝搬	40	－	M	－	40M	0	－
N進ツリー（バイパス構成）	40	－	M-1	－	40 (M-1)	$\frac{40\ (M-N)}{N}$	$\frac{3\ (M-1)}{N-1}$
N進ツリー（ｾﾚｸﾃｨﾌﾞ構成）	10	40	M-1	M	50M-10	$\frac{10\ (M-N)}{N}$	$\frac{3\ (M-1)}{N-1}$

＊：ＮＴＴ ＬＳＩ研究所開発のゲートアレイで構成する場合のセル数

以下に表２．１のハードウェア規模算出において前提とした伝搬演算器、ＰＯＥ等の回路構成を説明する。

（１）伝搬演算器の構成

伝搬演算器Ｐ１$_i$，Ｐ２$_i$の回路構成を図２．１４に示す。ゲート数は１９である。ＡＬＵのファンクション生成部は、４－１セレクタで構成している。マスタースレイブ型のＤフリップフロップ（ＭＳ－ＤＦＦ）は、逐次的に算術演算を行なう際に用いる桁上げレジスタである。

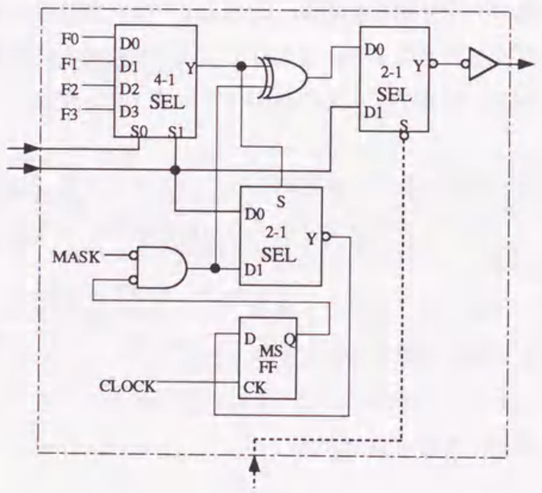


図２．１４　伝搬演算器Ｐ１$_i$，Ｐ２$_i$の構成

伝搬演算器Ｐ１$_A$，Ｐ１$_B$の回路構成は、基本的には、Ｐ１$_i$，Ｐ２$_i$と同一である。ただし、桁上げレジスタについては、Ａ，Ｂで共通に一個設け、セレクティブ伝搬終了後の確定値を格納する構成としている。Ｐ１$_A$，Ｐ１$_B$合わせたゲート数は３６である。

（２）セレクティブN進ツリー用ＰＯＥ

２－１セレクタをＯＲ－ＮＡＮＤの複合ゲートで構成する場合の回路構成を図２．１５に示す。ゲート数は１０である。

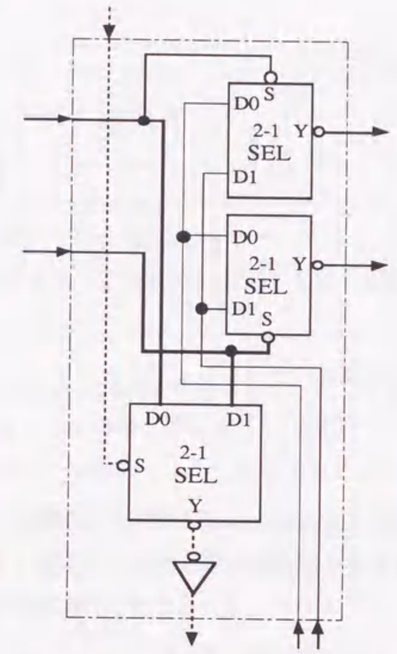


図２．１５　セレクティブN進ツリーのＰＯＥの回路構成

２．６　N進ツリー構成伝搬演算機構を付加した１次元ＰＥ配列のチップレイアウト

２進ツリーのレイアウト方式[28]を参考にレイアウトの基本的な検討を行なった結果を述べる。

２．６．１　N値の選択

ハードウェア規模低減の観点からは先に述べた通り、Ｎは大きいほどよい。しかし、高速化の観点からは、２．４．３で述べたように自然対数の底ｅ前後が望ましい。そこで、ここでは演算速度を重視し、Ｎ値を４に選ぶ。従来の２進ツリー構成と比べると、演算時間、ＰＯＵ間の結線数、ＰＥに一体化困難なゲート規模等がほぼ半分になる。

2．6．2　4^2 進ＰＯＵの構成

４進ＰＯＵの２段のツリー結合で、4^2進のＰＯＵを構成する。レイアウト構成の単純化をはかるためである。そのＰＯＵ構成を図２．１６に示す。ここで、上段のＰＯＵは、ＰＯＥ単位に分解し、下段のＰＯＵ間に分散配置することで、等価的に１段構成のレイアウトとし、集積度の向上をはかっている。

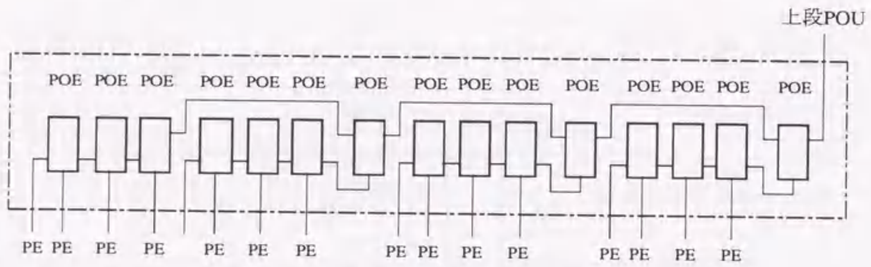


図２．１６　4^2進伝搬演算ユニットＰＯＵの構成

2．6．3　4^2 進ツリー構成のチップレイアウト

１段目の4^2進ＰＯＵを、１次元のＰＥ配列と組み合わせることによって構成した２段構成の4^2進ツリーのレイアウトを図２．１７に示す。この図から明らかなようにＰＯＵを構成する１５個のＰＯＥは、その下側のＰＥ配列に一体化することで、集積度の向上がはかれる。レイアウトは単体では、長さ２５６のＰＥ配列に対応するツリーであるが、複数結合することによって、２～３千までの配列サイズに対応可能である。

2．6．4　性能推定

このようなレイアウトによって、長さが２５６の１次元ＰＥ配列と伝搬演算機構が５～６年前より利用可能になっている0.8μm程度の製造技術のＣＭＯＳ ＬＳＩチップに搭載されるならば、ＰＯＥ間の伝搬経路の配線長は数ｍｍ以下となり、セレクタの平均の伝搬遅延t_{pdS}は１ｎｓ程度になる。従って、セレクティブ伝搬方式の場合、式（２．５）より、T_{OP}は、２０数ｎｓとなる。この値は、実現可能なＰＥの最小クロックサイクルと同程度であり、長さが２５６の配列に対する伝搬演算が、１～２クロックサイクルで実現されることになる。

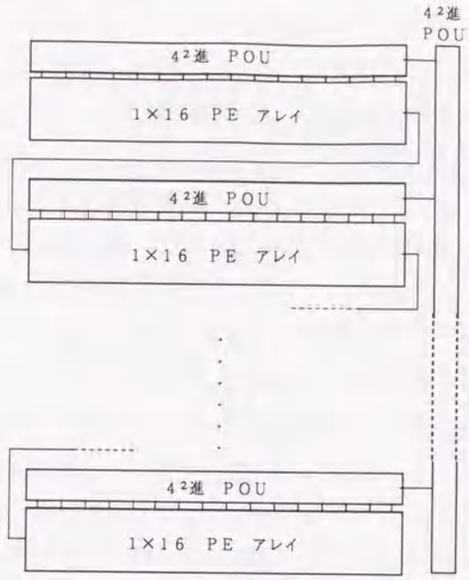


図２．１７　ツリー構成走査演算機構を備えた１次元ＰＥ配列のレイアウト構成

２．７　むすび

隣接結合ネットワークの遠隔ＰＥ間の演算あるいは転送を補強する伝搬演算機構の構成方式を検討し、Ｎ進ツリー構成の高速の伝搬演算機構を提案した。

はじめに、基本的な伝搬演算方式として、

①演算結果をＭ個のＰＥ間で直列に伝搬させる逐次伝搬方式

その高速化方式である

②Ｎ個のＰＥからなる局所ごとに並列に行う伝搬演算結果をバイパスを介して累算するバイパス方式

③演算単位を１ビットに限定し局所ごとに０,１の両方の入力に対する演算を先行的に行い、前段からの真の入力値により先行演算結果を選択するセレクティブ方式

のそれぞれについて、構成法を明らかにするとともに、伝搬演算時間を評価し、①の逐次伝搬方式ではＭ段分のＰＥの演算時間がそのまま積み重なるのに対し、②、③の高速化方式では、各々Ｎ段からなる局所ごとの演算時間と局所間のＭ／Ｎ段の演算時間との和にまで短縮されることを示した。

次いで、②、③のバイパス方式、セレクティブ方式について、階層化による一層の高速化を試み、ともにＮ進ツリーの階層的構成になること、特にセレクティブ方式では、

３個の２－１セレクタからなる伝搬演算エレメント（ＰＯＥ）のみでＮ進ツリーが構成されることを導いた。また、その伝搬演算時間についても評価し、ともに$N\log_N M$段分にまで低減されること、セレクティブ方式では演算がビット単位に制限されるものの途中セレクタのみを経由するために、演算器を経由するバイパス方式に比べ、さらに２倍近く高速になることを明らかにした。

Ｎ進ツリー構成については、ハードウェア規模の評価も行い、伝搬演算を１ビット単位で行なう場合には、４進のツリーにより、セレクティブ方式が従来のバイパス２進ツリーに比べ、半分以下の実効的なハードウェア規模で実現されることを示した。また、ＬＳＩチップレイアウト構成も検討し、このセレクティブ４進ツリー構成の伝搬演算機構が、稠密な構造で比較的容易にチップレイアウトできること、長さ２５６の配列データに対する伝搬演算を１ないし２クロックサイクルで実行できることを示した。

第３章　可変構造と演算アルゴリズム

３．１　はじめに

ＳＩＭＤプロセッサは、定型的な処理を得意とするといわれている。しかし一般的には、定型的な処理といっても、常に同一サイズの配列データのみを処理すればよいわけではない。ＰＥ配列のサイズをはるかに越えるような大規模な配列データの処理から、ＰＥ配列のサイズをかなり下回る配列データの処理まで、ダイナミックに変化する場合が大半である。一般にＰＥ配列のサイズを越えるような大規模な配列データの処理は、その配列をＰＥ配列のサイズに収るように分割し、それぞれを個別に処理することで容易に５０％以上の高いＰＥ配列の稼働率を得ることができる。これに対して、ＰＥ配列のサイズを大きく下回るような配列データの処理は、そのまま実行しようとすると、ＰＥ配列の大半が遊ぶことになり、ＰＥ配列の稼働率はかなり低下することになる。これでは、せっかくの並列処理による高い性能が生かせない。このサイズの小さな配列データに対する処理は、別に設けるスカラ処理プロセッサによって実行することも考えられる。しかし、その配列データが一連の配列データの処理の途中結果である場合には、それを外部のスカラプロセッサにいちいち転送しなければならないのに加え、場合によっては、スカラプロセッサの処理結果を再度ＰＥ配列にもどすことが必要になる。従って、スカラプロセッサにまかせる方法では、並列処理による高速化が期待できなくなるばかりか、データ転送のオーバーヘッドも加わり、処理性能を大きく低下させることになる。

このような被処理データの構造のダイナミックな変化に対応するために、代表的なセルラ配列型ＳＩＭＤプロセッサの１つとして知られているＩＣＬのＤＡＰ[12]では、ＰＥ配列の各行を１,２,４,８,１６,３２のいずれかの長さのＰＥ連結（以下ＰＵ［Processing Unit：処理単位］と呼ぶ）を、より高速な処理単位として動作させる機構を付加している。これによって、ＰＥ配列のサイズ（厳密にはＰＵ配列のサイズ）が可変となり、小さな配列データを処理する際のＰＥ配列の稼働率を改善している。

本章では、このようなセルラ配列型ＳＩＭＤプロセッサにおけるＰＥ配列の可変構造[29]を追究し、ＰＥ配列の一層の稼働率向上をはかる。また、この可変構造を利用することによって、汎用化に有効なＰＥ配列内の矩形の局所領域ごとの総和演算、ＬＳＩ－ＣＡＤの論理シミュレーションの基本処理等が効率的に実行できることを示す。

３．２　修飾型ＳＩＭＤ制御による適応的な可変構造の実現

ＰＥ配列を可変構造とするには、所定の命令に対し、各々のＰＥの動作を個別に設定

できることが必要である。このＰＥ毎の個別の動作制御の機能は、各ＰＥに制御レジスタを設け、これによって制御部から受ける全ＰＥ共通の命令を個別に修飾すること（修飾型ＳＩＭＤ制御）で実現される。

本節では、ＰＥ毎の個別の動作制御により得られる可変構造の基本機能のうち、セルラ配列型ＳＩＭＤプロセッサの性能向上に有効と考えられるものを示す。

３．２．１　ＰＥ配列への適合的なＰＵ割付け

ＰＵを構成するためのＰＥ間の連結は、各ＰＥを適当なサイズのビット幅で分割したレジスタ付きＡＬＵとして動作させることによって実現される。すなわち、算術演算における桁上げを、ＰＥ間で下位桁のＰＥから上位桁のＰＥに向かって伝搬させることによって実現される。ただし、この場合、最下位桁の（ＬＳＤ）ＰＥは、桁上げを下位側

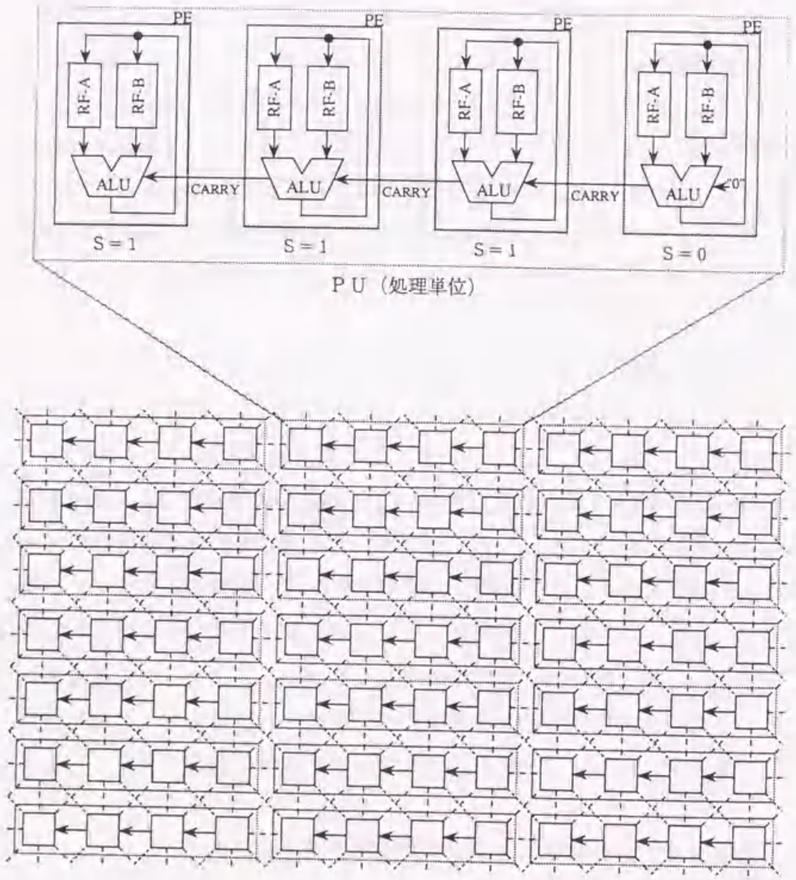


図３．１　ＰＵの構成

のＰＥ（存在しない）から受け取るのではなく、自身で加算の場合には０、減算の場合には１の固定値桁上げを発生できる必要がある。

この場合、ＬＳＤかそうでないかのＰＥ機能の違いをＰＥ毎に個別に設ける制御レジスタで切り換えることによって、任意の位置に任意のビット幅のＰＵを構成することができる[26]（図３．１参照。ここで、Ｓは制御レジスタの内容を示している。また前章の伝搬演算の場合とは矢印の方向をわざわざ逆に選んでいるのは、ＬＳＤ側を右、ＭＳＤ側を左側に配置して桁上げを左方向に伝搬させる加算器の記述慣習に従ったためである。）。セルラ配列型のＳＩＭＤプロセッサでは、ＰＥの構成がもともとビットスライスのレジスタ付きＡＬＵに近い構成を採っているため、ＰＥ連結により無駄なくビット幅の大きいＰＵを構成することができる。しかし、桁上げをＰＥ間で伝搬させるために、ビット幅の大きいＰＵを構成すると、演算時間がその分増加する。桁上げの伝搬経路を最適化してもクロック当たりＰＥ通過段数で５,６段程度が限界と考えられ、これ以上のビット幅のＰＵを構成しようとすると、処理語長と単位時間当たりの実行可能総演算量との積で表されるＰＥ配列全体の性能は、ＰＵのビット幅を大きくした分だけ低下することになる[30]。従って、処理データの語長が大きくとも、配列のサイズがＰＥ配列のサイズと同程度かそれ以上の場合には、ＰＥ毎に多倍長の演算（ＰＥが１ビット構成の場合にはビット直列演算）を繰り返すのが有利となる。

３．２．２　指定ＰＥでの直並列変換

各ＰＥは処理データを、通常、内蔵する演算系の演算単位に適合するように折り畳んで局所メモリ内に格納している。このため、特定のＰＥの保持データを演算幅の広いＰＵで処理するには、そのＰＥから、一旦、被処理データを取りだし、ＰＵ上に展開する必要がある。ＰＥが１ビット構成で、かつデータがビット直列形式で格納されている場合には、ＰＵで処理するデータをビット並列形式に変換することになることから、直並列変換と呼ぶ。

直並列変換は、図３．２に示すように各ＰＥの制御レジスタによる個別制御により送信ＰＥとシフトＰＥを設定し、送信ＰＥから取りだすビット直列データをＰＥ間の隣接結合で形成されるデータ転送パスに乗せて順次シフトすることによって実現される。ここで、送信ＰＥとは保持するビット直列データを順次シフトすることにより出力するＰＥであり、シフトＰＥとは、隣接ＰＥから受け取ったビット直列データをそのまま転送方向にシフト出力するＰＥである。

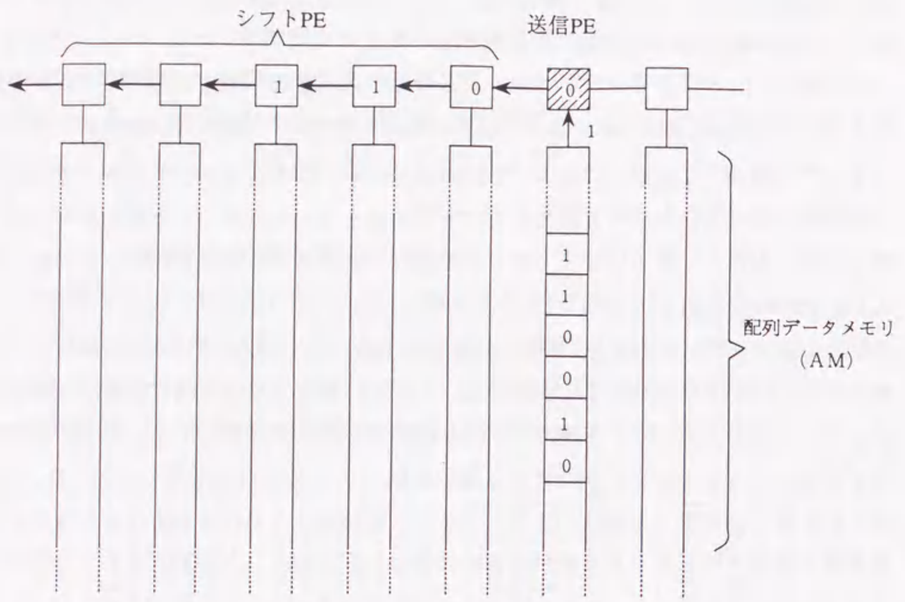


図３．２　１次元ＰＥ配列上での直並列変換

３．２．３　指定ＰＥでの転送パス間の乗り換え

並列に動作可能なＰＥ間転送パスを複数設け、その間でデータを往来させることによって、配列データのＰＥ配列上での９０度回転やミラー反転を効率良く実行することができる。

図３．３は、並列に動作する上から下方向と右から左方向に向かう転送パス間を左上角から右下角に向かう対角線上で乗り換えることによりミラー反転がかかった形の９０度回転が実現されることを示している。この転送と乗り換えの制御は、やはり制御レジスタによる個別の制御で、対角線上のＰＥのみを乗り換えＰＥに、それ以外のＰＥをパスＰＥに設定することによって実現される。この方法によれば、上下の辺間と左右の辺間をそれぞれ結ぶ標準的なトーラス接続だけで、Ｎ（配列の一辺の長さ）回の上から下と右から左への同時シフトにより９０度回転を実行することができる。もちろん、図３．４に示すように上下の辺のいずれかと左右の辺のいずれかを結ぶパスを新たに設け、それを介して転送パス間の乗り換えを行えば、同じシフト回数で９０度回転を実行することができる。しかし、この方法では、配列サイズが大きくなるに従って、プロセッサ配列の辺間を結ぶ結線の負担が重くなる欠点がある。

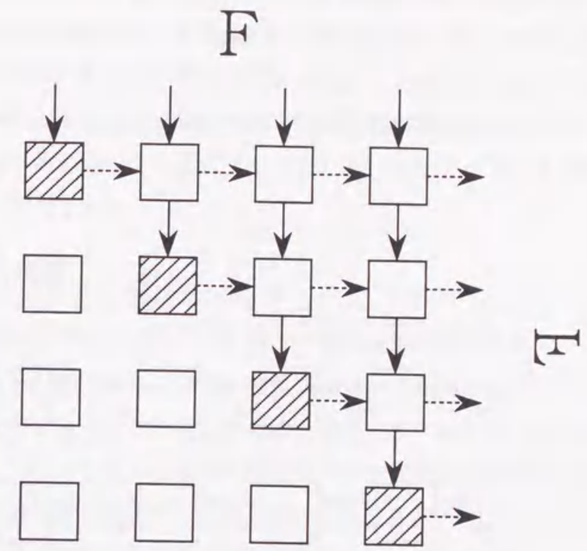


図３．３　対角線上でのパス乗換え機能を利用した９０度回転

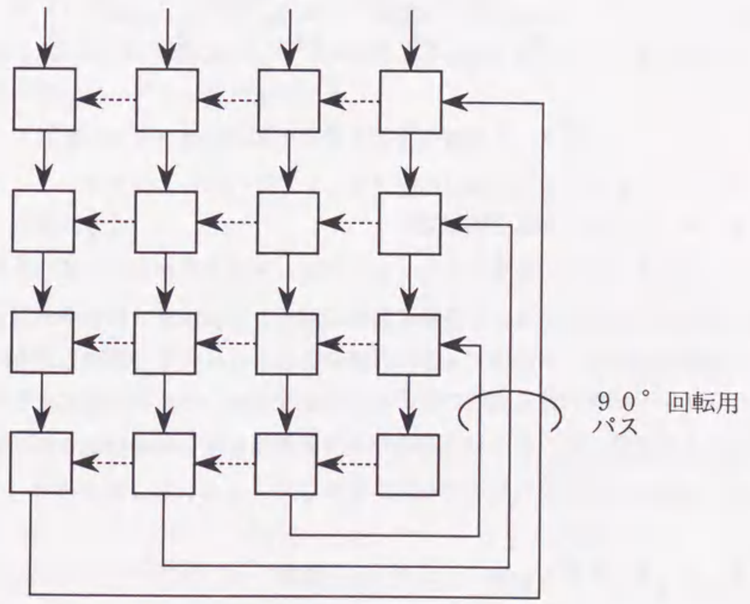


図３．４　下辺と右辺を結ぶパスを用いる９０度回転

図３．５はパス間の乗り換え機能によるミラー反転のイメージを示している。左右方向のＰＥ間結合を２重に設けて転送パスを２系統とし、それぞれの転送方向を逆に設定する。この状態で、配列の右端のＰＥのみ乗り換えＰＥに、それ以外をシフトＰＥに設定することによって、配列の右端でのパスの乗り換えが行なわれミラー反転が実現される。

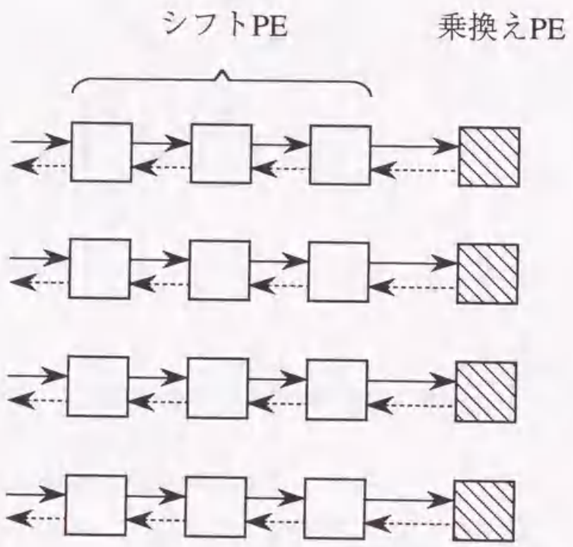


図３．５　パス乗換え機能を利用したミラー反転

３．２．４　ＰＥ毎の転送方向設定

バイトニックソート、論理シミュレーション、シストリックアルゴリズムを前提とした線形計算等[31]の応用では、ＰＥ毎に個別に転送方向を設定できるのが望ましい。この転送方向設定機能は、ＰＥ内で４方向の隣接ＰＥからの入力を選択して内部に取り込む入力セレクタの選択制御を個別の制御レジスタで行うことによって実現できる。２系統の転送パスを設け、先の転送パス間の乗り換え機能と組合わせれば、ＰＥ配列上でかなり自由度の高い転送経路の設定が可能になる。

３．２．５　ＰＥ毎のＡＬＵファンクション設定

論理シミュレーション、シストリック処理の実現には、局所的にＰＥの演算機能を変

更できることが必要である。特に論理シミュレーションにおいては、個別に演算機能を指定できれば、各ＰＥで割付けられた論理ゲートを直接エミュレートできるようになるので、高速化に大きく寄与するものと考えられる。

この演算機能の個別の指定は、従来の全ＰＥ共通の制御信号線による一括制御とは別に、各ＰＥに内蔵する制御レジスタによりＡＬＵのファンクションを個別に制御できるようにすることで実現される。

３．３ 可変構造を活用する効率的な演算法

３．３．１　伝搬加算を巧みに利用する矩形の局所領域の総和演算

前章では、１次元の配列データが複数の領域に区分けされた場合、その各領域に対する並列の総和演算が１次元のＰＥ配列上の伝搬演算により効率良く実行されることを示した。これに対し、ここでは、２次元の配列データが複数の矩形領域に区分けされた場合、その矩形領域毎の総和演算が、２次元構成のＰＥ配列上で、伝搬加算と可変構造によるＰＵ割付けを巧みに利用する階層的な演算により効率良く実行されることを示す。

図３．６はその階層的な総和演算の例である。１ビット構成のＰＥの１６ｘ１６の部分配列内で、１ＰＥに１要素（ビット直列形式）が割付けられる形で格納された同一形状の区分領域の総和をとる場合を示している。この概略の演算手順は、行方向にビット直列データに対する伝搬加算を実行したあと、直並列変換を行ない、得られたビット並列データに対して列方向の伝搬加算を行ない最終的な結果を得るというものである。以下、この演算手順について具体的に説明する。

ステップ１：１回目のビット直列データに対する伝搬加算

図３．６（ａ）に示すように各行でビット直列型の伝搬加算を行なう。

ステップ２：直並列変換

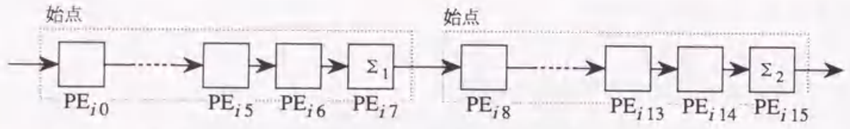
ステップ１で得られた各行の右端のＰＥに格納されている結果のビット直列データを、右端のＰＥのみ送信ＰＥに設定し、MSDの側から順に送出し、その送出ビットデータを各行で左方向に順次シフトすることによって、ビット並列データに変換する。

ステップ３：ビット並列データに対する伝搬加算

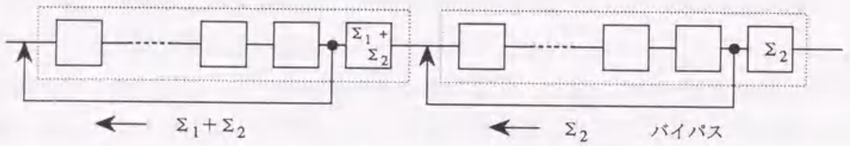
ステップ３で得られた各行のビット並列データ全体に対して、列方向の伝搬加算により総和をとり、行列要素全体の総和を得る（図３．６（ｃ））。ビット並列データに対する伝搬加算では、１６ｘ１６のＰＥ配列全体を一つのＰＵとして動作させる。各ＰＥは左方向に桁上げを下方向に和を出力する全加算器として機能させ、１６ＰＥからなる各行を１６ビット幅の加算器として動作させる。この結果、ビット並列データに対する

られる。ただし、この場合には、各行の最下位桁の指定の他に、上辺に伝搬の始点であることを指定することが必要で、これも各ＰＥの制御レジスタに所定の値を設定することにより実現する。

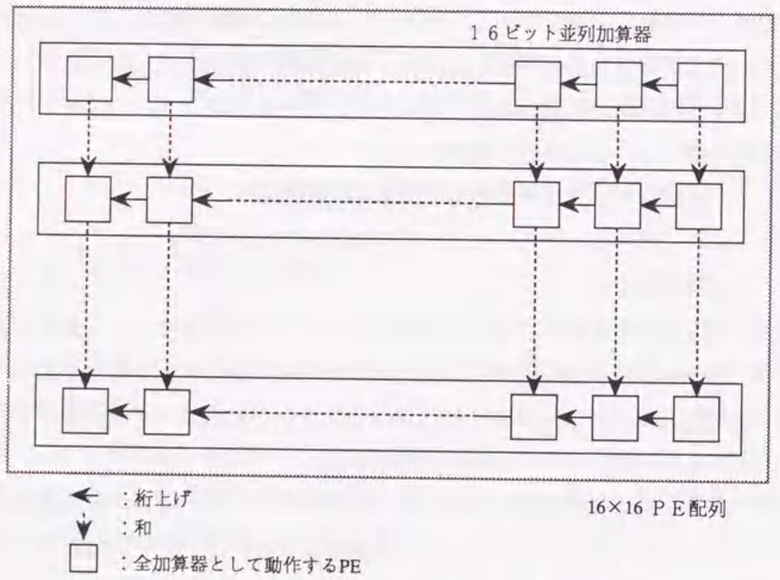
ステップ３でビット並列データに変換した後に伝搬加算を行なっているのは、ビット



（ａ）　ｉ行目における１回目のビットシリアル伝搬加算



（ｂ）　ｉ行目におけるバイパスを用いたビットシリアル伝搬加算



（ｃ）ビット並列データに対する下方向の伝搬加算

図３．６　矩形局所領域の総和演算法

直列データのままではビット単位で語長（Ｗ）の分だけ繰り返し行わなければならい列方向の伝搬加算が、パラレル形式で並列に実行することによって１回に低減されるからである。これにより短縮された演算時間ともともとのビット直列の伝搬加算を繰り返す場合の演算時間との比率Ｒは、制御のオーバーヘッドを無視すると、

$$R = \{(t_{sp} + t_c) \times W + t_{pp}\} \ / \ \{(2\,t_{sp} + t_c) \times W\}$$

となる。ここで、右辺の分子側は可変構造を利用する総和演算の演算時間で、分母側はビット直列の伝搬演算のみで得た総和結果に直並列変換を行い、最終的なパラレル形式の総和結果を得るまでの演算時間を表している。また、t_{sp}はビット直列データのビット単位の伝搬加算時間、t_{pp}は、ビット並列データに対する列方向の伝搬加算時間、t_cはビット単位の直並列変換時間である。この式から明らかなように、伝搬演算機構の構成が単純で、１６ｘ１６のＰＥ部分配列に対し、t_{sp}＝４クロックサイクル、t_c＝１クロックサイクル、t_{pp}＝８クロックサイクル、Ｗ＝１６とすると、Ｒは０．６１程度と、もともとの演算時間の２倍近くまで高速化されることがわかる。ただし、ツリー構成の高度の伝搬演算機構が組込まれている場合には、効果は小さくなる。例えば、セレクティブツリー構成の伝搬演算機構により、ビット単位の伝搬加算が、t_{sp}＝１クロックサイクルまで高速化されている場合には、Ｒは０．８３となる。また、当然ながらＰＥ間で直接ビット並列データに対する伝搬加算が実行できる機構が組込まれている場合には、効果はない。

３．３．２　論理回路シミュレーション

ＬＳＩの集積規模の拡大にともなう設計上の切実な要求の一つに、論理シミュレーションの高速化がある。セルラ配列型ＳＩＭＤプロセッサはこの論理シミュレーションの基本処理が、可変構造を利用することによって、エミュレーションの形で高速に実行できる。図３．７は、エミュレートに必要な構成要素のみを記述したＰＥのブロック構成である。図３．８（ａ）はこのブロック図を用いてＰＥ１個に論理ゲート１個を割り付けて２－１セレクタをエミュレートする場合の各論理ゲートのＰＥ配列への割付け方を示している。図（ｂ）はその論理ゲートを実現するためのＰＥ配列の内部接続構成を示している。

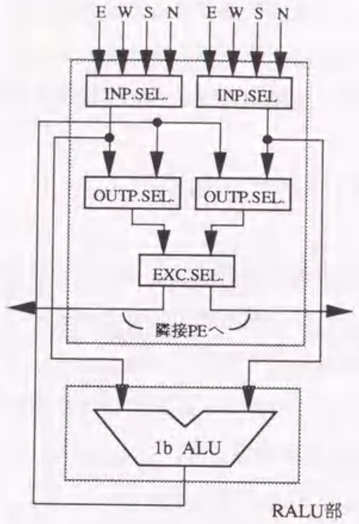
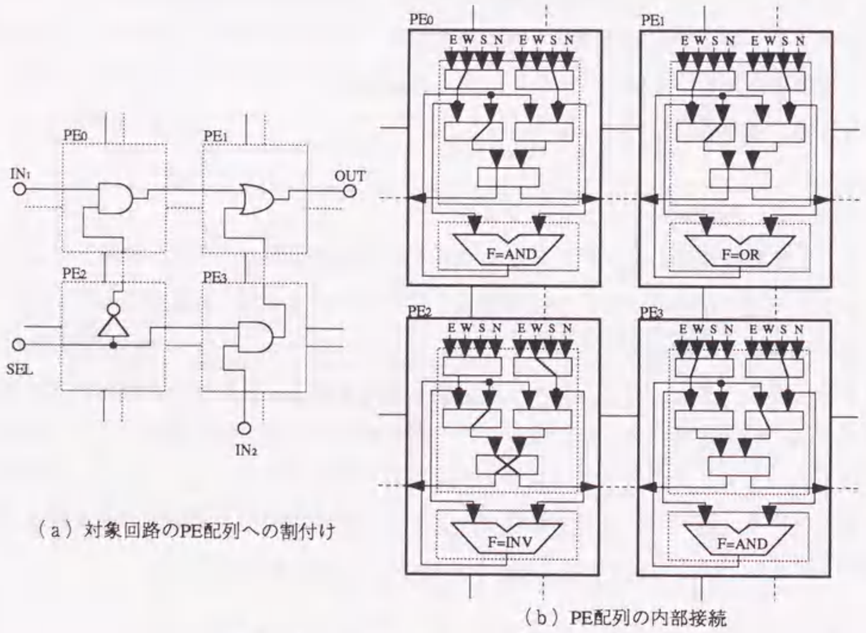


図３．７　ＰＥのブロック構成



（ａ）対象回路のPE配列への割付け

（ｂ）PE配列の内部接続

図３．８　論理回路のエミュレーション

ここで、ＡＬＵへの入力経路とＰＥ間の信号経路となる転送パスの設定は、転送方向設定機能すなわち制御レジスタによりデータ転送ユニット内の入力セレクタ（ＩＮＰ．ＳＥＬ．［input selector］）を適当に選択することにより実現される。これを図ではセレクタの箱の入力と出力を直線で結ぶことにより示している。また、ＰＥ２ではパス間の乗り換え機能（ＥＸＣ．ＳＥＬ．［exchange selector］）により、左側のパス１から入力された信号をそのまま右側のパス２に出力している。

以上説明した動作から明らかなように、この演算は論理回路をソフトウェアでシミュレートするのではなく、ハードウェアで直接的にエミュレートする。従って、配列の規模を大きくして、配列上に実際のハードウェアと等価な論理回路を実現すれば、汎用コンピュータ上のソフトウェアに比べ、２桁から３桁以上の速度の高速化が可能となる[32]。

３．４　むすび

全ＰＥ共通の命令を、ＰＥ毎に付加する制御レジスタで個別に修飾することによって得られる可変構造的なＰＥ配列の各種の機能について述べた。

はじめに、基本的な機能として

①ＰＥを連結して並列の加算単位を構成するモードで、固定値桁上げ入力の生成と下位桁ＰＥからの桁上げ入力を制御レジスタを用いてＰＥ単位で個別に制御することによって、任意のビット幅の加算単位を任意の位置に構成できること、

