情報処理学会論文誌　数理モデル化と応用　Vol. 2　No. 3　64–74 (Dec. 2009)

超並列プロセッサコアにおける PE 間データ転送効率の改善

中野光臣[†1]　弘田澄男[†1,*1]　兒玉章宏[†1]
飯田全広[†1]　末吉敏則[†1]

多数の PE（Processing Element）を持つ並列プロセッサにおいて，PE 間を接続する相互結合網は重要な構成要素であり，データ転送処理はアプリケーション実装において性能に影響を与える．しかし，応用によっては並列度が低下し，逐次処理が増加することで性能低下の要因となる．本研究では，株式会社ルネサステクノロジ社が開発した組み込み向け SIMD 型アクセラレータである MX コアにおいて，集中制御型の PE 間データ転送および分散制御型のデータ転送について配線構造の定式化を行う．また，PE 間データ転送問題を 2 種類の部分問題に分け，ルーティング手法を提案し評価する．規則的な転送処理および複雑なデータ転送が必要な処理を用いてルーティング手法の評価を行った結果，集中制御，分散制御の両制御においてそれぞれ効果的なデータ転送制御が行えたことを示した．また，複雑なデータ転送処理の評価において，分散制御の転送回数は集中制御時より 90%削減することができた．

Improvement of Data Transfer Efficiency between PEs in a Massively Parallel Processor

Mitsutaka Nakano,[†1] Sumio Hirota,[†1,*1]
Akihiro Kodama,[†1] Masahiro Iida[†1]
and Toshinori Sueyoshi[†1]

An interconnection network between PEs is an important component in parallel processors. The data transfer processing with it has a serious influence on the performance of applications. However, the performance deteriorates when an application has a only few parallelism of the data transfer because sequential processing increases. MX-Core developed by Renesas technology Corp., is a massively parallel SIMD type accelerator. We formulate the data transfer control of the MX-Core both of SIMD type and of MIMD type. The data transfer problem is divided into two kinds of partial problems, and we propose both solutions of them. Our proposal techniques are able to perform effective data transfer control by either two types. As an evaluation result, the MIMD control was reduced the number of transfer counts by 90% from SIMD control.

1. はじめに

多数の PE（Processing Element）を持つ並列プロセッサにおいて，PE 間を接続する相互結合網は重要な構成要素であり，データ転送処理はアプリケーション実装において性能に重要な影響を与える[1)]．そのうえ，一般に限られた配線領域を効率良く使うことは消費エネルギーの側面からも重要である．また，PE 間データ転送処理を実装する場合，多数の PE において手動で転送経路の最短となる組合せを導出するには膨大な時間がかかる．自動化のためには，PE 間データ転送における問題点の定式化と効率の良いルーティングアルゴリズムが必要である．

我々は，株式会社ルネサステクノロジが開発したマトリックスアーキテクチャによる超並列 SIMD（Single Instruction/Multiple Data）型プロセッサコアである MX コア[2),3)] に注目している．MX コアは 1,024 個の 2 ビット PE と制御回路，I/O からなり，各 PE には 2 本の独立した 512 ビットデータレジスタを持つ．MX コアは組み込み向けのアクセラレータとして開発され，それを搭載した MX-SoC[4)] が発表されている．

MX コアは SIMD 型プロセッサコアであるため，性能を出すには処理の並列性の抽出および PE 間のデータ転送の効率化が必要不可欠である．現状では MX コアのデータ転送の実装はほぼ手動で行われている．したがって，一般の並列プロセッサと同様に転送経路の組合せの探索に時間がかかり，結果として最適な経路設定ができないという問題をかかえている．本稿では MX コアへのアプリケーション実装において重要となる PE 間配線構造，およびデータ転送効率改善手法について議論する．以下，2 章で MX コアのアーキテクチャの詳細と処理方式を示す．3 章では PE 間データ転送の特性と問題点について述べ，4 章では PE 間相互配線の定式化および提案するデータ転送手法について述べる．5 章で提案手法の評価方法を示し，6 章で評価結果および考察，7 章で関連研究を述べ，8 章でまとめる．

†1 熊本大学大学院
Graduate School of Kumamoto University
*1 現在，株式会社日立製作所
Presently with Hitachi Ltd.

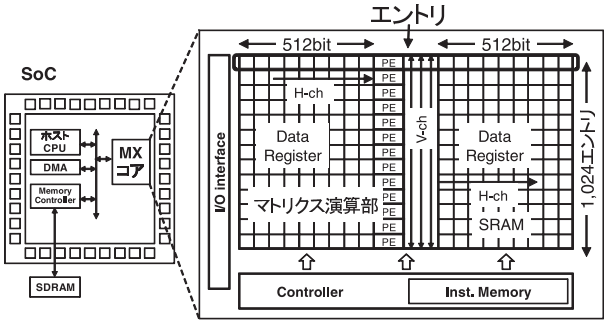

図 1　MX コアの構成
Fig. 1　Architecture of MX-Core.

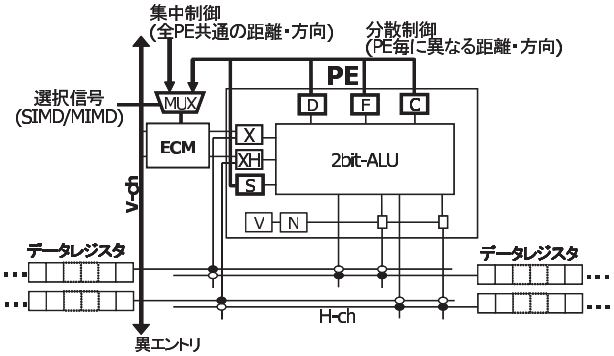

図 2　PE の構成
Fig. 2　Architecture of processing element.

