計算機アーキテクチャ 110−10
設 計 自 動 化 73−10
（1995．1．19）

多次元データ処理用SIMD型並列計算機のアーキテクチャ

金川 英一† 本沢 邦朗††
塩見 彰睦† 今井 正治† 松浦 一教†††

† 豊橋技術科学大学 情報工学系, 豊橋市
†† 東芝 情報・通信システム技術研究所, 青梅市
††† 新日本製鐵 LSI 事業部, 相模原市

本稿では，多次元データ処理用 SIMD 型並列計算機 RIPE/MD のアーキテクチャを提案する．RIPE/MD は，2次元プロセッサアレイ上で多次元データに対する空間並列性を持った局所処理を実行する．RIPE/MD は，実プロセッサ上で複数のプロセスをスレッド単位で並行的に実行するための仮想プロセッサ機構を持つ．これにより RIPE/MD は，処理対象に対する適応性が高くかつ実現性も高い構成となっている．

The Architecture of a SIMD Machine for Multi-dimensional Data Processing

Eiichi KANAGAWA†, Kunio HONSAWA††
Akichika SHIOMI†, Masaharu IMAI†, Kazunori MATSUURA†††

† Department of Information and Computer Sciences,
Toyohashi University of Technology, Toyohashi, 441 Japan
†† Information & Communication Systems Laboratory,
Toshiba, Ome, 198 Japan
††† Semiconductor Division,
Nippon Steel, Sagamihara, 229 Japan

This paper proposes the architecture of a SIMD type parallel processor RIPE/MD for multi-dimensional data processing. RIPE/MD has two-dimensional array structure for multi-dimensional data processing, where spatially parallel local window operations can be executed efficiently. RIPE/3D has a virtual processor mechanism which makes it possible to execute simultaneous parallel-processing of multi-dimensional data on each processor by threads, which makes RIPE/MD easy to adapt to the dimension and size of data, and easy to realize.

1 はじめに

近年，3 次元画像処理に代表される多次元データに対する高速処理の要求が増加している．例えば医療分野では，X 線 CT や MRI による高精度の 3 次元画像に対する処理の有効性を示した研究も発表されている [1][2]．しかし，3 次元画像は 2 次元画像よりも情報量が多いため単純な処理でも計算量が膨大となり，汎用逐次型計算機での高速処理は困難である．この問題を解決するには，画像処理の持つ並列性を活かし，画像の入出力バンド幅を考慮した専用並列計算機の実現が不可欠である [3][4].

並列計算機の持つ重要な要件として処理対象に対する柔軟性がある．並列計算機に柔軟性を持たせるために，仮想的に多数のプロセッサ (仮想プロセッサ) を持つとして処理を行なう方法がある [5]．従来は，この処理をソフトウェアによって実現していた．しかし，ソフトウェアによる処理では仮想プロセッサをエミュレートする際に生じるコンテクスト切換えを高速に行なうことは困難である．また，仮想プロセッサ間のデータ転送を効率よく行なうことも困難である．

本稿では，多次元データに対する空間並列性を持つ処理を高速に実行する専用並列計算機 (RIPE/MD) のアーキテクチャを提案する．RIPE/MD は，2 次元プロセッサアレイ方式を採用した SIMD 型並列計算機である．RIPE/MD は，実プロセッサ上で多数の仮想プロセッサをスレッド単位で高速にエミュレートするためのハードウェア機構を持つ．RIPE/MD は，柔軟性に優れ，入出力バンド幅と演算能力のバランスのとれた実現性の高い並列計算機である．

以下本稿では，まず 2. で RIPE/MD の概要を述べ，3. では RIPE/MD を用いた仮想プロセッサのエミュレーション方法について述べる．次に 4. でコンテクスト切換えの高速化について述べ，最後に 5. でプロセッサ間転送の高速化について述べる．

2 RIPE/MD の概要

2.1 設計方針

以下の設計方針に基づき文献 [6][7] の 3 次元画像処理用並列計算機 RIPE/3D のアーキテクチャを拡張して RIPE/MD のアーキテクチャを設計した．

