計算機アーキテクチャ　101－7
（１９９３．　８．　２０）

高並列計算機 EM-X のアーキテクチャ

児玉祐悦　佐藤三久　坂根広史　坂井修一†　山口喜教

電子技術総合研究所　　†新情報処理開発機構

EM-X は、細粒度データ通信に基づく並列処理を目指した高並列計算機である。このため EM-X では低レイテンシ高スループットのデータ転送や、データ駆動メカニズムに基づく高速な同期や命令発火機構を実現し、かつ、これらの機能を高性能な RISC パイプラインと融合している。

同様のコンセプトに基づき、我々は EM-4 を開発してきたが、EM-X は EM-4 の評価を基に、さらに高効率な並列処理を目指している。このため、パケットの入出力性能のバランスをとりつつ性能向上を図るとともに、リモートメモリ参照のハードウェアサポートや２つの入力パケットバッファなどにより動的レイテンシに対するサポートも行なっている。

The architecture of a highly parallel computer EM-X

Yuetsu KODAMA　Mitsuhisa SATO　Hirohumi SAKANE　Shuichi SAKAI†
Yoshinori YAMAGUCHI

Electrotechnical Laboratory　　†Real World Computing Partnership
1-1-4 Umezono, Tsukuba, Ibaraki 305, Japan

The EM-X is a highly parallel computer toward to parallel processing using fine grain data communication. The EM-X supports low-latency and high-throughput data communication, low-overhead matching and instruction invocation based on dataflow mechanism. It also fuse the packet communication with RISC pipeline execution.

We had already developed The EM-4 which was based on the above concept. We are developing the EM-X toward to more efficient parallel processing in consideration of the evaluation of the EM-4. The EM-X improves the performance of data input and output and balances them. The EM-X also tolerates some dynamic latency by non-blocked remote memory access and additional input packet buffer for high priority packets.

1 はじめに

並列処理による計算機の性能向上の研究が活発に行なわれており、データの独立性が高い問題では、すでに十分な性能向上を果たしている。しかし、より並列処理の可能分野を広げるためには、キメの細かい要素プロセッサ間の通信が不可欠である。このような処理では、大量のデータを一度に他のプロセッサに送るというよりは、小さなデータを送り、その応答結果を必要とすることがよくある。このため、データ通信のスループットと同様に、レイテンシが重要となる。

要素プロセッサ間の通信のレイテンシを考える場合、データの生成、ネットワークへの送出、ネットワーク内の転送、ネットワークからの受取のそれぞれがオーバヘッドとなる。最近の並列計算機では、高速なプロセッサと高速なネットワークにより、データの生成やネットワーク内の転送を、比較的高速に行なうことができる。しかし、プロセッサとネットワークとの間の制御がボトルネックとなり、要素プロセッサ間の通信レイテンシは、他の演算などの命令実行に比べると、多大の時間を必要としている。

プロセッサとネットワークとの間の制御を高速にするためには、データ転送と演算パイプラインとを融合し、それらの間のオーバヘッドを極力減らす方式が考えられる。我々は、このようなデータ転送と演算パイプラインとを融合した新しいアーキテクチャを持つ高並列計算機 EM-4[2, 1] をすでに開発した。EM-4 では要素プロセッサ (PE) 間の通信を１ワードのアドレス部と１ワードのデータ部からなる単純化したパケットを用いることにより、データ転送の演算パイプラインへの融合を実現している。例えば、EM-4 ではパケットの宛先を指定するオペランドと出力データを指定するオペランドを与えることにより、１命令で任意の PE に対してデータを転送することができる。さらに限られた PE に対しては、加算などの演算結果を直接転送することも可能である。データを受けとる側でも、実行すべき命令ブロックがパケットにより指示されるデータ駆動機構により、柔軟で高速なデータの処理が可能になっている。このような演算パイプラインと融合したデータ転送機構により、EM-4 は強力なネットワークと高速な演算パイプラインの両者の性能を十分に引きだし、効率の良い並列処理を実現することができた。

EM-X は、EM-4 のアーキテクチャを基礎として、より効率良く並列実行を行ない、並列処理の適用分野を広げるために開発中の並列計算機である。そのために、EM-4 の評価により明らかとなったいくつかの問題点を解決している。本稿ではこのような観点から EM-X のアーキテクチャについて述べる。まず、２章で EM-4 の問題点とそれに対する EM-X での解決策について述べるとともに、EM-X のアーキテクチャの概要について述べる。３章では、特に EM-X のパケット転送機構について述べる。４章では、EM-X で新たに付け加えられた機能について述べ、最後に今後の EM-X の開発計画についてまとめる。