2. MX コア

2.1 MX コアの構造と処理機構

図 1 に MX コアの構成を示す．MX コアは，主に PE とデータレジスタで構成される．データレジスタは，PE の両翼に 512 ビットずつの SRAM が配置される．これを MX コアの基本構成単位とし，PE とその両翼 512 ビットのデータレジスタをあわせてエントリと呼ぶ．MX コア全体で 1,024 エントリあり，これらの PE を SIMD 命令で実行することにより並列処理を行う．また，各 PE 間のデータ転送は，垂直方向のデータ通信用配線（V-ch）を使用する．V-ch は，上下に 1，4，16，64，256 といった 4 のべき乗の距離離れた PE 間で並列にデータ転送することが可能である．これらの SIMD 型処理は，すべて図中下部のコントローラ部により制御される．MX コアは 1 つのチップ上にホスト CPU となるメインプロセッサおよび DMAC（Direct Memory Access Controller），メモリコントローラ，SDRAM と合わせて実装され，様々なアプリケーションにおいてアクセラレータとして機能する．

2.2 MX コアの PE 構成と演算方法

図 2 に MX コアの PE の詳細を示す．PE は，1 ビットフラグ用レジスタ S，F，D，C，V，N，アキュムレータ X，XH および 2 ビット単位で処理できる ALU からなる．また，X，XH は PE 間通信を制御している ECM（Entry CoMmunicator）を介して V-ch とも接続されている．1 回の演算は，データレジスタから X，XH へデータを読み出し，2 bit-ALU で演算を行い，演算結果をどちらかのデータレジスタへ書き込む．C はキャリーフラグ，S，F および D は 2 ビット演算における各種処理レジスタとして動作する．各 PE に設けられた V と N は Valid フラグとして，H-ch，V-ch のデータ転送や演算時のマスクの指定に使用する．

PE 間データ転送において，コントローラからの一律制御である集中制御では，PE ごとに転送する距離が異なる場合，逐次的に転送を繰り返す必要があり，並列度の低下が性能低下の要因となる．そこで，我々はこの問題点の解決策として，各 PE が個々に制御可能な分散制御手法を提案している[5]．V-ch の転送距離に水平方向を加えた 11 通りの通信距離を表現するため，4 ビットのレジスタが必要となる．データ転送を行う際，ALU は動作しないことから，演算用のレジスタとして各 PE 内に設けられた S，F，D，C レジスタは使用されない．このことに着目して，これら 4 つのレジスタを通信先の距離と方向を制御する制御用レジスタとして利用する．図 2 中の太線部が分散制御部である．MUX はコントローラにより生成される SIMD/MIMD 選択信号に従い，コントローラからの制御あるいは各 PE の制御レジスタの一方を選択する．MUX で選択された信号に従い ECM で PE 間データ転送を制御する．このように，通信制御用ビットを格納するレジスタとして既存のレジスタを利用することで，追加リソースを極力抑えて分散制御における PE 間データ転送を実現している．追加リソースは 1PE あたり MUX と制御用レジスタからの配線であり，トランジスタの増加量は約 4%である．MX コアは 1,024PE を持つので，それらすべてに追加することになる．MX コアは 90 nm 7Cu CMOS Low Power プロセスにおいてコアのチップサイズが 2.26 mm^2 と小面積であるので，追加リソースによって回路規模が増大しても実用的で

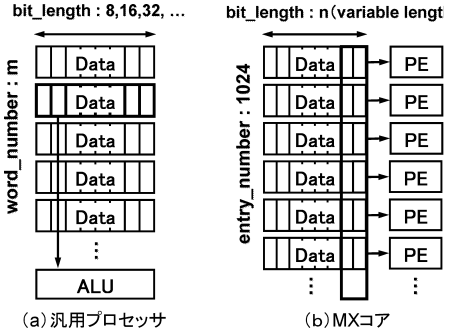

図 3　演算方法
Fig. 3　Operation method.

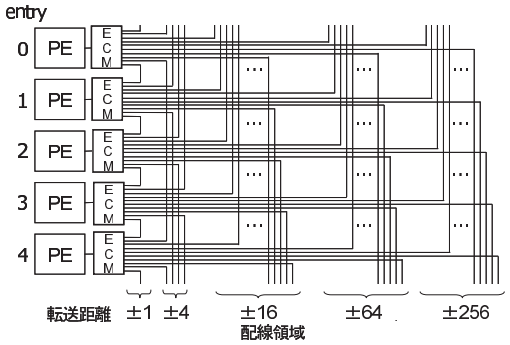

図 4　V-ch の構成
Fig. 4　Structure of V-ch.

ある．

図 3 に，MX コアと汎用プロセッサの演算方法の違いを示す．図 3 (a) のように汎用プロセッサの演算は逐次的にデータの数繰り返す必要がある．一方，MX コアは図 3 (b) に示すように 1,024 個のデータを同時に 2 ビットずつ演算を行う．このため，1 つのデータに対しては複数サイクルを要するが，超並列構造を生かした処理で総演算サイクル数を小さくできる．

3. PE 間データ転送

本章では，MX コアの集中制御および分散制御の PE 間データ転送の特性と問題点について述べる．

3.1 MX コアの PE 間データ転送

図 2 に示したように，MX コアの PE-データレジスタ間は H-ch により，PE 間は V-ch によりそれぞれ配線されている．H-ch は主に PE で演算するデータをレジスタから転送する配線である．また，V-ch は PE 間のデータ転送処理を行うための配線であり，これらの配線を使用してデータ転送処理を実現する．データ転送処理はアプリケーションの一部であり，データ転送処理も処理実装時に静的に実装を行う．**図 4** に PE 間の配線構造を示す．図 4 下部は，MX コアが転送距離 1，4，16，64，256 の配線領域を持つことを示している．各エントリは PE と PE 間データ転送を制御する ECM（Entry CoMmunicator）を持ち，データは ECM を経由してデータ転送を行う．PE はそれぞれ隣接した PE，4，16，64，256 離れた PE と直接接続され，各 PE は 1 度にこれらの距離離れた PE とデータ転送を行うことができる．PE 間データ転送の処理手順は以下のとおりである．