(1) 入出力と演算のバランス

筆者らは，これまでに 2 次元画像に対する処理を高速に行なう画像処理システムの研究開発を行なってきた [8]．その結果，多くの画像処理プログラムでは画像に対する演算を高速に行なえる一方で，画像の入出力が隘路になる場合が多いことが知られた [9][10]．多次元データに対する処理においても入出力が隘路になる傾向が強いと予測される．そこで，入出力バンド幅と演算能力のバランスのとれた構成とする．

(2) データの大きさや演算量に対する柔軟性

並列計算機の持つ重要な要件として処理対象に対する柔軟性がある．並列計算機でデータ処理を行なう場合は，データの大きさや演算量に応じた台数のプロセッサを用いて処理を行なうことが望ましい．しかし，各プロセッサはハードウェアで実現されているのでプロセッサ数を動的に変化させることは非常に困難である．そこでデータの大きさや演算量の変化に柔軟に対応できる構成とする．

(3) 実現性

一般にプロセッサ数を増やせば演算能力が向上するが，多数のプロセッサを持つ並列計算機の実現を試みようとすると 1 チップ当りのゲート数やピン数，プリント基板上への実装の面から実現が困難となる場合が多い．そこでこれらの制約を考慮して，実現性の高い並列計算機の構成とする．

2.2 RIPE/MD のアーキテクチャ

RIPE/MD は，多次元データに対する空間並列性を持った局所処理を高速に実行することを目的としている．RIPE/MD は，2 次元プロセッサアレイ方式を採用した SIMD 型並列計算機である．

2.2.1 システム構成

RIPE/MD のシステム構成例を図 1 に示す．RIPE/MD は，ホスト計算機のバックエンド・プロセッサとして動作する．プロセッサアレイは外部記憶装置 (External Storage) からの入力データに対して演算を行ない，外部記憶装置へ演算結果を出力する．また，システムコントローラはプロセッサアレイと外部記憶装置，命令メモリ (Instruction Memory) の制御を行なう．

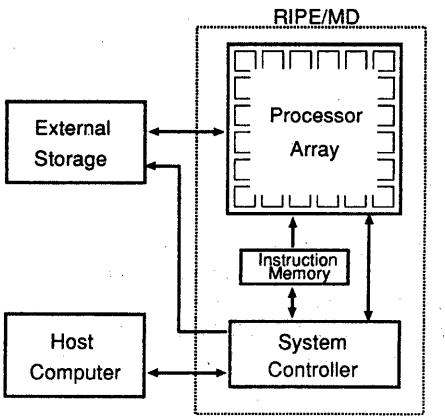

図 1: RIPE/MD のシステム構成例

2.2.2 実プロセッサ

実プロセッサ (PP: Physical Processor) は，プロセッサアレイを構成する基本単位である．PP は図2に示すように，シフトエレメント (SE: Shift Element)，ローカルメモリ (LM: Local Memory)，プロセッシングエレメント (PE: Processing Element) から構成されている．RIPE/MD では，処理の高速化を図るために，入力・演算・出力をパイプライン処理している．

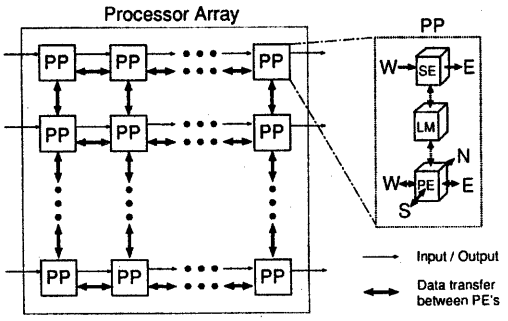

図 2: プロセッサアレイの構造

SE はデータの入出力を担当し，PE は演算を担当する．SE は隣接する 2 台の PP(East and West) の SE に接続され，データをシフト入出力する．PE は，隣接する 4 台の PE(North, South, East and West) と接続されている．

PP の内部構成を図 3に示す．PP は，まず SE からシフト入力したデータを LM へ格納する．次に，PE は LM に格納された入力データに対して演算を行ない，演算結果を LM へ格納する．最後に，格納した演算結果を SE を経由して出力する．SE と PE は LM を共有するため，LM に対してアクセスの競合が生じる．したがって SE が LM へアクセスしている間は，PE は LM に対するアクセスを行なわない．

