知能ロボット制御用再構成可能並列プロセッサの並列プログラミング

藤岡与周*・苫米地宣裕**

Parallel Programming of Reconfigurable Parallel Processors for Intelligent Robot Control

Yoshichika FUJIOKA and Nobuhiro TOMABECHI

Abstract

In the sensor feedback control of intelligent robots, the delay time must be reduced for a large number of multi-operand multiply-additions. To reduce the delay time, recofigurable parallel processors have been proposed. Since the direct connections between multiplers and adders can be dynamically changed every program step, the overhead for data transfer becomes greatly reduced. In this paper, we show the easiness of parallel programming for intelligent robot control, which is based on the reconfigurable parallel processor architecture. By using the developped programming environment, the performance evaluation shows the delay time for dynamic control of 6 degrees of freedom manipulator is reduced to about 1/4 in comparison with that of the parallel processor using several digital signal processors.

1. まえがき

自律的動作を目指した知能ロボット実現のためには，外界の環境の認識から柔軟なマニピュレータ制御に至るまで多種多様な処理が必要となる。これらの一連の処理は情報の流れが直列的であると共にセンサフィードバックが多いことから，外界環境の変化に高速に応答するためには，個々の処理に対してスループットの向上のみではなくセンサ信号が入力されてから制御出力が計算されるまでの，演算遅れ時間を十分に減少可能なVLSIプロセッサの開発が重要な問題となる[1)-7)]。

筆者らは，知能ロボット制御において必要となる種々の入力数の積和演算の遅れ時間減少を目的として，多入力積和演算器の再構成という概念に基づく，積和演算の高速処理用並列プロセッサをこれまでに提案している[8,9)]。多入力積和演算器の再構成とは，複数個の要素プロセッサ（PE）を用いて所望の個数の乗算器を含む多入力積和演算器を，スイッチ回路の切換えによりダイナミックに形成できることを意味している。複数個のディジタル信号処理プロセッサ（DSP）を用いた並列処理とは異なり，PE間通信がスイッチ回路の切換えによる乗算器や加算器等の各入出力間の直接接続に帰着されるため，データ転送時間が減少し乗算器や加算器などの稼働効率が向上する特長がある。複数個のDSPによる並列処理では，DSP間のデータ転送にソフトウェア手続きが必要となりデータ転送時間が増大するため，複雑なスケジューリングやアロケーションアルゴリズムを用いても，演算遅れ時間を大幅に減少できる並列プログラムの作成は困難であった。

本稿では，データ転送時間を大幅に減少できるとともに多入力積和演算の空間的並列処理が容易であるという，再構成可能並列プロセッサの有する特長に着目し，簡易なスケジューリングやアロケーションアルゴリズムでも十分に演算遅れ時間を減少可能な並列プログラムの作成が容易であることを示すとともに，構築した再構成可能並列プロセッサのソフトウェア開発環境をロボットマニピュレータの動的制御に適用した結果，DSPによる並列処理と比較して演算遅れ時間を約1/4に減少可能であることを明らかにしている。

2. 再構成可能並列アーキテクチャ

知能ロボット制御などに要求される種々の入力数の多入力積和演算の遅れ時間を減少するため，再構成可能並列アーキテクチャが提案されている。本アーキテクチャは，PEに備えられた乗算器や加算器の各入出力間の接続をプログラムステップ毎に動的に切換えることにより，所望とする入力数となるように多入力積和演算器の構造を変化させ，多入力積和演算の空間的並列処理を実行できる。

一例として，1個のPEチップに乗算器と加算器をそれぞれ2個ずつ備えた場合のアーキテクチャを図1に，また，4個の乗算器と加算器を備えた多入力積和演算器を再構成する場合を図2にそれぞれ示す。このように，乗算器と加算器間の直接接続を切換えるため，図3に示す