(1)　データレジスタから PE 内の X，XH レジスタへ H-ch を介してデータを読み出す．
(2)　集中制御または分散制御に従い，PE 間転送回路および V-ch を介してデータを転送先の PE 内のレジスタ X，XH へ転送する．
(3)　PE 内のレジスタ X，XH 内のデータを H-ch を介してデータレジスタへ書き込む．

これらの手順をデータのビット数分繰り返しデータ転送を終了する．また，任意の距離のデータ転送を行う場合は，上述の配線の組合せで実現する．

実際に V-ch を使用した通信を行う場合の経路設定例を**表 1** に示す．経路設定において，0 から 1,024 エントリまでの PE を縦 1 列に並べた状態を初期状態とし，各エントリの転送先エントリを目標状態とする．初期状態から目標状態のエントリまでの距離を転送距離と呼ぶ．初期状態から見てエントリ番号が増える方向を正，減る方向を負として表記する．また，転送距離を実現するための V-ch の組合せを転送処理要素とする．表 1 において初期状態が 1 エントリ，目標状態が 6 エントリの場合，転送距離は +5 となり，転送処理要素は V-ch の組合せから $\{+1, +4\}$ を決定する．このように，各エントリが目的状態への転送距離から通信距離要素を設定し，PE 間データ転送を行う．

集中制御による PE 間データ転送の例として，**図 5** にデインタリーブ処理を実行した際の PE 間データ転送の様子を示す．デインタリーブ処理とは，垂直方向に並んだデータに対し，偶数番目のデータを上部へ，奇数番目のデータを下部へと整列させる処理である．四角の中の数値はデータの番号を示し，src と dst は初期データと整列後のデータ用の領域，tmp は

表 1　PE 間データ転送の経路設定例
Table 1　Communication example.

初期状態	目標状態	転送距離	転送処理要素
0	1	+1	+1
1	6	+5	+1，+4
2	0	−2	−1，−1
4	3	−1	−1

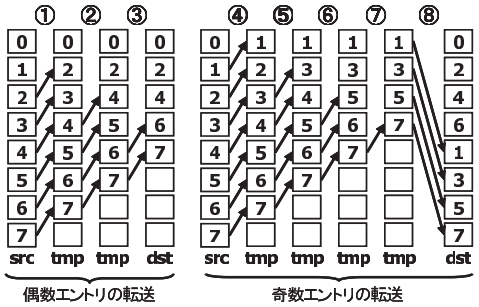

図 5　集中制御によるデインタリーブ処理
Fig. 5　De-interleave by the SIMD control.

一時保存用の領域である．丸数字の順に処理を進める．図 5 に示すように，偶数データの転送と奇数データの転送を一時保存用の領域を利用し別々に行い，最後に 1 列にまとめる．この場合の転送回数は 8 回となる．

3.1.1　集中制御におけるデータ衝突

MX コアは全エントリが一律に動作することで，処理の高速化を図っている．よって，データ転送においても全エントリが同じ動作を行う．もし，あるエントリに前処理のデータがあったとしても，全エントリが同じアドレスにストアするためデータが上書きされてしまう．したがって，そのデータが有効であったとしても，転送順序を考慮しただけでは必要なデータを失うことがある．表 1 の通信例を用いて，データ衝突の具体例を示す．**図 6** に表 1 における PE 間データ通信の様子を示す．

1 回目の転送においてデータ A，B を転送し，次に 2 回目の転送でエントリ 2 のデータ C に対して上方向に転送距離 1 の転送を行うとする．しかし，エントリ 2 の転送先に 1 回目に転送を行ったデータ A が存在しているため，上書きを回避するデータ転送処理を行う

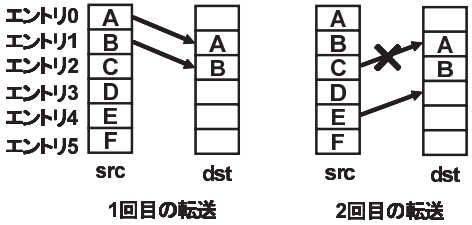

図 6　データ衝突の例
Fig. 6　Example of data collision.

必要がある．その場合，エントリ 2，4 を 1 度に転送できず転送回数が増加してしまう．

3.1.2　分散制御におけるデータ衝突

集中制御時は，1 つ前の処理において格納したデータが次の処理で上書きされるデータ衝突を回避することが必要であった．一方，データ転送の制御を分散することにより，各 PE が転送距離を選択でき，毎回全 PE が動作するため，集中制御のようなデータ衝突は起こらない．しかし，各 PE が転送距離を選択することで，複数の PE の転送先が同じ PE になる場合がある．このような，分散制御によって起こる異なるデータ衝突を回避してデータ転送を行う必要がある．動作する PE の数が多くなるほどこのデータ衝突が起こる確率が増し，回避することが困難となる．そこで，次章より本章で述べた集中/分散制御の特性や問題点を考慮に入れ，MX コアのデータ転送処理問題を定式化し，データ転送の効率化手法を提案する．

4. 提 案 手 法

本章では，PE 間のデータ転送回数を効率化する手法を述べる．データ転送では，前節のデータ衝突を回避する転送順序を考える必要がある．我々は PE 間データ転送の問題を 2 つの問題に分割し，それぞれについて解法を示す．第 1 の問題は，全エントリが初期状態から目標状態へデータを転送するために必要な転送処理要素を決定し，全体で最少の転送回数になる組合せを求める問題である．これを転送処理要素設定問題と呼ぶ．第 2 の問題は，転送処理要素設定問題によって得られた転送の組合せについて，データ衝突を回避し，全エントリのデータ転送が最少回数で終了するように転送処理要素の順序を決定するスケジューリング問題である．次節より，これらの問題の解法を集中制御，分散制御それぞれについて述べる．