②送信ＰＥとシフトＰＥを制御レジスタにより指定し、送信ＰＥの保持するシリアルデータをシフトＰＥに送りだすことによって直並列変換が実現できること、

③ＰＥ間シフト転送パスを２系統設け、その間の乗り換えの制御をＰＥ単位で個別に制御することによって、９０度回転、ミラー反転が実現できること、

④ＰＥ間の転送方向、ＡＬＵファンクションを制御レジスタにより個別に設定できること

を示した。

次いで、この①、②の機能を巧みに組合わせて階層化する伝搬加算法により、ＰＥ配列内に設定する任意の矩形領域内のビット直列データ配列の総和が、階層化しない場合に比べ２倍近い速度で実行できることを示した。また、④の機能を用い１個のＰＥに１ゲートを割付け、ゲート間の結線を伝搬転送と③、④の機能を利用してエミュレートすることによって、論理シミュレーションを高速に実行できることを明らかにした。

第４章　データの転送、回転、およびメモリアクセス機構

４．１　はじめに

近年に商用化されたセルラ配列型ＳＩＭＤプロセッサでは、ハードウェア量の大半を占めるプロセッシングエレメント（ＰＥ）配列部を、１チップに１６個から３２個程度のＰＥを搭載したプロセッサ配列型のＬＳＩで構成し、装置の小形化、経済化がはかられている。当然ながら、一層の性能／コスト向上には、このプロセッサ配列型ＬＳＩのチップ当たりの搭載ＰＥ数の拡大が望ましい。しかし、２次元あるいは３次元以上のＰＥ間ネットワークとＰＥ配列と一体化するＬＳＩ構成では、搭載ＰＥ数をこれ以上大きくすることは期待できない。これは、転送性能，メモリアクセス性能等を維持するのに必要となるＰＥ間パス用，局所メモリアクセス用等に必要な信号数が、パッケージの許容端子数を越えてしまうからである。

これに対し、１次元の接続構成のネットワークと一体化する構成では、面あるいは立体を単位とする高並列処理は困難になるものの、隣接ＰＥ間パス用に必要な端子が、搭載ＰＥ数によらず、左右の隣接ＬＳＩとの間を結ぶ２組分のパス用だけと少ない。従って、この分、搭載ＰＥ数が大きくでき、２次元構成のＳＩＭＤプロセッサに比べると、小形化、経済化の点で有利となる。この点が評価され、最近では、大容量メモリに１２８個のＰＥを一体化したＬＳＩの提案[33]、６４ないし１０２４個ものＰＥを１チップに搭載したＬＳＩの開発[34,35]等が行われている。しかし、これらのＬＳＩでは、汎用性よりも、搭載ＰＥ数の拡大を優先しており、

①間接アドレシングによるデータアクセス、

②２次元アクセス（２次元配列データから、行単位の１次元配列データ［以下、単純に行と呼ぶ］あるいは列単位の１次元配列データ［以下、単純に列と呼ぶ］のいずれでもアクセスできる機能[36-39]、

③遠隔ＰＥ間の高速転送

等の１次元ＳＩＭＤプロセッサの汎用化に必須とも言える機能が、全く組込まれていないか、組込まれていても同一チップ内に閉じる機能に限られていた。

さらに、本論文の主な研究対象である２次元構成のＳＩＭＤプロセッサにおいても、①から③の機能の強化は性能向上に大きく寄与する。これは、８章以降の文字認識処理への応用の経験から、並列度の低い小型の２次元ＳＩＭＤプロセッサでは、２次元の配列構成を活かした面単位の演算をベースに処理するよりも、ＰＥ間の転送パスを強化した１次元構成のＳＩＭＤプロセッサとみなし、各ＰＥが同時に動作することによって実現されるライン単位の演算をベースに処理を進める方が、プログラムの記述性に優れるばかりか、処理の効率も高いことが明らかになってきたからである。

そこで、本章では①、②の機能を強化するために、はじめに従来１次元構成のＳＩＭＤプロセッサに用いられていたメモリアクセス機構の問題点を明らかする。次いで、この問題を解決するメモリアクセス機構として、基本的な１次元ＳＩＭＤプロセッサに組込まれるルーティングレジスタ（ＲＴＲＧ）配列を、Ｗ個のＰＥからなる部分配列単位に、直並列変換型９０度回転器としても、外部メモリ用のアドレス生成器としても利用する構成（シフト転送と回転の機能を合せ持つことから転送回転型メモリアクセス機構と呼ぶ）を提案する。さらに、その性能、ハードウェア規模などを従来の行、列いずれのアクセスも１サイクルで実行する速度性能の点で理想的な構成の２次元アクセス機構（以下、従来型２次元アクセス機構と呼ぶ。）[39]と比較評価する。なお、③の問題を解決する構成については、２章で論じた伝搬演算機構が該当するので、ここでは触れない。

４．２　従来のメモリアクセス機構の問題点

４．２．１　メモリアクセス方式とその有用性

最も基本的なものは、各ＰＥが各々の局所メモリを、全ＰＥ共通のアドレスで並列にアクセスするメモリアクセス方式である。いわゆるＳＩＭＤプロセッサ向きといわれる規則性の高い処理のかなりの部分がこのメモリアクセス方式のみで実行できる。このため、このメモリアクセス方式のみサポートするＳＩＭＤプロセッサも数多く開発されている[11-15]。

しかし、このメモリアクセス方式だけでは、一見ＳＩＭＤ向きと思える処理でも、ＰＥごとでの関数計算のようにうまく処理できない場合がある。その例としては、ニューラルネットワークシミュレーションにおけるシグモイド関数、三角測量処理における三角関数、画像処理におけるヒストグラム解析等があげられる。この種の計算は、各ＰＥが局所メモリに格納される配列データの異なるアドレスを並列にアクセスすることで効率良く処理できるが、そのためには、各々のＰＥで異なるアドレスを生成するための間接アドレシングによるデータアクセス機能がＰＥごとに必要になる。

さらに、１次元ＳＩＭＤプロセッサには、非常に有用性の高いメモリアクセス方式として、２次元配列データに対し、行，列のいずれに対してもアクセス可能とする方式（２次元アクセス方式）がある。行あるいは列の一方のみの１次元配列データのアクセスでは、効率良く処理できない場合が少なくないからである。例えば、斜交軸変換による画像のアフィン変換[40]，行，列両方向の密度分布を求める文字の特徴抽出処理（８章

参照），行，列両方向の１次元ＦＦＴを繰り返す２次元ＦＦＴ[39]等では、一連の行単位の処理に続き、やはり一連の列単位の処理を行なう必要がある。

４．２．２ 実現上の問題点

より多くのＰＥを１チップに搭載しようとすると、集積度の制約から各ＰＥの局所メモリは、同一チップ内に全量を搭載するわけには行かない。この場合、チップ内のＰＥから外部に置く局所メモリを高速にアクセスする必要があり、そのためには多数のデータアクセス用の端子を設けねばならない。また、ＰＥの機能の向上をはかろうとすると、ＰＥ配列部の複数のＬＳＩによる分割構成が避けられず、そのために隣接ＬＳＩ間の接続に用いる信号端子も必要になる。これらの端子はもちろん、チップ全体の端子数の制約から無制限に設けるわけには行かない。従って、外部の局所メモリに対して、分割単位のＬＳＩごとに、より少ない端子数で各種メモリアクセスを効率良く行う仕掛けが必要になる。

実際、間接アドレシングによるデータアクセスについては、各ＰＥで完全に並列に行なう代りに、同一ＬＳＩ内のＰＥ同士が、一組の間接アドレス生成機構を時分割的に使用することによって、ＬＳＩの端子数とハードウェア規模の低減をはかる方法がある[17]。

しかし、２次元アクセスについては古くから検討されてはいるものの、ハードウェア規模低減の観点から理想的といえる方式、すなわちＰＥ部分配列を内蔵する同一構成の大規模ＬＳＩとそのＰＥごとの局所メモリの配列（メモリユニット）だけで分割単位を構成するような方式は見当たらない。例えば、従来型２次元アクセス機構[36-39]では、ＰＥ部分配列の１チップ化は現在の大規模ＬＳＩを用いたとしても、端子数の制限から実現は容易でない（表４．１参照）。これは、ＰＥごとに個別のアドレス端子を必要としたり、バレルシフタ、フリップネットワークのような分割構成時の相互の結線数の大きい高速ルーティングネットワークが必要となるからである。

４．３ 提案する転送回転型メモリアクセス機構の構成

転送回転型メモリアクセス機構は、基本的な構成のＳＩＭＤプロセッサがもともと備えている隣接ＰＥ間の転送パスを活用することによって、各種のメモリアクセスを効率的に実行することをめざす。従来型２次元アクセス機構のように高度なネットワークは用いないので、端子数、ハードウェア規模の増加は小さく、基本的な１次元ＳＩＭＤプロセッサの備えるＬＳＩ化時の搭載ＰＥ数の拡大容易性はほとんど低下しない。

転送回転型メモリアクセス機構を組込んだ１次元セルラ配列型ＳＩＭＤプロセッサの

並列演算部の構成を図４．１に示す。この並列演算部は、一点鎖線で囲まれるＰＥ部分配列とＲＴＲＧ部分配列にメモリユニット（局所メモリの部分配列）を外付けして１つの単位とし、これを規則的に並べることで構成される。ＰＥ部分配列とＲＴＲＧ部分配列は、図４．２に示すように、それぞれW個のＰＥとＲＴＲＧで構成され、幅Wのバスでメモリユニットと接続されている。ここで、ＲＴＲＧ部分配列は、ＰＥ部分配列の各ＰＥが縦続接続されることによってその内部に形成されるＲＴＲＧの配列を、抜き出して示したものである。従って、図４．２に示すようにレジスタ要素（ＲＥ）のＷｘＷ個の行列状の２次元配列が、左右の隣接レジスタ要素間のパスと列ごとのパスで結合される構成となっている。

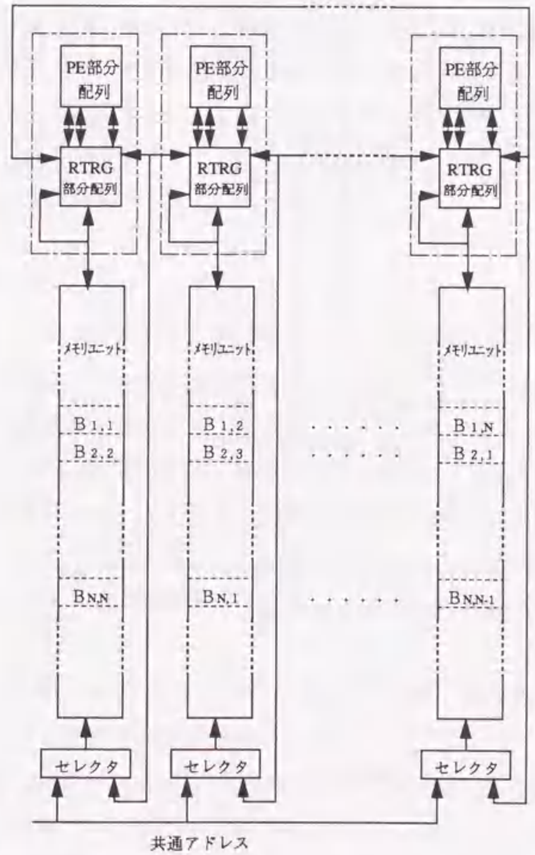


図４．１ 転送回転型メモリアクセス機構を組込んだ１次元ＳＩＭＤプロセッサのＰＥ配列部の構成

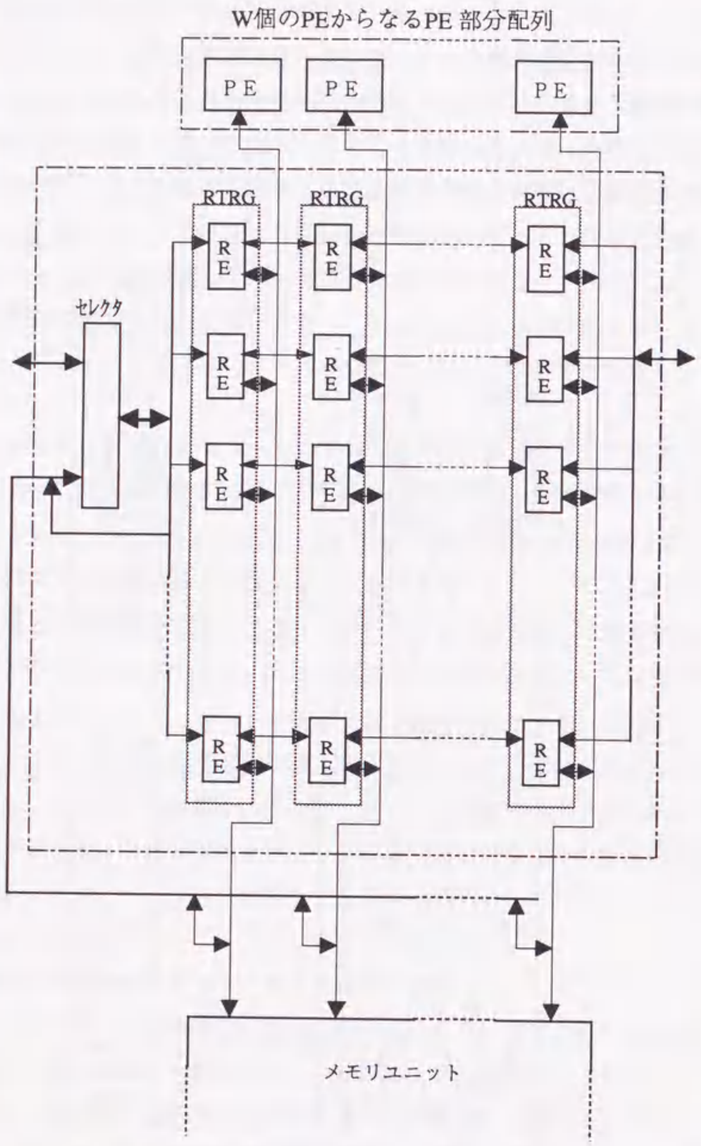


図４．２　ルーティングレジスタ（ＲＴＲＧ）部分配列の構成

転送回転型メモリアクセス機構は、メモリユニットに対して、

①ＲＴＲＧ部分配列右端からの出力を下側のセレクタを介して、メモリユニットのアドレスとして加えられるようにしたこと（図４．１）、

②ＲＴＲＧ部分配列の列ごとのバスを介する上下方向の入出力パスを設けると共に、ＲＴＲＧ部分配列の２次元配列の左端にセレクタを介した左右方向の入出力パスを設けたこと（図４．２）

に特徴がある。これらによって、部分配列ごとに、ＲＴＲＧ部分配列からの出力によるメモリユニットのアドレシング、メモリユニット内のＷｘＷのブロック単位の配列データに対する９０度回転等が可能となる。

４．４　メモリアクセス法

転送回転型メモリアクセス機構では、並列演算部のＲＴＲＧ間の本来のデータ転送機能に、ＲＴＲＧ部分配列ごとの回転機能、メモリユニットに対するアドレシング機能を組合わせることによって、間接アドレシングによるデータアクセスと、従来型、疑似型、間接型の３タイプの２次元アクセス方式が実現される。以下、これらのアクセス法について所要サイクル数の算出も含めて説明する。

４．４．１　間接アドレシングによるデータアクセス

間接アドレシングによるデータアクセスは、テーブル参照だけでなく、次節以降の各種２次元アクセスを実現するための基本機能として重要である[41]。転送回転型メモリアクセス機構では、この間接アドレシングによるデータアクセスがＲＴＲＧ部分配列の出力によるメモリユニットに対するアドレシング機能とブロック単位の配列データに対する９０度回転機能を組合わせることによって実現される。具体的には、図４．３に示すように、以下の①，②を並行して行うことで実現される。

①シフト転送によりＲＴＲＧ部分配列から出力される$A_W, A_{W-1}, \ldots, A_1$でメモリユニットをアドレシングする。

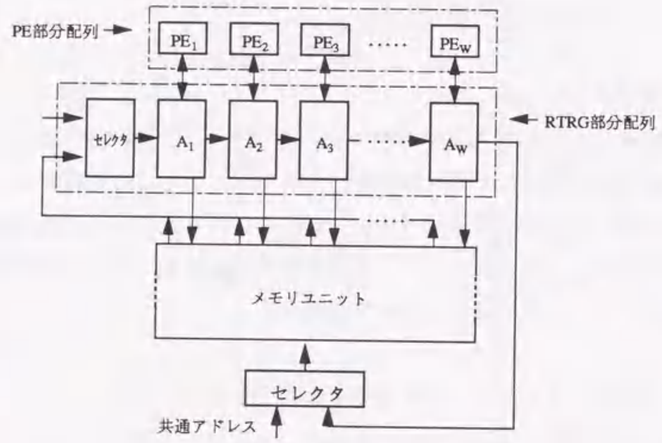
②①のアクセスデータ列［A_W］，［A_{W-1}］，....，［A_1］を、左端からシフト転送により９０度回転する形でＲＴＲＧ部分配列に戻す。

ここで、［A_i］はメモリユニットのA_i番地のワードデータであること、すなわちこの間接アドレシングによるデータアクセスがメモリユニットのワードデータの配列に対するものであることに留意されたい。

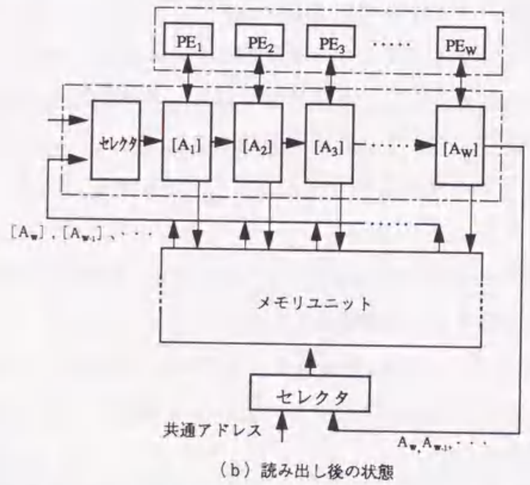
この間接アドレシングによるデータアクセス方式には、ハードウェア規模を低減する

上で、２つの利点がある。１つはテーブル参照処理において、テーブルの格納領域を１／Ｗにまで低減できることである。これは、間接アドレシングによるデータアクセスの対象となるワード単位のデータ配列が、ＰＥ部分配列単位で共有されるからである。もう１つは、従来方式[17]で時分割使用により間接アドレシングによるデータアクセスを実現していたＰＥ部分配列単位のデータバスとアドレスバスが不要になることである。

ところで、このアクセス法では、A_1，A_2，・・・・，A_Wのアドレス値の算出は、



（ａ）読み出し前の状態



（ｂ）読み出し後の状態

図４．３　転送回転型メモリアクセス機構での間接アドレシングによるデータアクセス

各ＰＥの並列処理により効率良く計算される。しかし、メモリユニットからのアクセスは１ワード（１ＰＥ分）単位と逐次的であり、この分についてのアクセス速度の向上はない。ただし、セルラ配列構成のチップのように、搭載ＰＥ数が大きく、端子数が制限されて、アクセス速度が端子の稼働率だけで決まる場合には、他の並列化可能な方式を採用したとしても意味はない。すでに、この方式でメモリユニットアクセス用の端子を、アドレス供給とデータのアクセスのために１００％稼働させており、並列化の効果を活かせないからである。

４．４．２　従来型２次元アクセス

従来型２次元アクセスは、従来型の２次元アクセス機構で用いているアクセス方法である[36-39]。転送回転型メモリアクセス機構では、このアクセス法をＲＴＲＧ配列によるＰＥ間シフト転送機能に前節の間接アドレシングによるデータアクセス機能を組合わせることによって実現できる。

ただし、アクセス対象は、行ごとに要素単位でずれる形式（スキュード配列［Skewed Array］形式）[37]の２次元配列データであることが必要となる。このため、メモリユニットの配列からアクセスされる各行がＰＥ配列側でねじれた形式となるように、メモリユニットに対してＰＥ部分配列ごとに回転をかけながら書込みあるいは読み出しを行う[42]。これによって、各メモリユニットに行の１要素が１ワードに入る形式で格納され、間接アドレシングによるデータアクセス機能により、任意の列を行同様に効率良くアクセスできるようになるからである。

このアクセス法での１行および１列のライン当たりの所要サイクル数C_{rc}，C_{cc}は、並列演算部の分割数をＮ、間接アドレシングによるデータアクセスのアドレス生成の所要サイクル数をＡ、サイクル当たりのシフト量をＳとすると、要素の語長がＷに等しい場合で、

$$C_{rc}=W+(W\times N)/(4S) \qquad (4.1)$$

$$C_{cc}=W+(W\times N)/(4S)+A \qquad (4.2)$$

となる。これらの式で、右辺第１項はメモリユニットからねじれた形式の行あるいは列をそのままアクセスするのに必要なサイクル数である。また、右辺第２項は、そのねじれをＰＥ間のシフト転送機能を利用して補正するのに必要なサイクル数であり、Ｎが偶数で逆方向のシフト転送も利用する条件でのずれ補正のための平均シフト量

（ＷｘＮ）／４に、サイクル当たりのシフト量Ｓを考慮して求められる。

４．４．３　疑似２次元アクセス

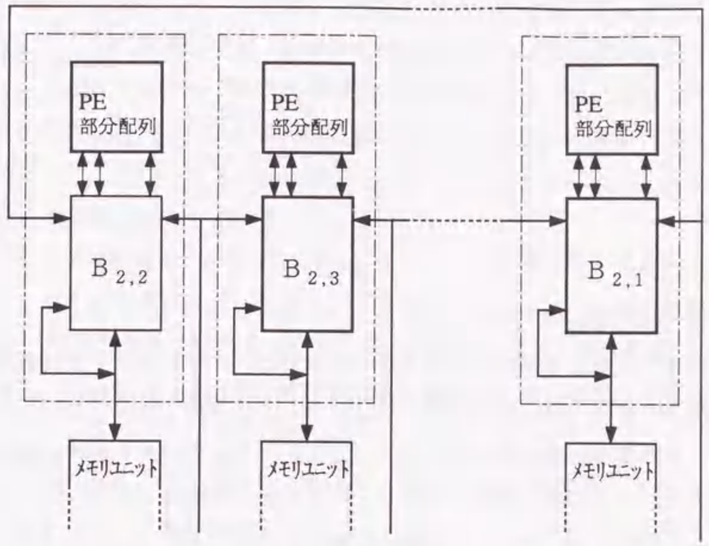
本アクセス方法は、従来型２次元アクセスのように行あるいは列を直接１本ずつアクセスするのではない。一旦、行あるいは列方向のビットライン（要素の語長が１ビットの１次元配列）Ｗ本からなる横長あるいは縦長の短冊状配列データをアクセスし、これを介して、行，列両方のアクセスを実現する。この疑似的な２次元アクセス（以下、疑似２次元アクセスと呼ぶ。）における短冊状２次元配列データのアクセスは、ブロック単位のスキュード配列形式配列データ（図４．１のメモリユニット配列に格納される$B_{1,1}$～$B_{N,N}$のブロック配列データ参照）に対し、ブロックを単位として従来と同様な２次元アクセスを行うことによって実現される。ここで、ブロックとは各メモリユニットに格納される幅Ｗのワードデータの Ｗ本の集りである。

以下、図４．１の配列データに対する短冊状２次元配列データのアクセスについて具体的に説明する。はじめに、この配列データに対し、各メモリユニット共通のアドレスで、例えば２段目のブロックを並列に読み出すと、ＲＴＲＧ部分配列の並びに図４．４（ａ）のブロック単位の配列データが得られる。この図から明らかなように、ＲＴＲＧ配列の転送機能により右方向に１ブロック分シフトし、ずれを補正すれば、目的の横長の短冊状配列データが得られる。また、次の２つの処理、

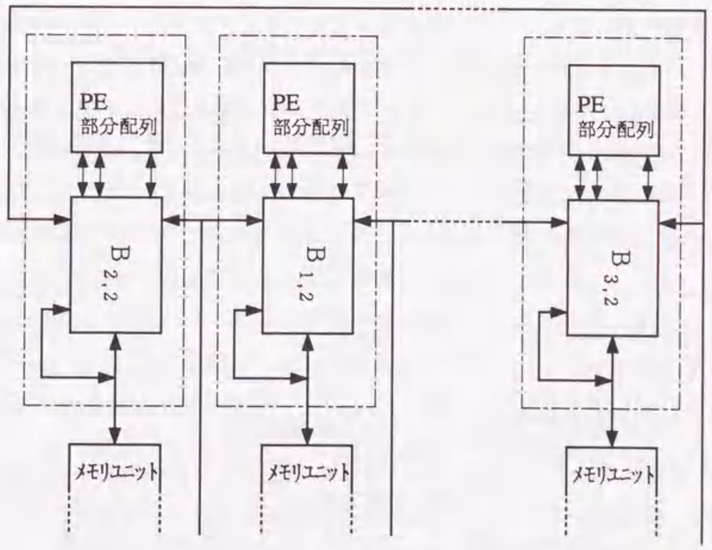
①右上がりの対角方向の$B_{2,2}$，$B_{1,2}$，$B_{N,2}$，$B_{N-1,2}$・・・・，$B_{3,2}$の各ブロックデータを、先の間接アドレシングによるデータアクセス機能を利用して(アドレス値は、各ＰＥの演算機能を利用して算出する）必要なアドレスを分割単位ごとに生成することによって、並列に読出す、

②①で読み出されたワード単位のデータを、左端のセレクタを介して、回転をかけながら列単位の配列データとして各ＲＴＲＧ部分配列に、順次、取込む

を同時に行うと図４．４（ｂ）に示すブロック単位の配列データが得られる。これを先の場合と同様に、２ブロック分だけ左方向にシフトして、ずれを補正すると２列目の縦長の短冊状配列データが横長に変換された形で得られる。ここで、補正のためのシフト量は、容易に導かれるように、Ｎが偶数の場合、逆方向のシフトも利用すると短冊状配列データ当たりの平均で（ＷｘＮ）／４となる。従って、短冊状配列データのアクセスとそれから行、列を切出してアクセスすることを並行して実行できるようにハードウェアを構成すれば、短冊状配列データを構成する行、列を一通りアクセスする場合の１行



（ａ）横長の短冊状配列データの読み出し



（ｂ）縦長の短冊状配列データの読み出し

図４．４　転送回転型メモリアクセス機構による疑似２次元アクセス

および1列のビットライン当たりの所要サイクル数C_{rp}, C_{cp}は、

$$C_{rp} = \lceil W+(W \times N)/(4S) \rceil /W$$
$$= 1+N/(4S) \quad (4.3)$$

$$C_{cp} = 1+N/(4S)+A/W \quad (4.4)$$

となる。

4．4．4　補正無しの2次元アクセス

これまでの従来型および疑似2次元アクセスでは、アクセスデータのずれをRTRG配列を利用したシフト転送により一々補正することを前提としていた。しかし、行あるいは列を、その並びの順に順次アクセスすることによって行なう線順次処理[43]においては、アクセスデータのずれ補正の代りに、演算の途中結果の方をずらすことでも対応可能であり、これによってシフト転送量を低減できる場合がある。

実際、従来型の2次元アクセスでは、この方法でメモリユニットからのデータアクセスと、演算の途中結果の左あるいは右方向の1要素分のシフトを同時に行うことにより、ずれ補正のオーバーヘッドをまるまるなくすことができる。これは演算対象の2次元配列データのスキュード配列が、1行あるいは1列ごとに1要素ずつずれていく形式であり、1要素分ずつのシフトで、アクセスデータのずれに合せられるからである。従って、この場合の1行および1列のライン当たりのアクセスの所要サイクル数C_{rc}, C_{cc}は、

$$C_{rc}=W \quad (4.5)$$
$$C_{cc}=W+A \quad (4.6)$$

となり、アクセス速度は従来型2次元アクセス機構に等しくなる。

しかし、疑似2次元アクセスについてはこの方法だけでは処理効率の改善はできない。これは、演算対象のスキュード配列形式の2次元配列データがW行あるいはW列ごとにW要素ずつ階段状にずれる形式であることから、演算の途中結果のシフトをW行あるいはW列分の処理ごとにW要素分まとめて行なう必要があり、ここまで前提としてきた1次元構成のPE配列では、結局、W／Sサイクルかかってしまうからである。

この疑似2次元アクセスを効率良く実行するには、本章で前提としている1次元の接続構成ではなく、2次元の接続構成が有効である。PEの2次元配列（例えば図1．1）の左右の端を一段ずつずらして接続し、各行を1つのPE部分配列として機能させれば、縦方向のPE間の隣接接続が、図4．1のW要素分のシフトをまとめて行うためのバイパスとして機能するからである。この縦方向の接続をバイパスとしてのみ用いる場合には、PE間の左右方向転送とは同時に動作することはないので、プロセッサ配列部の分割構成時にその接続に必要となる端子を、横方向の1次元配列を形成するために必要となる端子に多重化することができる。

従って、プロセッサ配列が2次元の接続構成を有し、かつ縦方向のPE間の接続のビット幅が、1である場合には、1行および1列のビットライン当たりの所要サイクル数C_{rp}、C_{cp}は、横長の短冊状配列データの1回のアクセス（W本の行あるいは列のアクセス）に対し、1ブロック分の割合で演算の途中結果をシフトすることになるので、

$$C_{rp} = (W+L/S)/W$$
$$= 1+L/(W \cdot S) \quad (4.7)$$

$$C_{cp} = 1+A/W+L/(W \cdot S) \quad (4.8)$$

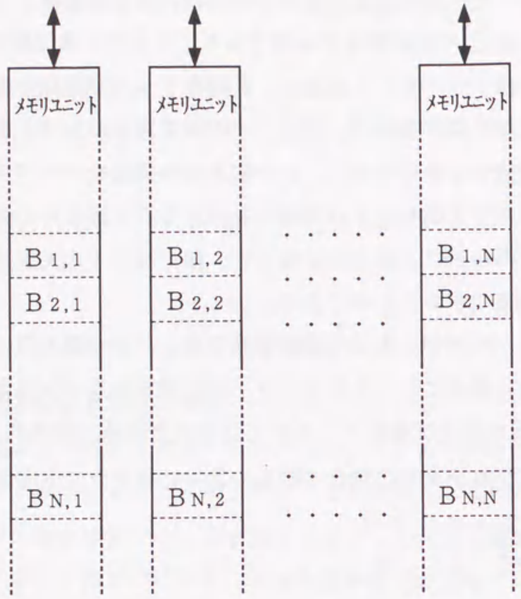
となる。ここで、Lは演算の途中結果の語長である。

4．4．5　間接的な2次元アクセスとそれに用いる90度回転

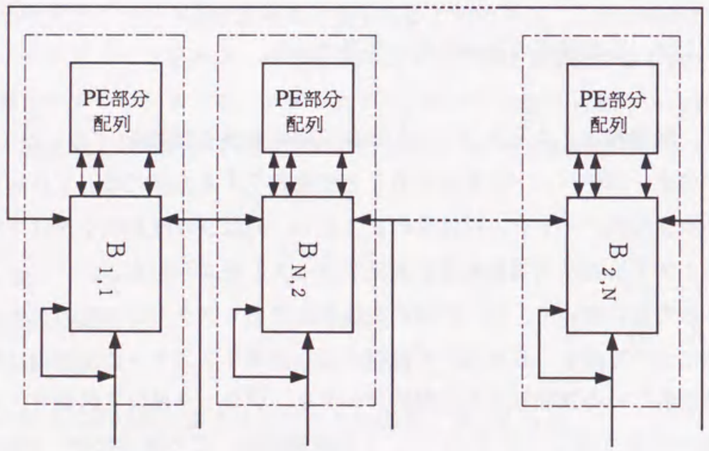
ずれ補正のオーバーヘッドを低減することで高速化する方法には、ずれ無しの形式で格納する2次元配列データを、行はそのままとし、列は90度回転をかけて行に変換してからアクセスする方法（間接的な2次元アクセスと呼ぶ）がある。

以下、この方法において、行，列間の変換手段でありアクセス速度の大半を決める90度回転について図4．5を用いて説明する。メモリユニットの配列内にずれ無しの形式で格納されている被回転2次元配列データを、ブロック単位に区切り、それぞれを$B_{1,1}$～$B_{N,N}$とする（図4．5（a））。90度回転は、この被回転データに対し、

①左からj番目のメモリユニットから、被回転2次元配列データの
$\lfloor mod((i-j+N+1)/N),j \rfloor$ の位置のブロックデータを、間接アドレシングによるデータアクセス機能を利用して、RTRG部分配列に回転をかけながら読み出す
（図4．5（b））。

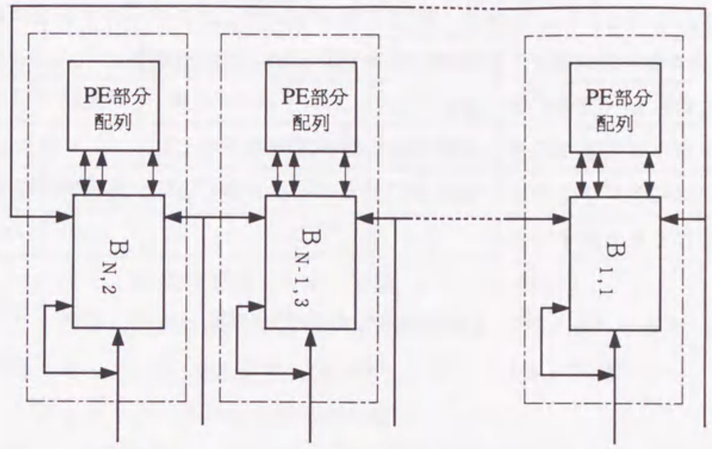


（a）メモリユニット配列内に格納される被９０度回転データ

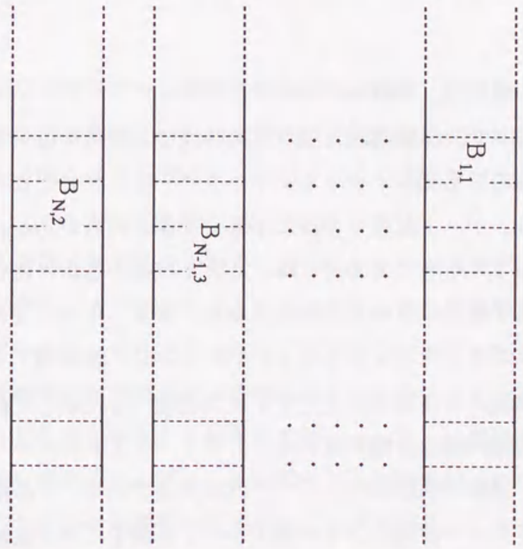


（b）ブロックデータB(mod((i-j+N+1)/N),jのＲＴＲＧ配列に対する回転をかけながらの読み出し。

図４．５　転送回転型メモリアクセス機構による９０度回転の手順



（c）ＲＴＲＧ部分配列に読み出したブロックデータの１ブロック（ｉ＝１）分左方向にシフトした結果。



（d）（c）のシフト結果の((mod((i+j)/N),j)の座標位置への書込み

図４．５　転送回転型メモリアクセス機構による９０度回転の手順

②ＲＴＲＧ部分配列間で、①のアクセス結果を左方向にｉブロック分シフトし（図４．５（ｃ））、結果を、間接アドレシングによるデータアクセス機能を利用して、メモリユニットの回転後の２次元配列データの格納先の ｜mod((i+j)/N),j｜ の位置に書込む（図４．５（ｄ））。

の処理を１からＮの各ｉに対して繰り返し行って実現される。

行あるいは列のビットライン当たりの所要サイクル数C_{90}は、処理の単位がブロック単位であることを考慮すると

$$C_{90} = 2 + (W \times N / (4S) + 2A + Q) / W$$
$$= 2 + N/(4S) + 2A/W + Q/W \qquad (4.9)$$

となる。式（４．９）で、右辺第１項はメモリユニットとの間の読出し、書込みのサイクル数である。右辺第３項でＡに２を乗じているのは読出し、書込みの両方で間接アドレシングによるデータアクセスが必要になるからである。右辺第４項のＱは、②の間接アドレシングによるデータアクセスでＲＴＲＧをアドレス生成に利用できるように、シフト結果を一旦ワークレジスタに転送するのに要するサイクル数である。

４．５　評価

４．５．１　構成の規則性，単純性

転送回転型メモリアクセス機構は、速度性能の点で理想的な従来型２次元アクセス機構に比べると、高速のＰＥ間ルーティングネットワーク、ＰＥごとの個別のアドレス修飾器等が不要であり、ゲート規模を格段に小さくできる利点がある。しかし、ＰＥ配列部の分割単位のＬＳＩ化を考える場合には、ＬＳＩ技術の進歩の著しい現在ではゲート規模よりもむしろ端子数の多さの方が問題になる。表４．１は、この端子数について、ＰＥ間のルーティングネットワークとして、バレルシフタを付加すると共に、アドレス修飾器をＰＥごとに設ける従来型２次元アクセス機構[37]と比較した結果である。配列サイズが２５６のＰＥ配列部を分割単位であるＬＳＩ（３２ＰＥ／チップ）８個で構成する場合の端子数を、実際の数値例として［］内に示している。この構成条件では、端子数が従来型２次元アクセス機構の３０４端子から９６端子にまで低減されている。この表の各式および実際の数値例から明らかなように、従来型データアクセス機構では、１チップ化のための端子数制限を越えてしまう場合でも、転送回転型メモリアクセス機構ならば容易に制限内に収められることがわかる。

表４．１　分割単位（ＬＳＩ）の端子数の比較

		従来方式	本方式	本方式／従来方式
分割単位の端子数		W (N+1)+a [304]	3W [96]	3／((N+1)+a／W) [0.32]
内訳	転送用	NxW [256]	2W [64]	2／N [1／4]
	ﾒﾓﾘｱｸｾｽ用	W [32]	W [32]	1 [1]
	ﾒﾓﾘｱﾄﾞﾚｽ用	a [16]＊２	0	－

注１）W:部分配列長，N:分割数，a:アドレスの幅

注２）処理対象配列データの要素語長は１ビット。

＊１：EOR Skewed Array方式[37]

＊２：ａ＝１０とした場合の値

表４．２　ビットライン当たりの所要マシンサイクル数ＭＳの比較

	従来機構	本機構	従来機構／本機構
２次元アクセス	1	1+N／(4S)+A／W	1／(1+N／(4S))
９０度回転	2	2+N／(4S)+2A／W+B／W	1／(1+N／(8S))