平成9年12月15日受理

* 八戸工業大学 電気工学科 講師
** 八戸工業大学 電気工学科 教授

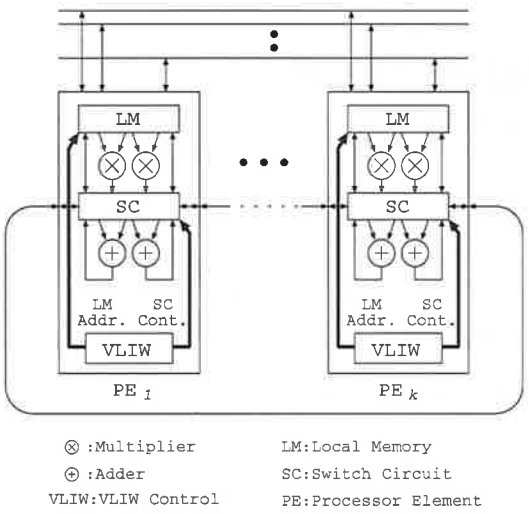

図1　再構成可能並列プロセッサ

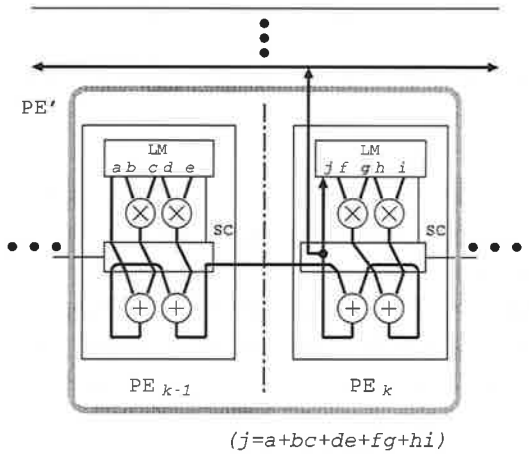

図2　多入力積和演算の実行例

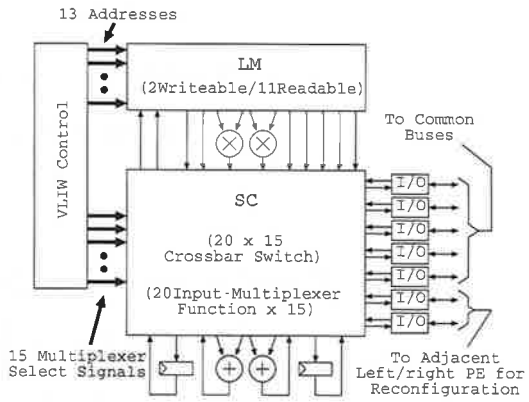

図3　PE の構成

ように各 PE にはクロスバスイッチにより構成されたスイッチ回路が備えられている。また，多入力積和演算器の再構成に伴う PE 間の局所的なデータ転送のため，隣接する PE のスイッチ回路が相互に接続されている。さらに，このスイッチ回路を Very Long Instruction Word (VLIW) プログラムで制御することにより，種々の入力数の多入力積和演算器をプログラムステップ毎に動的に再構成可能としている。

このように，再構成可能 VLSI プロセッサにおいては，多入力積和演算における中間結果の転送がハードウェアの直接接続によりなされるため，DSP を用いた並列処理におけるソフトウェア手続きによるデータ転送と比較すると，データ転送に要する遅れ時間を大幅に減少できるという特長を有する。すなわち，再構成可能 VLSI プロセッサは通信オーバヘッドが小さいため，乗算器や加算器の利用効率が大幅に向上し，演算遅れ時間を減少できる。また，演算器数と比べて処理可能なデータ数が十分多い場合には，演算器の個数に比例してほぼ理想的に演算性能を向上できる。一例として，再構成可能 VLSI プロセッサと，DSP による並列処理により，次式で表される多入力積和演算の並列処理を行う場合の実行ステップをそれぞれ表1と表2に示す。

$$c = a_1 b_1 + a_2 b_2 + a_3 b_3 \quad (1)$$

$$f = d_1 e_1 + d_2 e_2 + d_3 e_3 \quad (2)$$

再構成可能 VLSI プロセッサを用いる場合は，表1(a)と表1(b)に示すように，PE 数に比例して演算性能を向上できるため，1回あたりの多入力積和演算の遅れ時間が増加しない。

これに対し，DSP を用いた並列処理では，表2(a) に示すように，ソフトウェア手続きによる多入力積和演算の中間結果の転送が必要となる。このため，DSP 数が増

表1　再構成可能 VLSI プロセッサによる多入力積和演算の並列処理

(a) 式 (1) の実行

ステップ	PE_1	PE_2	PE_3
1	$c_1 = a_1 b_1$	$c_2 = a_2 b_2$	$c_3 = a_3 b_3$
2	$c = c_1 + c_2 + c_3$		

(b) 式 (1)，(2) の実行

ステップ	PE_1	PE_2	PE_3	PE_4	PE_5	PE_6
1	$c_1 = a_1 b_1$	$c_2 = a_2 b_2$	$c_3 = a_3 b_3$	$f_1 = d_1 e_1$	$f_2 = d_2 e_2$	$f_3 = d_3 e_3$
2	$c = c_1 + c_2 + c_3$			$f = f_1 + f_2 + f_3$		