2.2.3 システムコントローラ

一般に SIMD 型の並列計算機ではプロセッサアレイを制御する制御装置が必要となる．RIPE/MD ではシステムコントローラが制御装置に相当する．システムコントローラは，図 1に示すようにホスト計算機との通信，外部記憶装置の制御およびプロセッサアレイの制御を行なう．

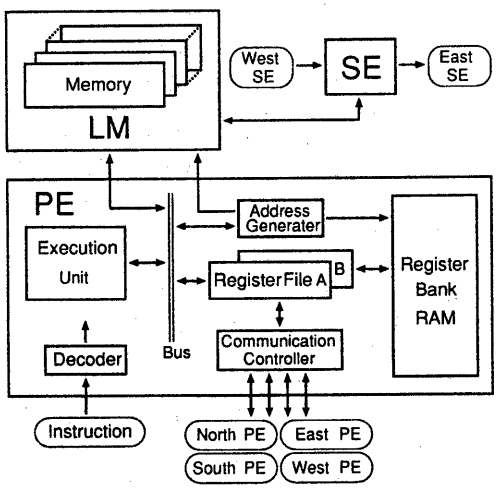

図 3: 実プロセッサの内部構成

3 仮想プロセッサのエミュレーション

3.1 仮想プロセッサ機構

仮想プロセッサ機構とは，データの大きさや演算量に応じて，実在する固定数のプロセッサ上であたかも多数の仮想プロセッサ (VP: Virtual Processor) が実在するかのようにふるまう演算機構である [5]．物理的に実在するプロセッサと仮想プロセッサを区別するために，実在するプロセッサを実プロセッサ (PP: Physical Processor) と呼ぶ．VP の台数は PP の台数よりもはるかに多いことが予想されるので，1 台の PP は複数の VP をエミュレートできる必要がある．

図 4に 1 台の PP で，2×2 台の VP をエミュレートする場合の PP と VP の対応関係を示す．仮想プロセッサ機構を持てば，データの大きさや演算量が変化する場合には VP の台数を変化させて対応することが可能になる．これによって処理対象に対して柔軟に対応できる．

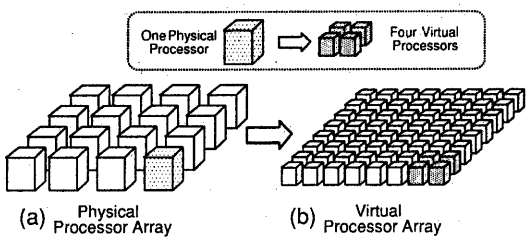

図 4: 実プロセッサと仮想プロセッサの対応関係

3.2 エミュレーションの実行方法

それぞれのVPがそのVPに割当てられたLM中のデータのみを用いて演算を行なう場合には，VPは個々に独立して演算を続けることが可能である．しかし，他のVP(VP')の演算結果を用いて演算を行なう場合には，VP'が演算結果を生成するのを待たなければならない．VP間のデータ転送は，PP間のデータ転送とPP内のデータ転送を用いて実現できる．データ転送については5.で詳細に述べる．

RIPE/MDでは，VP間データ転送で区切られた命令の部分列を1つの処理単位と考えて処理を行なう[1]．この処理単位をスレッド(thread)と呼ぶことにする．PPは，スレッドごとにVPを順次エミュレートする．

RIPE/MDの各PPはSEとPEをそれぞれ1個ずつしか持たないが，LMはVP台数分の容量を持つ．PEは，LMの番地とコンテクストを切換え，演算をスレッド単位で実行することによって複数のVPをエミュレートする．

以下では，1台のPPが，$(l_c \times l_r)$台のVPをエミュレートする場合を考える．VPのPP中での内部位置を表現するために，図5に示すようにVP[i,j]と書くことにする．ここに，[i,j]は，そのPPでエミュレートする仮想プロセッサアレイ内での座標$(0 \leq i < l_c, 0 \leq j < l_r)$である．