4.1　集中制御転送処理要素設定問題の定式化

まず，対象とする転送処理要素設定問題を次のように定式化する[6]．

$$D_n = 1x_{n1} + 4x_{n2} + 16x_{n3} + 64x_{n4} + 256x_{n5} - 1x_{n6} - 4x_{n7} - 16x_{n8} - 64x_{n9} - 256x_{n10} \quad (n = 1, \cdots, n) \tag{1}$$

$$x_{nm} \geq 0 \tag{2}$$

式 (1) の D_n は，n エントリのデータが目標状態へ達するまでの転送距離を表している．各項の係数は，データ転送を行う際に選択することができる転送距離を表しており，各項の変数 x_{nm} は各転送処理要素の個数を示す．m は，式 (1) におけるデータ転送距離の種類を示す．よって，式 (1) は各エントリの転送距離を転送処理要素の組合せに分解することを意味する．式 (2) は，各転送処理要素の個数 x_{nm} が 0 以上であることを示している．式 (1)，式 (2) は配線構造の制約条件なので，集中制御，分散制御共通である．

$$\text{minimize} \quad z = \sum_{m=1}^{10} y_m \tag{3}$$

$$\text{subject to} \quad y_m = \max\{x_{1m} \ldots x_{nm}\} \, (m = 1, \cdots, 10) \tag{4}$$

式 (1)，式 (2)，式 (4) は転送処理要素設定問題の制約条件である．式 (4) は，全エントリのデータが目標状態まで到達するために各転送処理要素を何回行うかを示している．式 (3) は本問題の目的関数である．データ転送を行う全データが目標状態へ到達するまでに処理する各通信要素の個数 y_m の総和を z とする．転送処理要素設定問題の目的は z を最少とする x_{nm} の値を割り当てることである．次節でこの転送処理要素設定問題の解法について述べる．

4.2　転送処理要素設定問題の最適解法

本稿において，あるエントリが目的状態に到達するとき使用する転送処理要素の組合せを転送パスと呼ぶ．各データの転送パスは指定できる転送処理要素の組合せで幾通りも考えられる．その組合せの中で目標状態に最少回数で到達できる転送パスを，そのエントリにおける最短転送パスとする．

集中制御における転送処理要素設定問題の制約について説明する．すべてのエントリにおいてデータ転送を行う場合，まず各エントリで最短パスを計算する．各エントリの最短パスの組合せによって全体の転送処理を決定し，集中制御によって処理を行う．このとき，集中制御では同距離同方向のデータ転送のみ一律に動作することができる．したがって，最短パスの組合せが集中制御にあわない場合，逐次処理によって全体の転送処理回数が増加する．また，最短パスどうしの組合せでなくても，同距離同方向のデータ転送を多く含むパスの組合せでは，集中制御により転送回数が抑えられる．以上から，最短パスのみでは不十分であり，最短転送パス以外のパスも考慮しなければならないことが分かる．そこで，転送処理要素設定問題を解決するために全探索法を用いる．

表 2　距離が 2 のときの転送処理要素
Table 2　Communication processing element when distance is two.

	転送パス	転送回数
最短	+1, +1	2
2nd	+4, −1, −1	3
3rd	+1, +1, +1, −1 +1, +1, +4, −4 +1, +1, +16, −16 +1, +1, +64, −64 +1, +1, +256, −256	4
4th	+1, +4, −1, −1, −1 +4, +4, −1, −1, −4 +4, +16, −1, −1, −16 +4, +64, −1, −1, −64 +4, +256, −1, −1, −256	5

まず，転送を行う全エントリが目標状態に達するまでの最短転送パスを組み合わせる．この組合せから同時に転送処理を行うことができる転送パスをグループ化し，転送回数を 1 回としてカウントする．1 つの組合せに対して全エントリが目標状態に達するまでの転送回数を算出し，暫定最少回数として設定する．次に，各データの最短転送パス以外の組合せを行い，最短パスどうしの組合せと同様の方法で転送回数を算出し，暫定最少回数の更新も行う．順次，各データに対し最短パス以外から最小の転送回数の組合せを算出し，これを移動処理設定問題の最適解とする．ここで，転送の組合せを行う際，効率良く探索を行うために，探索範囲を省く方法を述べる．

表 2 は，転送距離を +2 としたときの転送処理要素の組合せと転送回数を示している．表 2 の最短，2nd，3rd，4th はそれぞれ転送パスの短さの順である．最短が転送回数が一番少ない組合せであり，以降，転送回数が順次増える．表 2 において，3rd はすべて最短の転送パスを含んでいることが分かる．つまり 3rd を転送に用いた場合，必ず +1 を 2 回行うため，最短パスよりも転送回数が少なくなることはない．また，転送回数を増やすことは

データ衝突が発生する可能性が高くなる．そこで，全探索をする際，包含されている転送処理要素のみから構成される転送パスを用いる．同様に，4th も 2nd をすべて含んでいるため，探索しない．この例では，最短と 2nd のみで組合せを行い転送パスを求める．このように，転送回数の多いパスが転送回数の少ないパスを包含している場合と，4th 以降の転送パスは必ず 3rd までの転送パスを含んでいるので探索を行わない．以上のことを考慮し，各エントリの転送パスを候補の中から組み合わせ，全データが目的状態に到達するまでにかかる転送回数の最少値を求める．

4.3　集中制御スケジューリング問題の最適解法

転送パスの処理順序決定のアルゴリズムとして深さ優先探索[7),8)] を用いる．深さ優先探索は，探索木のなかでもより深いノードを優先的に探索し，バックトラックを行いながら探索するアルゴリズムである．また，深さ優先探索は分枝限定法と併用することができる．そのため，深さを転送回数とし，最初に全データが目標状態に達するまでの転送回数を下界値とすることで，下界値よりも良い解が得られないと分かった場合，探索を省くことができる．探索範囲を限定でき，最少転送回数で終わる転送処理順序を効率的に見つけることができるため深さ優先探索を用いた．**表 3** の転送例を用いて，転送パスのスケジューリングの具体例を示す．表 3 において，J_1 と J_2 は並列に処理できるので，これをグループ化し，G_1 とする．エントリ 0 は$+1(J_1)$ の転送を 1 回行い目標状態に到達する．エントリ 1 は$+1(J_2)$ と$+4(J_3)$，の転送をそれぞれ 1 回ずつ行い，エントリ 2 は $-1(J_5)$ の転送を 2 回行うことで目標状態に到達する．この最短転送パスを用いて，J_1，J_2，J_3，J_4，J_5，G_1 の全探索を行い，全エントリのデータが目標状態に到達するまでの最少転送回数を求める．全探索を行うときの探索木を**図 7** に示す．