４．５．２　従来型２次元アクセス機構との高速性比較

４．４で示した各メモリアクセス法および９０度回転の所要サイクル数を、従来型２次元アクセス機構のそれと合せて表４．２に示す。このままでは比較が困難なので、実際の数値を求める。

ＰＥ内のアドレス計算、ＲＴＲＧからＰＥ内のワークレジスタへの転送等が比較的簡単な演算であることから、Ａ＝２，Ｑ＝１のハードウェアを構成できるとする。これより、Ａ／Ｗ（＝１／１６）、２Ａ／Ｗ（＝１／８）、Ｑ／Ｗ（＝１／３２）等は小さいとして無視でき、従来型２次元アクセス機構に対する所要サイクル数の増分はほぼＮ／（４Ｓ）になることがわかる。これをもとに、４．５．１の構成条件でＳ＝１の場合の所要サイクル数を求めると、表４．３に示すように従来型メモリアクセス機構に比べ、１／２～１／３にアクセス速度は低下することがわかる。

しかし、増分のＮ／４Ｓが、すべて短冊状配列データのずれ補正に必要なＲＴＲＧ配列間のシフトに要する分であることから、サイクル当たりのシフト量Ｓを大きくできれば、アクセス速度は改善できる。このシフト量Ｓを大きくすることは、ＲＴＲＧの動作速度がサイクルを決めるメモリのアクセス速度、演算器の動作速度等より高速であるた

め、2～3程度までならば、比較的容易である。例えば、Sが1から2にできたとすると、表4．3に示すように、4．1の構成条件のアクセス速度の従来比は、2次元アクセスで1／3から1／2に、90度回転で1／2から2／3にそれぞれ改善される。

表4．3 本機構構成例と従来機構のビットライン当たりの所要サイクル数の比較

	従来機構	本機構		従来機構／本機構	
		S＝1	S＝2	S＝1	S＝2
2次元アクセス	1	3	2	1／3	1／2
90度回転	2	2	3	1／2	2／3

4．5．3 各アクセス法の高速性比較

前節では、従来型アクセス機構との比較を分りやすくするために、転送回転型メモリアクセス機構については、最も効率の良い場合の所要サイクル数を用いた。しかし、各アクセス法には、それぞれ制約があり、これを満たすか否かにより、アクセス速度が左右されるため、従来型2次元アクセス機構との速度性能の差はもう少し大きい。その制約を以下にまとめて示す。

①従来型2次元アクセスの制約

列あるいは行の要素の語長がメモリユニットのアクセス幅Wに一致していること。この制約は、メモリユニットの1ワードに1要素を格納することを前提としていることによる。

②疑似2次元アクセスの制約

列あるいは行の個別のアクセスが、一旦アクセスした短冊状配列データの範囲の中で繰り返し行われること。メモリユニットからの最小のアクセスの単位が横長あるいは縦長の短冊状配列データであることによる。

③ずれ補正なしの2次元アクセス

行あるいは列を順次アクセスして処理を進めること（線順次処理）を前提としているからである。

④間接的な2次元アクセスの制約

行あるいは列を、ある程度以上、一方的にアクセスすること。これによって、必要な90度回転の頻度が小さくなるからである。

これらの制約から、本転送回転型メモリアクセス機構において、単純に各アクセス法の高速性を比較することはできない。いずれのアクセス法が最適であるかは処理内容に

よって変るからである。しかし、実際の応用では、間接的な2次元アクセスが有利な場合が多いと推定される。これは、一般的な処理では行あるいは列の一方を続けてアクセスする傾向が強く90度回転の頻度を小さくできるからである。

いま、90度回転のオーバーヘッド分を含めた間接的な2次元アクセスによる列あるいは行の平均のアクセスサイクルをC_Iとすると、式（4．9）より、

$$C_I = 1 + r \times (2 + N/4S) \qquad (4．10)$$

のように表すことができる。ここで、右辺第1項は通常のメモリアクセスの所要サイクル数、右辺第2項は90度回転のオーバーヘッド分のサイクル数である。90度回転の付加サイクルである2A／WおよびQ／Wはやはり小さいとして無視している。また、rは回転対象の2次元配列データの構成本数分（N×W本）の行あるいは列をアクセスするごとに必要となる90度回転の回数であり、列あるいは行の1本分のアクセスに加わる平均的な90度回転のオーバーヘッドの割合を表している。図4．6は、この式（4．10）で与えられる間接的な2次元アクセスの所要サイクル数と従来型2次元アクセスあるいは疑似2次元アクセスの所要サイクル数が等しいことを条件としてrとNの関係を求めたグラフである。この図で、グラフの曲線の上側が、疑似2次元アクセスが速い領域、下側が間接的な2次元アクセスが速い領域をそれぞれ示している。この結果より、転送回転型メモリアクセス機構が、端子数低減、ハードウェア規模低減等で意味を持つ4以上の分割数Nに対して、間接的な2次元アクセスが高速になるのは、条件の悪いS＝2で、rが0．2以下の場合であることがわかる。従って、この値を目安に従来型2次元アクセスあるいは疑似2次元アクセスと間接的な2次元アクセスを使い分けることによって処理の高速化をはかることができる。

ところで、これまでのセルラ配列型1次元SIMDプロセッサの文字認識への応用の経験（8章，9章参照）からすると、一般的な処理では行あるいは列のいずれかを一方的に処理することが多く、rの値を0．2以下にすることは難しくない。従って、従来型2次元アクセス、疑似2次元アクセスとの整合性の高い場合、線順次処理が適用可能な場合等を除くと、一般には間接的な2次元アクセスの利用を前提に処理途中の90度回転の回数低減をはかるのが効率的な処理を実現する近道といえる。

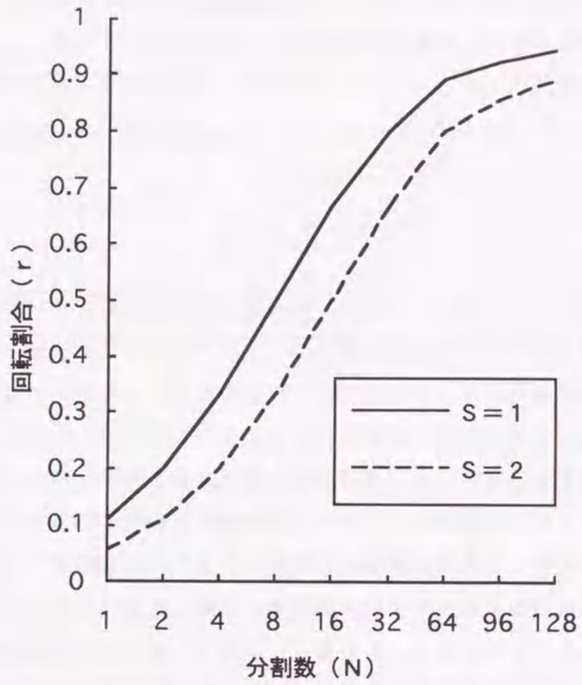


図４．６　疑似２次元モードと間接２次元モードのアクセス
速度が等しくなる回転割合（ｒ）と分割数（Ｎ）との関係

４．６　むすび

基本のＳＩＭＤプロセッサにわずかな改造を加えるだけで実現される転送回転型メモリアクセス機構を提案し、その各種のアクセス法を明らにすると共に、従来型の２次元アクセス機構と比較して、端子数、アクセス速度等を評価した。

はじめに、従来型のメモリアクセス機構が多くの端子数を必要とし、チップ当たりの搭載ＰＥ数拡大の妨げになることを示した。次いで、提案する転送回転型メモリアクセス機構として、ＰＥ配列を等分割したW個のＰＥからなるＰＥ部分配列の内部に形成されるWビット幅のルーティングレジスタW個の並び（ＷｘＷの２次元のレジスタ要素配列とみなせる）に対して、

①列，行両方向の入出力パスを設け、９０度回転器として利用できるようにすること、

②ルーティングレジスタの縦続接続からの出力を、ＰＥ部分配列に付加されるメモリユニットに対するアドレスとして供給できるようにすること、

の２点の改造を加えたＳＩＭＤプロセッサ向きメモリアクセス機構の構成を示した。

次いで、このメモリアクセス機構を用いたＰＥごとの間接アドレシングによるデータアクセス、従来型の２次元アクセス、横長あるいは縦長の短冊状の２次元配列データを単位とする疑似的な２次元アクセス、線順次処理を前提とする補正無しの２次元アクセス、２次元配列データの９０度回転とこれを用いる間接的な２次元アクセス等のメモリアクセス法を示すとともにその所要サイクル数算出式を求めた。また、これらの２次元アクセス法のいずれが高速であるかについても、所要サイクル数算出式をもとに簡単な解析を行い、行あるいは列のそれぞれを、一定回数（回転対象の２次元配列の構成行［列］数の５倍）以上続けてアクセスする形で行える一般的な処理では、９０度回転を利用する間接的な２次元アクセスが高速となることを導いた。

さらに、このメモリアクセス機構を、構成ＰＥ数２５６、分割数８の条件で、バレルシフタとＰＥごとの個別のアドレス生成器を付加する速度性能の点で理想的な従来型２次元アクセス機構と比較した。その結果、２次元アクセス速度は、２／３～１／２程度に低下するものの、端子数が１／３程度まで低減されることが明らかとなった。

このように、転送回転型メモリアクセス機構は、基本のＳＩＭＤプロセッサにわずかなハードウェアを付加するだけで構成される上に端子増も小さく押さえられるので、汎用性の高いセルラ配列型ＳＩＭＤプロセッサを小形、経済的に構成する上で極めて有用なＰＥ配列構成技術といえる。

第５章　１ビットセルラ配列型ＳＩＭＤプロセッサシステムの 設計と試作

５．１　はじめに

本章では、前章までの検討結果をもとにセルラ配列構成の利点をすべて引出すことを目指して試作した小型高並列プロセッサＬＩＳＣＡＲ（LIne Scannable Cellular ARray processor）と大規模セルラ配列型ＳＩＭＤプロセッサＡＡＰ（Adaptive Array Processor）の設計、構成について述べる。

設計に当たっては、ＬＩＳＣＡＲでは、小型経済的な高並列処理システムを目指し、ＰＥ数が10^2程度の規模のＰＥ配列を含む全体を１ボードに実装することを、ＡＡＰでは、超並列の利点を比較的小型のシステムで享受できるようにＰＥ数が10^4～10^5程度の規模のＰＥ配列を１台のラックに実装することを、それぞれ目標とした。これらの目標を達成するために、１．３節（３）の具体的な取り組みの方針に従い、アーキテクチャとしては面単位とライン単位の並列処理の両立が可能な２次元のＰＥ配列を、可変構造とバイパスで補強する最もシンプルなセルラ配列構成を採用することとした。

本章では、はじめに、このＬＩＳＣＡＲとＡＡＰで規模以外は共通であるＰＥ配列について、その構成要素であるセルラ配列型ＬＳＩの構成、設計を中心に述べる。次いで、ＬＩＳＣＡＲ、ＡＡＰのそれぞれについて、具体的なハードウェア構成と性能評価、プログラム開発環境等について述べる。なお、セルラ配列型ＬＳＩとしては、１９８１年にはじめて６４個の１ビットＰＥが搭載できることを示したＡＡＰ１（ｎＭＯＳ版）[44]に続き、ＡＡＰ１（ＣＭＯＳ版）[45]、ＡＡＰ２と試作を重ねてきたが、ここではそれらの集大成であるＡＡＰ２－ＬＳＩを中心に述べる。

５．２　ＰＥ配列とその構成要素であるセルラ配列型ＬＳＩの設計

５．２．１　ＰＥ配列とＬＳＩの構成

ＰＥ配列はセルラ配列構成の前提に従い、１チップに数十ＰＥを搭載するＬＳＩ（セルラ配列型ＬＳＩ：ＡＡＰ２－ＬＳＩ）とＬＳＩ毎に付加するメモリＩＣの対を、正方格子上に配置する単純で規則的な構成とする。図５．１は、そのＡＡＰ２－ＬＳＩとメモリＩＣの対とＰＥ配列構成との関係を示している。この図からも明らかなように、ＰＥ配列の構成・設計とは、セルラ配列型ＬＳＩの構成・設計にほかならない。また、ＬＳＩの構成自体も、必然的に８ｘ８のＰＥ部分配列とその周辺回路とからなる極めて単純なものになる。

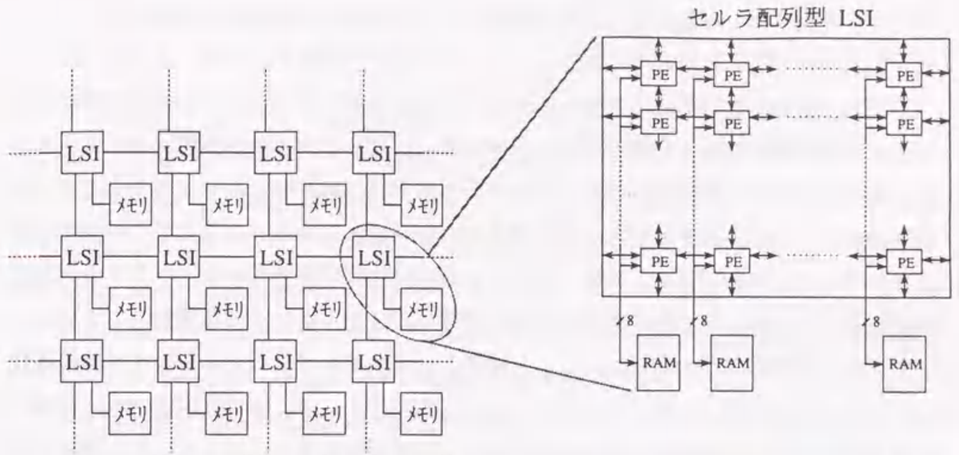


図５．１　セルラ配列型ＬＳＩとメモリＩＣの規則的配列で構成されるＰＥ配列

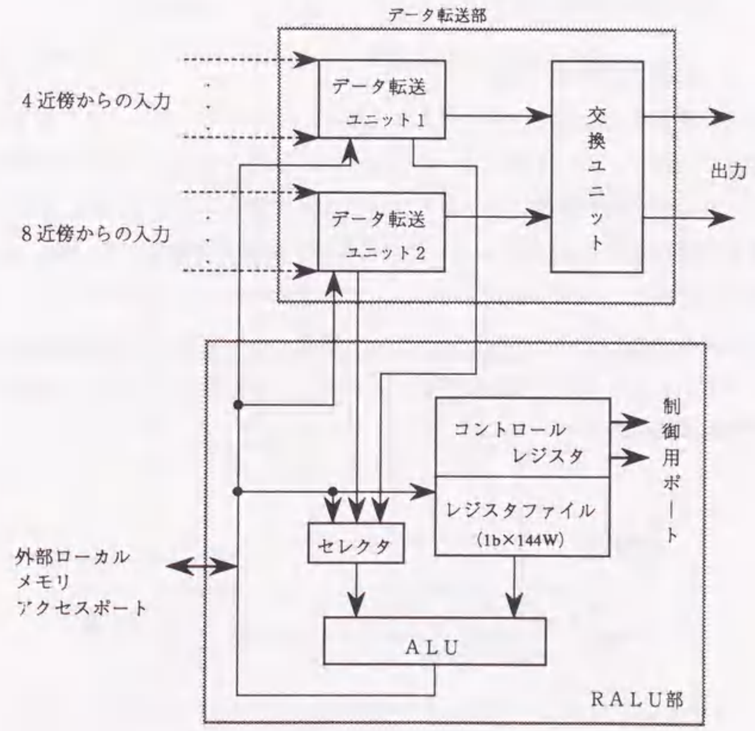


図５．２　ＰＥの概略構成

５．２．２　ＰＥ構成

図５．２のブロック構成図に示すようにＰＥは、レジスタ付きＡＬＵ部とデータ転送部とからなる。

レジスタ付きＡＬＵ部は１ビットのＡＬＵ、入力セレクタ、８ビット構成の制御レジスタ、一部制御レジスタを兼ねるマルチポートレジスタファイル、外部入出力ポート等からなる。８ビット構成の制御レジスタへのロードは、あらかじめレジスタファイルに格納されているデータを８ビット幅の読み出し専用ポートから一括して移す構造により１クロックサイクルで行なえるようにしている。外部入出力ポートは、ＬＳＩの入出力端子に直結されており、各ＰＥから直接外部の局所メモリをアクセス可能としている。

データ転送部は二つの転送ユニットを内蔵し、それぞれＰＥ間で４隣接、８隣接接続のネットワークを構成している。パスの転送方向は、３．２．４で述べたように制御レジスタによりデータ転送部の入力マルチプレスクサを切り換えることにより、各ＰＥで個別にプログラム可能としている。２系統のパス間のデータ交換についても、３．２．３で述べたように制御レジスタによって個別に切り換え可能な交換ユニットによってＰＥごとに選択的に指定できるようにしている。

５．２．３　伝搬演算機構の構成

伝搬演算の最も単純な高速化機構である１段のバイパスを、図５．３に示すようにＬＳＩ単位で、各行、各列に設けている。バイパスは端のＰＥから反対側の端のＰＥの出力セレクタに途中の７個のＰＥをスキップする構造として、図２．６に示されるバイパス系専用の演算器$P1_2$を単なるセレクタに置換する構成を採用している。この構成では、図２．６の$P1_1$だけでなく、$P1_2$、$P2_1$についてもＰＥでエミュレーションする必要が有るため、単なる伝搬転送ではなく演算を伴う場合には多少処理が煩雑になる欠点がある。しかし、ＰＥ配列にバイパスとセレクタを付加するだけで構成される点で、セルラ配列構成向きといえる。

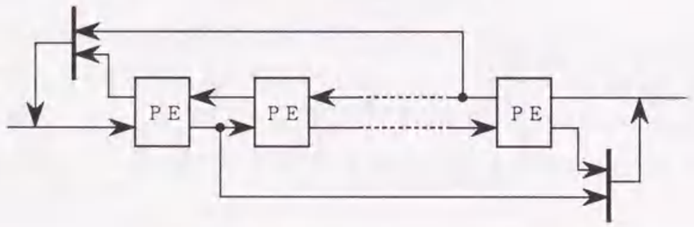


図５．３　行方向ＬＳＩレベルバイパス

５．２．４　回路技術

（１）出力レベル補償型ｎＭＯＳ構成パストランジスタ論理

人手によるレイアウトを前提に、片（ｎ）チャネル型ＭＯＳトランジスタ単独構成のパストランジスタによるセレクタ形論理をインバータで受けるタイプの回路（図５．４）をＰＥ論理部で多用した。例えば、ＡＬＵもパストランジスタ論理を用いた４－１セレクタで構成している[46]。従来からのフルＣＭＯＳ型のＮＡＮＤ、ＮＯＲ等のみで構成する場合に比べると、ゲート換算の集積度は数倍以上高められるものの、ＣＭＯＳとＰＭＯＳを対抗させる規則的な配置がとれないため、レイアウトは難しくなる[47]。しかし、ＰＥの回路規模は小さくかつ繰り返し使われるので、全体から見た設計の負担増はわずかであり、その割に大きなチップ面積低減効果が得られる。

ｎＭＯＳ構成のパストランジスタ論理では出力のハイレベルが十分上がりきらないため、次段のＣＭＯＳ回路に貫通電流が流れる問題があった。この問題に対して、入力が論理しきい値を超えた時点で自分自身で入力レベルを引き上げる能力を持つ回路（単安定回路）を、次段に配置することによって解決している[48]。図５．４に示すように、単安定化は、インバータの場合、ｐＭＯＳトランジスタを一個追加することによって簡単に実現されるので、それによる面積増加、速度劣化はわずかである。

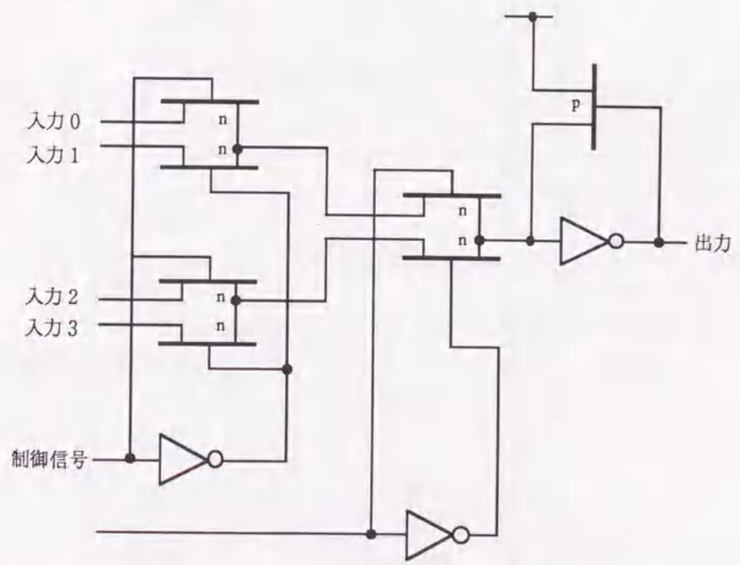


図５．４　片（ｎ）チャネルＭＯＳトランジスタ単独で構成するパストランジスタを用いた４－１セレクタ

（2）マルチポートレジスタファイル

メモリ配列部を9列x16行の標準的なSRAM（6トランジスタメモリセルの完全CMOS型）構成とするのに加え、アドレスデコーダ部を複数PE（16PE）で共通化することにより、占有面積の低減をはかっている。ディジット線のチャージアップはプリチャージによるダイナミック型とし、アドレスの多重選択の防止と読み出しの高速化をはかっている。

読み出し書き込みポートは、通常の1ビットアクセスRAMの読み出し書き込み用の第1ポートに加え、図5．5に示すようにメモリ配列の各列につながる8個のセンスアンプSAからなり行単位の並列読み出しが可能な第2ポート、16行目のメモリセルから直接制御信号として引き出される第3ポートの3つから構成している。この第2、第3のポートは、読み出し専用で、レジスタファイルの一部を制御レジスタとして利用するために設けてある。特に第2のポートは、制御レジスタの内容をバイト単位で書き換えることを可能にしており、これによって配列の構造を瞬時に切り換える高速の可変構造制御が実現される。

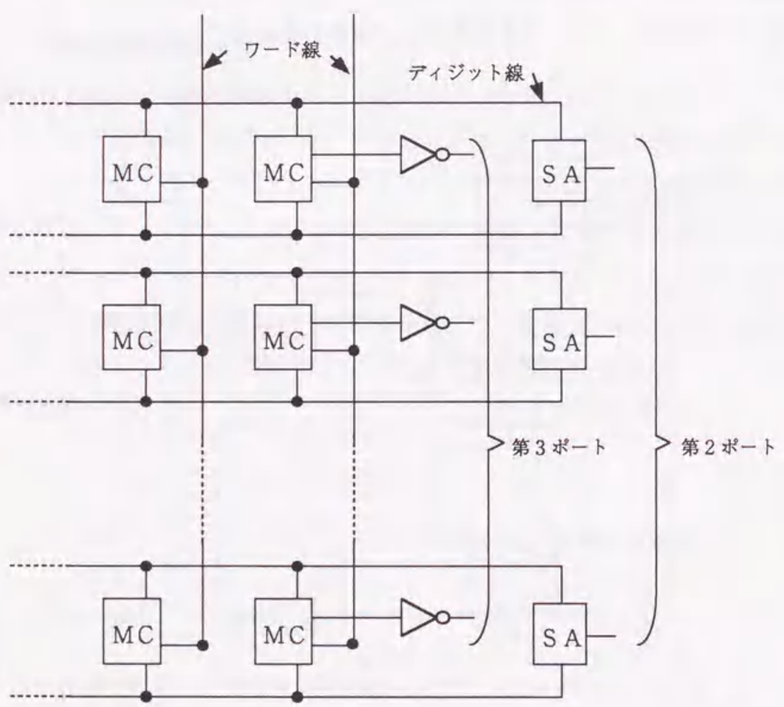


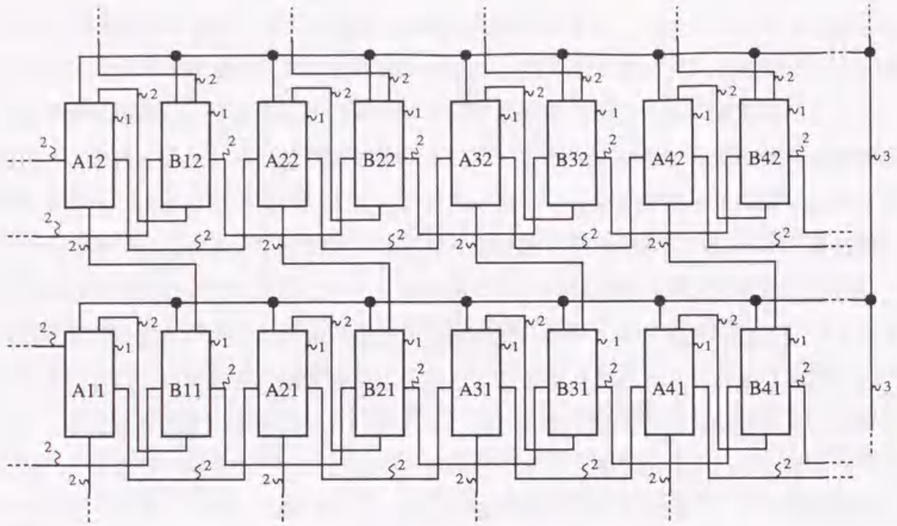
図5．5　レジスタファイルの第2、第3ポートの構成

5．2．5　チップレイアウト

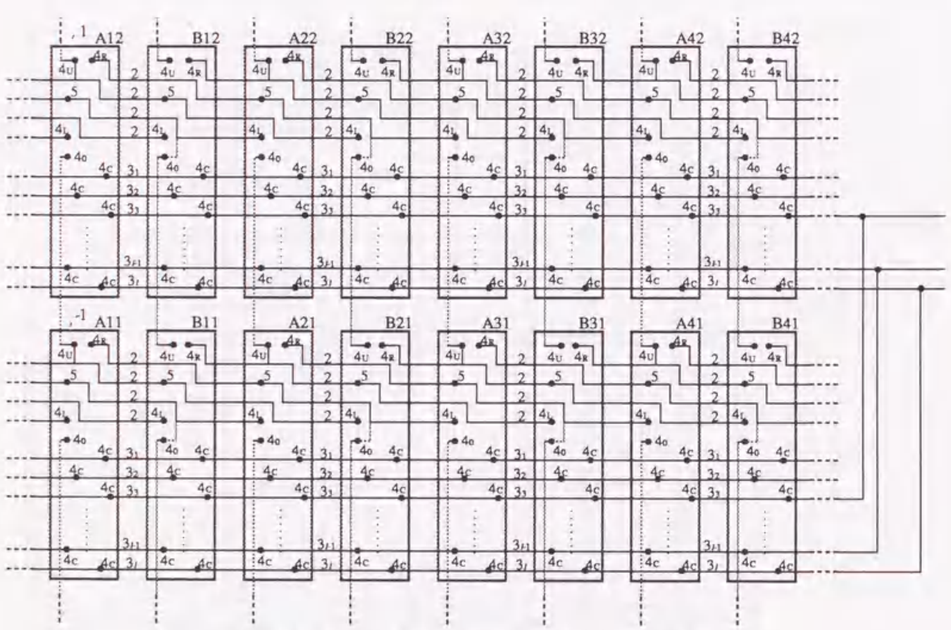
8x8の論理的配列を4x16の物理的配列で実現することにより配線のみに占有される領域を大きく低減している。図5．6にその概略レイアウト構成を示す。4x16のPE配列と、その左右両側のドライバー配列部およびAℓ2層配線からなる共通制御信号の束とで構成している。ドライバ配列部の各ドライバからの出力は、各PEの論理部の上を通過させ、16個のPEに対して制御信号を供給している。また、共通制御信号の束の下には、PE配列全体に関わる制御論理を埋め込んでいる。

この図5．6から明らかなように、このレイアウトの利点はPEの形状が細長くできるためにPE上の長手方向とは直角の方向にPE上を通過する配線トラックが多数確保できることにある[49]。この構成により、共通の制御信号線、PE間接続線等のみによって専有される配線領域は大きく低減され、PE配列部の集積度が大幅に高まる。一方、このレイアウトのかなめである4x16のPE配列は、PE間配線を複雑とし、かえって面積の増大につながるようにも見える。しかし、この推測は正しくない。図5．7は4x16の配列をとった場合のパス1に関するPE間接続の概略を示している。上PEと下PEを規則的に並べるだけで構成されており、配線の複雑化は生じていない。

図5．6　PE配列の概略レイアウト

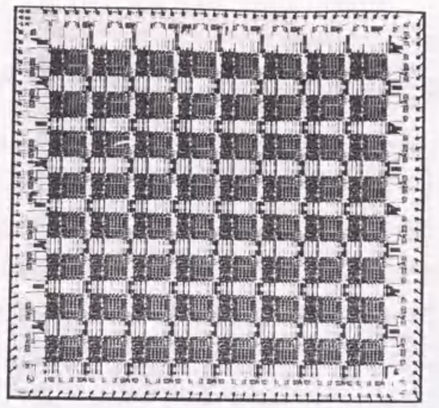


（a）ブロック構成図

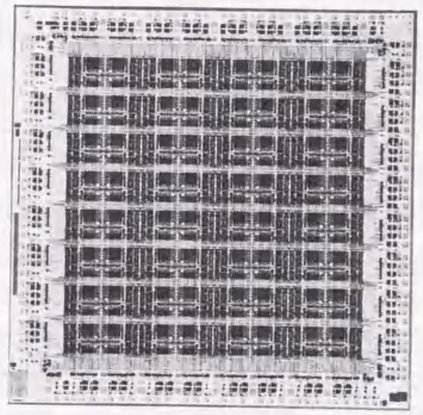


（b）マスクパターンの概略レイアウト図

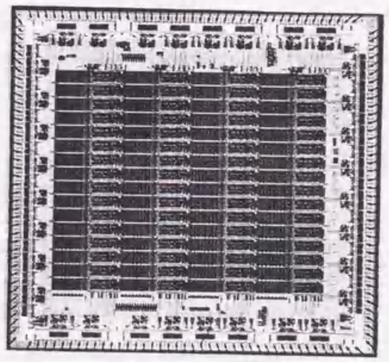
図５．７　パス１に関するＰＥ間接続のレイアウト構成



（a）ＡＡＰ１



（b）ＡＡＰ１改



（c）ＡＡＰ２

図５．８　各ＡＡＰ－ＬＳＩのチップ写真

5．2．6　設計・試作結果の評価

２μmｐウェル２層AℓのＣＭＯＳ技術を用いて実現した本ＬＳＩの写真と諸元を図５．８と表５．１に、ＡＡＰ１－ＬＳＩ（３μmAℓ１層ｎＥ／ＤＭＯＳ版）とそれに転送機能の一部を強化したＡＡＰ１－ＬＳＩ（２μmAℓ２層ｐウェルＣＭＯＳ版）と並べて示す。ＡＡＰ２－ＬＳＩは、ＡＡＰ１に比べ、データ転送機能の強化、ＡＬＵのファンクションの個別設定機能の付加、レジスタファイル容量の拡大（１．５倍）等の改良を加えている。それにも関わらず、チップサイズ自体は小さくなっている（表５．１参照）。これは、前節で述べたレイアウトの工夫が効いる。実際、チップ写真から明らかなように、ＰＥ配列の中でＰＥ間に走る信号配線の束のみで占められる面積が、ＡＡＰ１（３μmAℓ１層ｎＥ／ＤＭＯＳ版）では５０％程度、ＡＡＰ１（２μmAℓ２層ｐウェルＣＭＯＳ版）で３０％程度をそれぞれ占めるのに対し、ＡＡＰ２ではほぼ１０％程度にまで低減されている。この信号配線専用領域の低減により、ＡＡＰ２の集積密度は大きく向上している（表５．１参照）。また、このチップサイズ縮小には

表５．１　各ＡＡＰ－ＬＳＩの諸元比較

比較項目	ＡＡＰ１(nMOS版)	ＡＡＰ１(CMOS版)	ＡＡＰ２
搭載ＰＥ数	８×８	８×８	８×８
プロセス技術	３μmAℓ１層 ｎＥ／ＤＭＯＳ	２μmAℓ２層 ｐウェルＣＭＯＳ	２μmAℓ２層 ｐウェルＣＭＯＳ
素子数	８１，０００	１１１，９００	１０４，９００
ｻｲｸﾙﾀｲﾑ (typ.)	２００ｎｓ	５５ｎｓ	５５ｎｓ
電源電圧	５Ｖ単一	５Ｖ単一	５Ｖ単一
入出力レベル	ＴＴＬ	ＴＴＬ	ＴＴＬ
特徴的な回路技術	・nMOSパストランジスタ論理多用	・標準フルCMOS回路	・出力レベル補償型nMOSパストランジスタ論理多用 ・ダイナミックプリチャージ型SRAM
チップサイズ	10.0×9.4 mm²	9.85×9.75 mm²	8.8×8.16 mm²
集積密度	８６２素子 mm²	１,１７０素子 mm²	１,４０１素子 mm²
端子数	１４１	１４４	１６０
ＰＥ当たりの伝搬演算時間 (typ.)	５５ｎｓ	１７ｎｓ	１８ｎｓ

表５．２　ゲート当たりの素子数

ＰＥ論理部	１.８６
周辺回路	２.４８
ＬＳＩ全体	１.９７

表５．３　集積度一覧

チップ全体		１,４０１　素子mm²
RAM部を除外した場合	ＰＥ論理部	１,６３９　素子mm² (1,062 G/mm²)
	配列部全体	１,４０１　素子mm² (879 G/mm²)
	チップ全体	８０９　素子mm² (410 G/mm²)

出力レベル補償型のｎＭＯＳ構成パストランジスタ論理を多用したＰＥ部の人手設計も効いている。表５．２はゲート当たりのトランジスタ数を、ｎＭＯＳ構成パストランジスタ論理を多用したＰＥ論理部と、通常のフルＣＭＯＳ型の論理で構成した周辺部とで、比較したものである。周辺部に比べ、ＰＥ論理部が３０％近くもゲート当たりのトランジスタ数が少ないことがわかる。パストランジスタ論理はゲート当たりの実現機能が高いので、実際の素子数低減効果は、これ以上と考えられる。また、表５．３には、得られた集積密度を示している。人手設計と相まって、ＲＡＭ部を除外した配列部全体では、同程度のデザインルールのスタンダードセルベースのＬＳＩに比べ４～５倍高い集積密度が得られている。

一方、動作速度については、リードモディファイライトサイクルで５５ｎｓと、既存のコントローラＬＳＩと組み合わせるのに十分な速度性能が得られている。ＡＡＰ－ＬＳＩのように単純な構成の伝搬演算機構を採用する場合には、ＰＥ当たりの伝搬演算時間も性能上重要となるが、これについても表５．１に示すように１８ｎｓとサイクルタイムの１／３程度の演算時間を確保している。

5．3　小型システムＬＩＳＣＡＲ１の構成

パソコンレベルのホストコンピュータと組合わせる経済的なシステムを指向して、ＡＡＰ２－ＬＳＩを用いた小型の高並列プロセッサＬＩＳＣＡＲ１を試作した。本節では、そのハードウェア構成を中心に述べる。

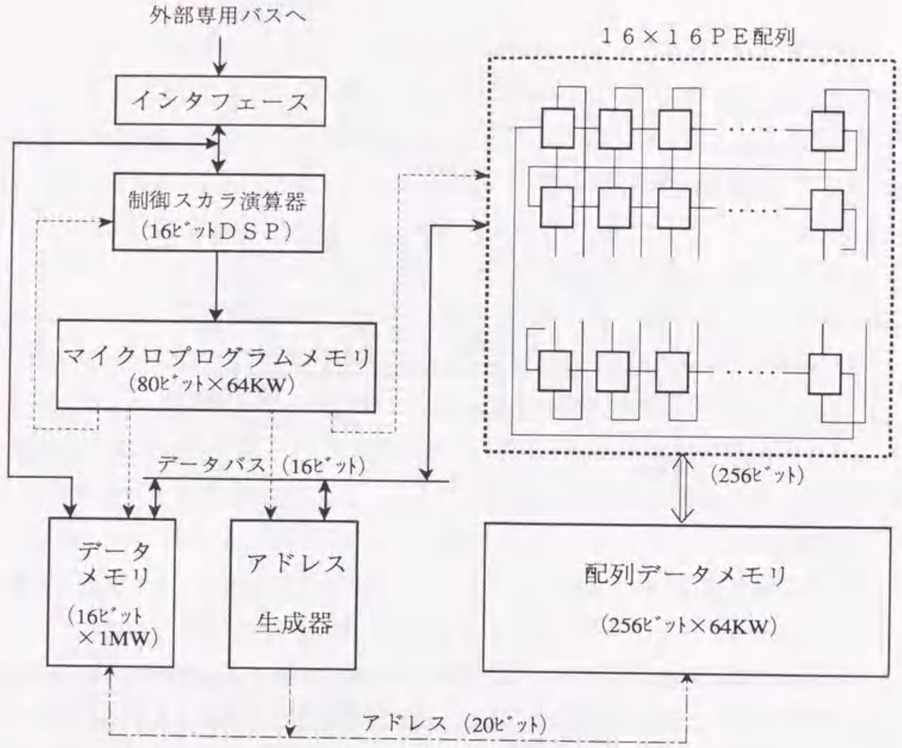


図5．9　ＬＩＳＣＡＲ１の構成

5．3．1　全体構成

ＬＩＳＣＡＲ１は、図5．9に示すように、１６ｘ１６ＰＥからなるＰＥ配列部と、制御スカラ演算器、アドレス生成器、プログラムメモリ、データメモリ、配列データメモリ、インタフェース等で構成している。ここで、制御スカラ演算器は、プログラムメモリと対で、全体を一括制御するためのマイクロ命令を生成するとともに、スカラ演算を分担する。アドレス生成器は、データメモリ、配列データメモリをアクセスするためのアドレスを生成する。データメモリはスカラ演算用のメモリであり、配列データメモリはＰＥ配列部の各ＰＥの局所メモリの配列である。