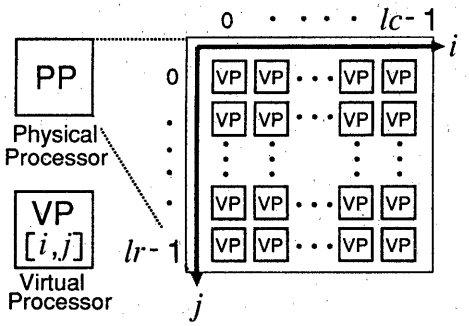

図5: 仮想プロセッサの内部位置の表現方法

RIPE/MDは，空間並列性のある処理を対象としているので，PPがエミュレートするVPの順序は固定でよい．n個のスレッドからなる処理をPPでエミュレートする場合を考える．ここでk $(0 \leq k < n)$はスレッド番号を表し，実行されるスレッドをthread [k]と書くことにする．VPのエミュレーションは，例えば図6の順序で実行できる．図6では，VPの内部位置の左から右，上から下の順序ですべてのスレッドの実行を行なう．

[1]制御構文が区切りになる場合もある

```
1: for k := 0 to (n-1) do
2:   for j := 0 to (lr-1) do
3:     for i := 0 to (lc-1) do
4:       begin
5:         thread[k] に対して
           VP[i,j] をエミュレートする
6:       end;
```

図6: 仮想プロセッサのエミュレーションの順序

3.3 エミュレーションの制御

PPがスレッド単位でVPをエミュレートするためには，各スレッドの開始番地と終了番地(またはスレッドの長さ)の情報が必要である．また，スレッドからスレッドへの分岐がある場合には，分岐先のスレッドを知る必要である．分岐や付加PP間転送の有無を知るために，フラグも必要である(付加PP間転送については5.5.2で述べる)．

RIPE/MDでは，これらのスレッド情報を表(スレッドテーブル)として保持する．スレッドテーブルには，図7に示すように，各スレッドの

- 開始番地
- 終了番地
- 分岐先スレッドまたは付加PP間転送
- フラグ

の情報を格納する．

Start Address	End Address	Branch / Transfer Data	Flags

図7: スレッドテーブル

システムコントローラはスレッドテーブルの情報を用いて，以下のようにして命令流をPPへ供給する．

(1) スレッドの開始番地から終了番地までの命令をPPへ順次供給する．
(2) コンテクスト切換えを行なう．
(3) 同じスレッドの命令をPPへ供給する．(1)~(2)の処理をVPの台数$(l_c \times l_r)$回分繰り返す．
(4) (1)~(3)を次のスレッドに対して実行する．

PPが複数のVPをエミュレートする場合，コンテクスト切換えを効率よく行なうことが重要である．また，プロセス間通信すなわちプロセッサ間転送の効率をあげることが処理の高速化を図る上で重要である．

以下，4. ではコンテクスト切換えの高速化について，5. ではプロセッサ間転送の高速化について述べる．

4 コンテクスト切換えの高速化

4.1 コンテクスト切換えの方法

PP がエミュレートする VP の実行順序は固定してよいと仮定すると，以下で述べる方法を用いることによって高速なコンテクスト切換えが可能である．

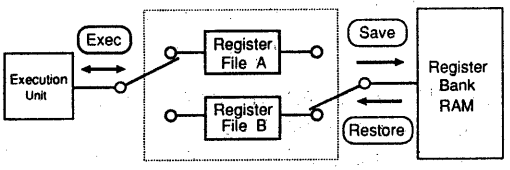

図 8: コンテクスト切換えの高速化

図 8に，PP にレジスタファイルを 2 セット持ちコンテクスト切換えを高速化するための機構の例を示す．レジスタバンク RAM は，レジスタファイルの内容を退避するためのメモリである．同図では，PE 中の Execution Unit が，レジスタファイルのうちの一方 (A) を用いて演算 (Exec) を行なっている間に，もう一方のレジスタファイル (B) の退避 (Save) とレジスタファイル (B) の復帰 (Restore) が行なわれている場合を示している．

図 8は，PP がレジスタファイルを 2 セット持つ例であるが，デュアルポートメモリを用いればレジスタファイルを 3 セット持ち，それぞれのレジスタに対して演算・退避・復帰を順次行なうことも可能である．この方法を採用するとハードウェアコストは上昇するが，より高性能化が図れる．