探索木のノードは転送パスのインデックスを表し，各インデックスに対応した転送処理を行う．親ノードは一番最初に転送を行う転送パスのインデックスを表しており，子ノードは未処理の転送パスのインデックスを表す．また表 3 の例で$+1(J_1)$ と$+1(J_2)$ は並列処理を行うことができるが，あえて逐次処理にすることも考慮に入れ，最短パスの探索を行う．表 3 の転送処理要素を全探索し，全エントリのデータ転送が完了するまで探索を続ける．すべての転送順序を用いて転送処理を行う際，最初に全データが目標状態まで到達したときの転送回数を下界値とする．繰り返し探索を行い，全データが目標状態に達するまでの転送回数が下界値よりも少なかった場合，その値を転送回数の下界値として再設定し探索を行う．下界値が，衝突を考慮せず目的状態に到達する転送回数と同値だった場合はその下界値を最適解とし，探索を終了する．下界値よりも転送が増える場合は枝刈りを行う．探索を終了した時点での下界値を最小転送回数として定める．

表 3　転送例
Table 3　Movement example.

初期状態	目標状態	転送距離	転送処理要素
0	1	1	$+1(J_1)$
1	6	5	$+1(J_2)$，$+4(J_3)$
2	0	-2	$-1(J_4)$，$-1(J_5)$

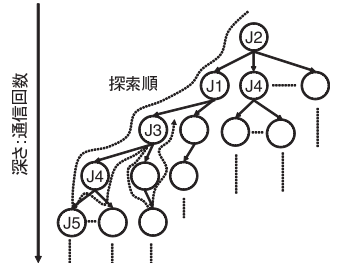

図 7　深さ優先探索によるスケジューリング
Fig. 7　Scheduling by depth priority search.

4.4　分散制御転送処理要素設定問題の定式化

分散制御における転送処理要素設定問題を次のように定式化する．

$$\text{minimize} \quad z = \max\{y_1, y_2, \cdots, y_n\} \tag{5}$$

$$\text{subject} \quad \text{to}$$

$$y_n = \sum_{m=1}^{10} x_{nm} \, (n = 1, \cdots, n) \tag{6}$$

集中制御の場合と同様に式 (1)，式 (2)，式 (6) が転送処理要素設定問題の制約条件である．分散制御において異なる部分は目的関数である式 (5) と制約条件の式 (6) となる．式 (6) は，各エントリにおけるデータが目標状態までに到達するために各距離の転送処理を何回行うかを示している．式 (5) は分散制御における目的関数である．データ転送を行う各エントリのデータが目標状態に到達するまでに処理する各転送要素の個数 y_m の中から最大の処理数を z とする．転送処理要素設定問題の目的は z を最小とする x_{nm} の値を割り当てることである．次節で分散制御における転送処理要素設定問題の解法について述べる．

4.4.1　転送処理要素設定問題の最適解法

各エントリの転送パスの組合せから，z を最小にする x_{nm} の組合せを探索する．ここで集中制御と異なる点は，各エントリが独立して転送距離を選択できるので，各エントリにおける最短パスの組合せが処理要素設定問題の最適解となることである．

4.5　分散制御スケジューリング問題の最適解法

分散制御におけるデータ転送のスケジューリングの問題として，複数エントリが 1 つのエントリへデータ転送を行い，データが衝突することがあげられる．本問題では，このデータ衝突を回避し z を増加させないことが重要である．

スケジューリングにおけるデータ衝突回避法として，最短転送回数が z であるエントリ以外のエントリに着目する．転送回数が z 以下のエントリにおいて，転送回数が z までであれば自由に処理要素の組合せを変更することが可能である．したがって，転送回数が z 以下のエントリにおいて，データ衝突の可能性があるならば転送処理要素の組合せを増やすことで，異なるエントリへ迂回しデータ衝突を回避する．

データ衝突の回避手法によって，転送回数が z 以下のエントリは最短パス以外のパスについても考慮する必要がある．これは集中制御と同様だが，何番目の短いパスまで考慮すればよいといったものはない．そのため，集中制御よりも探索範囲が広がることになり，解を導出することが難しくなる．

そこで処理要素に優先度を定め，優先度の高い処理要素から探索を行う．図 7 の探索木において，子ノードに対して優先度を定めて探索を行う．優先度は以下の条件により決定する[9]．

- 最短パスの処理要素
- 目標状態までの距離と転送回数
- そのデータと衝突する可能性のあるデータ数

まず，最短パスに含まれる処理要素に高い優先度を設定する．最短パスの処理要素は簡単な組合せ問題として解くことができる．すべてのデータが衝突を起こさず最短パスの処理要素のみで転送を行うことができれば最適である．また，最短パスの処理要素でデータ衝突が発生すると，最短パス以外の処理要素でデータの迂回を行う必要がある．この場合，転送回数 z 以内で目標状態に到達することができれば最適である．よって，最短パスの処理要素を基に転送パスを探索することにより良質の解が高速に発見できるので，最短パスの処理要素に高い優先度を設定する．

次に，転送処理を 1 回行った後の状態，つまり，図 7 の探索木において 1 つ下のノードの状態をもとに優先度を定める．1 つ下のノードにおける各エントリのデータが，目標状態に到達するまでの距離が短くなっていれば優先度を高く設定する．また，最短パスに含まれていない転送処理を選択した場合，目標状態までの最短パスが変化することがある．そこで，ノードの中間から目標までの最短パスを再度導出し，現在の最短パスと変化があるかを調べる必要がある．転送完了後の最短パスが現在の最短パスよりも転送回数が増加してしまう場合には優先度を低く設定する．