5．3．2　１次元／２次元両用のＰＥ配列

図5．１０に図示しているように、ＰＥ配列は、各行、各列のそれぞれで、一方の端ともう一方の端を結ぶループ接続により全体としてトーラス型の２次元隣接結合ネット

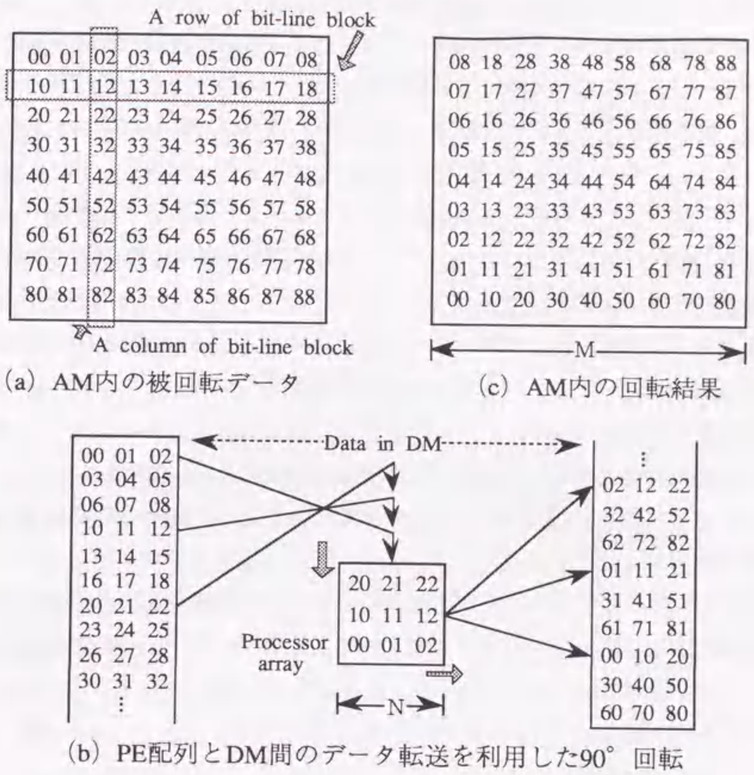


図5．１０　ＬＩＳＣＡＲ１における９０度回転

ワークを形成している。ただし、２系統の転送系の一方については、各行の端と端の接続を１段ずらす構成（ネスティッドトーラス）をとっている。これは、１６ｘ１６のサイズの２次元配列を単位とする動作モードと長さ２５６の１次元配列を単位とする動作モードの両方を可能とするためである。

5．3．3　９０度回転機構

４．１で述べたように１次元のＳＩＭＤプロセッサにおいて、メモリユニット配列内の２次元配列データに対し、行単位にでも列単位にでもアクセスできる２次元アクセス機能は、汎用化に必須ともいえる機能の一つである。しかし、ＬＩＳＣＡＲ１は１次元の動作モードをサポートしていながら、行／列の両方向のアクセスを可能とする２次元アクセス機構も、５章で明らかにした転送回転型メモリアクセス機構も搭載していない。このため、配列データメモリ（ＡＭ）から一旦データメモリ（ＤＭ）上に移した２

次元配列データを、２次元のＰＥ配列との間で入出力する際の１６ｘ１６のブロック単位の回転機能により９０度回転することによって、等価的に行、列両方向のアクセスを実現している。その回転手順は、以下の通りである（図５．１０参照）。

①縦、横の長さがＰＥ配列の長さ（２５６）に等しいＡＭ内のビットプレーンデータ（図５．１０（ａ））を行単位（行方向のビットライン単位）でＰＥ配列上に読みだしてから、ＰＥ間のシフト転送により１６ビット幅のワード単位に分割してＤＭに順次転送し、ＤＭ内に図５．１０（ｂ）の左側に示すように各行が折り畳まれた形式のデータ配列を得る。

②図５．１０（ｂ）に示すように、左側のＤＭから、ＰＥ配列のサイズに一致する１６ｘ１６のブロックの２次元配列を順次読みだし、図５．１０（ｂ）の中央のＰＥ配列内にＰＥ間のシフト転送を利用して入力する。

③ＰＥ配列上のブロックの配列データを、入力方向とは直角の方向にシフト転送により出力し、図５．１０（ｂ）の右側に示すように、ＤＭ内に各列が折り畳まれた形式で格納する。

④ＤＭ内に得られている列形式の配列データを、列単位でシフト転送によりＰＥアレイに転送し、行単位の配列データとして、図５．１０（ｃ）のようにＡＭにもどし、９０度回転後のビットプレーンを形成する。

以上の９０度回転の１行当たりの所要ステップ数Nsは、

$$Ns = (1+W) + W + W + (W+1)$$
$$= 4W + 2$$

となる。ここで、右辺の第１項～第４項は①～④に対応する１行当たりの所要ステップ数である。また、ＷはＰＥ配列を２次元配列として見た時の１辺の長さで、ＬＩＳＣＡＲ１ではデータメモリの語長でもある。ただし、ＰＥ間のシフト転送が１ＰＥ分のシフト当たり１ステップで実行可能で、かつＤＭのアクセスと並列に実行可能であるとしている。この所要ステップ数は、５章で述べた転送回転型メモリアクセス機構に比べると１桁近く大きく、回転の頻度の高い応用では、性能低下の原因になる。

５．３．４　制御構成

ハードウェア上での命令に階層を設けず、マイクロ命令で全ての構成要素を制御するマイクロコードマシンの制御構成を採用した。この理由は、１命令で複数の構成要素を並列に制御できる高速性の利点だけでなく、高速かつ大容量のメモリが比較的容易に入

手可能な現在では、開発コスト低減の点でも有利と判断したからである。

マイクロ命令の語長は８０ビットで、内１６ビットをコントローラ用の汎用ＤＳＰの制御フィールドに、残りをＰＥ配列、アドレスジェネレータ、各メモリ等、その他の構成要素の制御フィールドに割り当てている。

５．３．５　装置化

（１）１ボードプロセッサ構成

有力な応用先である文字認識プロセッサとして、小形経済性で十分な競争力の確保をめざしプロセッサ全体をＢ４版サイズのボード１枚に実現した。そのＬＩＳＣＡＲ１ボードの写真を図５．１１に、諸元を表５．４に示す。このコンパクトな実装は、

①ＰＥ配列を１チップに６４ＰＥを搭載したＡＡＰ２　ＬＳＩを４個で構成する、

②ＳＲＡＭに比べ１個当たりの容量が４倍以上と大きい標準ＤＲＡＭを高速アクセスモードで動作させる構成によって、メモリＩＣの使用個数を低減する、

③コントローラとスカラ演算器を兼ねる１チップＤＳＰを採用することで、制御スカラ演算部の部品点数を低減する

等により達成している。なお、採用したＤＳＰではアドレス空間が６４Ｋと狭く、データメモリの全領域（１Ｍ）を直接アクセスできない問題が生じた。そこで、６４Ｋを超える領域に対しては、１Ｍまでのアドレシング可能なレジスタ付きＡＬＵ ＬＳＩからなるアドレス生成器の側からアクセスできるようにして解決している。

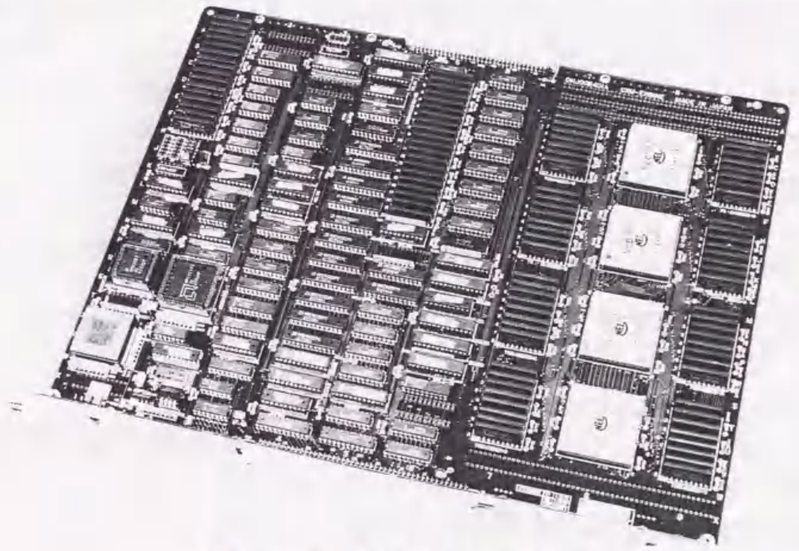


図５．１１　ＬＩＳＣＡＲ１ボード

表5．4　ＬＩＳＣＡＲボードの諸元

項目	内容
ＰＥアレイサイズ	１６×１６（２５６×１）
メモリ容量	
プログラムメモリ	６４０Kb
データメモリ	2MB
アレイデータメモリ	2MB（8MB）
最小命令実行時間	８０ns
電源電圧	5V±０．２５Ｖ
消費電力	２７．５Ｗ
ボードサイズ	３６６．７×２８０mm^2

配列データメモリは、各ＰＥがその１ビット分を占有できるように、アクセス幅をＰＥ数に等しい２５６ビットに選んでいる。この構成は、高速化には有効であるが、メモリＩＣの個数やコストがネックとなるために、メモリ容量の変更の自由度は低い。そこで、１６ビット構成の１Ｍと４ＭのＤＲＡＭを使い分けて、同一構成のボードで２Ｍバイトと８Ｍバイトの配列データメモリの容量を実現している。

速度性能については、最小命令実行時間が最初の試作機では２００ｎｓであったのに対し、回路構成の見直し、使用部品の高速化等により、１４３ｎｓ版を経て、最終的には表５．４に示す８０ｎｓ版までの高速化を達成している。

（２）装置構成

基本的な装置構成を、図５．１２に示す。ＳＣＳＩインタフェースとこれに専用バスを介してつながるＬＩＳＣＡＲ１ボードでＬＩＳＣＡＲユニットを構成し、これをＳＣＳＩ経由でホストコンピュータに接続する形である。ＬＩＳＣＡＲ１ボードは、専用バスに４枚まで接続可能で、この範囲でＬＩＳＣＡＲユニットの性能を要求に合わせることができる。また、複数のＬＩＳＣＡＲユニットを、ＳＣＳＩ経由でホストコンピュータに接続することによっても、性能の向上が可能である。

ここで、各ＬＩＳＣＡＲボードは、ホストコンピュータのバックエンドプロセッサとして機能し、ホストコンピュータよりプログラム、データを受け取り、必要な処理を実行した後、結果をホストコンピュータに転送する。

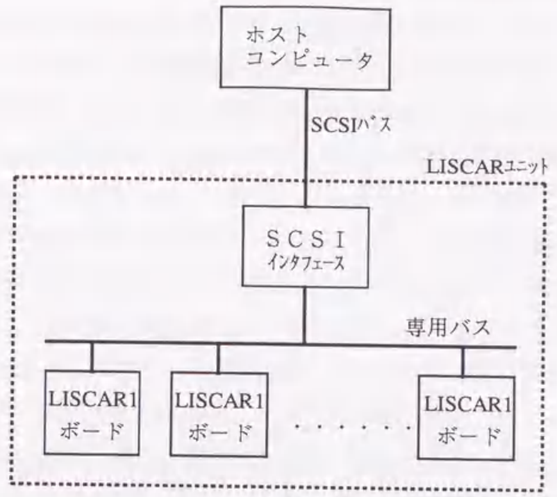


図５．１２　装置構成

５．３．６　ソフトウェア開発環境

プログラムのクロスコンパイル、ＬＩＳＣＡＲに対するコンパイル結果、処理結果等のロード・ストア、ＬＩＳＣＡＲの起動、対話型のデバッグ等を行なうことのできるソフトウェア開発環境を、ＶＡＸ上に構築した。

（１）プログラムミング言語

プログラミング言語は、ＦＯＲＴＲＡＮの部分ルーチンとしてコールするマクロ命令の集合で構成している。このため、プログラムは形式的にＦＯＲＴＲＡＮプログラムとなり、ＦＯＲＴＲＡＮ習熟者ならば容易に修得できる。マクロ命令は、やはりＦＯＲＴＲＡＮサブルーチンとしてコールするマイクロ命令で記述されたマイクロプログラムであり、必要に応じてユーザが作成できる。

変数形式としては、１６ビット長のワード変数に加えて、ＰＥ配列の１次元の演算モードのデータ形式に対応するライン変数（長さは２５６の倍数）をサポートしている。ＰＥ配列による並列処理は、このライン変数に対する演算として記述する。

（２）デバッガ

マクロ、マイクロ命令の両階層のデバッグに対応するシンボリックデバッガであり、マクロ、マイクロ命令の混在したプログラムのデバッグが可能である。走行中のＬＩＳＣＡＲに直接割り込んで、状態観測、状態設定を行なう方式を採用している。このため、十分な応答速度が得られ、スムースな対話型デバッグが可能となった。

5．3．7　基本性能評価

ＰＥ配列が１ビットプロセッサの配列であることから、特にビットラインに対する性能が高い。長さが２５６のビットライン間の基本演算では、例えば、算術論理演算および左右両方向のシフトで、実行時間が、それぞれ、９０ｎｓ（2,860MOPSに相当）、３２０ｎｓと非常に短い。グレイスケールライン（要素の語長が２ビット以上のライン）に対する演算でも、乗算が入らない場合、例えば8ビットグレイスケールライン間の差の絶対値の累積では、4.1μＳ（１８０MOPSに相当）と、かなり高い性能が得られる。これに対し、乗算の入る８ビットグレイスケールライン間の差の２乗の累積では、３６.４μＳ（２１.１MOPS）とかなり低くなる。ここで、グレイスケールライン間の差の絶対値の累積と差の２乗の累積は、それぞれ文字認識処理の識別処理に有用なシティブロック距離計算とユークリッド距離計算である。これらの性能値はいずれもピーク値であり、実際の応用で得られる性能はこれより低くなるが、それでも数十ＭＯＰＳのレベルにある汎用のＤＳＰ等と比べると、はるかに高い性能と言える。表５．５に基本的な２値画像処理の性能を示す。文書画像処理、文字認識処理等で多用する周辺分布、黒点面積等を求める処理では画素当たり数ｎｓと極めて高速に処理できることがわかる。ただし、すべての演算を論理算術演算、データ転送等の基本演算の繰り返しによって実現しているために、平滑化、９０度回転などでは処理時間が若干大きくなっている。

この低いほうの性能に注目すると、機能の高い演算でも画素当たり十数ｎｓ～数十ｎｓで実行する最近の画像処理プロセッサ[50]の方が優れているように思える。しかし、これらのプロセッサも、すべての演算を数十ｎｓで実行できるわけではない。専用の演算機構を組み込むことによって、高速化を達成しているので、高速化可能な演算は特定

表５．５　画素当たりの演算時間

演算内容	演算時間（ns）
論理演算	0.95
９０度回転	39.2
輪郭抽出	13.4
平滑化	62.7
周辺分布	3.3
黒点面積	3.4

注）対象画像：256 x 256の２値画像

のものに限られている。あらかじめ対応していない演算に対しては、結局、数十ｎｓかかる基本演算の組み合わせで実行せねばならず、基本演算を数ｎｓ以下で実行するＬＩＳＣＡＲ１に比べると、はるかに演算時間は長くなってしまう。従って、文書画像処理、文字認識処理等のように処理内容が多様で、サポートする演算の内容を絞り切れない場合にはＬＩＳＣＡＲのような汎用的な構成が適しているといえる。

5．4　大規模システムＡＡＰの構成

ＬＳＩ－ＣＡＤの自動配線における迷路法のように大規模な２次元配列データを直接処理する必要があり、逐次処理プロセッサでは十分な速度性能の得られない処理の超高速化を指向して、ＡＡＰ１－ＬＳＩおよびＡＡＰ２－ＬＳＩを用いた大規模セルラ配列型ＳＩＭＤプロセッサＡＡＰ１[51]およびＡＡＰ２を試作した。ＡＡＰ２は、ＡＡＰ１に対し、高速化と高機能化をはかった改良版であり、構成自体はほぼ同一である。そこで、本節では、ＡＡＰ２の構成および設計について述べる。

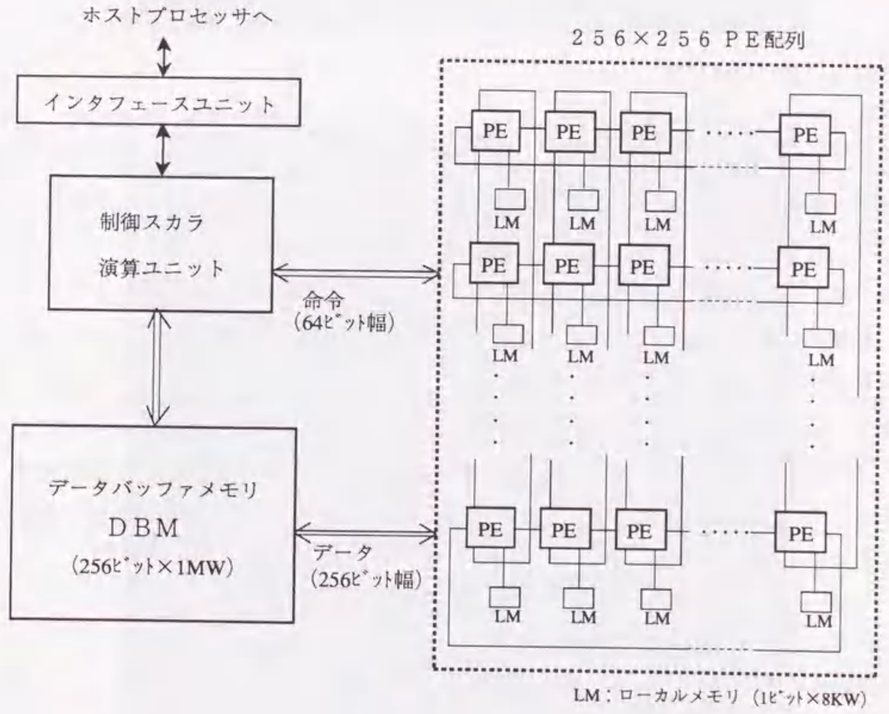


図５．１３　ＡＡＰ２の全体構成

表５．６　ＡＡＰ２のハードウェアの諸元

項目	内容
配列サイズ	２５６×２５６（ＡＡＰ２－ＬＳＩを６４個搭載したボード１６枚で構成）
メモリ容量	
データバッファメモリ　（ＤＢＭ）	３２Ｍバイト（２５６ビット×１Ｍワード）
インストラクションメモリ（ＩＭ）	１Ｍバイト（６４ビット×１２８Ｋワード）
配列データメモリ　（ＬＭ配列）	６４Ｍバイト（６５５３６ビット×８Ｋワード）
マシンサイクル（最小命令実行時間）	３３０ ns
基本演算性能	
８ビット固定小数点加算	7,３５５ MOPS
３２ビット固定小数点加算	2,００６ MOPS
ラックサイズ	１．３×１．５５×０．９５ m^3

５．４．１　全体構成

ＡＡＰ２の全体構成を図５．１３に、ハードウェアの諸元を表５．６に示す。２５６ｘ２５６のＰＥの２次元配列からなるＰＥ配列、制御スカラ演算ユニット、デュアルポート構成のデータバッファメモリ、インタフェースユニット等で構成している。

５．４．２　制御スカラ演算ユニット

各種の搭載メモリに対するアドレス・ＰＥ配列用の制御信号等の発生、およびスカラ演算等を行うユニットであり、シーケンサ、インストラクションメモリ、スカラプロセッサ、２ポートレジスタファイル、カウンタ、シフタ等で構成している。本ユニットの各部を効率良く制御するため、ＬＩＳＣＡＲ１同様、階層的な制御構造は採用せず、水平型のマイクロ命令に近い形式の６４ビット長の命令でハードウェアの各部を直接制御する方式を採用している。

５．４．３　ＰＥ配列とその実装技術

大規模なＰＥ配列をコンパクトに実装するために、ＡＡＰ２－ＬＳＩチップ１個と局所メモリ（ＬＭ）用にＳＯＰ(Small Outline Package)タイプの６４ＫビットＳＲＡＭ

８個を表と裏に搭載したＱＦＰ(Quad Flat Package)タイプのセラミックのモジュール（図５．１４参照）を開発した。このモジュールを６４個載せた図５．１５に示すプリント基板１６枚で、２５６ｘ２５６のサイズのＰＥ配列を、わずか０．２m^3のスペースに実現している。このモジュールは、高密度実装が可能なだけでなく、不良時の交換も比較的容易であるという利点ももっている。

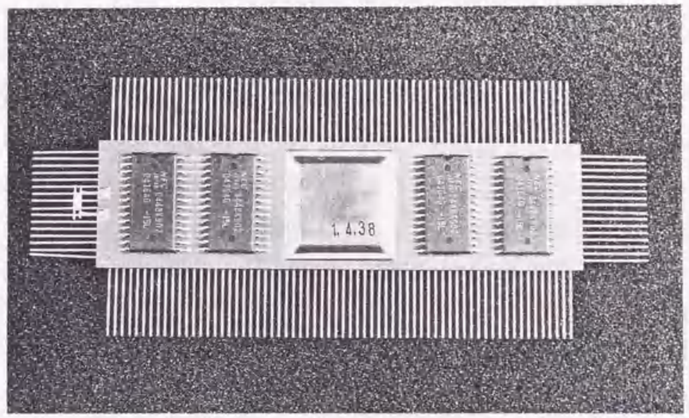


図５．１４　ＡＡＰ２モジュール

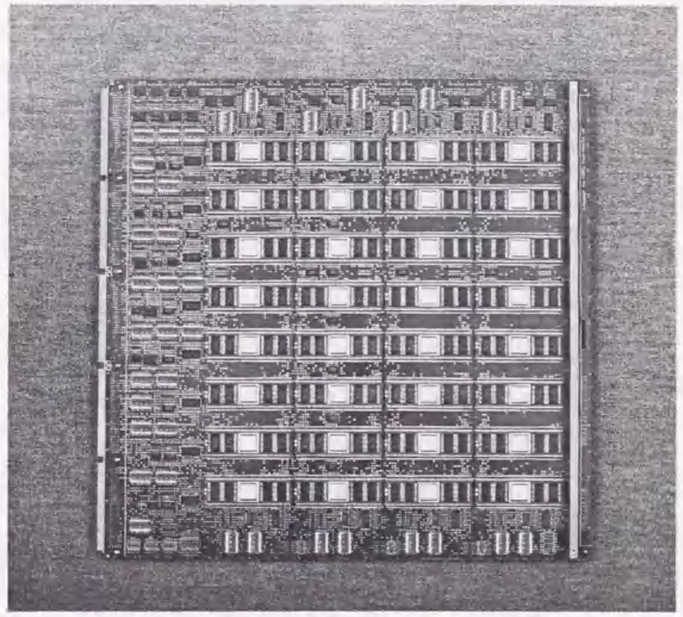


図５．１５　ＡＡＰ２　ＰＥ配列ボード

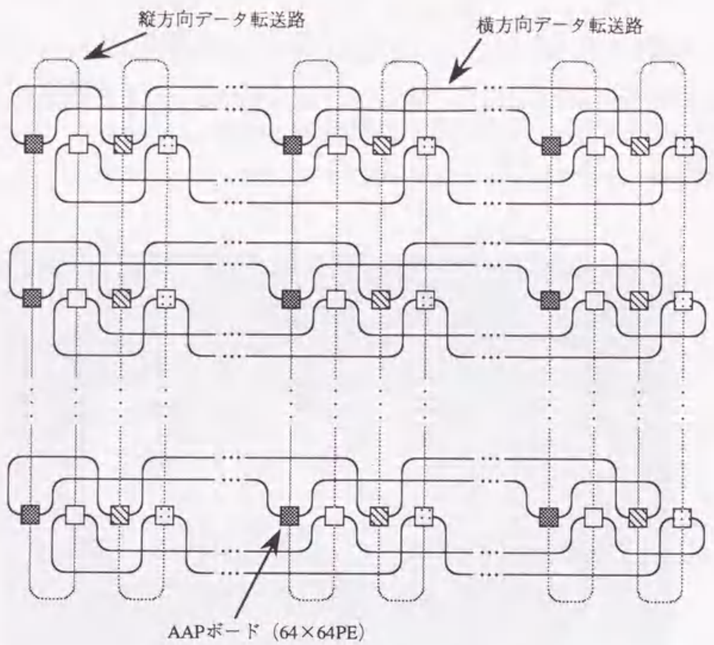


図５．１６　ボード間接続図

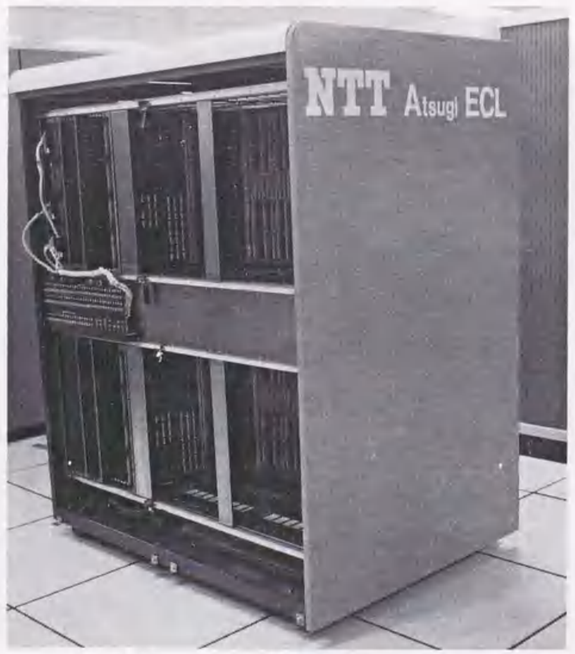


図５．１７　ＡＡＰ２システム

基板間の接続は、モジュラリティ確保のために、図５．１６のように接続している。この接続構成により、実装時に特定部分の接続距離増大のないボード間接続距離均一の条件を満たすトーラス結合を実現している。また、接続の規則性が高く、ボードの増設により、ＰＥ配列のサイズを容易に拡大することができる。

試作した装置のきょう体（図５．１７参照）は、制御ユニット、データバッファメモリに加え、２５６ｘ５１２までのＰＥ配列が実装できるようにしている。

５．４．４　ソフトウェア開発環境

ＡＡＰ２は専用指向が強く、高速化可能な処理は限定される。しかし、その配列データに対する超高速性をうまく引出すことができれば、効果は非常に大きい。このための試行錯誤が効率良く行えることが必要である。このような観点から、ＡＡＰ２のプログラム開発環境として、ＡＰＬ風の専用言語ＡＡＰＬとＩＢＭの汎用コンピュータ上で動作するコンパイラを開発した[52]。ＡＡＰＬは、ＡＡＰのビットプレーン単位の並列演算と伝搬演算機能を、ＡＰＬにならい、それぞれ２次元配列間の演算として簡潔に表現可能とするとともに、伝搬演算機能を『走査』の一部として表現できるようにしている。また、ＡＡＰ２の役割の異なる４つのメモリ領域に対応する変数形式を設け、効率のよいプログラムが記述できるようにしている。

５．４．５　基本性能

各ＰＥは局所メモリ内にある２つのオペランドの間でビット直列型の論理あるいは算術演算を行い結果を局所メモリ内の別の領域に得る基本的な演算をビット当たり３クロックサイクルで実行できる。これをもとに例えば８ビットの固定小数点データに対して、ＰＥの稼働率が１００％の場合の加算速度を求めると表５．６の試作したハードウェアの諸元に示すように７，３５５ＭＯＰＳ(Mega Operation Per Second)となる。

５．５　むすび

１ビットプロセッサの配列からなるセルラ配列型ＬＳＩ（ＡＡＰ２－ＬＳＩ）、さらに、このＡＡＰ２－ＬＳＩを用いた２種類のセルラ配列型ＳＩＭＤプロセッサＬＩＳＣＡＲ１（LIne Scanable Cellular ARray processor 1）およびＡＡＰ２（Adaptive Array Processor 2）を、設計、試作した。

ＡＡＰ２－ＬＳＩは、セルラ配列構成の設計上の利点をすべて引出すことを主眼に以下の３つの設計手法

①１ビット構成で、かつ算術論理演算機能、ＰＥ間のシフト転送機能、最も単純なバ

イパス型伝搬演算機構等の必要最小限の機能、機構を組込むにとどめ、ＰＥの単純性およびＰＥ配列の規則性を確保する方法、

②トランジスタ当たりの実現機能の高いｎチャネルのＭＯＳトランジスタ単独で構成するパストランジスタ論理を多用する方法、

③ＮｘＮのＰＥ配列を（Ｎ／２）ｘ（２Ｎ）配列としてＰＥのレイアウト形状を縦長とすることによって共通の制御信号をＰＥ上にはしらせ、配線専用領域を低減する高密度のレイアウト方法、

を採用して、スタンダードセルベースのフルカスタムＬＳＩの４～５倍の高集積度を達成し、２μｍのＣＭＯＳ技術で8.8ｘ8.16mm^{2}のチップに８ｘ８ＰＥの搭載を実現した。

ＬＩＳＣＡＲ１は、小型のセルラ配列型ＳＩＭＤプロセッサをめざし、４個のＡＡＰ２－ＬＳＩに１６ビットの１チップＤＳＰ、ビットスライスＡＬＵ、標準ＤＲＡＭを組合わせることによって、Ｂ４版サイズのボード１枚に実現した。その性能は、長さが２５６のビットライン単位の算術論理演算で2,860ＭＯＰＳの性能を、８ビットのグレイスケールライン間の差分絶対値累積処理（シティブロック距離計算）では１８０ＭＯＰＳの性能をそれぞれ得ている。２値画像に対する性能も非常に高く、基本的な演算では数ｎｓ／画素以下で、平滑化や９０度回転でも数十ｎｓ／画素で実行できる。

ＡＡＰ２は、超高速のセルラ配列型ＳＩＭＤプロセッサをめざしたもので、1,024個のＡＡＰ２－ＬＳＩと8,192個のＳＲＡＭからなるＰＥ配列部に、主にディスクリートのＳＳＩ、ＭＳＩで組み上げた制御部、ＳＲＡＭで構成したインストラクションメモリ、データバッファメモリ等を組合わせることで構成している。ＰＥ配列部は、ＡＡＰ２－ＬＳＩチップとＳＲＡＭを直付けするＱＦＰタイプのセラミックモジュールを両面実装した多層ボードと、そのボード間を均等距離でつなぐ接続構成とで、わずか0.2m^{3}のスペースに実現した。性能は８ビットの固定小数点データに対する加算で最大7,355ＭＯＰＳの性能を得ている。

第６章　８ビットセルラ配列型ＳＩＭＤプロセッサシステムの設計と試作

６．１　はじめに

ＬＩＳＣＡＲ１、ＡＡＰ２で採用した単純で規則性の高い１ビット構成のＰＥ配列は、大規模かつ高集積のＬＳＩが容易に設計できる点で優れている。しかし、処理データの語長が大きくなると演算の効率が低下する、回転や間接アドレシングなどの汎用化に必須とも言える機能が効率良く実行されない等の欠点が有り、応用対象が限られたり、性能／コストの面で並列処理の利点を活かせない場合がある。

これに対し、最近ではＬＳＩの設計技術が着実に進歩し、そのおかげで、多ビット構成のＰＥ配列も比較的容易に設計できるようになっており、ＡＡＰ２－ＬＳＩ程には単純な構成にこだわる必要がなくなってきている。そこで、ＡＡＰ２－ＬＳＩの欠点を克服すべく、従来からのＡＡＰの可変構造に加え、第２章で導いたセレクティブツリー構成伝搬演算機構、第４章で示した転送回転型メモリアクセス機構を組込んだ８ビット構成のＰＥ配列を搭載したＡＡＰ３－ＬＳＩを開発した。また、種々の機器へ手軽に組込み可能な程度までの小型化、経済化を指向したセルラ配列型ＳＩＭＤプロセッサＬＩＳＣＡＲ２を試作した。

本章では、このＬＩＳＣＡＲ２の実現上の鍵となったＰＥ配列部用ＬＳＩを中心に、ハードウェア構成、基本性能評価等について論じる。

６．２　小型経済化のためのアプローチ

小型経済性を確保するには、ハードウェア規模の低減に加え、部品コストの低減も必要である。そこで、アーキテクチャを設計する上で、次の３つの構成条件を設定した。

①ＰＥ配列部は、一種類の専用ＬＳＩと局所メモリ用のＤＲＡＭを規則的に並べるだけで構成する。

②各メモリは原則としてＤＲＡＭで構成する。

③ＰＥ配列部以外の構成には極力汎用のＬＳＩを活用する。

ここで、①は、全構成ゲート数の大半を占めるＰＥ配列部のハードウェア規模低減のための条件である。この条件を満たすには、プロセッサ間転送用ネットワークや２次元アクセス機構等もＰＥ配列とともに１チップに搭載しなければならない。このために、端子数やハードウェア規模が制限されるが、許される範囲内で、極力、性能や汎用性の向上をめざす。

②は、大容量のメモリを小型経済的に構成するための条件である。ＤＲＡＭは、

ＳＲＡＭに比べ、速度性能が劣るので、性能向上の制約となる。この制約に対しては、アクセス幅の拡大、メモリアクセスと演算の並列実行等により対応する。それでも、プロセッサの性能に追い着かない分については、演算性能がアクセス速度に見合うまで、プロセッサ側の構成を単純化する。高性能化よりも小型経済化を優先するからである。

③は、ハードウェア規模低減の効果が小さい部分について、開発コストを低減することで、間接的に経済化をはかるためである。

６．３　ＳＩＭＤプロセッサ構成

６．２節の方針に基づいて設計した本ＳＩＭＤプロセッサの全体構成を図６．１に示す。本プロセッサは、並列演算部であるＰＥ配列部と、制御部、データメモリ、アドレス生成器等からなる標準的な構成のＳＩＭＤプロセッサである。制御部は、コントローラに１６ビット固定小数点ＤＳＰを用い高速の制御機能とスカラ演算器機能を小型経済的に実現している。マイクロプログラムメモリ（８０ビットｘ６４ＫＷ）とデータメモリ（１６ビットｘ１ＭＷ）は１Ｍないし４ＭビットのＤＲＡＭを用いることで小型化を図っている。以下、本プロセッサの性能上のかなめであるＰＥ配列部の構成とその構成要素であるプロセッシング要素（ＰＥ）の構成について説明する。

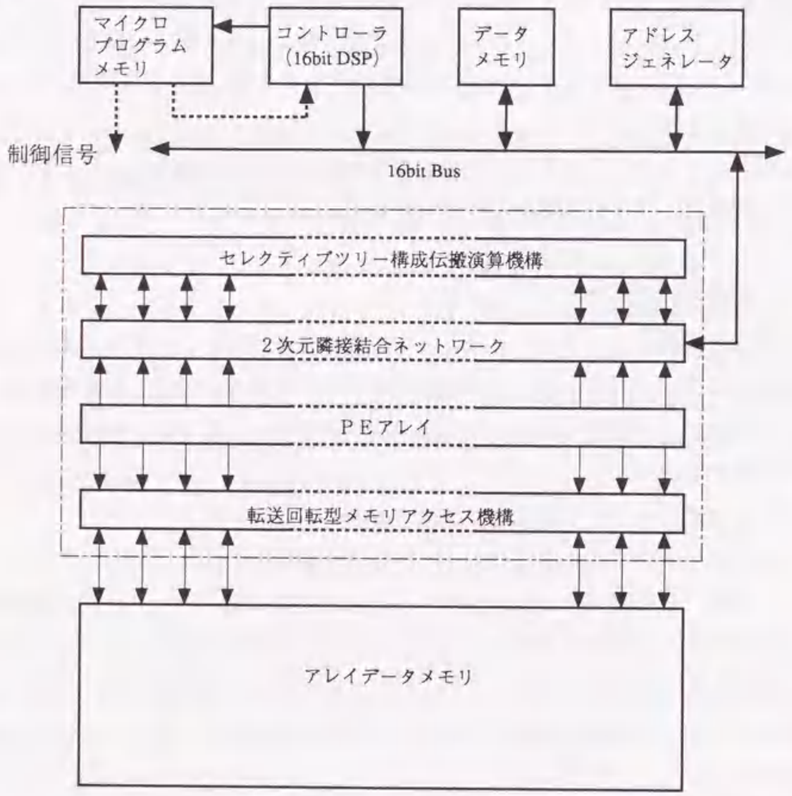


図６．１　ＬＩＳＣＡＲ２のＰＥ配列のブロック構成

６．３．１　ＰＥ配列部の構成

このＰＥ配列部は、図６．１に示されるように８ｘ８の２次元接続のＰＥ配列にセレクティブツリー構成の伝搬演算機構と転送回転型のメモリアクセス機構を付加した構成を採っている。セレクティブツリー構成の伝搬演算機構は、図２．１２に示す３個の２入力１出力のセレクタからなる伝搬演算エレメントの縦続接続である伝搬演算ユニット（ＳＰＵ）をＡＡＰ３－ＬＳＩ内でツリー状に２段接続して構成している（図６．３のＳＴＮ部参照）。ＬＳＩ間については、外部の部品点数を減らすために、ツリー型ではなく、単純な縦続接続としている。また、転送回転型メモリアクセス機構（ＴＲＭＡＰ）については、図４．３に示す転送回転型メモリアクセス機構をそのままＡＡＰ３－ＬＳＩごとに組込んでいる。

６．３．２　プロセッシングエレメント（ＰＥ）の構成

図６．２にＰＥのブロック構成を示す。２個の転送演算部が互いに共有するレジスタファイルＲＦ、ＩＯＢで結合される複合的な構成を採っている。ここで、ＲＦは主に演算に用いるのに対し、ＩＯＢはＬＳＩ外部に設ける配列データメモリを効率良くアクセスするのに用い、配列データメモリからのロードストアを他の演算と並行して実行することができる。