4.2 命令パイプラインとコンテクスト切換え

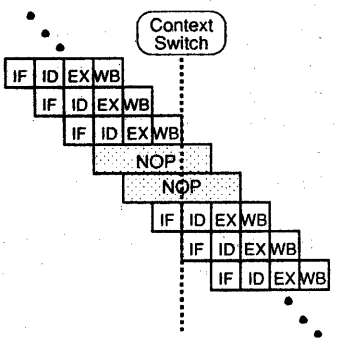

図 9: パイプラインハザードへの対応方法

RIPE/MD の命令パイプラインは，命令フェッチ (IF)・命令デコード (ID)・実行 (EX)・ライトバック (WB) の 4 段からなる．

4.1 で述べたようなコンテクスト切換えの手法を用いると，命令パイプラインに生じるハザードを最小限 (2 サイクル) に抑えることができる．このハザードには，図 9のように，命令パイプラインへ 2 命令分の NOP を入れて対応する．

5 プロセッサ間転送の高速化

5.1 プロセッサ間転送の種類

ある VP と他の VP との間のデータ転送を **VP 間転送**と呼ぶ．また，ある PP と他の PP との間のデータ転送を **PP 間転送**と呼ぶ．さらに，同一の PP 内の 2 つの VP 間のデータ転送を **PP 内転送**と呼ぶ．

VP 間転送は，一般に PP 間転送と PP 内転送の組み合わせによって実現できる．

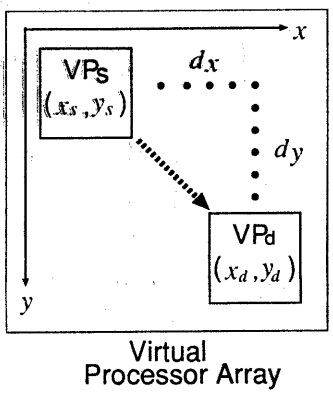

図 10: 記法

以下では，図 10に示すように，2 つの仮想プロセッサ $VP_s(x_s,y_s)$ および $VP_d(x_d,y_d)$ の相対的な位置関係を (dx,dy) で表す．dx，dyは，式 (1)~(2) のように定義される．

$$dx = x_d - x_s \quad (1)$$

$$dy = y_d - y_s \quad (2)$$

5.2 PP 間転送

VP_s(送信元 VP) から (dx,dy) の位置にある VP_d(受信先 VP) への VP 間転送に対応する PP 間転送について考察する．

VP_sの PP での内部位置を $[i,j]$ とする．この時，$|dx+i| \geq l_c$または$|dy+j| \geq l_r$の場合には，VP_sの存在している PP から，VP_dの存在している PP への PP 間転送が必要である．VP_sをエミュレートする PP のVP_dをエミュレートする PP に対する相対位置を (DX,DY) とすると，VP 間転送を実行するためには $(|DX|+|DY|)$

回の PP 間転送が必要となる．DX,DYは，dx,dyとl_c,l_rと VP$_s$の内部位置$[i,j]$を用いて式(3)(4)で表現できる．ここに$\lfloor z \rfloor$はzを越えない最大の整数で，$\lceil z \rceil$はzより小さくない最小の整数である．

$$DX = \lfloor \frac{dx+i}{l_c} \rfloor \quad (3)$$

$$DY = \lfloor \frac{dy+j}{l_r} \rfloor \quad (4)$$

式(3)(4)は式(5)(6)のように変形できる．

$$DX = \begin{cases} \lfloor \frac{dx}{l_c} \rfloor + \lfloor \frac{mod(dx,l_c)+i}{l_c} \rfloor & (dx \geq 0) \\ \lceil \frac{dx}{l_c} \rceil + \lfloor \frac{mod(dx,l_c)+i}{l_c} \rfloor & (dx < 0) \end{cases} \quad (5)$$

$$DY = \begin{cases} \lfloor \frac{dy}{l_r} \rfloor + \lfloor \frac{mod(dy,l_r)+j}{l_r} \rfloor & (dy \geq 0) \\ \lceil \frac{dy}{l_r} \rceil + \lfloor \frac{mod(dy,l_r)+j}{l_r} \rfloor & (dy < 0) \end{cases} \quad (6)$$