2 EM-X

EM-4 における問題点としては、まず第一に、ネットワーク利用率が高い時に起きるネットワークホットスポットの問題があげられる。これはパケットが要素プロセッサに取り込まれず、ネットワーク上に溢れることによって発生しており、この影響はパケットを出力しようとする他のプロセッサの演算パイプラインにまで広がるため、全体の実行効率に大きな影響を与える。EM-4 ではプロセッサ内部に１６パケット分の入力パケットバッファを内蔵して、パケット取り込みの性能を向上させているが、ネットワークが混んでくるとそのバッファが溢れてしまうためにホットスポットが発生する。EM-X では、ネットワークが混雑している状況でもパケット処理の各部のバランスを崩さないように設計されているため、各部の性能を向上しても、ホットスポットは起きにくくなっている。このパケット転送機構については３章で詳しく述べる。

次にあげられるのが、メモリバンド幅の不足である。EM-4では待ち合わせメモリやパケットバッファなど命令やデータとは独立してアクセスすることのできるメモリも単一のメモリ空間に配置している。これは高並列計算機を目指すためには、各要素プロセッサ(PE)当たりの物理的スペースを極力減らさなければならないためであり、また、すべてのメモリ空間をメモリ参照命令で参照できる空間に配置することにより、柔軟で効率良いメモリ管理を目指したためである。しかし、このことにより特殊用途のメモリ空間にアクセスする際にも命令フェッチやデータ参照とのアクセス競合が起き、それらの調停の複雑化と実行効率の低下を招いていた。

通常のRISCプロセッサのように、EM-Xにおいても命令キャッシュの内蔵やメモリバンド幅の多ワード化を検討したが、プロセッサのピン数やゲート数の制限などにより困難であった。また、外部キャッシュについても、異なるメモリチップが必要になるため、PE当たりの物理スペースの制限から困難であった。そこで、EM-Xでは時間方向にメモリバンド幅を拡張している。すなわち、1システムクロックサイクルを2つのフェーズに分け、各フェーズでメモリ参照を行なうことにより、1クロック当たりのメモリバンド幅を2倍に拡大している。EM-4の演算パイプラインは2段のパイプラインであり、命令フェッチはシステムクロックの前半で完了していなければならないため、後半サイクルでのメモリ使用は待ち合わせ処理やstore命令によるデータの書き込みに限られていた。このため、後半サイクルでのメモリの参照を有効に利用するのは、妥当な拡張であると考えられる。

このようなメモリバンド幅の拡張により、パケットバッファへのアクセスを命令実行と独立に行なうことが可能になり、また、他の要素プロセッサからのリモートメモリ参照要求を実行中のスレッドにブロックされることなく実行することも可能になった。これらの機構については4章で詳しく述べる。

3番めの問題は、パケット処理の実行スケジューリングが固定化されていることである。EM-4ではネットワークからやってきたパケットは、1つの入力パケットバッファに保持されPEに到着した順に処理される。このようにFIFO性を保つことにより、余分な同期を省くことができ、ブロックデータ転送などを簡略化することができた。一方、どんなパケットでも、すでに到着しているすべてのパケットの処理の終了を待たなければならないため、到着してから実際に実行されるまでの待ち時間が大きくなってしまうという問題点があった。

例えば、EM-4ではネットワークが動的負荷分散をサポートしており、特殊なパケットをネットワーク上を巡回させることで、負荷最小のPEを数クロックで探索することができる。けれども、その負荷情報を持ったパケットがプロセッサに到着しても、実際にそのデータを利用できるのは、入力パケットバッファ中の全パケットを処理した後であるため、時間的ギャップが生じ、そのデータを有効に利用することが難しかった。EM-Xでは異なる優先度を持つ2種類の入力バッファを用意することにより、柔軟なパケットスケジューリングを可能にした。これについても4章で詳しく述べる。

4番めの問題は、待ち合わせ処理の機能が制限されている点である。EM-4では2つのパケットの待ち合わせを高速に処理できるが、これはリモートメモリ参照パケットのような特殊パケットに対しては適用できない。このため、ほとんど待ち合わせ処理のみで済むI-structureのような構造体メモリの同期処理も、ソフトウェアでエミュレーションしなければならず、効率が低下していた。また、例えば右入力パケットが続けてやってきた時のような待ち合わせエラー時には全体の実行を中断してしまうため、エラーの復帰やそれを利用したキューイングつきの待ち合わせの実現は不可能であった。EM-Xでは、待ち合わせ処理の機能を見直し、特殊パケットに対しても待ち合わせを行なえるようにするとともに、待ち合わせエラー時には特定の