一方の転送演算部は、画像処理や文字認識処理において、処理の大半を占める演算精度に合せ、演算幅を８ビットに設定している。もう一方は、ＰＥ内のビット処理と、ＰＥ間のビット転送をスムーズに行うために設けたもので、演算幅を１ビットに設定している。１ビットの転送演算部は、もちろん８ビットの転送演算部に組込むことが可能であるが、このように独立した構成を採ることにより、元になったＬＩＳＣＡＲ１との互換性を考慮した高度のビット処理機能、ビット転送機能を比較的簡単に設計することができる。

８ビットの転送演算部は、ＡＬＵ、シフタ、ルーティングレジスタ（ＲＴＲＧ）、乗算器等で構成している。乗算器は、ＡＬＵ、ＲＦ等による累積処理と並行に動作できるようにしており、識別処理やニューラルネットワークのシミュレーションに必要な積和演算を効率良く実行できる。

１ビットの転送演算部の特徴は、２個の１ビットＡＬＵを持つことである。これは、

セレクティブツリー構成伝搬演算機構に対し、先行的な２組の演算結果を供給するために設けたものである。

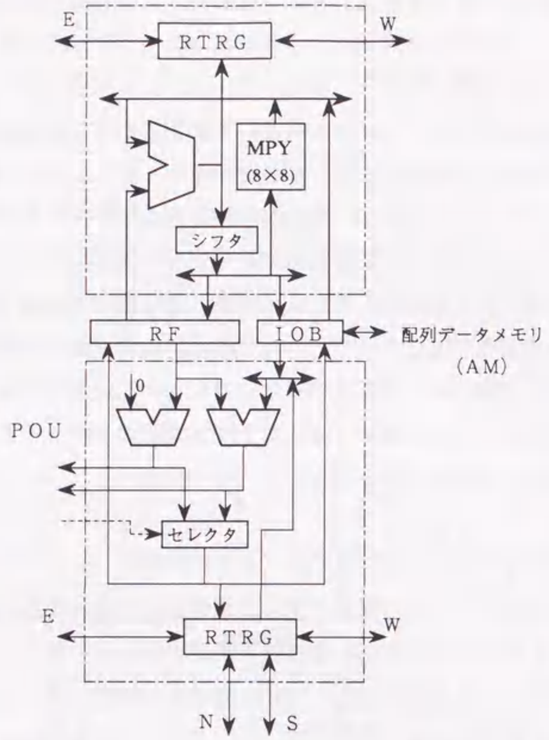


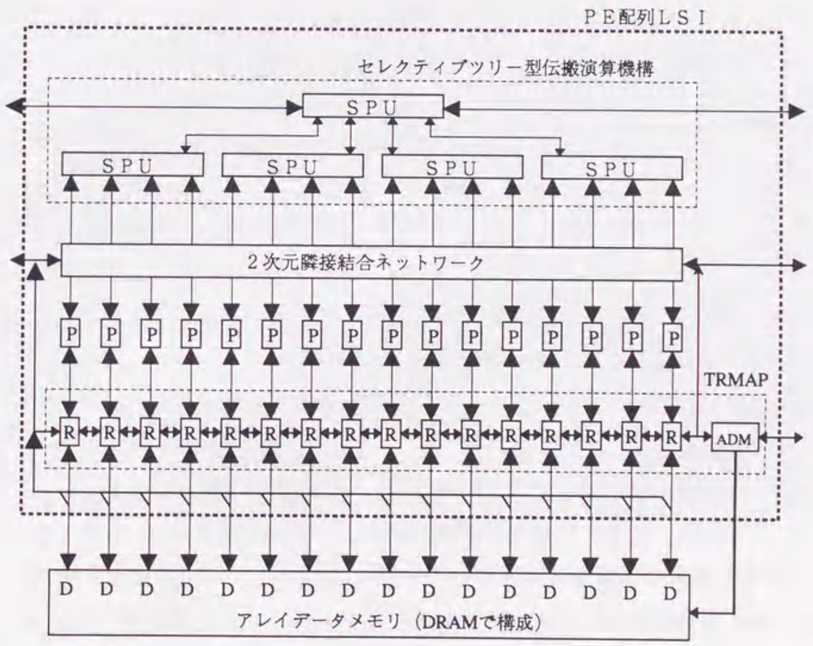
図６．２　ＰＥ構成:８ビットバス :１ビットあるいは２ビットバス

６．４　ＰＥ配列ＬＳＩの設計

この種のＳＩＭＤプロセッサのＰＥ配列部は、通常、フルカスタムＬＳＩで構成される。これは、ＰＥ配列部が、性能、小型経済性の決め手となるからである。しかし、標準ＤＲＡＭの使用を前提にすると、データのアクセスネックからフルカスタム構成により得られるＬＳＩ内部の高い演算性能を活かし切ることは容易でない。そこで、開発コスト、開発期間の点で有利なゲートアレイを用い、標準ＤＲＡＭのアクセス速度にバランスする性能の達成をめざした。ゲートアレイは、ローゲート数１２８Ｋの0.8μｍＡｌ２層のＣＭＯＳ　ＳＯＧ（Sea Of Gates）を用いた。ＬＳＩの最小サイクルタイムは、標準ＤＲＡＭを高速ページモードで動作させる場合のサイクルタイムに適合する５５ｎｓ（保証値）が達成された（表６．１参照）。

表６．１　ＬＳＩの諸元

ＰＥ数	１６
サイクルタイム	５５ns　（保証値）
電源電圧	５　±　０．５　Ｖ
パッケージ	１６０　pin QFP
プロセス技術	0.8μm　Aℓ　２層 CMOS
基本セル構成	ＳＯＧ
セル数	１２８　ｋセル
チップサイズ	11.76 × 11.82　mm^2



ＳＰＵ　：セレクティブ伝搬ユニット(Selective Propagation Unit)
Ｐ　　：８ビットプロセッサ
Ｒ　　：ルーティングレジスタ(Routing Register)
ＡＤＭ　：アドレス修飾器(ADdress Modifier)
ＴＲＭＡＰ：転送回転型メモリアクセス機構
(Path for Transmission,Rotation,and Memory Access)

図６．３　ＡＡＰ３－ＬＳＩの構成

6．4．1　全体設計

ＰＥ配列ＬＳＩの構成を図６．３に示す。このＬＳＩを規則的に並べるだけで、ＰＥ配列部全体を構成できるように、プロセッサ部分配列と共に、セレクティブツリー構成伝搬演算機構、２次元の隣接接続ネットワーク、転送回転型メモリアクセス機構等をすべて搭載している。ただし、この図では理解し易くするため、図６．１と同様に、プロセッサ部分配列と、２次元隣接接続ネットワーク、転送回転型メモリアクセス機構を別々に示しているが、実際にはハードウェア規模低減のために、ＰＥ部分配列に一体化する形で構成している。

6．4．2　ＰＥ設計

ゲートアレイでは、ハードマクロやＰＥ配列の規則性を利用したレイアウトの高密度化は困難であり、同一チップサイズのフルカスタムＬＳＩに比べると、搭載可能なゲート規模は数分の１以下と小さい。従って、全体のゲート数の大半を占めることになるＰＥ配列部の構成単位であるＰＥについては、ハードウェア規模低減が必須といえる。ここでは、高速化との両立の観点から特徴的なレジスタファイルと乗算器の構成について説明する。

レジスタファイルは、８ビットｘ３２ワード構成と通常のレジスタを組合わせる構成法ではＬＳＩ全体で見るとかなりのゲート数になる。そこで、次の２つの方策、

①アクセス機能を制限して、同一サイクルに１オペランドのみ読みだす構成とし、レジスタファイルの単純化をはかる、

②６４ビット幅のハードウェアマクロのＲＡＭ２個を、各ＰＥが８ビット分づつ分割して用いる

によりゲート規模の低減をはかっている。

また、乗算器は、大半の処理で被乗数を配列データメモリからロードするために、そのロード速度以上の演算速度は不要との判断に基づいて、ハードウェア規模低減をはかっている。具体的には、８ビットの加算器とブースのリコーダを用いたシリアル型の構成とし、８ｘ８の乗算を、被乗数のロード時間に等しい４クロックサイクルで実行できるようにしている。

6．4．3　セレクティブツリー構成伝搬演算機構の設計

図６．３に示されるように、伝搬演算をビット単位で行うビット直列型のセレクティブツリー構成の伝搬演算機構を搭載している。ＬＳＩ内で閉じたツリーとすることによって、ＬＳＩ間を単純に隣接結合するだけで、伝搬演算系を構成できるようにしている。なお、このビット直列型のセレクティブツリー構成伝搬演算機構を各ビットに設ける並列構成により、伝搬論理演算の高速化が可能であるが、ここではゲートアレイのハード規模制約から採用していない。

6．5　並列プロセッサシステムの実現

外部バスインタフェース部を含む並列プロセッサ全体を、ＶＭＥ規格のボード１枚に実現した。さらに、この１ボードプロセッサ複数枚を、ワークステーションのバックエンドプロセッサとして動作させる実験システムを開発した。

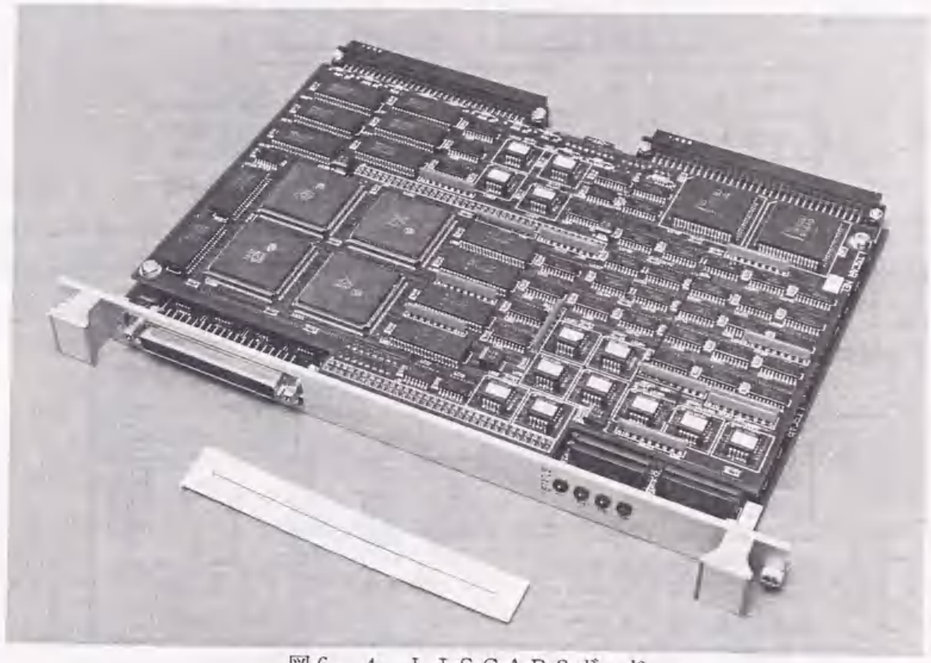


図６．４　ＬＩＳＣＡＲ２ボード

6．5．1　ボード

図６．４に試作したボードの写真を示す。主に、外部バスインタフェース部、制御部を搭載したマザーボードと、ＰＥ配列部、アドレス生成部、マイクロプログラムメモリ等を搭載したドーターボードとを、積み重ねて構成している。配列データメモリと８０ビット幅のマイクロプログラムメモリには、高速ページモードで動作させる１６ビット幅の４Ｍおよび１ＭビットＤＲＡＭを用いて、経済化をはかっている。また、アドレス生成部は１６ビットスライスのＡＬＵを用いて構成し、アドレス生成以外の演算にも利用できるようにしている。ボードの諸元を表６．２に示す。

表6．2　ボードの諸元

サイクルタイム	62.5　ns
搭載ＬＳＩ数	4
メモリ容量	
配列データ	4 MB
データ	2 MB
プログラム	640KB
ボードサイズ	233.3 × 16　mm^2
ボード規格	VME

6．5．2　システム

図6．5に、試作したシステムのブロック構成を示す。ワークステーションからVMEバスを介して、複数のプロセッサボードを動かす構成を採っている。適当な周辺ボードと組合わせることによって種々の用途に適用可能である。ワークステーションには、ＬＩＳＣＡＲ用から記述性の向上をはかったマクロアセンブリ言語に対応するマクロアセンブラとシンボリックデバッガを搭載している。

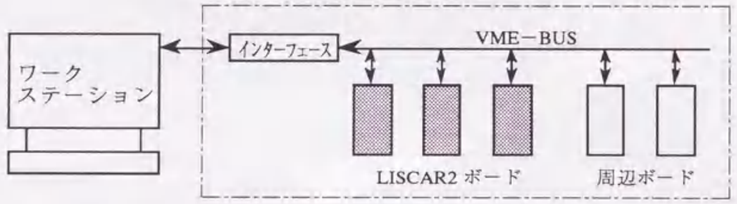


図6．5　ＬＩＳＣＡＲ2システムの構成

6．6　基本性能評価

基本演算性能を示すとともに、パターンデータ処理の重要な演算の1つである９０度回転性能を明らかにした。

6．6．1　基本演算

表6．3に基本性能をまとめた結果を示す。配列に対する算術演算、論理演算の性能は、並列処理のおかげで非常に高い。ＡＬＵと並列に設けた乗算器により、高い積和演算性能を確保している。表6．4に示す例からも明らかなように、これらを組合わせて

行う配列演算の処理時間もかなり短い。乗算器の効果の出るユークリッド距離計算では、５３３ＭＯＰＳとＬＩＳＣＡＲ1の２５倍以上の性能が、乗算器の効果の出ないシティブロック距離計算でも３７９ＭＯＰＳとＬＩＳＣＡＲ1の2倍以上の性能が得られている。この高い基本演算性能を活かすことによって、ニューラルネットワーク処理では２０ＭＣＵＰＳの性能が得られる[53]。

また、伝搬演算についても、セレクティブツリー構成伝搬演算機構のおかげで、ゲートアレイを用いながら、２ｎｓ／ＰＥと、ＰＥ間を直接伝搬させるＬＩＳＣＡＲ1に比べ1桁高速化されている。

表6．3 配列データに対する基本性能

演算内容	性　能	精　度
算術論理演算	1024 MOPS	8ビット固定小数点
乗累算	512 MOPS	乗算：8ビット固定小数点 累算：32ビット固定小数点
伝搬演算	2 ns/PE	ビット単位

表6．4 基本的な配列演算に対する処理時間

配列演算の内容		演算性能 (MOPS)	演算時間 (ms)	演算の条件
距離計算	ｼﾃｨﾌﾞﾛｯｸ距離	379	8.3	配列要素の語長：8bit ベクタの次元数：256 参照ベクタの数：4,096
距離計算	ﾕｰｸﾘｯﾄﾞ距離	533	5.9	
行列・ベクタ間の乗算		357	2.2	配列要素の語長：8bit 行列のサイズ　：1,536×256 ベクタの次元数：4,096

6．6．2　９０度回転処理

９０度回転は、4．4．5で述べたように転送回転型メモリアクセス機構を用いて、効率良く行える。図6．6は、その処理アルゴリズムの概要を示したものであり、

①図の上部に示す６４ｘ６４の2値データ配列（ビットプレーン）から、対角方向に並ぶ１６ｘ１６の部分ビットプレーン4組（図ではＡ，Ｎ，Ｋ，Ｈ）を、９０度回転をかけながら、ＴＲＭＡＰのルーティングレジスタ（ＲＴＲＧ）配列に読みだす、

②読み出した部分配列4個を、左方向に1個分シフトする、

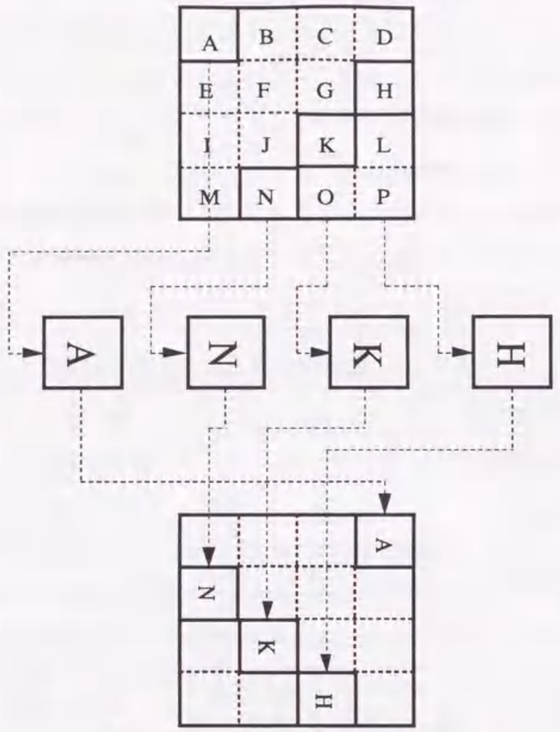


図６．６　９０度回転アルゴリズム

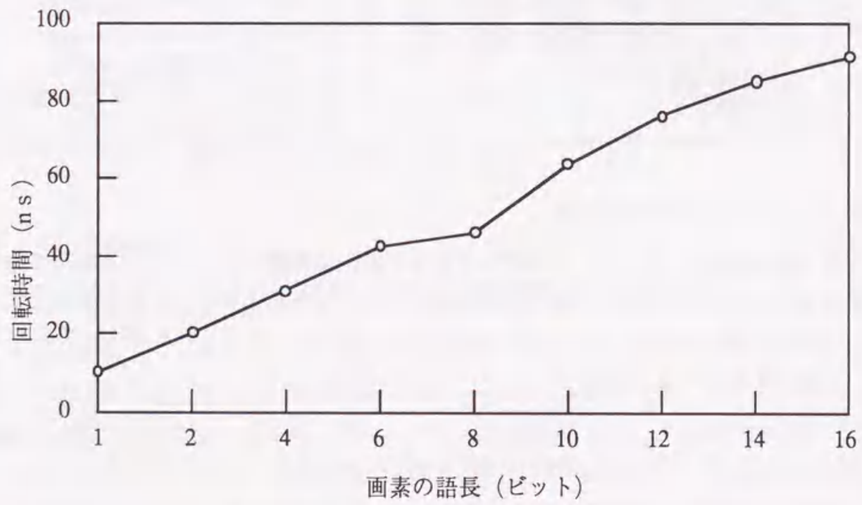


図６．７　画素当たりの９０度回転時間

③シフト結果（図ではN，K，H，A）を読みだし時の対角方向と直交する方向に書き戻し、図下部の回転後の６４ｘ６４のビットプレーンを形成する

の３ステップを、読みだし位置と書込み位置を更新しながら、４回繰り返すことで行う。図６．７は、このアルゴリズムをもとに作成した回転マクロについて、６４ｘ６４のプレーンに対する９０度回転の処理時間を測定した結果である。この図から明らかなように、転送回転型メモリアクセス機構は、従来の転送機構にわずかなハードウェアの追加で実現されているにもかかわらず、６４ｘ６４のビットプレーンが回転器への読みだしの６４クロックサイクル、回転器からの書込みの６４クロックサイクルのみで実行される理想的な構成の９０度回転器の１／３～１／４の速度で、対象画像のグレイトーンによらず回転されている。この結果、ＬＩＳＣＡＲ１の４倍以上の回転速度が達成されている。

しかし、図６．７から明らかなように、要素の語長が大きくなるにつれて速度がかなり低下する。この原因は、転送回転型メモリアクセス機構の部分配列単位の回転が１６ｘ１６の部分ビットプレーンであるためである。従って、必要に応じて、各ＰＥでＲＴＲＧを複数持つ構成とし、部分配列単位で複数の部分ビットプレーンを同時に回転することによって高速化可能である。

６．７　むすび

ＰＥ配列部をゲートアレイＬＳＩ（ＳＯＧ）４個と標準ＤＲＡＭのみで構成することによって、小型化、経済化をはかった１ボードＳＩＭＤプロセッサＬＩＳＣＡＲ２を実現した。

はじめに、各メモリは原則ＤＲＡＭで構成する、ＰＥ配列部以外は汎用のＬＳＩで構成する等の小型経済化、開発コスト低減のための方針を明らかにした。次いで、これらの方針に基づいて設計したプロセッサ構成、ＰＥ配列部用ＬＳＩ（ＡＡＰ３－ＬＳＩ）の構成、ボード構成、システム構成を示した。プロセッサ構成は基本的にＬＩＳＣＡＲ１のそれを踏襲しながら、性能向上の鍵となるＰＥ配列部について、

①ＰＥを１ビット転送演算部と８ビットの乗算器内蔵の転送演算部の複合構成とする、

②ＰＥ間の２次元の隣接結合ネットワークにセレクティブツリー構成の伝搬演算機構と転送回転型メモリアクセス機構を組合わせる

の抜本的な改良を加えるとともに、プロセッサアレイのサイズを２５６から６４に縮小している。ＡＡＰ３－ＬＳＩは、伝搬演算機構、転送回転型メモリアクセス機構を一体化した１６個のＰＥの部分配列からなる構成とし、０．８μｍのＳＯＧ型ゲートアレイ

の技術により実現している。ボードは、４個のＡＡＰ３－ＬＳＩと８個の１６ビット幅の４ＭＤＲＡＭでＰＥ配列を構成して、プロセッサ全体をＢ５版サイズのボードに実現している。

さらに、このＬＩＳＣＡＲ２の性能を評価し、基本的な演算に対しては、算術論理演算性能が１０２４ＭＯＰＳ、伝搬演算速度が２ｎｓ／ＰＥと非常に高い性能の得られること、複合的な演算に対しても、乗算器が有効に働くユークリッド距離計算で、５３３ＭＯＰＳとＬＩＳＣＡＲ１の２５倍以上の性能が、乗算器の効果の出ないシティブロック距離計算でも３７９ＭＯＰＳとＬＩＳＣＡＲ１の２倍以上の性能がそれぞれ得られることを示した。また、転送回転型メモリアクセス機構による９０度回転の性能を評価し、ＬＩＳＣＡＲ１の４倍以上、理想的な構成の従来型９０度回転器と比べても、１／３～１／４の回転速度の得られることを示した。

このようにＬＩＳＣＡＲ２は、単に算術演算、積和演算の性能が高いだけでなく、セレクティブツリー構成伝搬演算機構や転送回転型メモリアクセス機構を利用することによって、ＰＥ配列の局所ごとの総和演算、ＰＥ配列全体に対する放送、９０度回転等を効率良く実行できるので、配列演算を主体とする応用では、専用構成のハードウェアに匹敵する小型化、経済化が達成されよう。

第７章　大規模セルラ配列型ＳＩＭＤプロセッサＡＡＰのＬＳＩ－ＣＡＤへの応用

７．１　はじめに

５．３節で述べたＡＡＰは、そのセルラ配列構成から明らかなように、大規模な２次元配列データの処理に適している。この意味で、規模の大きな配列データを扱うＬＳＩ－ＣＡＤは、ＡＡＰに適した応用分野の一候補といえる。しかし、配列データを扱うといっても、ＡＡＰの得意とする短語長の２次元配列データに対する演算のみで処理できるわけではない。むしろ、従来はたとえ処理対象データがもともと短語長の２次元配列データであっても、処理中は、ベクトルデータに変換して処理することが多かった。これは、逐次型の汎用プロセッサでは、データ構造の規則性よりも、データ量の少なさ、すなわち演算量が少ないほど、高速に処理できるからである。このような状況は、２次元配列データのまま直接処理を進めるアルゴリズムの発展を疎外してきた。従って、ＡＡＰの性能を十分に引出すには、処理アルゴリズムの見直しから行わなければならない。

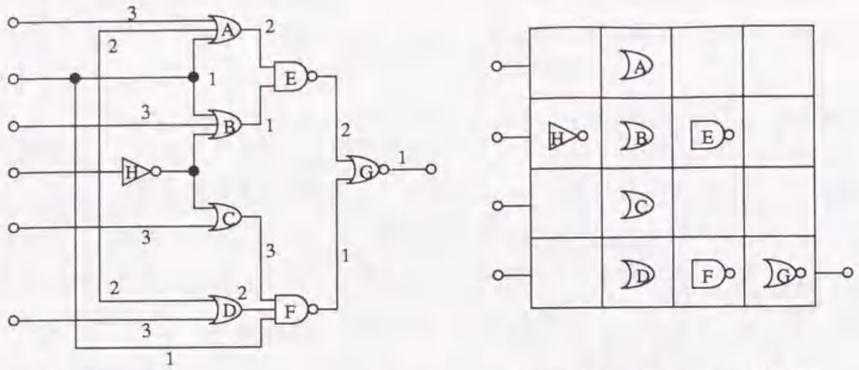
本章では、ＬＳＩ－ＣＡＤの中でも完全性を追究しようとすると汎用プロセッサでは処理時間が大きくなりすぎる論理シミュレーションと、セルラ配列型ＳＩＭＤプロセッサに特に適すると考えられるマスクパターンレイアウト設計における自動配線処理について、１ビット構成のＰＥからなるＡＡＰに適した処理アルゴリズムを提案する。さらにこれらのアルゴリズムに基づき、ＡＡＰ１を用いて行った処理実験の結果について述べる。

７．２　論理シミュレーション

論理ゲートをＰＥで、論理ゲート間の結線をＰＥ間の伝搬転送でそれぞれエミュレートすることにより、ＡＡＰ上で論理シミュレーションを行うことができる。

図７．１は、８ゲートからなる論理回路を４ｘ４のＰＥ配列に割付ける例を示している。ここで、それぞれの升目はＰＥを表している。また、未割付のＰＥは、データ転送専用のＰＥである。各ＰＥは、次式で示す論理演算を実行することによって、３入力までの任意のゲートをエミュレートすることができる。

$$G_{out}(T+1) = IV_{out} \oplus \bigwedge_{i=1}^{3} \{(ID_i(T) \oplus IV_i) \vee MA_i\} \qquad (7.1)$$



(a) 論理回路

ラベル1,2,3は論理シュミレーションにおける伝搬転送順を示している。

(b) PEアレイへのマッピング

空白のPEは、論理値の転送に使用している。

図７．１　論理回路のＰＥ配列へのマッピング

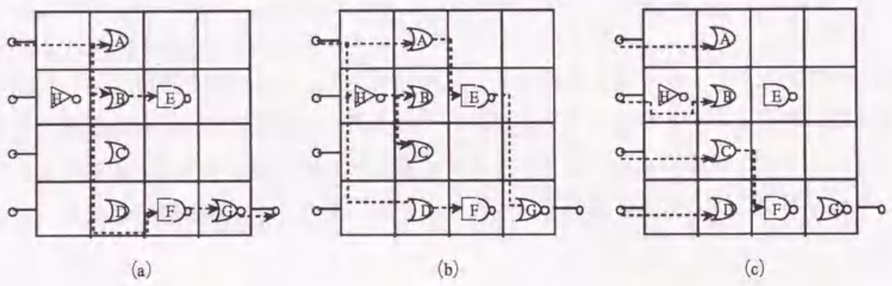


図７．２　伝搬転送を利用しゲート間の結線のエミュレーション
（ａ），（ｂ），（ｃ）は図７．１（ａ）のラベル付けされた
１，２，３の結線に対応

ここで、IV_iとIV_{out}は、入力信号と出力信号に対する反転制御用の係数データである。IDi(T)は、時刻Ｔにおけるｉ番目の入力端子に対する入力データ、MA_iはｉ番目の入力端子を活性化するか否かを示すマスクデータ、⊕は排他的論理和である。また、$G_{out}(T+1)$は、時刻Ｔ＋１における論理ゲートの出力である。論理ゲート間の結線は、図７．１（ａ）で１，２，３のラベルで指定される結線に対応し、右方向と下方向の２回の伝搬演算を組として、図７．２に（ａ），（ｂ），（ｃ）の順で繰り返すことで、

表７．１　シミュレーション時間の比較

回路の名称		加算器	浮動小数点データ正規化回路
ゲート数		２,１０５	３３７
シュミレーション時間（ユニット遅延）	ＡＡＰ１	４.０ ms	０.８９ ms
	７ＭＩＰＳ汎用コンピュータ	MIN　３２.０ ms MAX ２６０.０ ms	MIN　４.２ ms MAX ４９.０ ms

エミュレートする。

以上の論理ゲートのエミュレーションと論理ゲート間の結線のエミュレーションを繰り返すことによって単位遅延の論理シミュレーションが実現される。

このアルゴリズムの原理確認のプログラムを、クロック周波数０.６７ＭＨｚのＡＡＰ１に組込み、シミュレーション実験を行った。その結果得られたシミュレーション性能と７ＭＩＰＳの汎用コンピュータでのシミュレーション性能との比較を表７．１に示す。汎用コンピュータではイベントの生起に関わるゲートのみ論理値を評価するイベントドリブン型のシミュレーションアルゴリズムを採用しているので、シミュレーション時間のイベントの発生頻度が高い場合の値（上側）と少ない場合の値（下側）は、１０～１００倍も変化している。これに対し、ＡＡＰ１は無条件に全ゲートを評価するアルゴリズムなので、イベントの発生頻度に関わらずシミュレーション時間は一定で、汎用コンピュータの５から６０倍の性能が得られている。ＡＡＰ１のクロック周波数が低いため高速化の程度は、それほどでもないが、汎用コンピュータ並みの１０Ｍ～２０ＭＨｚのクロック周波数が実現されれば、さらに１０倍以上の高速化が可能である。また、ＡＡＰ２では転送方向とＡＬＵファンクションのＰＥ毎の個別設定機能を利用する４．３．２の方法で、もう一段の高速化が可能になる[32]。

７．３　自動配線

マスクパターンレイアウト設計における自動配線は、パターンデータを扱う点で自動配置[54]とともにセルラ配列型ＳＩＭＤプロセッサに適した処理といえる。しかし、従来は、汎用コンピュータの使用を前提に、基本的にベクトルデータの世界で逐次処理のアルゴリズムが発展してきた。従って、これらの自動配線処理をそのままＡＡＰに持ってきても高い性能を得ることは難しい。それでも、中には本来並列処理向きで、有効性は明白であっても演算量の多さから局部的にしか使用できなかったり、あるいは簡略化せ

ざるを得ないアルゴリズムも存在した。その代表的なアルゴリズムが最短経路を見つけることのできる迷路法であり、古くから専用ハードウェアあるいは並列プロセッサによる高速化の検討がなされてきた[55,56]。

本節ではこの迷路法と処理量の関係から従来より広く用いられてきたライン探索アルゴリズム[57]を組合わせた新しい配線アルゴリズム[58]と、それによる自動配線のＡＡＰ１での実験結果について述べる。

７．３．１　新配線アルゴリズムの概要

はじめに新アルゴリズムについて３つの端点間の配線を例に説明する。この例の配線処理は次の７ステップを順に実行することで実現される。

ステップ１：端点Ｔ1から、昇順のラベルを振りながら波面を最も近い端点Ｔ2に到達するまで広げる［図７．３（ａ）］。

ステップ２：ステップ１とは逆に端点Ｔ2から、降順のラベルを振りながら波面を端点Ｔ1に到達するまで広げる［図７．３（ｂ）］。

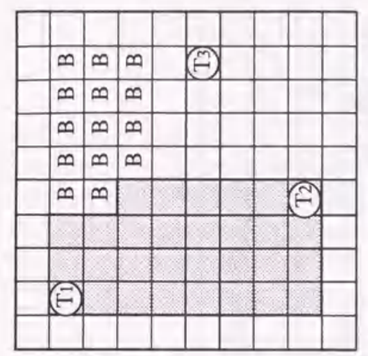
ステップ３：ステップ１，２で得られた昇順と降順のラベルの論理積をとり、互いのラベルが同一の領域（網点部）を抽出する［図７．３（ｃ）］。この領域は、波面の伝搬の性質から明らかなように、Ｔ1、Ｔ2間の最短経路の集合である。

ステップ４：ステップ３で得た網点領域の周辺から、昇順ラベルの波面を端点Ｔ3に到達するまで広げる［図７．３（ｄ）］。

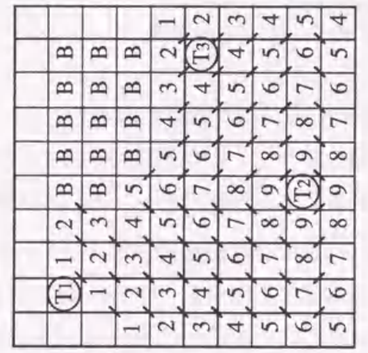
ステップ５：ステップ４とは逆に端点Ｔ3から降順ラベルの波面を網点領域の周辺に到達するまで広げる。

ステップ６：ステップ３同様、ステップ４，５で得た昇順と降順のラベルの論理積をとり、互いのラベルが同一のハッチ領域（ライン）を抽出する。このハッチのラインは、網点領域と端点Ｔ3の間の最短経路である。このラインに接する網点領域の点をＰとする［図７．３（ｅ）］。

ステップ７：網点領域、ハッチ領域に限定し並列のライン探索アルゴリズムによって、点Ｐから、端点Ｔ1，Ｔ2，Ｔ3に対する経路を見つける［図７．３（ｆ）］。



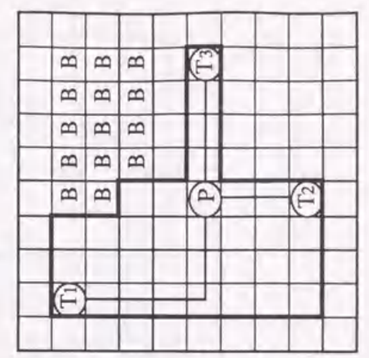
（ａ）Ｔ1から最近傍のＴ2に向かう波面の拡張



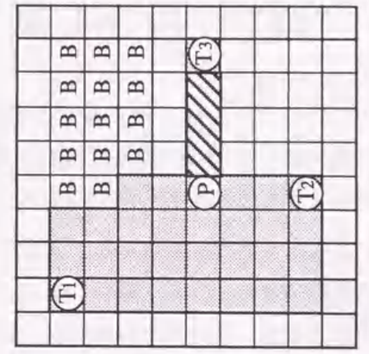
（ｂ）Ｔ2からＴ1に向かう波面の拡張

（ｃ）Ｔ1、Ｔ2間の最短経路領域（網点部）の抽出

（ｄ）網点部からＴ3に向かう波面の拡張



（ｅ）Ｔ3との間の最短経路領域の抽出



（ｆ）網点部内でのライン検索アルゴリズムによる配線

図７．３　新アルゴリズムの配線手順

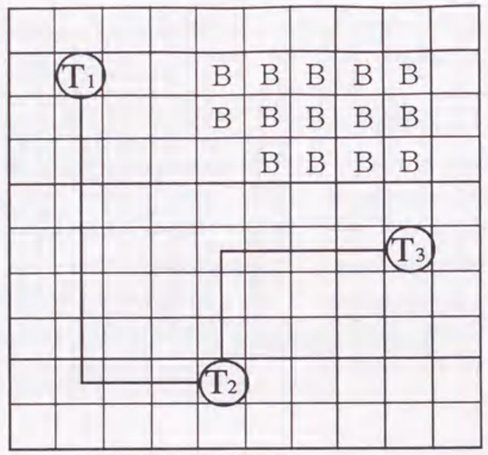


図７．４　ライン探索アルゴリズムによって引かれた結線

ライン探索アルゴリズムは、折れ曲がり数最小の経路を見つける能力を持っている。ハッチ内は長さ最小の経路の集りであることから、図７．３（ｆ）の配線は、長さが最小でかつ折れ曲がり数最小の経路で引かれている。すなわち、配線パターンとしては最も望ましいと考えられている直線状のスタイナツリー[59]の配線経路が得られている。配線経路の品質の高さは、ライン探索のアルゴリズムのみで求めた場合（図７．４）と比べるとはっきりする。この配線経路では、図７．３で得られた経路とは異なり、経路長も折れ曲り数も最小ではない。なお、新アルゴリズムは、ｎ個の端点間の配線に対しても、最短経路の集りからなる領域の抽出とそれに続くライン探索による配線を繰り返すことによって、３点間の場合同様に品質の高い経路を見つけることができる。

７．３．２　ＡＡＰ１の配線処理速度評価

ＡＡＰでは、７．３．１で示した新アルゴリズムを高速に処理できる。これは、迷路法の並列に波面を広げる処理をＰＥ間のシフト転送とビットプレーン間で、ライン探索法のライン単位の拡張を伝搬演算で、それぞれ効率良く処理できるからである。

図７．５はＡＡＰ１による迷路法の処理速度を、１ＭＩＰＳの汎用コンピュータと比較して示した結果である。ＡＡＰ１のクロック周波数は０．６７ＭＨｚと低いにもかかわらず、２５６ｘ２５６の配線領域に対し、１００倍以上の処理速度が得られている。