ここで，

$$DX_{est} = \begin{cases} \lfloor \frac{dx}{l_c} \rfloor & (dx \geq 0) \\ \lceil \frac{dx}{l_c} \rceil & (dx < 0) \end{cases} \quad (7)$$

$$DX_{opt} = \lfloor \frac{mod(dx,l_c)+i}{l_c} \rfloor \quad (8)$$

$$DY_{est} = \begin{cases} \lfloor \frac{dy}{l_r} \rfloor & (dy \geq 0) \\ \lceil \frac{dy}{l_r} \rceil & (dy < 0) \end{cases} \quad (9)$$

$$DY_{opt} = \lfloor \frac{mod(dy,l_r)+j}{l_r} \rfloor \quad (10)$$

とすると，DX_{est}，DY_{est}は VP$_s$の PP 内での位置$[i,j]$に依存しない項であり，DX_{opt}，DY_{opt}は VP$_s$の PP 内での位置$[i,j]$に依存する項である．すなわち PP 間転送は，どの VP でも必ず実行される PP 間転送と，VP の内部位置によって実行が左右される PP 間転送の 2 種類に区別できることになる．前者を**必須 PP 間転送**と呼び，後者を**付加 PP 間転送**と呼ぶ．

5.2.1 必須 PP 間転送

RIPE/MD では，PP 間転送を PP 間転送命令で実現している．PP 間転送回数と PP 間転送命令は 1 対 1 に対応している．VP 間転送を実現するためには，$(|DX_{est}|+|DY_{est}|)$ 回の必須 PP 間転送が各 VP で必要である．

RIPE/MD では必須 PP 間転送を効率よく実行するために，**裏転送機構**が用いられる．裏転送機構とは，

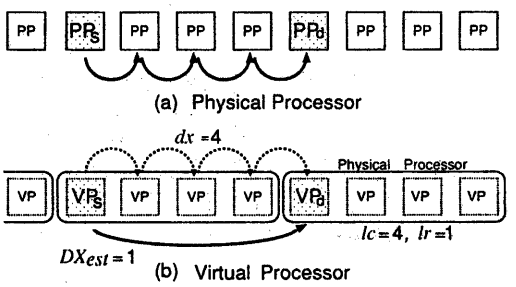

図 11: 必須 PP 間転送

PP 間転送命令と，PP 間転送命令以外の基本演算命令を並列に実行する機構である．

必須 PP 間転送の回数について考える．図 11(a) は，仮想プロセッサ機構を用いない場合 ($l_c = 1, l_r = 1$) の行方向の PP 間転送を示している．それぞれの PP が右方向に 4 回転送を行なう場合には，PP 間転送が 4 回必要である．

図 11(b) は，仮想プロセッサ機構を用いて 1 台の PP が 4 台 ($l_c = 4, l_r = 1$) の VP をエミュレートする場合を示している．それぞれの VP が右方向に 4 回転送を行なう場合 ($dx = 4, dy = 0$) の必須 PP 間転送は各 VP ごとに 1 回となる ($|DX_{est}| + |DY_{est}| = 1 + 0 = 1$)．

仮想プロセッサ機構を用いない場合と用いる場合を比較すると，後者では各 VP ごとの PP 間転送回数は前者の場合よりも減少する．図 11の例では，PP 間転送命令は 1/4 に減少している．このように，プロセッサを仮想化すると，プログラムに対する PP 間転送命令の割合が減少することになる．プログラムに占める PP 間転送命令の割合が減少すると，裏転送機構を用いてより多くの PP 間転送命令を基本演算命令と並列に実行することができる．これによって，データ転送を効率よく実行できる．

5.2.2 付加 PP 間転送

VP 間転送を実現するためには，$(|DX_{est}| + |DY_{est}|)$ 回の必須 PP 間転送の他に $(|DX_{opt}| + |DY_{opt}|)$ 回の付加 PP 間転送が必要である．式 (8)(10) より，付加 PP 間転送のために各 VP で必要な PP 間転送はたかだか 2 回であることが知られる．