エラーハンドラに制御を移すことにより、同期つきメモリの処理を柔軟にかつ効率良く行なえるようになった。これについても4章で詳しく述べる。

この他、EM-4はI/Oノードを持たずファイルなどのI/O処理はすべてパケットとしてネットワークに出力し、そのパケットをホスト計算機が処理していた。このネットワークとホスト計算機のインターフェースを汎用のマイクロプロセッサを用いて行なっていたため、多大のオーバヘッドを要していた。EM-Xでは新たにホストとのパケット交換を行なうハードウェアを開発しホスト計算機との通信性能を大幅に向上した。また、I/Oノードをネットワーク上に接続するためのルーティング機構を持たせることにより、80 PE版のEM-Xで最大16 I/Oノードを接続できるようにした。

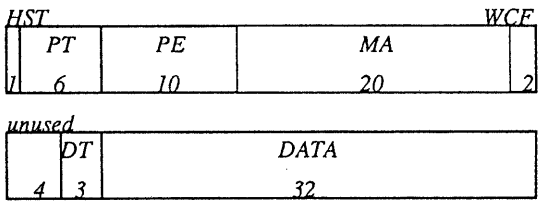
また、要素プロセッサ(PE)の逐次実行性能を向上させ、システム全体の性能向上を図っている。このために、システムクロックの高速化、浮動小数点演算回路の内蔵、命令アーキテクチャの再構成を行なっている。例えば、上に述べたメモリバンド幅の拡張および埋め込み型即値の利用により、レジスタ相対アドレッシングによるメモリ参照命令の実行性能を3倍以上に向上させている。また、EM-4ではパケット出力命令の直後にはレジスタ演算命令しか置くことができないという制限があったが、このような命令スケジューリングに対する制限を削減し、命令コードの最適化を容易にしている。

3 データ転送

要素プロセッサ(PE)間のデータ転送を考える場合、

- 演算パイプラインでのパケット生成、および出力
- ネットワークでのパケット転送
- ネットワークからのパケット取り込み

といった段階が考えられるが、全体のデータ転送性能を向上させるためには、各部の性能を向上させるとともに、どこもボトルネックとならないように各部の性能のバランスを考えることが重要である[4]。



HST: send to host　　MA: matching address
PT: packet type　　WCF: matching condition
PE: destination PE　　DT: data type
DATA:data

図 1: Packet Format of EM-X

3.1 パケット生成・出力

EM-Xではパケットを演算パイプラインで生成している。EM-Xのパケットは図1に示すようなアドレス部1ワード、データ部1ワードの2ワードからなっている。このような非常に単純な構造をしているため、RISCパイプラインで生成することができる。さらに、アドレス部とデータ部の生成を独立に行ない、演算パイプラインからは2ワード幅でパケット出力部に転送している。これにより、1クロックで1パケットの生成が可能で、RISCパイプラインを滞らせることなく、連続してパケットを生成することが可能である。

パケット出力命令は、2つのオペランドをとり、1つはアドレス部の指定に、もう1つはデータ部の指定に使用される。アドレス指定のオペランドと命令内の指定領域の組合せにより、4通りのアドレス指定が行なえるため、柔軟なパケット生成が行なえる。また、アドレス部をレジスタに格納する命令(Load Packet Address)により、continuationの生成が容易である。さらに、現在実行中の命令ブロックと同じパケットベースアドレスを用いることにより、加算などの演算結果を直接パケットとして出力することも可能である。

演算パイプラインで生成されたパケットは、出力バッファに保存された後、演算とは独立にネットワークに出力される。この出力バッファは入力ポートを2ワード幅とすることにより、演算パイプラインの性能を落さないようにしている。出力はネットワーク幅と同じ1ワード幅であり、出力バッファによりパケット出力命令の頻度のバラツキを吸収し、ネットワークの性能を引き出すことが可能である。

3.2 パケット転送

EM-X では、各 PE はサーキュラオメガ網 [1] により結合されている。ネットワークに出力されたパケットは、セルフルーティングにより各ノード上を転送されるため、演算部とは独立にパケット転送が可能である。ネットワークトポロジやデッドロックフリーな機構 [1] などは EM-4 と同じであるが、ルーティング戦略を改良し [4]、トポロジ上最適なルーティングを実現している。例えば 80 PE システムでは、EM-4 のルーティングと比較して、平均 PE 間距離で 5.53 から 5.06 へ、最大 PE 間距離で 11 から 8 へと、2 割から 3 割ほど向上させている。