最後に，時間計算量が増加する要因としてデータ衝突の発生があるので，優先度決定をする際にデータ衝突を考慮に入れる必要がある．転送を行うエントリが局所的に存在すると，データ衝突が発生する確率は高くなる．そこで，データが局所的に固まって存在している場合には，事前にデータを広域に転送させる転送処理の優先度を高く設定しデータの衝突確率を抑える．転送処理を行うデータと衝突する可能性のあるデータ数に応じて転送処理に優先度を設定する．衝突する可能性のあるデータ数が多い場合は優先度を低く設定する．

これらの条件により次に探索するノードを決定し，時間計算量を短縮する．転送パスの処理順序決定のアルゴリズムとして集中制御と同様に深さ優先探索を用いる．

5. 評 価 方 法

5.1　評 価 環 境

評価は，ルネサステクノロジ社提供の MX シミュレータ Version 0.03.01，および，このシミュレータに分散制御手法を追加したものを用いる．このシミュレータでは，図 1 における入出力部および制御部の所要時間は考慮しない．以上の条件で，集中制御および分散制御データ転送処理に対する評価を行う．

5.2　評 価 方 法

評価方法として，実際のアプリケーションに現れるデータ転送処理を用いる．評価対象データ転送パターンは RSA 暗号のデータ転送処理，および，MP3 デコーダ[10] 内に含まれるアンチエイリアス処理，IMDCT（Inverse Modified Discrete Cosine Transform）処理に必要なデータ転送処理の 3 種類である．また，単純で規則的な転送パターンである RSA およびアンチエイリアスでは，転送を行う各エントリに 20%，50%，70%の確率でランダムに転送先を変更したケースの評価をとる．このように規則的な転送から一部を確率的に変更することでランダムなデータ転送への耐性を評価する．

評価項目として，初期データ配置から目標状態のデータ配置へのデータ転送処理における通信回数を比較する．また，評価のうち RSA およびアンチエイリアスのランダム付加評価

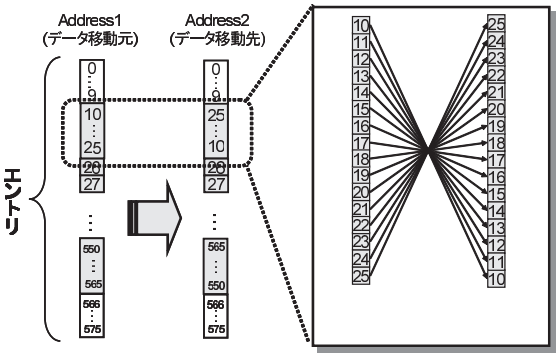

図 8　アンチエイリアスのデータ転送
Fig. 8　Data transmission of the anti-aliasing.

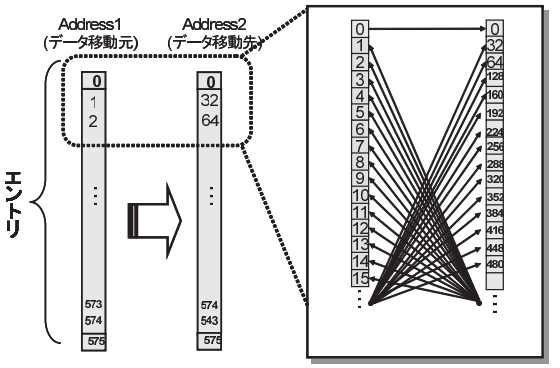

図 9　IMDCT のデータ転送
Fig. 9　Data transmission of the IMDCT.

では，各確率ごとに 10 パターンずつエントリの目標状態を変更した転送パターンを用意し，その平均値をとる．

5.3　規則性のあるデータ転送処理

規則的なデータ転送処理評価について説明する．RSA のデータ転送は，複数のデータを同距離同方向に転送するのみである．次に，アンチエイリアス中のデータ転送処理を説明する．アンチエイリアス処理および IMDCT 処理は，MP3 で使用される 1 グラニュールである 576 データに対して行う．本評価では，576 データを 576 エントリに配置して演算する．アンチエイリアスにおけるデータ転送を**図 8** に示す．具体的には，エントリ番号 12 のデータを 23 のエントリに，13 のデータを 22 のエントリに移動する．すなわち，10 から 25 までのエントリでの逆順の並べ替えになっている．また，2 エントリ挟んで 28 エントリから 33 エントリまでの逆順をとる，これを 576 エントリまで行う転送処理となる．

これら規則的なデータ転送において，各確率におけるランダム性を付加することで集中制御および分散制御における特性を評価する．

5.4　複雑なデータ転送処理

規則的だが複雑なデータ転送として，IMDCT のデータ転送を**図 9** に示す．IMDCT におけるデータ転送処理は，576 個のデータを 32 個おきに上から集めている移動，すなわちバタフライ演算，特に FFT（Fast Fourier Transform）と同じ移動になっていることが分かる．前述したアンチエイリアス処理に必要な転送はすべて近辺のエントリの転送で完結していたが，この転送は 576 個のエントリ全域にわたり転送が起こる．

6.　評価結果および考察

6.1　規則性のある転送処理における評価

図 10 に，RSA 処理中のデータ転送およびアンチエイリアス処理中のデータ転送処理における転送回数を示す．図中の縦軸は転送回数，横軸はランダム転送処理の付加確率である．ランダム転送を付加していないのが 0 であり，以降，動作エントリに対し 20%，50%，70%とランダム転送の割合を多くしている．