ＡＡＰ１の新配線アルゴリズムに対する処理速度を、迷路法の処理速度と比較して示す。新アルゴリズムは、迷路法より２倍程度高速に処理されている。これは、新アルゴ

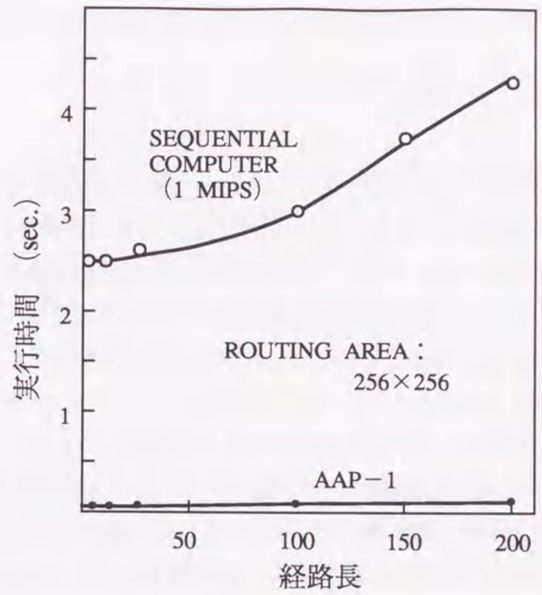


図７．５　迷路法による配線時間

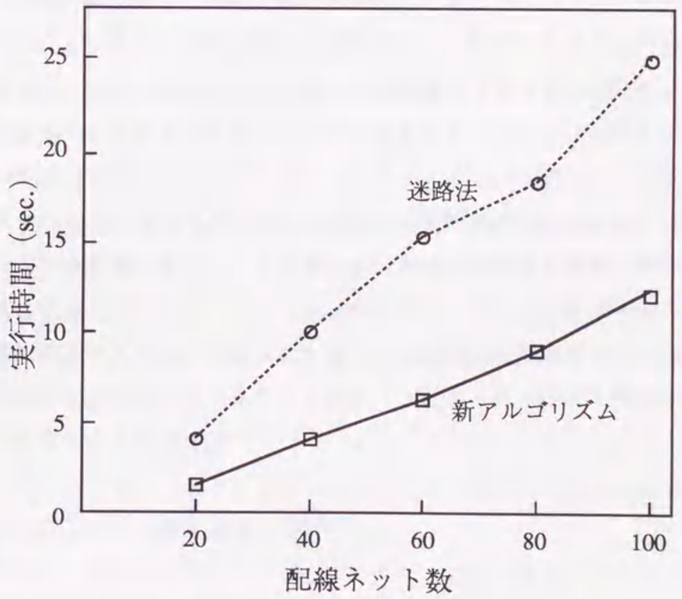


図７．６　ＡＡＰ１による多端子配線の実行時間（３端子／ネット）

リズムの方が、バックトレース時に必要となる逐次的な演算の量が少なくなるからである。

7．4　むすび

ＬＳＩ－ＣＡＤの中でも汎用コンピュータの能力では処理速度や処理品質の要求に答えることが難しい論理シミュレーションとマスクパターン設計における自動配線のＡＡＰ１での処理方法を明らかにし、実機を用いてその処理性能を評価した。

論理シミュレーションについては、３入力までの任意の論理ゲートを１個のＰＥで、論理ゲート間の結線を伝搬転送で、それぞれエミュレートするシミュレーション方法を明らかにし、実機上で汎用コンピュータの５～５０倍（クロックサイクルが同一の条件では５０～５００倍以上）の性能の得られることを確認した。

自動配線については、迷路法にライン探索アルゴリズムを組合わせたセルラ配列型ＳＩＭＤプロセッサ向きの新アルゴリズムを示した。次いで、その新アルゴリズムがＰＥ間のシフト転送と伝搬演算の活用により効率良く実行され、迷路法が１ＭＩＰＳの汎用コンピュータの１００倍程度の速度で実行されるのに対し、さらに２倍程度高速に実行できることを実機により確認した。

ＡＡＰ２については、このような実機での性能評価は行っていない。しかし、ＰＥの構成を改良したこと、クロックサイクルが５倍速くなっていること、外付けの局所メモリをＰＥ当たり８Ｋビット付加したこと等によって、ＡＡＰ１より１桁以上高速に処理できる見通しが得られている[52]。論理シミュレーションについては、新たな処理アルゴリズムの検討も行われており、それを用いることでさらに１桁以上の高速化が可能になる[32]。

このようにＡＡＰはその超並列構成を活かすことによって、自動配線、論理シミュレーション等の処理を高速に実行することができる。しかし、本章で行った評価は、それぞれ処理の核となる部分に対してのみである。周辺の処理には、処理量自体は多くなくとも、煩雑でかつ並列化困難な処理が存在する。実用的なシステムを構築するには、これらの周辺処理をどのように実行し、全体として高い性能を得るかを解決して行く必要がある。

第８章　小型セルラ配列型ＳＩＭＤプロセッサ ＬＩＳＣＡＲ１の文字認識処理への応用

8．1　はじめに

ＯＡの進歩と共に重要性が益々高まっている文書画像処理，文字認識処理，文書検索処理等は、２次元あるいは１次元の大規模な配列データを扱う割合の高い処理であり、汎用プロセッサのみでは実用的な処理性能の達成が難しい。このため、布線論理の組合わせを主体とする専用的なハードウェア構成により処理装置（例えば光学的文字読取り装置［ＯＣＲ］[60,61]）の性能向上が、はかられてきた。しかし、この方法では、ハードウェア構成を特定のアルゴリズムに適合化しているために

①アルゴリズムの変更に対応できず、異なるアルゴリズムの搭載が困難である、

②ハードウェアの開発を、アルゴリズムと独立には進められない

等の欠点があり、装置開発におけるコスト低減、期間短縮等の障害となっていた。

専用ハードウェアによらぬ高速化手法としては、複数のマイクロプロセッサによる並列処理を利用する方法があり、この分野でもいくつかの装置が開発されている。この中には数個の画像処理プロセッサによる汎用的構成の装置の報告もある[62]ものの、大半は高性能化のためにプロセッサ間でパイプラインを組む専用的構成をとっており、搭載アルゴリズムの可変性は小さい[63,64]。

これに対し、小型のセルラ配列型ＳＩＭＤプロセッサであるＬＩＳＣＡＲは、

①文書画像処理，文字認識処理で大半を占める８ビット以下の短語長の配列データ処理を得意とする、

②汎用的な構成でも構成プロセッサ数を増やすことによって性能向上が可能である

等の特性を備えており、小型・高速性と搭載アルゴリズムの可変性を両立できる可能性がある。このような観点からＬＩＳＣＡＲへの認識アルゴリズムの異なる複数の文字認識処理の搭載を試みた。本章では、これらの処理をＬＩＳＣＡＲ上に搭載する際の具体的な処理方法を示すとともに、処理性能を評価し、問題点を明らかにして行く。

8．2　手書き英数カナ認識処理[65]

専用イメージスキャナやファクシミリを入力手段とする手書き文字の認識処理システムへの適用をめざし、次の要求条件を設定した。

①ファクシミリ入力の文字パターンデータに対し、汎用スキャナ入力の文字認識装置並みの認識精度を達成すること。すなわち、不特定多数が丁寧に筆記した数字に対

して９９％程度以上、カナに対して９５％程度以上の正読率を達成すること。

②認識速度は、ファクシミリから帳票を入力する際の実行的な文字入力速度を上回る１０文字／秒以上であること。

要求条件の①は、解像度が低い上に白黒判定のしきい値を調整できないために、専用スキャナに比べ画品質がかなり劣るファクシミリ入力の文字認識装置を実用に供するための条件であり、極めて厳しく、数ある認識アルゴリズムの中でもこの要求にこたえられるものはごく少ない。そこで、これまでの専用ハードウェアに搭載した実績から唯一見通しの立つ輪郭特徴を組合せた位相構造化法[66]に、ファクシミリ対応の前処理を付け加えた認識アルゴリズムを用いることにした。ただし、このアルゴリズムは、専用ハードウェアによる逐次的な処理機構の使用を前提に開発されてきたものであり、ＬＩＳＣＡＲとの適合性は高いとはいえず、今度は要求条件の②をクリアすることが難しくなる。だからといって、アルゴリズムをＬＩＳＣＡＲ向きに修正することも許されない。これまでにかなりの期間をかけて作成してきた認識辞書に再度手を加えなければならなくなるからである。結局、アルゴリズムに内在する並列化可能な処理を極力ＬＩＳＣＡＲの並列処理部で実行することで、処理効率の向上をはかることが必要になる。

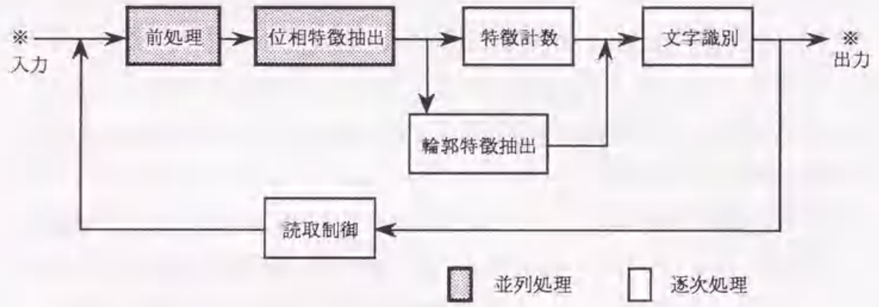


図８．１　位相構造化法による認識処理の構成

８．２．１　認識処理の構成

輪郭特徴を組合わせた位相構造化法の処理構成を、図８．１に示す。この図で前処理は、入力の文字パターンに対するノイズ除去、縮小等の処理である。位相特徴抽出処理は、文字パターン上の全画素について、黒点である場合には、その黒点が輪郭点でないか、輪郭点であればその傾きが縦、横、斜のいずれであるか、白点である場合には、上下左右のいずれの方向に黒点が存在するかを求める処理である。特徴計数処理は、得られた位相特徴を分類計数する処理である。輪郭特徴抽出処理は、文字ストロークの輪郭

部の屈曲度と屈曲方向の分布を求める処理である。文字識別部は、得られた特徴を辞書と照合し、認識結果を得る処理である。

８．２．２　並列処理と逐次処理の切り分け

前処理、位相特徴抽出処理は、文字パターンをビットプレーンとして直接処理することから、ＬＩＳＣＡＲのライン単位の並列処理で効率良く処理することが可能と考えられる。

一方、特徴計数、輪郭特徴抽出、文字識別については、並列処理の適用は容易ではない。特に文字識別については、場合分けを繰り返し、辞書を順次たぐることで正解を探す全くの逐次的な照合アルゴリズムを用いている[61]ため、ＳＩＭＤ型の並列処理による高速化は期待できない。これに対し、特徴計数、輪郭特徴抽出の一部については、並列処理適用の可能性はあっても、規則性が低い、処理が複雑化する等の理由により、高速化の効果はそれほど期待できないことが明らかとなった。

そこで、特徴計数処理以降については、ＬＩＳＣＡＲのアドレス生成部と制御スカラ演算部を利用した逐次処理で実行することとした。

８．２．３　並列処理の内容

（１）前処理

前処理で行う主要な処理は、ノイズ除去と縮小である。いずれも文字パターンをビットライン（要素の語長が１ビットの１次元配列）の集りとして配列データメモリＡＭに格納し、ビットライン単位の並列演算の繰り返しにより実現可能なＬＩＳＣＡＲ向きの処理である。実際、ノイズ除去は３ｘ３のマスク演算であり、ビットラインの左右シフトとビットライン間の論理演算の組合わせで実行できる。縮小についても、その要求が文字パターンの高さ方向のみであることから、文字パターンの構成ビットラインを適当な間隔で間引くことによって対応できる。ただし、画素１個分の厚みしかない細線が消失しないように、間引き対象の前後のビットラインを論理演算により加工する。

（２）位相特徴抽出

位相特徴抽出は、１次位相特徴抽出、統合、高次化の順で行う。１次位相特徴を数字の５を例に説明する（図８．２）。この図で、Ｑは白点、Ｐは黒点を示している。白点、黒点はそれぞれ次のような特徴を持つ。白点の特徴は、例えば ┣が、その白点の上下と左側に黒点が存在することを、 ┗がその白点の下側と左側に黒点が存在することを表す。また、黒点の特徴は、Ｉがその黒点が文字線の内点であることを、Ｖ、Ｈ、

Sがその黒点が文字線の輪郭点でかつ傾斜が、それぞれ垂直、水平、斜めの方向であることを、表す。

この１次位相特徴の抽出はＬＩＳＣＡＲでは次のように行う。すなわち、白点の特徴は、左右方向については黒点の有無を各黒点を始点とする伝搬演算により、上下方向の黒点の有無をビットライン間の論理演算により実現される上下方向の伝搬演算により、

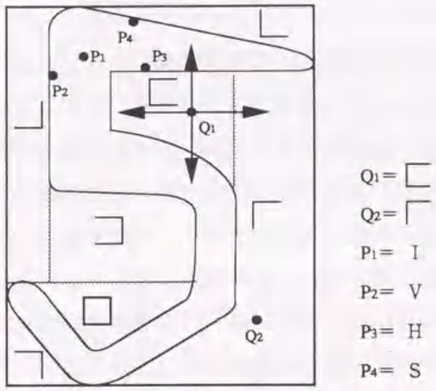


図８．２　１次位相特徴

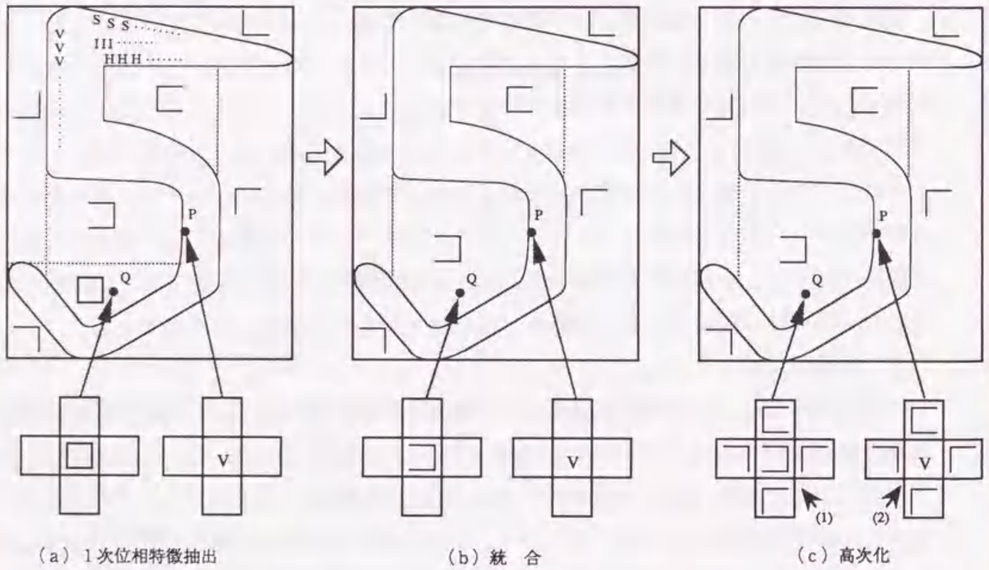


図８．３　位相構造化法による特徴抽出

各白点に通知することによって、黒点の特徴は３ｘ３のマスク演算を繰り返すことによって、それぞれ抽出する。

統合処理は、図８．３（ｂ）に示すように、隣接する白点の特徴を統合する処理であり、ＬＩＳＣＡＲでは上下左右方向の伝搬演算を繰り返して行う。

高次化処理は、各画素の上下左右方向の１次位相特徴を集積する処理である。図８．３（ｃ）に示すように、黒点に関しては、その点から上下左右方向を眺め、最初に見出される各白点特徴をその黒点特徴に付加して高次化特徴とする。また、白点については、その点から上下左右方向を眺め、文字線を越えて最初に見出された白点特徴を、その白点特徴に付加して高次化特徴とする。この高次化処理についても、ＬＩＳＣＡＲでは上下左右方向の伝搬演算を繰り返して行う。

８．３　印刷漢字読み取り

既存の印刷文書を専用のイメージスキャナを介して短時間で読み取って、文書データベースを効率良く作成することを目標に、日本語文書に現れる文字（以後印刷漢字と呼ぶ）に対し、認識精度９９％以上、認識速度２０～３０字／秒以上を目指した。印刷漢字では、漢字のカテゴリ数が３，０００～４，０００にもなるため、認識率の点で望ましい特徴ベクトルの全数照合を辞書との間で行おうとすると処理量が膨大になってしまう。従って、この膨大な照合処理の高速化と認識精度の向上の両立を、ＬＩＳＣＡＲの並列処理活用により、いかに実現するかがポイントになる。

８．３．１　認識処理の構成

一般に漢字認識のアルゴリズムは、演算量の大半を占める照合処理で、認識対象の文字毎に抽出する特徴ベクトルと辞書内の標準ベクトルとの間の識別計算（距離計算、類似度計算等）を繰り返し行うことから、並列処理向きと言える。それでも、カテゴリ数が大きく、複雑な識別関数を選ぶと目標の認識速度は達成できなくなる。さらに、全体の高速化には、前処理、特徴抽出にも並列処理がうまく適用できることが必要になる。以上の高速化の条件から、特徴としては、ＬＩＳＣＡＲのビットライン単位の並列処理により比較的容易に抽出処理可能な傾斜別とペリフェラル[66]を、識別関数としては演算量の少ないシティブロック距離を選択した。この印刷漢字の認識処理系の構成を図８．４に示す。なお、前処理は、１２本／ｍｍのページ型の専用イメージスキャナからのページ入力に対応できるものと、１６本／ｍｍのハンドスキャナからの帯状入力に対応できるものを用意する。

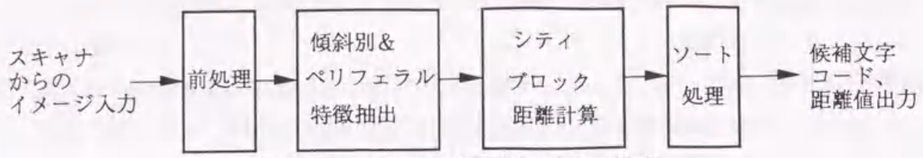


図８．４　印刷漢字認識処理系の構成

８．３．２　並列処理の概要

（１）前処理

ページ入力の場合は、文書イメージデータの縦方向、および横方向の周辺分布を求め、周辺分布の周期性から文字列を抽出する。次に、文字列と垂直方向の周辺分布を求め、黒の部分と白の部分の位置関係をもとに文字を検出して切出す[67]。ここでは、長さが２５６のビットライン単位の並列処理によって高速化を図っている。横方向の周辺分布はＬＩＳＣＡＲの９０度回転機能を用い、ページ全体を９０度回転したイメージデータに対して行っている。

ハンドスキャナの帯状入力の場合は、帯状のイメージ領域を複数の区間に分割し、その区間毎に、文字列方向に黒画素の周辺分布をとる。周辺分布で一定値以上の値を持つ領域を切出し領域の候補として抽出する。次に、その先頭の区間の周辺分布でセンサ視野中心部に最も近い部分の切出し領域候補を文字列の切出し領域の先頭と見なし、その

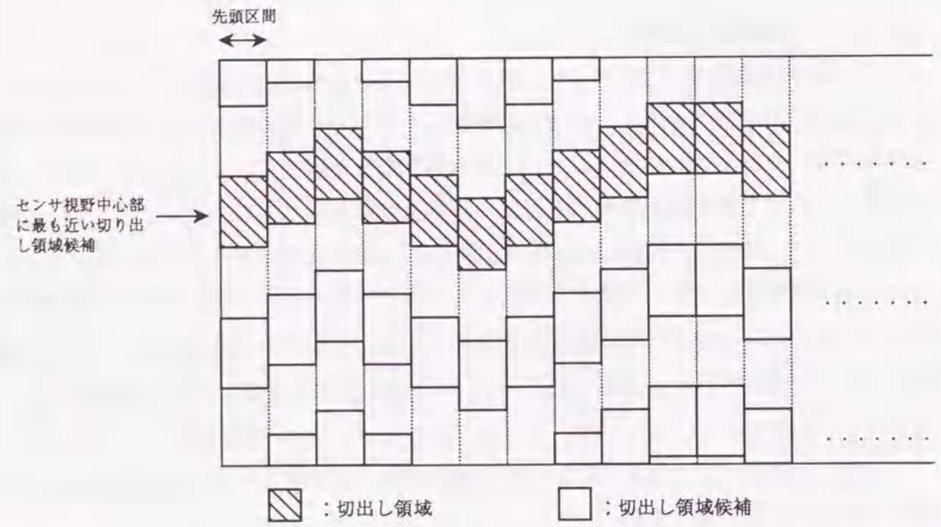


図８．５　ハンドスキャナ入力の文字列切出し

切出し領域が区間毎に順に移動して行くのを、隣接する区間の間では切出し領域が必ず重なりを持つとして、切出し領域候補を追跡する。ＬＩＳＣＡＲではこの追跡処理を、伝搬演算を利用して効率良く実行できる。

切出した文字は、３ｘ３のマスク演算によりノイズ除去（平滑化）し、６４ｘ６４のサイズに正規化する。演算方法は、手書き英数カナ文字の場合と同様に長さが２５６のビットライン単位の並列処理で実行する。

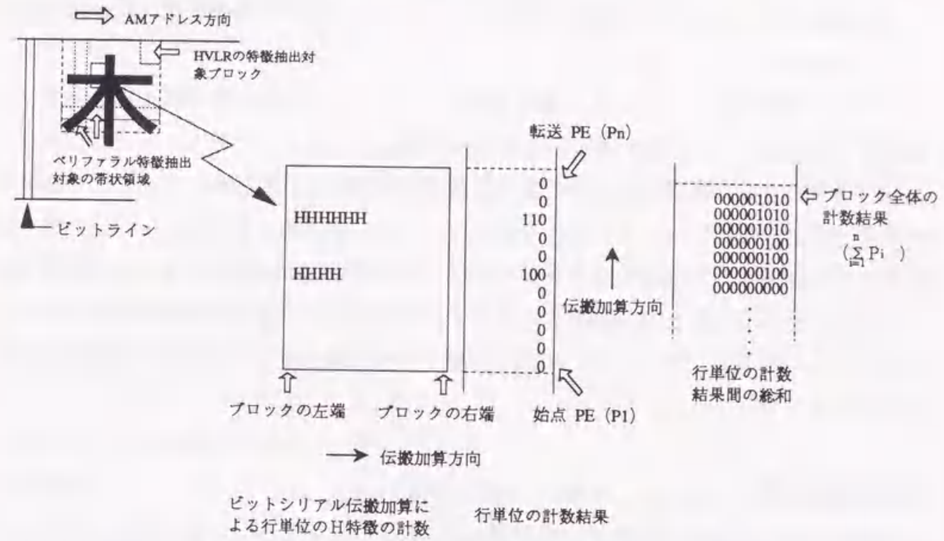


図８．６　特徴抽出とその計数処理のイメージ

（２）特徴抽出

傾斜別特徴は、文字の輪郭部の黒点を連結方向別に４種類（水平：Ｈ、垂直：Ｖ、左上がり：Ｌ、右上がり：Ｒ）に分類し、６４ｘ６４の文字パターンを４ｘ４に等分割したそれぞれの方形ブロック毎で集計したものである。ＬＩＳＣＡＲによるこの傾斜別特徴抽出の方法を、図８．６に文字「木」に対するＨ特徴抽出を例に示す。この特徴抽出の手順は、図３．６の総和演算法に基づいたもので、以下の通りである。

①ＡＭ内に格納している文字パターンに対し、ビットライン単位の演算により、輪郭を抽出する。

②輪郭に対し、３ｘ３のマスク演算によりＨ特徴の黒点を抽出しＡＭ内に格納する（図中央左より参照）。

③Ｈ特徴をブロックごとに行単位でビット直列伝搬加算により計数する。その結果は

図中央右の枠内に示すように得られる。

④その計数結果を、ビット並列データに変換した後、ブロック単位で並列に総和をとる。（図右側参照）。

この他のＶ，Ｌ，Ｒの特徴についても同様に求めている。

一方、ペリフェラル特徴は、縦横各々から外接枠内を８本の帯領域に等分割し、その各帯領域毎に外接枠と１本目の文字線間あるいは２本目の文字線間に存在する白点数を計数した値である。ＬＩＳＣＡＲによるこの特徴の抽出は、

①各帯状領域ごとの計数対象白点領域をビットライン単位の論理演算と伝搬演算により求める、

②その計数領域をビットライン単位で読みだし、ＰＥ配列内のグレイスケールラインに加え込む、

③その結果を伝搬加算により帯状領域毎に集計する

の手順で行っている。

以上の特徴抽出の結果得られる特徴の数は、傾斜別が各傾斜毎に１６で４方向分を合せて６４、ペリフェラルが方向毎に１本目と２本目の分の１６で４方向分を合せて６４、さらに単純にブロックごとの黒点の数を計数するメッシュ特徴１６が加わり、全体では１４４となる。

（３）距離計算

距離計算はＪＩＳ第１水準の漢字を含む3,３２８カテゴリ分の標準特徴ベクトルからなる辞書に対して行う。具体的には、特徴抽出によって得られた特徴ベクトルを放送により各ＰＥの局所メモリである配列データメモリ（ＡＭ）に格納し、あらかじめ各ＰＥの局所メモリに分散配置しておく１３カテゴリ分の標準の特徴ベクトルとの間で、各ＰＥごとに並列に距離計算を行い距離値の２次元配列を得る。

（４）ソート処理

各カテゴリを距離の小さい順に並べ直すソート処理は次に示す近似的な方法を用いる。

①各ＰＥの局所メモリに得られている距離値の最小値とその次に小さい距離値を選び出す。

②各ＰＥの最小値の配列（グレイスケールライン）の全体の中から、最小値検索により小さい順に所定の数だけを選び出す。

③各ＰＥの最小値の次に小さい距離値の配列の全体の中から、小さい順に所定の数だけ選び出し、その中に②で選び出した配列より小さい値があれば、それを②の配列に組入れ、それを最終的な候補距離値配列とする。ただし、②の配列への組入れがあった場合には、そのことがわかるようにフラグをたてる。

このソート処理は高速ではあるものの、処理手順から明らかなように、③で入れ替えが起る場合には、１位以外は、不正確である可能性が残る。しかし、類似した文字の標準の特徴ベクトルがＰＥ間で分散配置されるように、辞書をＡＭに適切に割付けることによって、ソート処理誤りの低減が可能であり、これによる認識率低下は無視できる程度に小さくなるものと考えられる。

８．４　手書き漢字読み取り

英数カナに加え、漢字をも扱える帳票読み取り装置実現を目指し、手書き漢字認識機能のＬＩＳＣＡＲへの搭載を試みた。手書き漢字は、印刷漢字に比べると字形の変形の程度が激しく、実用的な認識精度を得るには高度の認識アルゴリズムの搭載が要求される。このため処理の規則性が低下し、ＳＩＭＤタイプの並列処理による高速処理が困難になる。従って、この問題をいかに克服して実用的な認識速度と精度を確保するかが、漢字認識機能実現上のポイントとなる。

８．４．１　認識処理の構成

認識アルゴリズムとしては、比較的規則性の高い処理のみで、高い認識率の得られる外郭方向寄与度特徴[68]を次元圧縮した後、シティブロック距離計算あるいはユークリッド距離計算で識別する方法を採用する。処理系の構成を図８．７に示す。

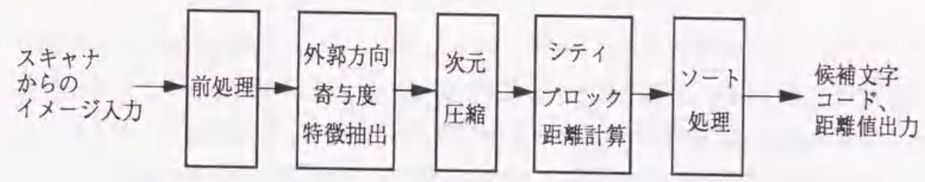


図８．７　処理系の構成

８．４．２　並列処理の概要

前処理とソート処理の内容は、基本的に前節の印刷日本語文書認識処理と同一である。従って、ここでは、特徴抽出、次元圧縮、距離計算について説明する。

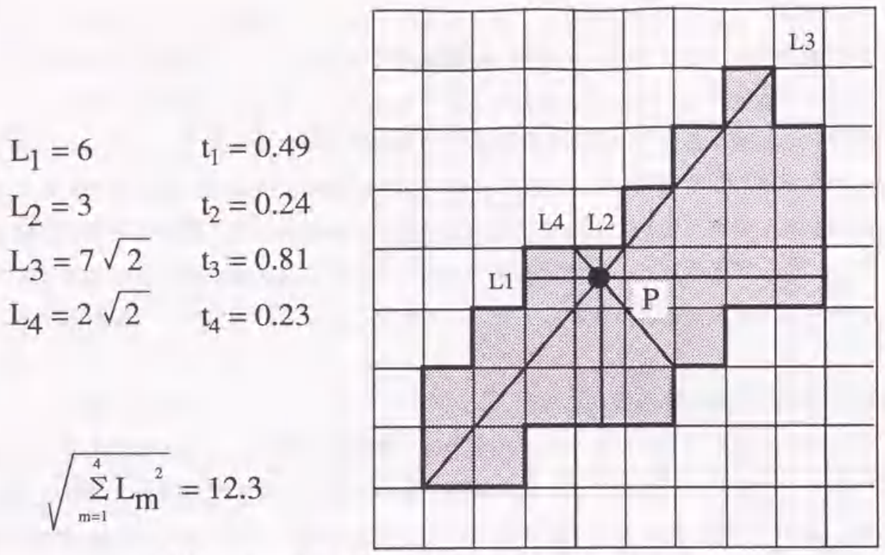


図８．８　文字パターンと４方向寄与度との関係

（１）外郭方向寄与度特徴抽出

処理内容の説明に入る前に外郭方向寄与度特徴について説明する。文字線の外郭点の方向寄与度ｔ［＝（t_1, t_2, t_3, t_4）］は、次式で定義される。

$$t_m = L_m / \sqrt{\sum_{m=1}^{4} L_m^2} \qquad (8.1)$$

ここで、L_1, L_2, L_3, L_4は、図８．８に示されるように対象とする外郭点の所属する４方向の黒点連結の長さである。ただし、斜め方向については実際の黒点連結の長さに$\sqrt{2}$を乗じた値としている。外郭方向寄与度特徴とは、周囲の８方向から見て、白から黒に変る最初の外郭点（１次外郭点）、２番目の外郭点（２次外郭点）、３番目の外郭点（３次外郭点）の４方向寄与度を、それぞれ１次から３次の投影軸に取り出し、この投影軸を８区間に分割し、その区間毎の方向寄与度の総和をとったものである。従って、この特徴抽出によって得られるベクトル（原特徴ベクトル）の要素の数は、８投影方向、４方向寄与度、３次外郭、８区間がかかって、７６８となる。

この外郭方向寄与度特徴のＬＩＳＣＡＲ１による抽出処理は次のように行う。はじめに次の手順で黒点連結の長さの投影軸グレイスケールラインを求める。

①文字パターンとそれを９０度回転したパターンに対し、ビットライン単位で上下方向と斜め方向に走査し、４方向の黒点連結の長さのテーブルをＡＭ内に作成する。

②それぞれの黒点連結の長さのテーブルを９０度回転し、回転無しの黒点連結の長さのテーブルの組と回転した黒点連結の長さのテーブルの組をＡＭ内に作成する。

③文字パターンとそれを９０度回転したパターンに対し、上から下方向および斜め左上から右下方向にビットライン単位で走査し、１～３次の外郭点の検出処理を実行する。この処理により外郭点が見つかった場合に、その点の４方向の黒点連結の長さをＡＭ内から読み出し、対応する次数の投影軸グレイスケールラインにコピーする。ここで、斜め方向走査は、ｉ番目の文字パターンのビットラインの外郭点の検出処理で用いるｉ－１番目の文字パターンと投影軸のグレイスケールラインを左右方向に１ビットづつシフトすることで等価的に実現する。

次に外郭別に黒点連結の長さの投影軸グレイスケールライン間の算術演算を行うことで、方向寄与度のグレイスケールラインを算出する。続いて、このグレイスケールラインを８等分し、その区間毎で伝搬加算により方向寄与度を累積し、最終的な外郭方向寄与度特徴を求める。

（２）次元圧縮

係数行列と特徴ベクトルとの乗算となる次元圧縮処理は、以下の①、②の処理を特徴ベクトルを構成する全ての要素に対して繰り返すことによって行う。

①特徴抽出によって得られた特徴ベクトルの要素を全ＰＥに放送する。

②放送した要素と、それに対応する係数行列のグレイスケールラインとの間で乗算を実行し、その結果を累算用のグレイスケールラインに加え込む。

ただし、係数行列は、あらかじめグレイスケールラインの本数が特徴ベクトルの要素数に等しくなる方向で格納しておく。

８．５　速度性能の改善

高速化前の１４３ｎｓ版ＬＩＳＣＡＲ１に搭載した各認識処理の精度と速度性能を表８．１に示す。認識精度については、高水準の認識アルゴリズムを搭載しただけあって、従来の専用ハードウェア構成のＯＣＲと同等かそれ以上の性能が得られている。しかし、認識速度については、専用ハードウェア構成の２０～３０字／秒以上の性能に比べ見劣りするばかりか、当初目標も満たせていない。そこで、本節では処理方法の見直しによる速度性能の改善を試みる。

表８．１　各文字認識処理の性能

		認識精度		認識速度（字／秒）
		正読率	エラー率	
手書き英数カナ	数字	９９.６６	０.０６	７.７
	英字	９８.２５	０.１４	－
	カナ	９６.２３	０.４６	－
印刷漢字*1		９８.０５	０.１６	２５.５
手書き漢字*2		９８.０	－	４.６

＊１ カテゴリ数：３,３２８　　＊２ カテゴリ数：４,０９６

８．５．１　高速化手法

実際に試みた３つの高速化法について述べる。

（１）複数文字の同時並列処理による高速化

ＬＩＳＣＡＲの構成プロセッサ数は２５６で、ビットラインの長さは２５６である。これに対し、認識対象の文字パターンのサイズは６４ｘ６４以下であり、文字パターンをビットライン単位の順次走査により直接処理する前処理や特徴抽出処理では、ＰＥ配列の３／４が無駄になってしまう。この無駄をなくすには、同時に複数の文字を並列に処理するのが有効である。並列処理可能な文字数としては、文字パターンを無駄なく割り付けるとすると、最大４まで可能である。ただし、手書き認識処理では外郭方向寄与度特徴抽出においては、もともと文字パターンのサイズのエリアの２倍の処理幅の１２８を用いて処理しているので、２文字並列に制限される。

（２）語長低減による距離計算の高速化

ＬＩＳＣＡＲ１では、シティブロック距離計算を各ＰＥのビット直列演算で行う。このため、演算時間はほぼ特徴ベクトルの要素の語長に比例することになる。従って、要素の語長を低減できれば、その分演算時間を短縮できる。

実際、傾斜別特徴では、１６ｘ１６の方形ブロックを横切る線の数がせいぜい２本程度であり、現れる特徴値は最大６４程度であると考えられることから、８ビットの語長を確保するのはもともと精度過剰といえる。逆に、与えられた語長分を完全に利用するようにすれば、認識率の低下なく、要素の語長が短縮できるものと考えられる。このような考えに基づき、次式の定比例的なスケール変換による４ビットまでの語長短縮を試み、印刷漢字認識処理の読み取り精度をシミュレーションにより評価した。

$$Y_i = a + \{[X_i - MIN(X_i)] / [MAX(X_i) - MIN(X_i)]\} \times (b-a) \qquad (8.2)$$

ここで、Yiは変換後のｉ番目の特徴値、X_iはｉ番目の変換前の特徴値である。ＭＩＮ（X_i），ＭＡＸ（X_i）は特徴値Xiの最小値、最大値をそのまま用いるのではなく、それぞれ１.０２、０.９８を乗じた値を用いる。

評価結果を、単純に４ビット足切りした場合と比較して、表８．２に示す。この結果より、定比例的なスケール変換によれば語長を４ビットまで短縮しても、認識精度がほとんど低下しないことがわかる。

表８．２　特徴値の語長短縮の認識率への影響

	語長	語長低減手法	正読率
５０６５号未学習データ	８	－	９９.７６
	４	定比例法	９９.７６
		足切り	９９.６３
７０１７ポイント未学習データ	８	－	９９.３０
	４	定比例法	９９.２７
		足切り	９９.０２

（３）逐次処理の低減

以上の（１）、（２）では、主に並列処理部の高速化法である。しかし、もともとＤＳＰ部やアドレス生成部での逐次的な処理の割合の大きい手書き英数かな認識処理においては、並列処理部が高速化すると、今度は逐次処理の部分が目立ってくる。そこで、手書き英数カナ文字の認識アルゴリズムを見直し、逐次処理部分の処理量を低減した。具体的には、位相特徴のみで識別可能かどうかを判断し、可能な場合には輪郭特徴抽出処理をバイパスするようにして高速化をはかった。

８．５．２　高速化の結果と考察

それぞれの認識処理の高速化の結果を処理時間の内訳と合せて表８．３～８．５に示す。８．５．１節の高速化法の適用と一部処理方法の見直しだけで、当初の１.８倍～２.１倍の高速化に成功している。手書き数字認識と手書き漢字認識の結果は、並行して行った高速化のためのマイナチェンジの結果である８０ｎｓ版のＬＩＳＣＡＲ１で測定したものである。これらについては、１４３ｎｓ版からハードウェアが実行的に

1．6倍程度高速化されたことも加わり、表8．1に示した認識速度に比べ、全体で3倍程度高速化され、専用ハードウェアに匹敵する性能が達成されている。印刷漢字認識については、143nS版での性能を示しているが、80nS版では同様に1．6倍程度高速化され、70文字程度の認識速度が見込まれる。

表8．3　手書き数字認識の高速化前後の処理時間の内訳と認識速度

処理内容		処理時間（ms）		高速化法
		高速化前	高速化後	
前処理	文字パターン入力	1.2	同左	4文字並列
	ノイズ除去	2.9	0.74	
位相特徴抽出	位相特徴抽出	32.2	7.6	4文字並列
	シリパラ変換	5.0	5.0	
輪郭特徴抽出		12.8	1.3	逐次処理バイパス
位相特徴係数		16.0	同左	
文字識別		10.8	同左	
その他		1.7	同左	
全処理時間		82.6	44.3	
認識速度（文字／秒）		12.1	22.6	

表8．4　印刷漢字認識の高速化前後の処理時間の内訳と認識速度

処理内容	処理時間（ms）		高速化法
	高速化前	高速化後	
前処理	8.9	6.7	4文字並列
位相特徴抽出	4.4	1.1	4文字並列
識別　距離計算（ｼﾃｨﾌﾞﾛｯｸ）	24.4	12.5	定比例法による語長短縮
ソート処理	1.5	1.5	
全処理時間	39.2	21.8	
認識速度（文字／秒）	25.5	45.9	

表8．5　手書き漢字認識の高速化前後の処理時間の内訳と認識速度

処理内容	処理時間（ms）		高速化法
	高速化前	高速化後	
前処理	8.4	4.3	2文字並列
特徴抽出　黒点連結の長さ	87.2	29.6	2文字並列、処理法見直し
方向寄与度	8.5	8.5	
区間内累積	1.3	1.3	
次元圧縮処理	11.3	11.3	
識別　距離計算（ｼﾃｨﾌﾞﾛｯｸ）	18.8	18.8	
ソート処理	1.5	1.5	
全処理時間	137.0	75.3	
認識速度（文字／秒）	7.3	13.3	