EM-X のネットワークノードは 3×3 のクロスバースイッチとなっており、実際のパケット転送時間は、上記の静的レイテンシに加えて、出力先が衝突することによる遅延が必要となる。パケット衝突の確率は PE 間距離の指数乗に比例するため、PE 間距離のわずかな短縮もネットワークが混雑している場合には大きな短縮となって現れることもある。

3.3 パケット取り込み

EM-X ではネットワークからやってきたパケットは、いったん入力パケットバッファへ保持され、順に処理される。

EM-4 ではプロセッサ内にバッファを持つことによりパケット取り込みの性能を上げていたが、このプロセッサ内のバッファが溢れると、パケットはメモリ上のパケットバッファに退避される。メモリパケットバッファは命令やデータと同じメモリ上にあるため、メモリバッファの参照には命令フェッチやデータ参照との調停が必要となる。EM-4 ではメモリバッファの参照プライオリティが低かったため、パケットバッファへ入れないパケットがネットワーク上に溢れ、ホットスポットとなり性能低下を招いていた。

EM-X ではこれを防ぐため、メモリバンド幅を広げ、かつ命令フェッチとメモリパケットバッファのメモリアクセスが競合しないようにして、パケット取り込み性能をプロセッサの状況やネットワークの負荷に依存しないようにした。これにより、パケットがネットワーク上に溢れ、システム全体の性能低下を引き起こすことを防ぐことができる。

EM-X では、さらに、パケットに2種類の優先順位をつけることにより、実行順序の制御を行なっている。また、実行中の命令にブロックされることなく、他のプロセッサからのリモートメモリ参照を処理する機構を入力パケットバッファに持たせているが、これらについては4章で述べる。

4 並列性能向上のための機能拡張

4.1 リモートメモリ参照

EM-X では他のプロセッサのリモートメモリを参照するためには、図2のような特殊パケットを用いる。通常のパケット (パケットタイプが0) は待ち合わせアドレスにより実行する命令ブロックを指定するが、特殊パケットでは、パケットタイプにより実行するルーチンを指定し、待ち合わせアドレスは参照すべきメモリアドレスの指定に使用できる。

リモートメモリ書き込みの場合は、データ部に書き込むデータを指定して、パケットを出力する。リモートメモリ書き込みの書き込み完了を確認する必要がない場合には、現在実行中のスレッドを中断する必要がなく、高速なリモートメモリ書き込みが行なえる。書き込みの完了を確認するためには、以下のリモートメモリ読み出しと組み合わ

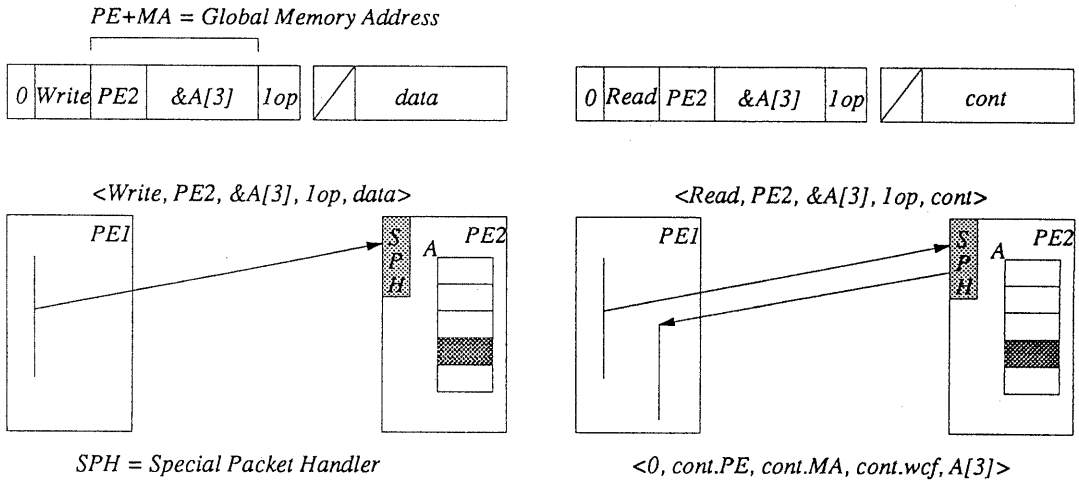


図 2: リモートメモリ参照パケット

せて行なう。

リモートメモリ読み出しの場合は、データ部に読み出したデータを処理するスレッドの continuation を指定してパケットを出力する。EM-X ではパケットデータを受けとることができるのは、スレッドを起動する時のみであるため、現在実行中のスレッドを一旦中断する必要がある。