図 12に 1 台の PP が，16 台 ($l_c = 4, l_r = 4$) の VP をエミュレートする場合を示す．この時にそれぞれの VP が右方向に 1 回 VP 間転送を行なう場合 ($dx = 1$, $dy = 0$) を考える．付加 PP 間転送が行なわれるのは，網目の入っていない 4 台の VP($i = 3$) だけである ($|DX_{opt}| + |DY_{opt}| = 1 + 0 = 1$)．網目の入った 12 台の VP($0 \leq i \leq 2$) では PP 間転送を行なう必要はない ($|DX_{opt}| + |DY_{opt}| = 0$)．

したがって，図 12の例の場合には，付加 PP 間転送を行なう 4 台の VP と，付加 PP 間転送を行なわない

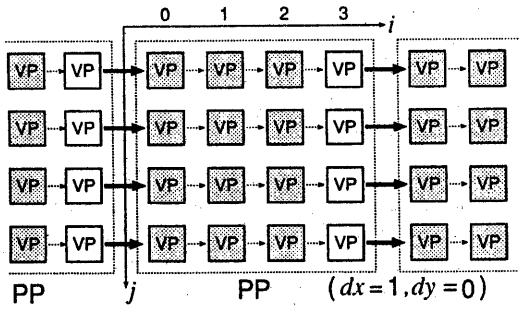

図 12: 付加 PP 間転送

12 台の VP で，供給する命令列を変化させる必要がある．VP へ供給する命令を変化させるための機構を検討した．

(1) PP 間転送命令を NOP に変換する方法

付加 PP 間転送が不要な VP には，システムコントローラが PP 間命令を NOP へ変換して命令を供給する．システムコントローラが命令メモリから付加 PP 間転送ための命令を読み出し，VP に PP 間転送命令としてを供給するか，あるいは NOP を供給するかを判断する．

(2) 必要な PP 間転送命令だけを生成する方法

付加 PP 間転送の必要な VP をエミュレートする場合は，システムコントローラが VP に付加 PP 間転送に対応する PP 間転送命令を供給する．付加 PP 間転送の不要な VP をエミュレートする場合には，PP 間転送命令の供給を行なわない．

(1) の方法は，付加 PP 間転送の不要な VP にも PP 間転送命令の代わりに NOP を供給するので実行サイクル数が増大する．(2) の方法では，不要な NOP 命令を供給しなくてよいので，実行サイクルは増大しない．したがって (2) の方法を採用する．(2) の方法を実現するための機構を図 13に示す．付加 PP 間転送が必要な VP には，命令メモリからの命令読み出しを一時中断し，システムコントローラが PP 間転送命令を生成しプロセッサへ命令を供給する．システムコントローラは，図 7に示したように，付加 PP 間転送を実行するための情報をスレッドテーブルに保持する．このハードウェアを設けることにより，データ転送を効率よく実行できる．

5.3 PP 内転送

コンテクスト切換えが発生する時，4.1 に示した方法を用いて，レジスタファイルのデータをレジスタバンク RAM へ退避する．また退避されたデータは，レジスタバンク RAM からレジスタファイルへ復帰する．

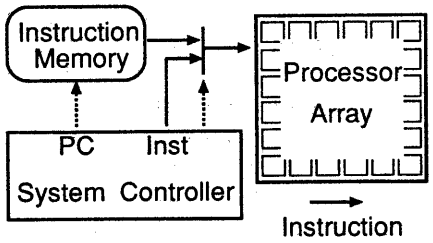

図 13: 付加 PP 間転送機構

PP 内転送は，レジスタファイルの中のプロセッサ間転送用レジスタ (CR: Communication Register) の復帰時に行なう．

図 14に，1 台の PP が 4 台 ($l_c = 4, l_r = 1$) の VP をエミュレートする場合を示す．それぞれの VP が右方向に 1 回 VP 間転送を行なう場合 ($dx = 1, dy = 0$) の PP 内転送の実行例を示す．それぞれの VP の CR には図 14(a) のようなデータが入っているとする．

この場合に PP 間転送を実行するのは VP[3,0] だけである．そこで，VP[3,0] の CR のデータを 1 つ右の PP の VP[3,0] の CR へ転送する．転送を行なった結果を図 14(b) に示す．続いて，図 14(c) のようにレジスタバンク RAM へレジスタの退避を行なう．最後に図 14(d) のように，レジスタバンク RAM のアドレスを操作して CR にデータを復帰する．