しかし、高速化の内容を個別に見てみると、並列処理が必ずしも有効に機能していない部分があり、これが高速化のネックになっていることがわかる。その典型は、手書き英数カナ文字の処理時間内訳に見られ、並列化困難な、位相特徴係数と識別の処理時間が全体の60％も占めている。また、一見高速化がうまくいっていると思われる印刷漢字、手書き漢字認識処理においても、SIMD型の並列処理がうまく機能しないことから、十分な高速化が達成できていないところがある。例えば、手書き認識処理の特徴抽出では2文字並列化をはかった上に処理の最適化をはかっても、その処理時間は全体の52％も占めている。これは、外郭方向寄与度特徴の抽出処理では、大半の処理を文字線の輪郭部分に対してのみ行えばよいのに、LISCAR1の単純なSIMD処理ではこのような選択的な処理が困難で、文字パターン全体をくまなく処理しているためである。また、全体に占める処理時間自体は大きくはないが、文字パターンの正規化処理では文字毎の拡大縮小率が異なるため、文字単位の並列処理が利用できず、LISCAR1の256プロセッサの3／4が無駄になってしまう。印刷漢字認識、手書き漢字認識処理では、実はさらにもう一つ大きな問題がある。それは、認識精度の点でシティブロック距離より優れているといわれるユークリッド距離がLISCAR1では、シティブロック距離計算の4倍程度の処理時間を要するために、利用できないことである。速度が大きく低下する原因は、LISCAR1ではユークリッド距離計算に表れる乗算もビット直列に処理せねばならず、一回の乗算に語長の2乗のオーダーのサイクル数を要するためである。

8．6　むすび

本章ではＬＩＳＣＡＲ１の文字認識処理への応用について述べた。

はじめに、手書き英数カナ文字、印刷漢字、手書き漢字の３種の代表的な文字認識処理に対し、前処理、特徴抽出、識別のそれぞれをどのように実行するかを示した。具体的には、手書き英数カナ文字認識処理については、並列処理に不向きな特徴計数処理以降はＬＩＳＣＡＲの逐次処理機能を利用して実行することとし、残りの前処理と位相特徴抽出をビットライン間の論理演算と上下左右方向の伝搬演算によって実行する方法を示した。印刷漢字認識処理については、文字切出しを含む前処理から、傾斜別＆ペリフェラル特徴抽出、シティブロック距離計算、ソート処理までの主な文字認識処理の全てをビットライン単位あるいはグレイスケールライン単位の算術論理演算あるいは伝搬演算によって実行する方法を示した。また、手書き漢字認識処理については、印刷漢字認識処理とは異なる処理である外郭方向寄与度特徴抽出をＬＩＳＣＡＲ１の単純なＳＩＭＤ処理機能によって実行する方法と、抽出した特徴ベクトルの次元圧縮処理のための行列とベクトルの乗算の方法を示した。

次いで、これらの処理方法による実機での認識精度と認識速度の測定から、認識精度は９６～９９％と汎用コンピュータ上でアルゴリズムを評価した通りの値が得られることを示した。しかし、認識速度は５～２０文字／秒程度と、当初目標を達成できていないことから、速度性能の改善を試みた。具体的には、高速化法として複数文字の同時並列処理、距離計算における被演算データの語長低減、一部逐次処理の条件に応じたスキップの３つの方法を各認識処理に可能な限り組込むことを試みるとともに、ＬＩＳＣＡＲ１をクロックサイクル１４３ｎｓの版から８０ｎｓの高速版に置き換えた。その結果、当初に比べ3倍程度の高速化が達成され、専用ハードウェアに匹敵する１３文字／秒～４６文字／秒の認識速度が得られることを確認した。

最後に、高速化後の処理時間の内訳を分析し、

①手書き英数カナ認識処理では、並列処理部分の高速化に伴い、逐次処理部分が全処理時間の６０％を占め高速化の障害になっていること、

②逐次処理の占める割合を数パーセント以下にまで低減可能な手書き漢字認識処理においても、外郭方向寄与度の特徴抽出処理が、ＬＩＳＣＡＲ１のＳＩＭＤ処理の制約により高速処理困難なため、全体の５２パーセントの処理時間を占めていること、

③認識精度向上の観点から、要求の強いユークリッド距離計算が、ＬＩＳＣＡＲ１の乗算性能の低さから利用できないこと

等を明らかにした。

第９章　ＬＩＳＣＡＲ２の漢字認識処理への応用

９．１　はじめに

前章では、ＬＩＳＣＡＲ１により最高水準の認識精度の文字認識処理アルゴリズムが専用ハードウェア並みの速度で実行できることを示した。しかし、最高水準の認識精度と言え、人間に比べるとまだまだ低く、印刷漢字や手書き漢字では、文字品質によっては確認修正の手間が大きくなりすぎ使いものにならない場合さえある。このため認識精度向上の要求は強く、より高い正読率の得られる高度の認識アルゴリズムの検討が続けられている。

残念ながらこれらの高精度の認識アルゴリズムは、ＬＩＳＣＡＲ１の得意とする傾斜別特徴やシティブロック距離をベースとした単純な処理のみでは構成できない。例えば、最近行われた文字認識アルゴリズムのコンテストで上位を占めたアルゴリズムは、特徴として輪郭特徴を用いていたり、識別関数としては積和演算を多用する高度なものが利用されている[69]。このような認識処理をＬＩＳＣＡＲ１のシンプルなＳＩＭＤ処理でカバーするのはかなりの無理があり、十分な速度性能が得られる見通しはたたない。

このような状況のもとに新たに改良試作したのがＬＩＳＣＡＲ２である。本章では、このＬＩＳＣＡＲ２上に、８方向の外郭方向寄与度特徴に投影距離[70]を組み合わせた最新の手書き漢字認識アルゴリズムを組込む際に、高速化に工夫を凝らした特徴抽出処理、次元圧縮処理、大分類処理、識別処理の処理方法を明らかにし、速度性能を評価する。

９．２　認識処理の構成

図９．１にＬＩＳＣＡＲ２に搭載する手書き漢字認識処理の手順を示す。図８．７のＬＩＳＣＡＲ１の手書き漢字認識処理との違いは、外郭方向寄与度特徴の方向寄与度を４方向から８方向に拡張している点、識別処理を大分類と識別に分け大分類でユークリッド距離による全数照合を、識別で大分類で上がった上位候補に対する投影距離による照合を行う点である。

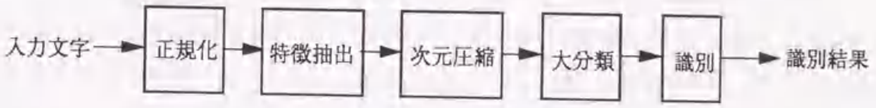


図９．１　手書き漢字認識処理の構成

9．3　並列処理の概要

9．3．1　特徴抽出処理

（1）高速化の方針

ＬＩＳＣＡＲ１では、外郭方向寄与度特徴の抽出時間が長く、これが手書き漢字認識全体の速度を低下させる主因になっている。２文字並列や処理の最適化を行っても特徴抽出処理の高速化が十分達成できないのは、外郭部の黒点連結の長さを投影軸のグレイスケールラインに投影する処理が、ＬＩＳＣＡＲ１の単純なＳＩＭＤ制御では効率よく実行できないからである。具体的には、次の３つの処理を文字パターンを構成する各ビットラインごとに毎回繰り返し実行しなければならないからである。

①１～３次の外郭部の検出、

②検出された外郭部黒点連結の長さの１～３次、４方向成分（計３ｘ４本）の投影軸グレイスケールライン（要素のデータ語長６ビット）へのコピー、

③斜め方向走査の場合の全投影軸グレイスケールラインの１ビット左右方向シフト。

これらの３つの処理の中で②と③の処理が特に重い。ビット直列処理ベースのＬＩＳＣＡＲ１では、総計７２ビットのコピーあるいはシフトを、ビット処理に分解して実行しなければならないからである。

これに対してＬＩＳＣＡＲ２ではコピーあるいはシフトの単位が８ビットあるいは１６ビットと大きく②と③の処理の負担はかなり軽減される。それでも、ＬＩＳＣＡＲ２では配列サイズが１／４の６４になっている上、前節で説明したように方向寄与度の方向を４から８に拡大すると、コピーあるいはシフトの対象の総計ビット数が拡大前の２倍の１４４ビットにもなるため、ＬＩＳＣＡＲ１と同一の処理アルゴリズムを用いるのでは大幅な処理速度の向上は望めない。　そこで、ＬＩＳＣＡＲ２では新たに組込んだ転送回転型メモリアクセス機構による間接アドレシング機能を駆使することによって、文字パターンを構成するビットライン毎の②と③の処理を不要として、特徴抽出処理の高速化をはかる。

（2）８方向寄与度への拡張

文字線の外郭点の方向寄与度　ｔ［＝（$t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8$）］を、次式で示すように８方向に拡張する。

$$t_m = L_m \Big/ \sqrt{\sum_{m=1}^{8} L_m^{\,2}} \qquad (9.1)$$

ここで、Ｌ1,Ｌ2,...,Ｌ8は、図９．２に示すように対象とする外郭点から８方向に向かう黒点連結の長さである。斜め方向については実際の黒点連結長に$\sqrt{2}$を乗じた値とす

$L_1 = 5$　　$t_1 = 0.47$
$L_2 = 2$　　$t_2 = 0.19$
$L_3 = 4\sqrt{2}$　　$t_3 = 0.53$
$L_4 = 4\sqrt{2}$　　$t_4 = 0.53$
$L_5 = 1$　　$t_5 = 0.09$
$L_6 = 3$　　$t_6 = 0.28$
$L_7 = 1\sqrt{2}$　　$t_7 = 0.13$
$L_8 = 2\sqrt{2}$　　$t_8 = 0.27$

$\sqrt{\sum_{m=1}^{8} L_m^{\,2}} = 10.6$

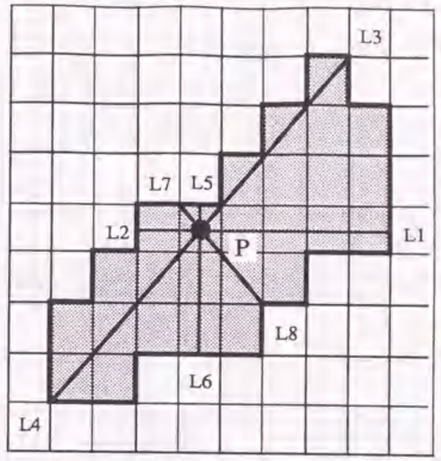


図９．２　文字パターンと８方向寄与度との関係

るのは、４方向寄与度の場合と同じである。方向寄与度が４方向から８方向に拡張される点を除くと、外郭方向寄与度特徴抽出の考え方は、ＬＩＳＣＡＲ１の場合と全く同一である。従って、この特徴抽出によって得られるベクトル（原特徴ベクトル）の要素の数は、８投影方向、８方向寄与度、３次外郭、８区間がかかって、１,５３６となる。

（3）特徴抽出処理の内容

高速化のポイントは、ＡＭ内に形成する黒点連結の長さのテーブルを、間接アドレシング機能を利用して、文字パターン走査より得られる外郭アドレス値で読み出すことにより、投影軸グレイスケールラインを繰り返し処理なしで効率良く求めることにある。以下、この間接アドレシングによる投影軸グレイスケールラインの算出を実現するための、黒点連結の長さのテーブルの作成法と１次から３次の外郭点アドレスグレイスケールラインの算出法を中心に説明する。

図９．３は、あらかじめ演算器配列の上側に示すレジスタファイルに格納しておく文字パターンを、上側から下に向かってビットライン単位に走査し、黒点連結の長さを計算し、その結果を配列データメモリ（ＡＭ）内に格納して順次黒点連結の長さのテーブルを作成して行く際のレジスタファイルとＡＭ内のデータの割付けを示している。この図から明らかなように、L_1～L_8の黒点連結の長さは連続して格納している。これは、間接アドレシングにより黒点連結の長さ一式をまとめてアクセスできるようにするためである。

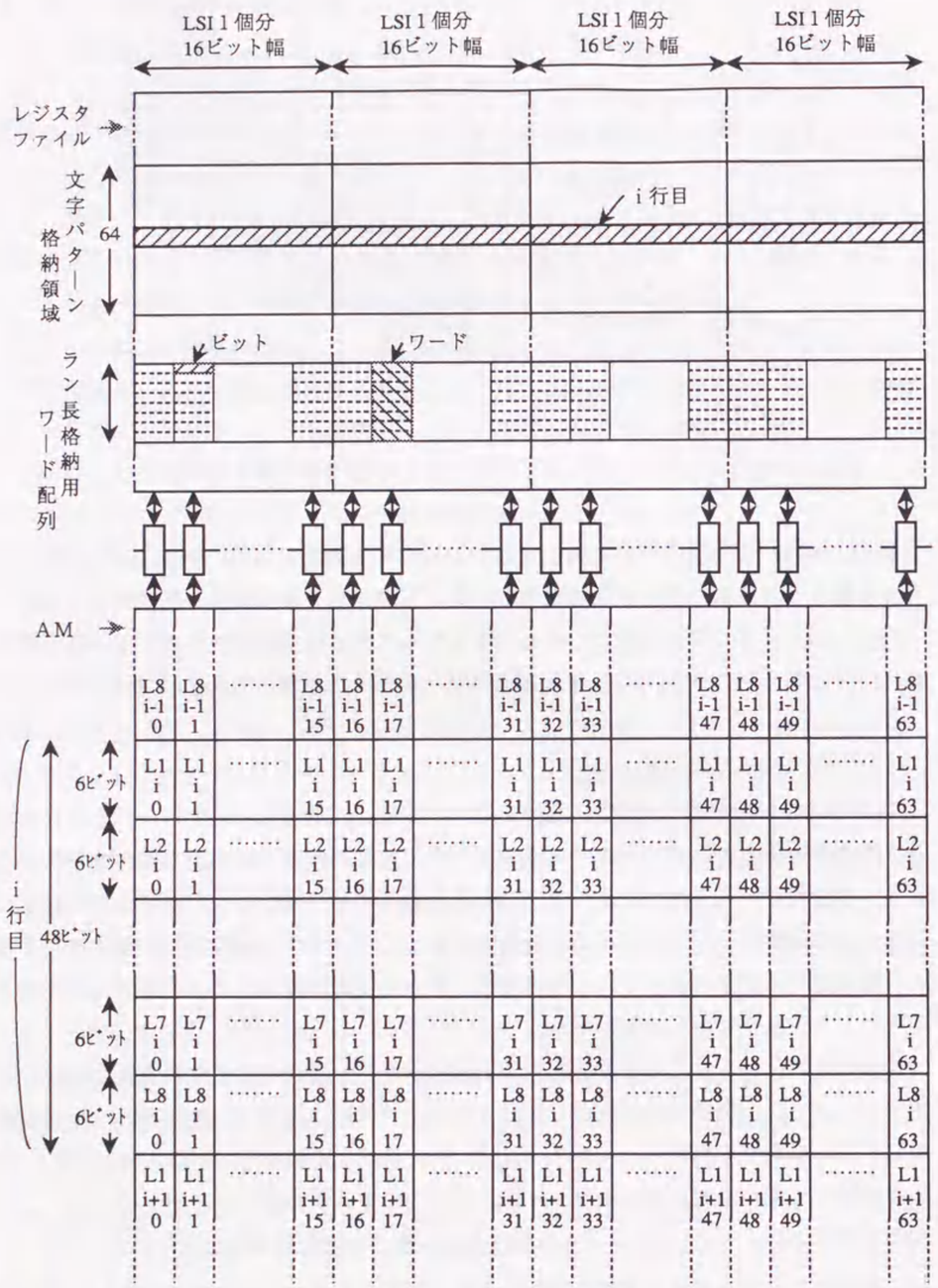


図９．３　文字パターン走査による黒点連結の長さのテーブル作成時のメモリ割付け

図９．４は、ＬＩＳＣＡＲ２の間接アドレシングによるデータアクセス機能を利用して外郭部の黒点連結の長さをアクセスできるように、黒点連結の長さのテーブルの格納形式をビット直列型からビットパラレル型に変換した様子を示している。この直並列変換は、ＬＩＳＣＡＲ２の転送回転型メモリアクセス機構をＬＳＩ単位で動作させることで実現する。

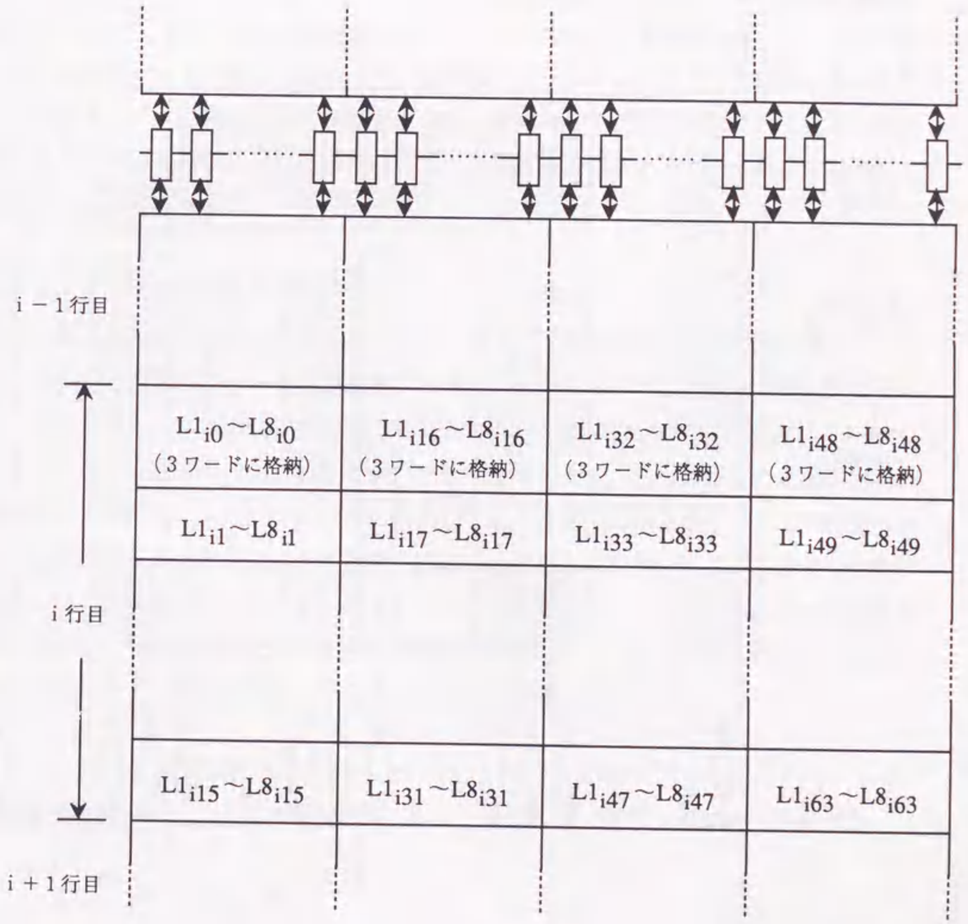


図９．４　直並列変換後の黒点連結の長さのテーブル

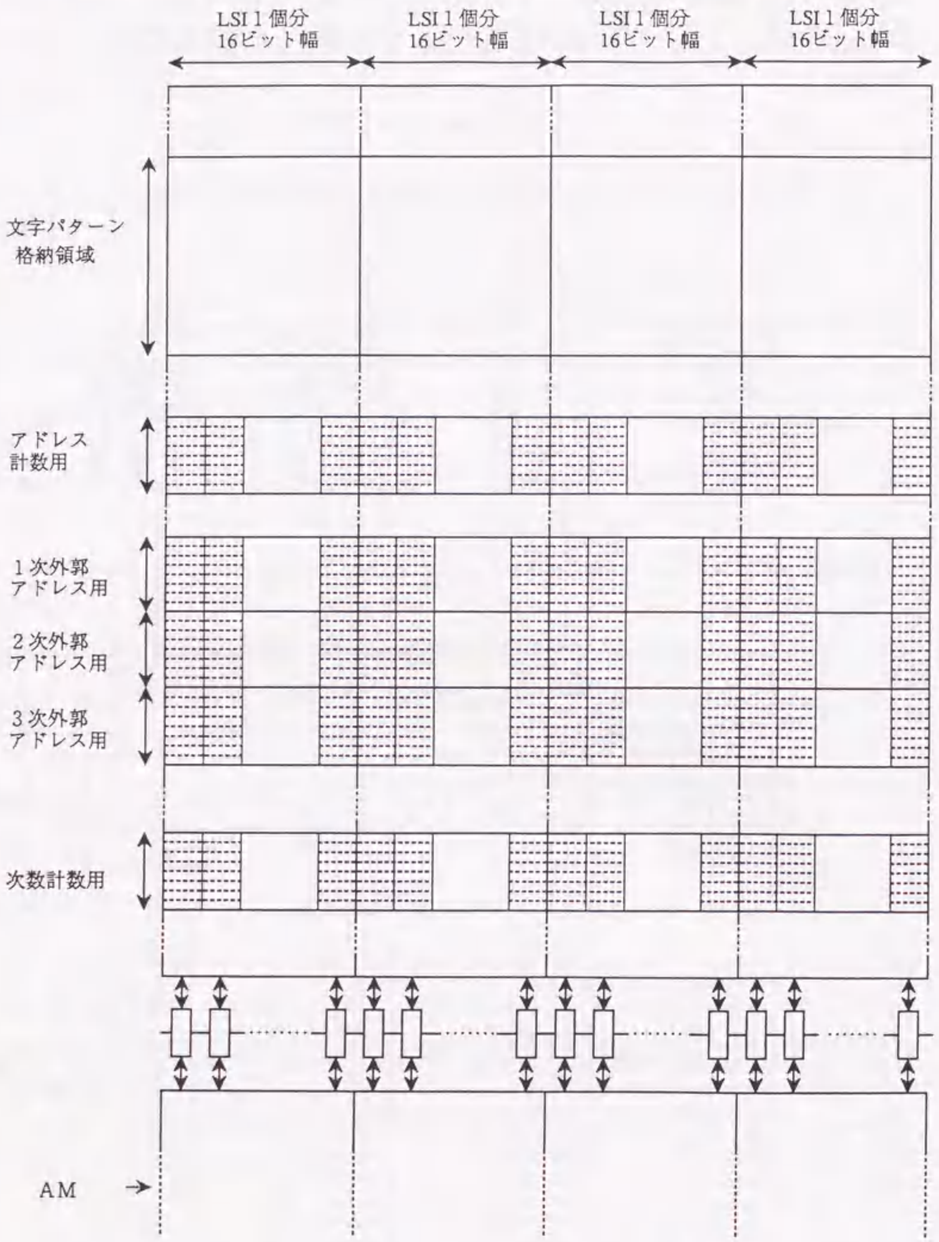


図９．５　外部アドレス抽出処理におけるメモリ割付け

図９．５は、１次から３次の外郭アドレスを求める際のメモリへのデータ割付けを示している。この処理は、文字パターンをビットライン単位で走査し、外郭部を検出したならば、並行して求めておく走査中のビットラインのアドレス値を、外郭アドレスのグレイスケールラインに、やはり並行して求めておく外郭次数値を参照して選択的にコピーすることによって実現する。これによって得られたアドレス値は、黒点連結の長さのテーブルのアドレス値そのものではない。黒点連結の長さが３ワード単位でまとまっていることから、３倍するとともにベースアドレスとして黒点連結の長さのテーブルの先頭アドレス値を加える必要がある。

この最終的なアドレス値のグレイスケールラインを用いて図９．３に示される黒点連結の長さのテーブルを、転送回転型メモリアクセス機構による間接アドレシングを利用して、直接アクセスすることによって、投影軸グレイスケールラインを一気に求めることができる。これ以降の方向寄与度算出処理、区間内の累積処理はＬＩＳＣＡＲ１と同様に実行する。もちろん、ＬＩＳＣＡＲ２ではこれらの処理を８ビット単位で処理する分だけ、高速化される。

９．３．２　次元圧縮処理

１５３６次元の原特徴ベクトルと1,536ｘ２５６の係数行列との積をとり、２５６次元の特徴ベクトルを求める処理である。ＬＩＳＣＡＲ１の場合と異なりシストリックアルゴリズム[7]を用いる。図９．６は、その処理手順の概要を、３次元の特徴ベクトルと３ｘ３の係数行列との積を例に示したものである。あらかじめ、係数行列を紙面の上下方向にねじった形でＡＭ内に格納しておき、要素の語長が８ビットの特徴ベクトルを転送回転型メモリアクセス機構の転送パスを介し左方向にシフトしながら係数行列との間で乗算を行い、結果をレジスタファイルで累算することにより、次元圧縮処理を行う。ＬＩＳＣＡＲ２では、ＡＭからの係数行列のアクセスと、乗算あるいは特徴ベクトルのシフトを並列に行えるので極めて高い処理効率が得られる。

９．３．３　大分類処理

辞書側の標準ベクトル$\overline{x}$と特徴ベクトルｘとの間のユークリッド距離の２乗Ｄを

$$D=\sum_{i=1}^{256}(xi-\overline{xi})^2 \qquad (9.1)$$

を求める処理であり、次元圧縮処理の場合同様に実行する。すなわち、辞書側の標準ベクトルをあらかじめ左右方向にねじった形で格納しておき、次元圧縮処理で得られた

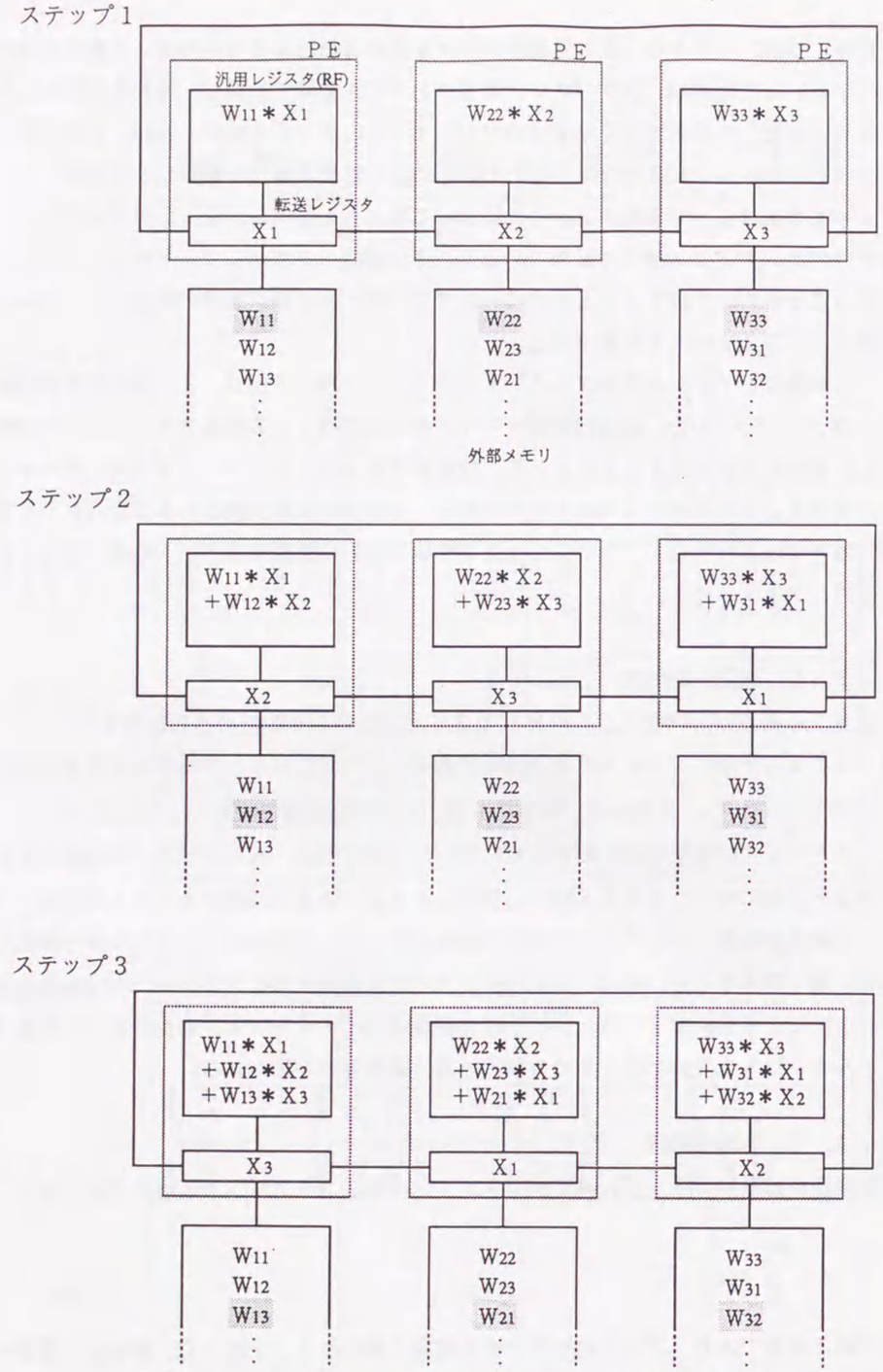


図９．６　シストリックアルゴリズム処理における次元圧縮処理

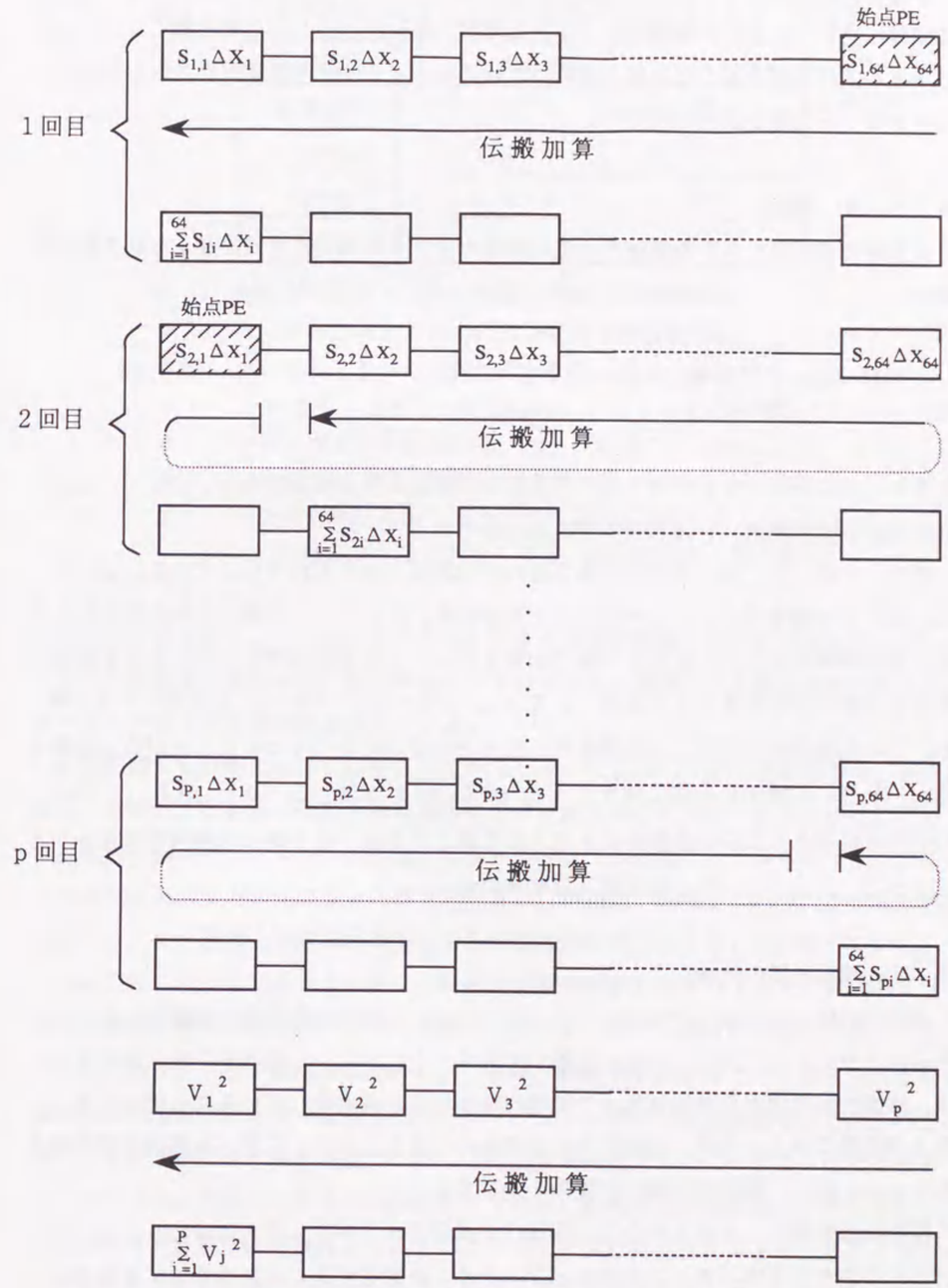


図９．７　投影距離の演算法

２５６次元の特徴ベクトルをＰＥ配列の右から左にシフトしながら、ＡＭの各ＰＥ対応の領域に格納されている標準ベクトルとの間で、差分をとったあとで２乗し、レジスタファイル内に累算することを繰り返す。これによって６４個の標準ベクトルとの間のユークリッド距離を並列に求める。

９．３．４　識別

大分類処理で得られた距離値の小さい候補カテゴリに対し、次式で与えられる投影距離Ｈ

$$H = D - \sum_{j=1}^{p} \left| \sum_{i=1}^{256} s_{jic} \left(x_i - \overline{x_i} \right) \right|^2 \qquad (9.2)$$

を求め、その距離の小さいカテゴリを最終的な候補とする処理である。ここで、s_{jic}は部分空間行列の要素、Ｐは部分空間の次元数である。

図９．７は（９．２）式の右辺第２項の積和演算の実行方法を示している。はじめに、各ＰＥで標準ベクトルとの差分△ｘを求める。次いで、その値とＡＭ内に格納しておく部分空間行列Ｓの１行目との間で乗算を行い、その乗算結果を右端のＰＥを始点とする左方向の伝搬演算により累積し、$\Sigma\ s_{1ic}(x_i - \overline{x_i})$［＝Ｖi］を左端のＰＥに得る。この伝搬演算を利用した乗累算を、始点を右方向に１つずつずらしながらｐ回繰り返し、ＰＥの左端から右側に向う$\Sigma\ s_{1ic}(x_i - \overline{x_i}) \sim \Sigma\ s_{pic}(x_i - \overline{x_i})$［＝Ｖp］の配列を得る。これらの累算結果をさらに２乗してから、左方向の伝搬演算を行うことによって、（９．２）式の右辺第２項を求める。

９．４　速度性能の評価

前節の処理方法に基づいたプログラムを作成し、各処理の所要時間を測定した。その測定結果をＬＩＳＣＡＲ１の値と比較して表９．１に示す。この表から明らかなように、特徴抽出処理から識別処理まで処理内容はかなり高度化しているにも関わらず、個々の処理では３～９倍、全体では５倍程度のＬＩＳＣＡＲ１に比べた高速化が達成されている。以下、個別に高速化の要因を分析する。

特徴抽出処理は、９倍とかなりの高速化が達成されている。これには、９．３．１（１）で述べた文字パターンの構成ビットラインを走査するごとに繰り返す必要のあった②と③の処理が、転送回転型メモリアクセス機構を用いた間接アドレシングにより黒点連結の長さのテーブルをアクセスすることによってほとんど一回分の処理にまで短縮

表９．１　漢字認識に必要な主要な処理の所要時間（ｍｓ）の比較

処理内容	ＬＩＳＣＡＲ１	ＬＩＳＣＡＲ２
特徴抽出	３９.４*1	４.５*2
次元圧縮	１１.３*3	２.２*4
大分類	１８.８*5	５.９*6
識別	－	０.７*7

＊１　方向寄与度が４方向の外郭方向寄与度特徴抽出
＊２　方向寄与度が８方向の外郭方向寄与度特徴抽出
＊３　７６８ｘ２５６の係数行例と７６８次元の特徴ベクトルとの積
＊４　1,５３６ｘ２５６の係数行例と1,５３６次元の特徴ベクトルとの積
＊５　シティブロック距離計算、カテゴリ数：４,０９６
＊６　ユークリッド距離計算、カテゴリ数：４,０９６
＊７　大分類で識別困難な入力文字の２割の文字に対してのみ行う。
　　詳細識別対象文字数：６４、部分空間次元数：８

できたのが最も寄与している。もちろん８ビット単位の演算、ＡＭに対するロード、ストアと内部演算の並列実行、転送回転型メモリアクセス機構による黒点連結の長さのテーブルの直並列変換等も寄与している。

次元圧縮、大分類については演算単位を８ビットに拡張し、かつ乗算器を各ＰＥに内蔵し、乗算とＡＭからの乗数、被乗数のロードを並行実行可能としたことが効いている。転送回転型メモリアクセス機構の隣接ＰＥ間シフト機能によりシストリックアルゴリズムが利用できるようになった効果も大きい。識別は、乗算とロードの並行実行機能に加え、ツリー構成の伝搬演算機構により伝搬加算が高速化されたのが効いている。

ところで、ＬＩＳＣＡＲ２で新たに組込んだ転送回転型メモリアクセス機構とツリー構成伝搬演算機構の高速化の効果を比較してみると、ツリー構成の伝搬演算機構の効果が以外に小さいことが明らかになった。これは、転送回転型メモリアクセス機構が特徴抽出処理の高速化に大きく貢献したのに対し、ツリー構成伝搬演算機構は方向寄与度特徴の区間内累積処理と識別処理の一部に伝搬加算が利用できたにとどまったためである。このように利用できる処理が少なかったのは、総和あるいは累算について１ビット単位の実行が必要な伝搬加算より、８ビット単位で実行可能なシストリックアルゴリズムが有利と判断し、極力シストリックアルゴリズムを利用するようにしたからである。しかし、前節で示した並列処理法の検討の過程で、シストリックアルゴリズムのＬＩＳＣＡＲ２への組み込みは、伝搬加算に比べるとかなり難しいことも判明した。こ