EM-X では、特殊パケットを用いた 1 ワードのデータ転送のみをサポートしている。このため、ブロック領域の転送にはその領域サイズだけのパケットを必要とするが、EM-X では同一の PE 間では先に出力したパケットが必ず先に到達するというパケットの FIFO 性を保証しているため、効率良く転送を行なうことができる。例えば、メッセージ通信を行なう場合には、まず相手 PE 上にメッセージサイズ分の領域を確保し、その領域に対して 1 ワード転送を繰り返すことによりブロック転送を行ない、最後にメッセージ領域のアドレスを転送すれば良い。個々のパケット転送は高速に行なえるため、十分なスループットを引き出すことが可能であるが、メッセージの出力とメッセージの書き込みの双方に命令実行時間をとられるため、メッセージが大きくなるにしたがってその時間を無視できなくなる。

そのため、EM-X ではリモートメモリ処理をパケット入力処理部で行なうことにより、リモートメモリ書き込みに必要となる時間を隠蔽する機構を備えている。すなわち、パケットがリモートメモリ書き込みの特殊パケットである場合、そのパケットを入力パケットバッファに格納する代わりに、パケットのアドレス部で指定されるアドレスに格納する。この処理は他のメモリ書き込み命令以外の命令実行と並行して実行されるため、メッセージ書き込みに必要となる命令実行時間を削減することが可能となる。

ただし、この処理はパケット処理の FIFO 性、および実行スレッド (強連結ブロック) の持つ排他性を損なうため、使用できる状況を制限する必要がある。また、このような優先処理を行なわず、FIFO 性や排他性を保持するリモートメモリ書き込みの特殊パケットも用意する必要がある。

リモートメモリ書き込みとほぼ同様の処理を入力パケットバッファで行なうことにより、実行中の命令やバッファ内のパケットにブロックされない、リモートメモリ読み出しを行なうことができる。共有メモリ型プログラムを分散メモリ計算機で実行する場合、リモートメモリ読み出しのレイテンシが全体の実行時間に与える影響は大きい。たとえパケット転送自体のレイテンシが短くても、そのリモート PE で他の処理が実行されており、

その終了を待っていると実際のレイテンシは大きくなってしまう。これを防ぐためには各命令ブロックの大きさを調整するなどの命令ブロックのスケジューリングが必要となるが、これは極めて複雑な問題となる。それに対し、リモートメモリ読み出しを入力パケット部で行なえば、そのような見かけのレイテンシを気にすることなくリモートメモリ読み出しを行なえる。

しかし、この場合もFIFO性および排他性を損なうため、読み出しに同期が必要な場合には、他の手段(barrier同期や優先処理しないリモートメモリ読み出しなど)により保証する必要がある。

4.2 入力パケットバッファ

EM-Xではネットワークにおいて、パケット順序のpoint to pointのFIFO性、つまり、あるPEnからあるPEmへ複数のパケットを送った時、PEmへ到着するパケットの順序は、PEnから出力されたパケットの順序に等しいことを保証している。さらに入力パケットバッファでもFIFO制御を行なうことにより、実行順序のFIFO性も保証している。これにより多引数の関数の実行時の引数の到着の保証や、ブロックデータの転送完了の保証を容易に行なえる。

しかし、中には優先的に処理を行なった方が良いパケットも存在する。例えば、負荷を制御するパケットは、実際の処理を行なうパケットよりも優先的に処理を行なわないと、そのデータはまとはずれなものになってしまうか、大きなオーバヘッドを伴うものとなってしまう。

また、分散配置した共有メモリを参照するようなマルチスレッドプログラムでは、リモートメモリ参照の優先順位を他のパケットの処理よりも高くすることにより、動的レイテンシ、つまり他のパケットにより影響を受けないレイテンシにパケット衝突やPEでの処理待ち時間を加えたレイテンシを低く抑えることができる。このような動的なレイテンシは、リモートメモリ参照のプリフェッチやマルチスレッド化によっても簡単には解決されない[5]ため、ハードウェアでサポートしなければならない問題である。