このような機構を持つことにより，PP 内転送がレジスタの復帰時に実行できるので，データ転送処理の高速化が図れる．

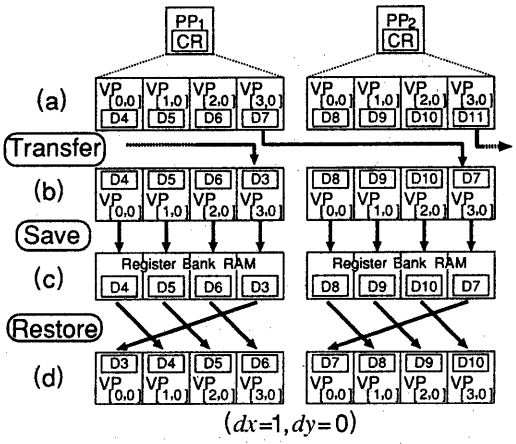

図 14: PP 内転送の実行例

6 むすび

本稿では，多次元データに対する処理を高速実行する多次元データ処理用並列計算機 RIPE/MD のアーキ

テクチャを提案した．RIPE/MDは，画像処理や行列演算や有限要素法など空間並列性のある局所処理に応用可能である．

RIPE/MDは仮想プロセッサ機構を持ち，コンテクスト切換えとプロセッサ間転送を効率的に行なえる．また，仮想プロセッサ機構を持つことで，RIPE/MDは柔軟性に優れ，入出力バンド幅と演算能力のバランスのとれた現実性の高い並列計算機であると予測される．

今後はRIPE/MDの詳細設計を行ない，性能を見積もる予定である。

謝辞 日頃から御指導頂く中京大学の情報科学部の長谷川純一教授，豊橋技術科学大学知識情報工学系の山本眞司教授に深謝します．また，討論して頂いた豊橋技術科学大学VLSI設計研究室の諸兄および貴重な御意見を頂いた並列計算システム研究室の貴島寿朗氏，渡邊 誠也氏に感謝致します．

参考文献

[1] 長谷川純一，森健策，鳥脇純一郎，安野泰史，片田和廣：3次元ディジタル画像処理による胸部連続CT像からの肺がん候補領域の自動抽出，電子情報通信学会論文誌，Vol. J76-D-II, No. 8, pp. 1587–1594 (1993).

[2] 赤松茂男，山本眞司：3D-MRI画像からの脳内組織自動抽出，電子情報通信学会春季大会講演論文集，第7巻，pp. 7–334 (1992).

[3] 前田明：画像処理マシン，情報処理，Vol. 28, No. 1, pp. 19–26 (1987).

[4] 前田明：VLSI画像処理プロセッサ，情報処理，Vol. 31, No. 4, pp. 480–484 (1990).

[5] Philip J. Hatcher and Michael J. Quinn: Data-Parallel Programming on MIMD Computers, The MIT Press(1991).

[6] 松浦一教: 3次元画像処理用並列計算機RIPE/3Dの構成方法に関する研究，豊橋技術科学大学修士学位論文 (1994).

[7] 古沢浩美，"RIPE/3D上での3次元空間フィルタの実現とその評価"，豊橋技術科学大学修士学位論文 (1994).

[8] 本沢邦朗，佐藤淳，冨田穣太，今井正治，長谷川純一：超高速画像処理システムRIPEのアーキテクチャ，電子情報通信学会技術研究報告，Vol. CPSY89-12, pp. 43–50 (1989).

[9] 本沢邦朗，平岡久和，松浦一教，金川英一，中西弘泰，塩見彰睦，今井正治，長谷川純一：試作結果に基づく高速画像処理用並列処理計算機RIPEの性能評価，情報処理学会研究会資料，Vol. CV93-83-6, pp. 41–48 (1993).

[10] Honsawa, K., Hiraoka, H., Imai, M. and Hasegawa, J.: Configuration and Performance of RIPE: a Parallel Processor for Very High Speed Image Processing, *Asian Conference on Computer Vision(ACCV'93)*, pp. 700–704 (1993), Osaka, Japan.