ランダム転送のない RSA のデータ転送は，規則的に 1 回の転送が行われるだけなので，両制御においてデータ転送回数は 1 である．しかし，集中制御の制御メモリが 1 回の転送につき 1 ビットに対し，分散制御の方は 4 ビットのメモリを必要とするので，制御に使用するメモリが多い．規則的であった RSA のデータ転送に対し，ランダム転送に変更したとき集中制御はランダムの割合が多くなるに従って転送回数が多くなっている．これは，ランダム転送が集中制御の組合せに合わず，逐次処理が増加するからだと考えられる．分散制御においては，ランダム転送の割合が多くなったとしてもほとんど増加にはつながっていない．RSA では，分散制御の方が少ない転送回数で処理を終える．しかし，前述のように分散制御のデータ転送は集中制御の 4 倍制御用メモリを使用するので，メモリリソースの面も考慮に入れると，集中制御の方が効率的な場合もある．

アンチエイリアス処理中のデータ転送における転送回数においても同様の結果が出ていることが分かる．また，集中制御の転送回数は分散制御の転送回数に比べ，ランダム転送の

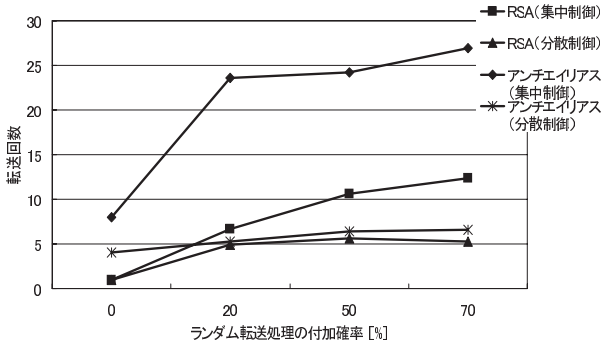

図 10　RSA およびアンチエイリアスのデータ転送
Fig. 10　Data transmission of RSA and antialiasing.

付加によって 4 倍以上の差が出ている．この場合，メモリ使用量においても分散制御の方が集中制御より少なく処理ができる．また，規則的な転送処理の場合，集中制御および分散制御におけるデータ転送の解の導出にかかる時間は数秒から数分で導出することが可能である．これは，RSA やアンチエイリアスにおけるデータ転送が近距離のエントリでデータ転送が行われるので，探索領域が発散することなく処理できるからだと考える．

6.2　複雑な転送処理における評価

表 4 に，IMDCT 処理中のデータ転送における転送回数と制御ビット数を示す．IMDCT におけるデータ転送処理では，集中制御に対し分散制御は約 90%の転送回数削減，および 60%のメモリ使用量の削減を達成した．これは，IMDCT におけるデータ転送が複雑であるため，集中制御の組合せよりも逐次処理が多くなっているからだと考えられる．一方，分散制御は毎回すべての PE が稼動してデータ転送を行うため処理回数が抑えられる．

一方，複雑な転送処理における提案手法の解を導出するまでの処理時間は，分散制御においては数分で解を導出することができる．しかし，集中制御によるデータ転送では，表 4 の結果からも分かるとおり，分散制御の 10 倍の転送回数を導出するため処理時間は数日から数週間を要している．これは，規則的なデータ転送に比べ，多くの PE が遠距離の PE 間データ転送を行う必要があるため探索領域が広がっているからだと考える．また，データ転送を行う PE 数が増加するに従って，集中/分散制御ともさらに探索時間が増加すると予想される．したがって，転送処理の効率化を行うことはできたが，探索方法の改良や処理時間の短縮を行う必要がある．

表 4　IMDCT における転送回数とメモリ使用量
Table 4　The number of times and memory usage in IMDCT.

	転送回数	制御ビット (bit)
集中制御	91	91
分散制御	9	36

これらの結果から，分散制御によるデータ転送処理は複雑なデータ転送が起こっても安定して効率の良い性能が出ていることが分かる．規則性が高いデータ転送になると集中制御においても十分な性能が出ており，メモリの使用量等を考えると集中/分散制御を切り替えて利用することで効率的なデータ転送が実現すると考える．

6.3　アプリケーションにおけるメモリ使用量の考察

MX コアは 1 エントリに保持できるデータ量が限られている．また，MX コア外部との入出力が頻出すると性能の低下となるため，外部との入出力が少なくなるように 1 度に多くのデータを入力する方が有利である．処理やデータ転送制御に必要なメモリ使用量の削減を行うことでより多くのデータを入力することができ，外部との入力を抑えることが可能となる．

RSA では入力データが 1,024 ビットを超えるため，複数エントリに入力データを配置する．1 エントリあたりのデータレジスタの領域を**表 5** に示す．これらの領域に加え，システム領域が 16 ビットと合計 1,023 ビット使用している．これには集中制御時の制御用ビットも含んでいる．したがって，分散制御ではさらに多くのメモリを必要とするので 1 エントリに収めることができない．複数エントリにまたがるデータは，またがるエントリ数が多くなればなるほど並列度が下がるため性能の低下につながる．したがって，RSA では必要メモリが少ない集中制御によるデータ転送が効果的である．

MP3 デコーダでは，本稿で述べたアンチエイリアス，IMDCT に加え，逆量子化，ミドルサイドステレオを MX コアで処理している．1 エントリあたりのデータレジスタの領域を表 5 に示す．システム領域を加え，合計 843 ビット使用している．MP3 デコーダの場合グラニュール単位で処理するため，使用メモリの削減により複数グラニュールを入力できることが可能となると考える．したがって，転送回数の削減とともに制御メモリ量の削減を行うことは処理全体の高速化につながる．

7. 関 連 研 究

近年，専用 SIMD マシンの系列は，LSI の高集積化にともない 1 チップに多数の PE を

表 5　集中制御によるアプリケーションのメモリ使用量
Table 5　Memory usage by concentrated control.