EM-Xではこのような柔軟なパケット制御を可能にするため、異なる優先度を持つ2種類の入力バッファを用意した。1つは通常のユーザパケットを保持するバッファで、もう一つが主に負荷制御などのシステム処理を行なうパケットを保持するバッファである。これらの優先度はパケット出力時にパケットタイプを用いて指定する。優先度は完全に静的で、システムパケットバッファ内のパケットが優先して実行され、ユーザバッファのパケットが実行されるのはシステムパケットバッファが空の時のみである。ただし、各スレッドの実行は排他的であり、優先度が低いパケットの処理を行なっている最中に、優先度が高いパケットが到着しても、割り込まれることはない。各入力バッファ内ではFIFO性を保証し、パケット到着の保証を容易にしている。

また、ユーザパケットバッファに関しては動的な確保を可能にしている。つまり、ユーザパケットバッファをメモリの高位から確保し、関数フレーム領域を含むヒープ領域はメモリ低位から確保することにより、両者の領域が重ならない範囲で自由に拡張可能としておくと、現在確保したバッファが溢れた時に起きるトラップハンドラにより、ユーザパケットバッファを拡張することができる。一方、システムパケットバッファは固定長領域を確保しているため、溢れた時には、トラップハンドラでパケットのユーザパケットバッファへの組替えなどが必要となる。

4.3 I-structure

並列処理を手続き型言語で明示的に記述する際には、処理の流れの同期などを指定するよりは、同期機構を持った変数を参照する方が容易な場合がある。EM-Xではこのような同期機構を持った変数を実現するのにI-structureを用いている。1回書き込み1回読み出しのI-structureは、基本的には、EM-4で行なっている2入力待ち合わせと

パケット出力命令との組合せで実現できる。ただし、各々のパケットは、待ち合わせアドレスとデータ、あるいは待ち合わせアドレスと continuation アドレスというように、1つのリクエストに対して2つのオペランドが必要となる。このため、I-structure の各リクエストを特殊パケットで表すことが必要となる。

EM-X では、あらたに特殊パケットに対しても2入力待ち合わせを可能にすることにより、ほぼ待ち合わせ処理の時間のみで I-structure の処理が行なえる。さらに、右入力が続けてやってくるような待ち合わせミス時には、特定のエラーハンドラを起動することができる。このエラートラップハンドラで、パケットのキューイングなどをソフトウェアエミュレーションすることにより、読み出し・書き込みともにキューイングを行なうような構造メモリ Q-structure[3] も容易に実現することができる。

5 おわりに

現在開発中の EM-X のアーキテクチャ、特にパケット転送性能の改良、リモートメモリ参照の優先実行、2つの優先順位つき入力パケットバッファ、I-structure のサポートといった新たな機能について述べた。

現在 EM-X の要素プロセッサである EMC-Y を設計中である。同時にクロックレベルのシミュレータを開発中で、本稿で述べた改良点や新たな機能について検証するとともに、バッファサイズなどの各種パラメータの評価を行ない、実際のパラメータを決定する予定である。EMC-Y を 1993 年 12 月末に、80 PE からなる EM-X を 1994 年 4 月末に完成させる予定である。

謝辞

本研究を遂行するにあたりご指導、ご検討いただいた太田情報アーキテクチャ部長ならびに研究室の同僚諸氏に感謝いたします。

参考文献

[1] Sakai,S., Yamaguchi,Y., Hiraki,K., Kodama,Y. and Yuba,T. An Architecture of a Dataflow Single Chip Processor, Proc. 16th Annual Symp. on Comp. Arch., (1989), 46-53.

[2] Yamaguchi,Y., Sakai,S., Hiraki,K., Kodama,Y. and Yuba,T. An Architectural Design of a Highly Parallel Dataflow Machine, Proc. of IFIP 89, (1989), 1155-1160.

[3] 佐藤三久, 児玉祐悦, 坂井修一, 山口喜教. 並列計算機 EM-4 における分散データ構造を用いたマルチスレッドプログラミング, 情報処理学会 計算機アーキテクチャ研究会 92-7, (1992), 1-8.

[4] Kodama,K., Koumura,Y., Sato,M., Sakane,H., Sakai,S. and Yamaguti,Y. EMC-Y : Parallel Processing Element Optimizing Communication and Computation, to appear Proc. of ICS'93, (1993).

[5] Hiraki,K., Shimada,T. and Sekiguchi,S. Empirical Study of Latency Hiding on a Fine-grain Parallel Processor, to appear Proc. of ICS'93, (1993).