れは、シストリックアルゴリズムでは、係数行列をねじった形式のスキュード配列形式でＡＭ内に格納する必要があるだけでなく、配列の任意の区間での総和が困難なためである。このシストリックアルゴリズムの問題を解決するには、バイパス型の伝搬演算機構を用いて伝搬加算の単位を８ビットあるいはそれ以上に拡張するのが、最も効果的であると考えられる。

９．５　むすび

本章では、前章の手書き認識処理に対し、

①方向寄与度を４方向から８方向に拡張する、

②識別を大分類、詳細識別の２段階で実行するように変更する、

③大分類の距離計算をシティブロック距離からユークリッド距離に変更する、

④詳細識別に投影距離を採用する

等の改良を加えた新しい手書き認識処理を、ＬＩＳＣＡＲ２で効率良く実行する処理法を明らかにするとともに、その処理速度を実機を用いて評価した。

はじめに、ＬＩＳＣＡＲ１での高速化のネックが、外郭方向寄与度特徴抽出において、外郭部の黒点連結の長さを投影軸のグレイスケールラインに投影する処理が効率良く実行できないことにあることを指摘し、ＬＩＳＣＡＲ２ではこの処理を転送回転機構による間接アドレシング機能を用いて高速化する方針を示した。続いて、この方針に基づく、文字パターンの走査によりあらかじめ求めておく外郭点のアドレス値で、配列データメモリ内に形成する黒点連結の長さのテーブルを間接アドレシングすることにより直接アクセスする投影処理法を示した。

さらに、次元圧縮の行列とベクトルの乗算をシストリックアルゴリズムを用い転送のオーバーヘッドなく実行する方法、ＬＩＳＣＡＲ２の各ＰＥの乗累算機能とツリー構成伝搬演算機構の高速の伝搬演算機能を組合わせて投影距離を効率良く求める方法を示した。

最後に、ＬＩＳＣＡＲ２にこれらの処理を搭載し処理速度を評価し、方向寄与度が８方向に拡張されたり、シティブロック距離からユークリッド距離に変更されたりして、処理量がかなり増加しているにも関わらず、ＬＩＳＣＡＲ１の速度に比べ、それぞれの処理で３～９倍、全体の平均では５倍程度の高速化が達成されていることを確認した。

第１０章　結論

１０．１　研究の内容と主な成果

ＬＳＩの集積能力をほぼ完全に引出せる単純かつ規則的なセルラ配列構成を前提としたＳＩＭＤプロセッサを取上げ、そのＰＥ配列部構成法、ＬＳＩおよびシステム構成法、演算処理法等を追究するとともに、ＬＳＩ－ＣＡＤおよび文字認識処理への応用を試みた。その結果、次の成果を得ることができた。

①隣接結合ネットワークの遠隔ＰＥ間の転送性能を補強する２種類のツリー構成の伝搬演算機構を提案したこと。

②効率的な９０度回転、ＰＥ単位の自律的なアドレシング等を可能とする転送回転型メモリアクセス機構を提案したこと。

③ＰＥ単位で個別に設ける制御レジスタにより実現される可変のＰＥ配列構造とそれによって実現される各種の機能を示したこと。

④１ビット構成と８ビット構成の２種類のＰＥ配列部用ＬＳＩの構成法、設計法を明らかにし、実チップを実現したこと。

⑤３台のセルラ配列型ＳＩＭＤプロセッサ（小型機：ＬＩＳＣＡＲ１，ＬＩＳＣＡＲ２，大型機：ＡＡＰ２）の構成・設計法を明らかにし、試作機を実現したこと。

⑥ＡＡＰ１のＬＳＩ－ＣＡＤへの応用で、クロックサイクル同一の条件で、汎用コンピュータの５０～５００倍の速度性能が得られることを実証したこと。

⑦ＬＩＳＣＡＲ１の文字認識処理への応用で、主要な認識処理アルゴリズムを専用ハードウェアに匹敵する１３～４６文字／秒の速度で実行できることを実証したこと。さらに、ＬＩＳＣＡＲ２により、より高度な手書き漢字認識処理がＬＩＳＣＡＲ１の手書き漢字認識処理に比べ５倍程度高速化されることを示したこと。

以下、これらの本研究で得られた成果の内容を詳しく述べる。

（１）ツリー構成伝搬演算機構の提案

隣接結合ベースのネットワークの遠隔ＰＥ間の転送性能を改善する手段として、バイパス，セレクティブ型の伝搬演算機構を多段に階層化することから得られるＮ進ツリー構成の伝搬演算機構を提案した。伝搬演算機構は、送信ＰＥから受信ＰＥまでの各ＰＥが同一の演算を行い、その結果を順次伝搬させて受信ＰＥに最終的な演算結果を得る機

構であり、隣接結合のみのネットワークでは最終的な結果を得るまでに送信ＰＥから受信ＰＥまでの距離Mに比例した時間がかかる。これに対し、本ツリー構成伝搬演算機構では、伝搬演算時間が、Ｏ（Ｎｌｏｇ$_N$Ｍ）と高速化され、かつ、Ｎを大きくすることによってハードウェア規模が低減されることを示した。さらに、バイパス構成とセレクティブ構成の比較から、セレクティブ構成が、算術演算のビット幅については１ビットに限定されるものの、同程度のハードウェア規模で２倍の速度性能の得られることを示した。

（２）ＰＥ配列の可変構造とその演算アルゴリズムの提案

ＰＥ配列の稼働率の向上を狙って、複数ＰＥの協調的動作により構成される演算ユニット（PU:Processing Unit）を問題向きに再構成する適応的（Adaptive）な可変構造を提案した。また、ＰＵの再構成は、全ＰＥ共通に配る命令を個別に修飾するＰＥ毎の制御レジスタを必要に応じて書き換えて、各ＰＥの役割分担を変更することによって実現され、送信、受信ＰＥの任意設定が可能な伝搬演算機構と組合わせることで、ＰＵごとの総和演算、行列計算等が効率的に実行されることを示した。

（３）転送回転型メモリアクセス機構の提案

ＰＥごとに１個ずつ設けるルーティングレジスタの並びを、本来のＰＥ間転送パスとしてだけでなく、一定個数のＰＥ（ＰＥ部分配列）毎に、局所メモリ構成用のメモリユニットに対するアドレス生成器、あるいは部分ビットプレーンの９０度回転器として利用するデータアクセス機構を提案した。そして、それらのデータ転送機能、アドレス生成機能、９０度回転機能を組合わせることで、ＳＩＭＤプロセッサの汎用化に有効な多値プレーンの９０度回転、２次元アクセス、間接アドレシング等を効率よく実効できることを示した。また、セルラ配列型ＳＩＭＤプロセッサで本来必要とする隣接ＰＥ間転送パスがほぼそのまま流用できるため、わずかの端子数、ゲート数の増加でＰＥ配列に組込めることも示した。

（４）大規模ＰＥ配列ＬＳＩチップの実現

外付けのメモリＩＣとの対を規則的に並べるだけでＰＥ配列の構成が可能な単純で規則性の高いＰＥ配列ＬＳＩ（ＡＡＰ２－ＬＳＩ）の構成、設計法を明らかにし、２μｍのＣＭＯＳ技術を用いて１ビットＰＥ６４個からなるチップを実現した。また、その構成、設計法である以下の３つの技術、

①トランジスタ当たりの機能の高い出力レベル補償型ｎＭＯＳパストランジスタ論理

の多用、

②８ｘ８の論理的なＰＥ配列を縦長ＰＥの４ｘ１６の配列とする高密度レイアウト、

③単純かつ規則的なセルラ配列構成によって可能となったＰＥ部マスクパターンの人手設計

により、スタンダードセルベースのフルカスタムＬＳＩに比べ、４～５倍の高集積度を実現し、目論見通りＬＳＩの集積能力を十分に活かせることを実証した。

さらに、１ビット構成のＡＡＰ２－ＬＳＩの限界を克服すべく、０．８μｍのＣＭＯＳゲートアレイ技術を用いて、８ビット構成ＰＥの配列を内蔵するＰＥ配列ＬＳＩ（ＡＡＰ３－ＬＳＩ）を試作した。各ＰＥのレジスタファイルを機能制限し、ハードウェアマクロのＲＡＭを８分割することで構成する、演算能力を外付けの標準ＤＲＡＭのアクセス速度に合せる等の設計手法により、８ビット乗算器、ツリー構成伝搬演算機構、転送回転型メモリアクセス機構等を組込みながら、１チップに１６ＰＥの搭載に成功している。

（５）大規模システムおよび小形経済的システムの実現

８ビットの固定小数点加減算で７,３５５ＭＯＰＳものピーク性能が得られる配列サイズが２５６ｘ２５６の超並列構成セルラ配列型ＳＩＭＤプロセッサ（ＡＡＰ２）を試作した。この中で、高密度の両面実装技術を開発し、１,０２４個のＡＡＰ２－ＬＳＩと８,１９２個の６４ＫビットＳＲＡＭからなるＰＥ配列をわずか0.2ｍ^{3}の容積に実装できることを示した。また、新しいボードおよびラック間の接続構成技術を開発し、均一な接続距離を保ったままで配列サイズを拡張できることを示した。

ビットライン単位の算術論理演算に対し２,８６０ＭＯＰＳものビット演算性能が得られる小型のセルラ配列型ＳＩＭＤプロセッサ（ＬＩＳＣＡＲ１）も試作した。このプロセッサは、４個のＡＡＰ２－ＬＳＩに汎用のＤＳＰ ＬＳＩと標準ＤＲＡＭを組合わせることで、Ｂ４サイズのボード１枚に実装することに成功している。

さらに、同様の構成の１ボードプロセッサ（ＬＩＳＣＡＲ２）をＡＡＰ３－ＬＳＩ４個を用いて試作した。ＬＩＳＣＡＲ１に比べ、配列サイズが全体で８ｘ８と１／４になっているにも関わらず、乗算を用いるユークリッド距離計算で２５倍、乗算を用いないシティブロック距離計算、９０度回転で、それぞれ２倍および４倍以上の性能が得られることを示した。

（６）ＡＡＰ１のＬＳＩ－ＣＡＤへの応用での高速性の実証

ＡＡＰの応用先として超高速処理が必要でかつＳＩＭＤ処理向きのＬＳＩ－ＣＡＤで

ある論理シミュレーションとマスクパターン設計の自動配線を取上げ、それぞれの処理方法を明らかにし、ＡＡＰ１を用いてその処理性能を評価した。

その結果、ＬＳＩ－ＣＡＤについては、論理シミュレーションが１ゲートを１ＰＥで、ゲート間の結線を伝搬転送でエミュレートするシミュレーション法を用いることによって、クロックサイクル同一の条件で汎用コンピュータの５０～５００倍の速度性能が得られることを明らかにした。また、自動配線が、迷路法にライン探索アルゴリズムを組合わせるセルラ配列型ＳＩＭＤプロセッサ向きアルゴリズムを、ＰＥ間のシフト転送と伝搬演算でを活用することにより、クロックサイクル同一の条件で汎用コンピュータの１００倍以上の速度で実行されることを明らかにした。

（7）ＬＩＳＣＡＲの文字認識処理への応用での高速性の実証

小型経済性の要求される文字認識装置への組込をねらい、従来の専用ハードウェアでは困難であった複数の認識アルゴリズムのＬＩＳＣＡＲ１への搭載を試みた。その結果、前処理、特徴抽出、識別の各処理の並列処理方法に加え、２ないし４文字並列処理、距離計算の語長短縮、逐次処理バイパス等の高速化手法を明らかにし、処理内容の大きく異なる手書き英数カナ文字、印刷漢字、手書き漢字に関するこれまで開発してきた主要な認識アルゴリズムの全処理の実行が、それぞれ２３、４６、１３文字／秒の速度で実現されることを実機上で確認した。

さらに、処理速度の劣る手書き漢字認識処理に対し、高速化と一層の認識精度向上の両立を目指し、ＬＩＳＣＡＲ２の適用を試みた。外郭方向寄与度特徴を４方向寄与度から８方向寄与度に、識別処理をシティブロック距離単独からユークリッド距離による大分類と投影距離による詳細識別の２段階構成へと認識処理を高度化しているにも関わらず、ＬＩＳＣＡＲ１に比べ５倍程度の高速化が可能になることを実機上で確認した。

１０．２　今後の研究の方向と課題

本研究は開始時よりほぼ１５年が経過した。その間、ＡＡＰ１，２は試作レベルに留ったのに対し、ＬＩＳＣＡＲ１は、その小型経済性とプログラムの変更により種々の認識処理が搭載できることが評価され、文字認識装置の認識処理エンジンとして実用化され、一定の成功を納めるにいたった。

これに対して、逐次処理プロセッサの進歩は予想以上で、本研究の開始と期を同じくして、やはりＬＳＩの設計ネック回避を背景に登場した汎用処理指向のＲＩＳＣプロセッサが著しい進歩をとげ、ＣＩＳＣプロセッサも巻き込む形で、大幅な性能向上を果たしている。この逐次処理プロセッサの飛躍的発展には、登場の背景からは想像もつか

ぬ程、複雑化したＲＩＳＣプロセッサを短期間に実現可能とするまでの設計ツールや設計技術の進歩が大きく貢献している。この結果、現在では機器組込み用のＲＩＳＣプロセッサが、各種の専用プロセッサを駆逐したばかりか、専用の逐次処理プロセッサともいうべき信号処理用プロセッサ（ＤＳＰ）の領分まで脅かす存在となっている。
ＬＩＳＣＡＲに対する現状も同様であり、ＲＩＳＣプロセッサに対する性能上の優位性は急速に失われ、今後の発展は有りえないようにさえ見える。

もちろん、このＬＩＳＣＡＲの競争力の低下の原因は主に開発力の差であって、セルラ配列型ＳＩＭＤプロセッサ自体の本質的な優位性が失われたからではない。しかし、汎用性やプログラミングの容易性で劣り、巨額の開発投資に耐えるだけの需要が見込めない以上、これまでと同じようなアプローチ、同じような応用を指向しているのでは、ＲＩＳＣプロセッサに太刀打ちできなくなるのは明らかである。以下では、このような状況のもとで今後とるべき研究の方向と解決すべき課題についてまとめる。

（1）新たな応用分野の開拓

これまでＬＩＳＣＡＲの有力な応用先であった文字認識処理の分野においても、一部の高精度の認識アルゴリズムを除き単独のマイクロプロセッサで必要な速度と精度を満たせるようになって来ている。現在では、１００ＭＩＰＳを越えるローコストのＲＩＳＣマイクロプロセッサが利用できるようになったからである。こうなるとコストで劣るＬＩＳＣＡＲのような専用プロセッサを文字認識処理にわざわざ使う必要はなくなってくる。

しかし、これを持って、専用プロセッサの応用分野がしだいに狭まって来ていると考えるのは早計である。マルチメディア処理のようにＳＩＭＤプロセッサ向きの定型的な処理を大量に必要とする分野が新たに出現しているからである。事実、ＴＩ社で開発された１次元構成のセルラ配列型ＳＩＭＤプロセッサ[34]は、テレビ受像機の前処理用ＬＳＩとして大きな成功を治めている。また、ＨＰやＳＵＮに用いられているＲＩＳＣプロセッサには、画像圧縮の符号化、複合化に対応するためにＳＩＭＤ型の処理機構が組込まれている[71]さらに、最近では、ＳＩＭＤ処理機構を組込むことでマルチメディア処理全般への応用を目指したメディアプロセッサなる専用プロセッサＬＳＩが出現している。今後は、このような分野にすばやくアプローチし、必要に応じてアーキテクチャを適合化することで応用領域を広げていく努力がますます重要になると考えられる。

（2）技術の進歩に応じて、逐次的に高性能化が可能なアーキテクチャの追究

汎用プロセッサの世界では、一度作ったアーキテクチャを１０年２０年と発展させて行くのは常道であり、アーキテクチャの開発時にその発展性に十分な考慮が払われている。しかし、ＬＩＳＣＡＲ１では、アーキテクチャを維持したままで小型化、高性能化をはかることが困難で、１から２への移行時に全面的なソフトウェアの変更を余儀なくされた。これは製品開発の側からすると受入れがたいものであった。専用ハードウェアに対し、バージョンアップや新製品開発のコストが低いという専用プロセッサの利点が半減されてしまうからである。

このような状況に至った原因を考えてみると、文字認識処理の特性に対するアーキテクチャ上の考慮が不十分であっただけでなく、開発当初に数年後のバージョンアップのビジョンまで描き切れていなかったことによるところが大きい。もっとも、後者の問題については、いまもって解決の方向が見えているわけではない。応用やアーキテクチャが変れば小型化、高性能化のための対応も大きく変ってくるからである。従って、今後、アーキテクチャの大幅な見直しをはかったり、異なる応用を目指す際には、互換性を維持したまま小型化、高性能化できるようにすることは、セルラ配列型ＳＩＭＤプロセッサの専用プロセッサの開発を進める上でも基本的な課題といえる。

（３）セルラ配列型ＰＥ配列ＬＳＩ構成の見直し

セルラ配列構成ＬＳＩの基本的な考え方は、規則性の高いＰＥ配列のみ搭載するというものである。これは、本研究開始時には設計が単純化される、容易に高い集積度が得られる、ＬＳＩを規則的に並べるだけで任意のサイズのＰＥ配列を構成できる等大きな利点があると考えられた。しかし、現在ではこれらはあまり利点とは言えなくなってきている。なぜならば、設計の生産性が上がり設計の単純性に固執する必要がなくなってきたり、小型のプロセッサを目指す場合には集積度が上がり必要な数十程度の並列度は十分１チップに入るようになってきているからである。また、並列度自体を大きくできたとしても、文字認識処理における２文字並列や４文字並列のような手法を使わないと、その能力を引出せないのならば、プログラミングを難しくするばかりで、利点が少ないことも明らかになってきたからである。これは、換言すると、プログラミングの観点からは、高速化の程度が同程度ならば、並列度を大きくするよりもＰＥ自体の性能改善をはかる方が利点が大きいといえる。

こうなると１チップ化可能な範囲の並列度とし、周辺回路を極力取込んで、１チップマイクロプロセッサ同様な使い方ができるようにする方が、使いやすさや経済性の点で利点が大きいと考えられる。今後はこういった方向でのアーキテクチャの追究が成功のポイントとなろう。

（４）ソフトウェア開発に対する拒否反応の払拭

本論文では専用プロセッサ指向と言うことで、ソフトウェアの開発容易性の点には、触れないようにしてきたが、実はかなり深刻な問題である。何が深刻かといえば、マクロアセンブリレベルの言語に対する周囲の拒否反応が強いことである。確かに、このような専用言語をマスタするには、１，２ケ月の期間を要する。また、習熟してもＣやＦＯＲＴＲＡＮのような言語のように抽象的な記述ができるわけでわない。しかし、ひとたび高速あるいはスタティックステップ数の小さいプログラムを作ろうとした場合には汎用プロセッサと比べそれほど生産性に違いがでるわけではない。逐次型の汎用プロセッサでは、自由度が高い分、最適な処理法を見つけるのに試行錯誤を繰り返す必要があるからである。同一内容のプログラムの高速化の経験からすれば、生産性は、マクロが整備されている条件では、高級言語による汎用プロセッサのプログラミングに比べ、高々２，３分の１に低下するだけであった。もちろん、これには、インタラクティブなデバッガを利用すれば逐次型の汎用プロセッサと同様な感覚でデバッグできるＳＩＭＤプロセッサのデバッグ容易性も寄与している。

いずれにしても、生涯に記述されるプログラムの総ステップ数が数十Ｋにも及ばない可能性のある専用プロセッサに対して、高性能の高級言語コンパイラを用意するわけにはいかない。専用言語ベースのプログラム開発環境を提供するだけでもかなりの負担である。今後はより開発コストの低いプログラム開発環境の構築方法を明らかにするとともに、習熟期間の短い専用言語の仕様を追究し、専用言語に対する拒否反応を払拭していく必要がある。

謝　辞

本論文をまとめるに当たり、懇切なご指導とご教示を賜った名古屋大学工学部稲垣康善情報工学科教授、鳥脇純一朗情報工学科教授、島田俊夫電気工学科教授、ならびに高木直史情報工学科助教授に厚く感謝いたします。また、筆者が名古屋大学大学院在学中の心温まる御指導とその後の変らぬ励ましにより本論文の執筆へと導いて下さった故赤尾保男元名古屋大学工学部電気工学科教授に深く感謝いたします。

本論文は、筆者が１９７９年から１９９３年にかけてＮＴＴ武蔵野電気通信研究所集積回路研究部、厚木電気通信研究所カスタム化技術研究部、横須賀電気通信研究所宅内機器研究部、ＬＳＩ研究所ＬＳＩ開発研究部、設計システム研究部、ヒューマンインタフェース研究所マルチメディア処理研究部、第５プロジェクトチームにおいて行った研究の成果をまとめたものであります。これらの各所で本研究の機会を与えて頂いた須藤常太元カスタム化技術研究部部長、小森和昭ＮＴＴインテリジェントテクノロジー株式会社社長（元宅内機器研究部統括役）、酒井保良ＬＳＩ研究所長（元設計システム研究部長）、徳永幸生映像処理研究部部長（元マルチメディア処理研究部グループリーダ）、北村正元第５プロジェクトチームリーダ、武田和光元設計システム研究部グループリーダ、中川亨元マルチメディア処理研究部グループリーダに感謝いたします。

本研究を進めるに当たり、直接ご指導いただきました中島孝利ＮＴＴエレクトロニクステクノロジー営業本部副本部長（元集積回路研究部研究室長）、青木誠元交換方式研究部主幹研究員、宮原末治メディア応用システム研究部主幹研究員、多田俊吉関連企業本部担当部長（元言語メディア処理研究部主任研究員）、川谷隆彦元第５プロジェクトチーム主幹研究員、杉山吉関連企業本部担当部長（元カスタム化技術研究部主幹研究員）に心からお礼申し上げます。

また、本研究は多くの方々の支援、協力のもとで行ってきたものであります。特に、大規模セルラ配列型ＳＩＭＤプロセッサＡＡＰの試作とＬＳＩ－ＣＡＤへの応用に関しては、渡辺琢美高速ＬＳＩ研究部主幹研究員、北村美宏広帯域研究部主幹研究員、および土屋敏雄関連企業本部担当課長から、小型高並列プロセッサＬＩＳＣＡＲ１，２の試作と文字認識への応用に関しては、宮原末治メディア応用システム研究部主幹研究員、多田俊吉関連企業本部担当部長、木村義政メディア応用システム研究部主任研究員、宮本信夫メディア応用システム研究部主任研究員、下原勝憲画像通信処理研究部グループリーダ、および曽根原登映像処理研究部グループリーダから、多大の支援と協力をいただいたことに深く感謝いたします。

最後に、本論文執筆の機会を与えて下さった唐津高機能ＬＳＩ研究部部長、笠井高機能ＬＳＩ研究部グループリーダに感謝いたします。

参考文献

(1) S. H. Unger: "A computer oriented toward spatia problem", Proc. IRE. Vol.46, No.10, pp.1744-1750(1958)

(2) B. H. McCormick: "The Illinois pattern recognition computer ILLIAC III", IEEE Trans. Electronic Computers, EC-12, No.12, pp.791-813 (1963)

(3) G. H. Barnes, et al.: "The ILLIAC IV Computer", IEEE Trans. computers,Vol.C-17, No.8, pp.746-757 (1968)

(4) 加藤, 苗村: "並列処理計算機", オーム社(1976)

(5) M.Griffin, G.Tahara, K.Knorpp, R.Pinkham, B.Riley: "An 11-Million Transistor Neural Network Execution Engine", ISSCC'91 Digest of Tecnical Papers, pp.180-181(1991)

(6) D. A. Patterson and C.H.Sequin: "Design consideration for single chip computer of the future", IEEE Trans. C-29, No.2, pp.108-115(1980)

(7) H. T. Kung: "Why systolic architecture?", IEEE Computer, Vol.15, No.1, pp.37-46(1982)

(8) M. J. B. Duff and D. M. Watson: "The cellular logic array image processor", Comp. Journal, Vol.20, pp.68-72(1977)

(9) 宮田裕行: "高速画像処理向きセルラ・アレイ・プロセッサ", 情処学会計算機アーキテクチャ研究会資料, CA53-6(1984)

(10) R. Brooks and R. Lum: "Yes an SIMD mashine can be used for AI", Proc. of IJCAI, pp.73-79(1985)

(11) K. E. Batcher: "STARAN parallel processor system hardware", Proc. of National Computer Conference, pp.405-410(1974)

(12) P. M. Flanders, D. J. Hunt, S. F. Reddaway, D. Parkinson: "Efficient high speed computing with the distributed array processor", in High Speed Computer and Algorithm Organization, New York:Academic, pp.113-128(1977).

(13) M. J. B. Duff et al.: "Review of the CLIP image processing system", Proc. of National Computer Conference, pp.1055-1060(1978)

(14) K. E. Batcher: "Design of a massively parallel processor", IEEE Trans.Computers, C-29, No.9, pp.836-840(1980)

(15) K. E. Batcher: "Bit-serial parallel processing systems", IEEE Trans.Computers, Vol.C-31, No.5, pp.374-384(1982)

(16) W. Daniel hillis: "The connection machine", An ACM Distinguiushed Dissertation, The MIT Press, London(1985)

(17) T. Blank: "The MasPar MP-1 architecture", CompCon Spring 90, San Francisco, pp.20-24 (1990)

(18) H. J. Siegel, J. D. Armstrong and D. W. Watson: "Mapping computer-vision-related tasks onto reconfigurable parallel-processing systems", IEEE Computer, vol.25, No.2, pp.54-63 (1992)

(19) A. H. Karp, D. Heller and H. Simon: "1993 Gorden bell prize winners", IEEE Computer, vol.27, No.1, pp.69-75(1994)

(20) J. Beetem, M. Denneau and D. Weingarten: "The GF11 supercomputer", 12th symposium on Computer Architechture Conf. Proc., pp.108-115(1985)

(21) T. Feng: "A survey of interconnection network", IEEE Computer, Vol.14, No12, pp.12-30 (1981)

(22) A. P. Reeves: "A Systematically designed binary array processor", IEEE Trans. Computers, C-29, No.4, pp.278-287(1980)

(23) G. E. Blelloch: "Scans as primitive parallel operations", Proc., Int.Conf.on Parallel Processing, pp.355-362(1987)

(24) G. E. Blelloch and J. J. Little: "Parallel solutions to geometric problems on the scan model of computation", Proc., Int. Conf. on Parallel Processing, pp.218-222(1988)

(25) J. J. Little, G. E. Blelloch and T. A. Cass: "Algorithmic techniques for computer vision on a fine-grained parallel machine", IEEE Trans. Pattern Anal. & Mach. Intell., Vol.11, No.3, pp.244-257(1989)

(26) "Connection machine model CM-2 technical summary", Thinking Machines Corporation Version 5.1(May 1989)

(27) K. Hwang(堀越監訳):"コンピュータの高速演算方式", 近代科学社, pp82-88(1979)

(28) J. D. Ullman: "Computational aspects of VLSI", Computer Science Press (1984)

(29) 岡田, 田島, 森: "新しい可変構造プロセッサアレイ －ＥＴＬ ＡＮＭＡ－", 情処学会研報, CA-24, pp.95-104(1977)

(30) R.W. Hockney, C.R. Jeshope(奥川, 黒住訳): "並列計算機", 共立出版, pp.141- 156 (1984)

(31) L. Snyder: "Introduction to the configurable, highly parallel computer", IEEE COMPUTER Vol.15, No.1, pp.47-56(1982)

(32) Y. Kitamura, T. Watanabe and K. Matsuhiro: "CAD oriented 2-dimensional array processor －part2:routing and logic simulation", IFIP Workshop on CAD Engines(1987)

(33) 藤田善弘: "画像演算メモリ", 情処学会計算機アーキテクチャ研究会資料 ,91-ARC89-1, pp.1-8(1991)

(34) J. Childers, P. Reinecke, H. Miyaguchi, Y. Takahashi, Y. Yaguchi and M. Takeyasu: "SVP: serial video processor", IEEE 1990 CICC, 17.3.1(1990)

(35) N. Yamashita et al.: "A 3.84GIPS integrated memory array processor LSI with 64 processing elements and 2Mb SRAM", IEEE ISSCC Digest of Tecnical Papers, pp.260-261(1993)

(36) K.E. Batcher: "The Multidimensional access memory in STARAN", IEEE Trans.C-26, No.2, pp.174-177(1977)

(37) 元岡達, 田中英彦, 上森明, 鈴木達郎: "２次元記憶を用いた連想記憶システム", 信学技報, EC76-80(1976)

(38) D. Van Voorhis and T. Morrin: "Memory systems for image processing", IEEE Trans. Computers, C-27, Vol.2, pp.113-125(Feb.1978)

(39) D. Hayes and B. Strawhorns: "The application of multi-dimensional access memories to high performance signal processing systems", Proc. ICASSP, pp.1412-1415(1985)

(40) 田畑邦晃, 武田晴夫, 町田哲夫: "ラスタ走査とテーブル参照による画像回転の高速処理", 信学論(D), J69-D, No.1, pp.80-90(1986)

(41) E.R. Komen: "Efficient parallelism using indirect addressing in SIMD processor arrays", Pattern Recognition Letters 12, pp.279-289(1991)

(42) C.L. Kuszmaul: "Rapid transpose methods on massively parallel SIMD Computers", The society for industrial & applied mathematics annual meeting(July 1990)

(43) 中山晶, 木村文隆, 吉田雄二, 福村晃夫: " パイプライン型並列画像処理アルゴリズムの一提案" , 信学論(D), J71-D, No.9, pp.1879-1882(1988)

(44) T. Sudo, T. Nakashima, M. Aoki and T.Kondo: "An LSI adaptive array processor", IEEE ISSCC Dig. Tech. Papers, pp.122-123(1982)

(45) T. Kondo, T. Nakashima, M. Aoki and T. Sudo: "An LSI adaptive array processor", IEEE J. Solid-State Circuits, C-18, Vol.2, pp.147-156(1983)

(46) C. Mead and L. Conway: "Introduction to VLSI systems", Addison-Wesley (1980)

(47) 佐々木, 矢野, 力野, 関: "パストランジスタ回路集積化技術：ＬＥＡＰ", 1994年信学会秋期全国大会, A-64, pp.64(1994)

(48) 近藤: "ＣＭＯＳトランスファゲート回路の比較検討", 昭和５９年信学会総合全国大会 589, pp.2-352(1984)

(49) W. R. Iversen: "New CMOS chip processes data in parallel", Electronics Week, November 12, pp.17-18(1984)

(50) 前田: "ＶＬＳＩ画像処理プロセッサ", 情処学会誌 Vol.31, No.4, pp.480-485 (1990)

(51) T. Kondo, T. Nakashima, T. Tsuchiya, Y. Sugiyama and T. Sudo: "A Large scale cellular array processor: AAP-1", Proc., 1985 ACM Computer Science Conf., pp.100-111(1985)

(52) 近藤, 杉山, 中島: "２次元アレープロセッサ(ＡＡＰ２)とプログラミング言語", 信学論, Vol.J71-D, No.8, pp.1399-1406(1988)

(53) 曽根原, 近藤, 木村, 内田, 平岩: "ニューロ処理指向並列プロセッサ（Neurocessor）及びこれを用いた学習型個人適応システムの開発",1993年信学会秋期全国大会, D-31, pp.6-33(1993)

(54) 杉山, 渡辺: "ＡＡＰによる論理モジュール並列配置", 信学技報 CAS84-12, pp.25-32(1984)

(55) M. A. Breuer and K. Shamasa: "A hardware router", Journal of Digital Systems, Vol.4, No.4, pp.393-408(1980)

(56) T. Blank, M. Stefik and W. Vancleemput: "Parallel bit map processor architecture for DA argorithms", IEEE 18th Design Automation Conference, pp.837-845(1981)

(57) W. Heyns et al.: "A line-expansion problem with a guaranteed solution", Proc. 17th Design Automation Conference, pp.243-249(1980)

(58) T. Watanabe, H. Kitazawa and Y. Sugiyama: "Parallel adaptable routing algorithm and its implementation on a two dimensional array processor", IEEE Trans. Comput.-Aided Des. Integrated & Syst., CAD-6, Vol.2, pp.241-250(1987)

(59) M. Hanan: "On steiner's problem with rectilinear distance", SIAM J. Appl.Math.14, pp.255-265(1966)

(60) 赤松, 塩, 飯田, 川谷:"手書き漢字用文字読取装置", 研実報, Vol.36,No.4, pp.579-587(1987)

(61) 宮本, 堤田, 中島, 川谷: "英数カナ用高精度文字読取装置", 研実報, Vol.37, No.2(1988)

(62) 上, 岡本, 天満, 浅井: "階層化判別法とその文字認識システムＰＣ－ＯＣＲ", 信学技報 PRU-86-76, pp.49-55 (1987)

(63) 松村: "卓上型手書き漢字ＯＣＲの開発", 信学会総合全国大会1540, pp.6-124 (1985)

(64) 吉本, 安部, 小森: "高速印刷文字認識装置の開発", 信学技報 PRU88-16, pp.1-7(1988)

(65) 多田, 宮原, 木村, 川谷: "ＬＩＳＣＡＲを用いた手書き帳票読取り", NTT R&D, Vol.39 No.11, pp.1603-1612(1990)

(66) 目黒, 梅田: "マルチフォント印刷漢字認識装置", 信学論 J67-D, No.8, pp.908-915 (1984)

(67) 宮原, 木村, 豊田, 宮田: "部分パターンによる可変ピッチ文書からの文字切出しと認識", 信学論 J72-D-II, No.6, pp.846-854(1989)

(68) 萩田, 内藤, 増田: "外郭方向寄与度特徴による手書き漢字の識別", 信学論 J66-D, No.10,pp.1185-1192(1983)

(69) 能見他:"手書き数字認識における誤読・リジェクトパターンの分析 第２回文字認識コンテストの実施結果より",PRU93-46, pp25-32(1939)

(70) 池田，田中，元岡:"手書き文字認識における投影距離法", Vol.24, No.1, pp.106-112 (1983)

(71) R.B.Lee:"Accelerating Multimedia with Enhanced Microprocessors", IEEE Micro, Vol.15, No.2, pp.22-32 (1995)

研究業績リスト（＊は本論分に関連する業績を示す。）

１．学会誌論文

(1*) T. Kondo, T. Nakashima, M. Aoki and T. Sudo: "An LSI adaptive array processor", IEEE J.Solid-State Circuits, C-18,2, pp.147-156(1983)

(2*) 近藤, 杉山, 中島: "２次元アレープロセッサ(ＡＡＰ２)とプログラミング言語", 信学論（Ｄ）, Vol.J71-D, No.8, pp.1399-1406(1988)

(3*) 近藤: "ＬＳＩ化に適したツリー構成走査演算機構", 信学論(C-II), J74-C-II, No.5, pp.388-397(1991)

(4*) 近藤, 木村, 曽根原: "１次元ＳＩＭＤプロセッサ向きデータアクセス機構", 信学論 (D-II), J74-D-II, No.6, pp.269-278(1993)

(5*) T. Kondo, Y. Kimura and N. Sonehara: "Single-Board SIMD Processors using Gate-Array LSIs for Parallel Processing", IEICE Transactions on Electronics, E76-C, No.12, pp.1827-1834(1993)

(6*) 多田俊吉, 近藤利夫, 宮原末治: "小形高並列プロセッサと文字認識への応用", 信学論(D), J71-D, No.8, pp.1546-1552(1988)

(7) 宮原, 近藤, 多田: "ＳＩＭＤ型並列プロセッサを用いたフルテキスト検索", 情処学会論文誌, Vol.33, No.3, pp.397-404(1992)

２．査読付き国際会議論文

(1*) T. Kondo, T. Nakashima, T. Tsuchiya, Y.Sugiyama and T. Sudo: "A Large Scale Cellular Array Processor: AAP-1", Proc., 1985 ACM Computer Science Conf., pp.100-111(1985)

(2*) T. Kondo, T. Tsuchiya, Y. Kitamura and Y. Sugiyama: "Pseudo MIMD array processor -AAP2", 13th Symposium on Computer Architechture Conf. Proc., pp.330(1986)

(3*) T. Kondo, S. Tada and S. Miyahara: "Kanji character recognition unit with hand-scanner using SIMD processor", SPIE Vol.1001 Visual Communication and Image Processing, pp.476-483(1988)

(4*) T. Sudo, T. Nakashima, M. Aoki and T. Kondo: "An LSI Adaptive Array Processor", ISSCC Digest of Tecnical Papers, pp.122-123(1982)

(5) T. Watanabe, Y. Sugiyama, T. Kondo and Y. Kitamura :"Neural Network Simulation on a Massively Parallel Cellular Array Processor: AAP-2", Proc. of Int'l Joint Conf. on Neural Networks(IJCNN-89), pp.II-155-161(June 1989)

5．解説論文等

(1*) 近藤, 中島: "ＡＡＰ（Adaptive Array Processor）とその応用", Ｏ plus Ｅ, No.55, pp.68-75(1984)

(2*) 近藤, 中島: "１ビット２次元ＳＩＭＤプロセッサＡＡＰ", bit臨時増刊:並列コンピュータアーキテクチャ Vol.21, No.4(1989)

(3*) 近藤，多田，宮原，杉山: "小型並列プロセッサＬＩＳＣＡＲの構成", NTT R&D Vol.39, No.11, pp.1585-1592(1990)

(4*) 宮原，多田，鈴木，近藤: "ＬＩＳＣＡＲを用いた印刷文書読取り", NTT R&D Vol.39, No.11, pp.1593-1602(1990)

(5) 近藤，木村，曽根原: "ニューロ処理指向並列プロセッサ：ニューロセッサ", NTT R&D Vol.42, No.3, pp.-(1993)

略語、定義語一覧