	入力データ	テンポラリ	マスクビット
RSA	480	516	11
MP3 デコーダ	448	283	96

集積したアクセラレータとして開発されている．95 年の NEC の IMAP-2 [11] を皮切りに，最近では ClearSpeed の CSX600 [12] のような大規模な SIMD 型アクセラレータが商用化されている．

これらの SIMD 型アクセラレータでメモリアクセスを含むデータ転送を行う場合，各 PE がそれぞれメモリのアドレスを指定しデータを転送する必要がある．しかし，MX コアではエントリが縦に 1 次元的に配列されることにより，全エントリのロードおよびストアのメモリアドレスを 1 度に指定することができる．このことにより，より高速なメモリアクセスとデータ転送処理が可能となり，リソースも抑えることができる．メモリアドレスを 1 度に指定することによりデータ衝突が発生することもあるが，本提案により効率的なデータ転送を行うことが可能である．

8. まとめと今後の課題

本稿では，手動に頼っていた MX コアの PE 間データ転送の経路決定において，データ転送効率の改善として定式化とその問題解決手法について議論を行った．評価結果として，集中/分散制御それぞれにおいて提案手法から効果的な転送処理の導出およびメモリ利用を行うことができた．分散制御では，すべての転送処理で全エントリが稼動し，並列性の抽出ができた．データ転送が複雑な場合，集中制御と比較して約 10 倍の速度向上が見込める．

今後の課題として，スケジューリングを深さ優先探索で行ったが，処理時間に問題がある．そこで，スケジューリングの計算時間の短縮化が課題である．また，本稿では現状の MX-SoC に則った配線構造の評価を行ったが，配線構造を変更した場合の影響は評価していない．そこで，データ転送時間と配線リソースのトレードオフも課題として残る．最後に，本稿ではデータ転送に限った処理の効率改善を行ったが，アプリケーション実装におけるデータのメモリ配置問題も含めた評価を行う必要があると考える．

謝辞　本研究を遂行するにあたり，貴重なご助言をいただいた（株）ルネサステクノロジ東田基樹氏，水本勝也氏，山崎博之氏に深く感謝いたします．なお，本研究はルネサステクノロジとの共同研究による．

参 考 文 献

1) 富田眞治：並列コンピュータ工学，昭晃堂 (1996).
2) Noda, H., et al.: The Design and Implementation of the Massively Parallel Processor Based on the Matrix Architecture, *IEEE J. Solid-State Circuits,* Vol.42, pp.182–192 (2007).
3) Noda, H., et al.: The circuits and robust design methodology of the massively parallel processor based on the matrix architecture, *IEEE J. Solid-State Circuits*, Vol.42, pp.804–812 (2007).
4) Mizumoto, K., et al.: A multi matrix-processor core architecture for real-time image processing Soc, *A-SSCC*, No.6-2, pp.180–183 (2007).
5) 溝上雄太，中野光臣，飯田全広，末吉敏則：SIMD 型プロセッサ MX コアにおける PE 間データ通信の高度化，信学技法 CPSY2007-64, Vol.107, No.415, pp.19–24 (2008).
6) 西原清一：制約充足問題の基礎と展望，人工知能学会誌，Vol.12, No.3, pp.351–358 (1997).
7) 伊庭斉志：探索のアルゴリズムと技法，サイエンス社 (2002).
8) 杉原厚吉：データ構造とアルゴリズム，共立出版 (2002).
9) 兒玉章宏，溝上雄太，中野光臣，飯田全広，末吉敏則：MX コアの MIMD 型 PE 間データ通信における経路決定手法の提案，信学技報 RECONF2008-6，Vol.108, No.48, pp.31–36 (2008).
10) 浦田敏道：詳細 MP3 マニュアル，エム研 (1999).
11) 藤田善弘，山下信行，木村　亨，中村和之，岡崎信一郎：メモリ集積型 SIMD プロセッサ IMAP，電子情報通信学会論文誌 D-I，Vol.J78-D-I, No.2, pp.82–90 (1995).
12) ClearSpeed Technology Inc.
http://www.clearspeed.com/（参照 2009-03）

(平成 21 年 2 月 5 日受付)
(平成 21 年 3 月 26 日再受付)
(平成 21 年 4 月 22 日採録)

中野　光臣（学生会員）

2005 年熊本大学工学部数理情報システム工学科卒業．同年同大学大学院自然科学研究科数理科学・情報システム専攻博士前期課程入学．2007 年博士前期課程修了．同年同大学院自然科学研究科情報電気電子工学専攻博士後期課程に入学，同課程に在学中．現在，コンピュータアーキテクチャの研究に従事．IEEE，電子情報通信学会各会員．

弘田　澄男

2007 年熊本大学工学部数理情報システム工学科卒業．同年同大学大学院自然科学研究科情報電気電子工学専攻博士前期課程入学．2009 年博士前期課程修了．同年株式会社日立製作所入社．

兒玉　章宏

2008 年熊本大学工学部数理情報システム工学科卒業．同年同大学大学院自然科学研究科情報電気電子工学専攻博士前期課程入学，同課程に在学中．現在，コンピュータアーキテクチャの研究に従事．電子情報通信学会会員．

飯田　全広（正会員）

1988 年東京電機大学工学部電子工学科卒業．同年三菱電機エンジニアリング（株）入社．1995 年同社を休職し，九州工業大学大学院情報工学研究科に入学．1997 年博士前期課程修了．同年同社に復職．1999 年熊本大学大学院自然科学研究科に入学．2002 年博士後期課程修了．2003 年熊本大学工学部助教授．2007 年同大学大学院自然科学研究科情報電気電子工学専攻准教授．博士（工学）．現在，リコンフィギャラブルシステム，VLSI システム設計等の研究に従事．著書『リコンフィギャラブルシステム』（共著）．電子情報通信学会会員．

末吉　敏則（正会員）

1976 年九州大学工学部情報工学科卒業．1978 年同大学大学院工学研究科情報工学専攻修士課程修了．同年九州大学工学部情報工学科助手．同大学院総合理工学研究科助教授，九州工業大学情報工学部助教授を経て，1997 年熊本大学工学部数理情報システム工学科教授．現在，同大学大学院自然科学研究科情報電気電子工学専攻教授．工学博士．コンピュータアーキテクチャ，リコンフィギャラブルシステム，VLSI システム設計等の研究に従事．著書『並列処理マシン』（共著），『リコンフィギャラブルシステム』（共著）等．IEEE，電子情報通信学会，電気学会各会員．