表2　DSPによる多入力積和演算の並列処理

(a) 式 (1) の実行

ステップ	DSP_1	DSP_2	DSP_3
1	$c_1=a_1b_1$	$c_2=a_2b_2$	$c_3=a_3b_3$
2	out c_1	in c_1	
3		$c'_2=c_1+c_2$	
4		out c'_2	in c'_2
5			$c=c_3+c'_2$

(b) 式，(1)，(2) の実行

	DSP_1	DSP_2	DSP_3	DSP_4	DSP_5	DSP_6
1	$c_1=a_1b_1$	$c_2=a_2b_2$	$c_3=a_3b_3$	$f_1=d_1e_1$	$f_2=d_2e_2$	$f_3=d_3e_3$
2	out c_1	in c_1				
3		$c'_2=c_1+c_2$		out f_1	in f_1	
4		out c'_2	in c'_2		$f'_2=f_1+f_2$	
5			$c=c_3+c'_2$		out f'_2	in f'_2
6						$f=f_3+f'_2$

加すると，表2(b)に示すように中間結果の転送回数の増加により，1回あたりの多入力積和演算時間が増加することから，演算性能をDSPの個数に比例して向上させることは困難である。

3. ソフトウェア開発環境の構成

再構成可能並列プロセッサは，所望とする入力数の多入力積和演算の遅れ時間を大幅に減少できるが，処理の基本はスカラ演算が単位となる。一方，知能ロボットなどの制御アルゴリズムは主に行列やベクトルを用いて記述されることが多いとともに，マニピュレータ関節数に応じた繰り返し演算が多用される。そこで，構築したソフトウェア開発環境の機能は，図4に示すように，ベクトルや行列を用いた記述からスカラの多入力積和演算に変換する部分と，その結果に基づきスケジューリングやアロケーションを行う部分とから構成されている。

まず，知能ロボットの制御アルゴリズムの特長に基づき，ソースファイルを行列やベクトルを用いて記述可能としているとともに，図5に示すようにループ文を利用して繰り返し演算のプログラム記述ステップを大幅に減少可能としている。また，ループ文を展開し行列/ベクトル演算をスカラ演算に図6に示すように変換した後，各変数のデータ依存関係を調査して冗長な代入文や変数の除去や，定数どうしの計算の畳み込みなどを実行すると，図7に示すように，データ依存グラフの深さ毎にグループ化された多入力積和演算のリストが得られる。図7における多入力積和演算の式の左側の数字はデータ依存グラフの深さを示しており，例えばtau_1を求める多入力

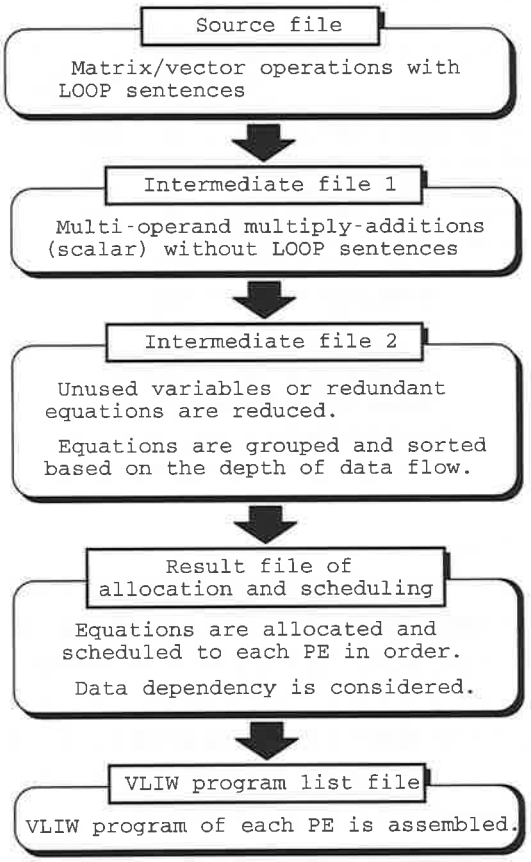

図4　ソフトウェア開発の流れ

積和演算のデータ依存グラフの深さは29となる。これは，多入力積和演算を処理単位とした場合，どれほど演算器を用意しても最低で29回の多入力積和演算の並列処理がtau_1を求める準備段階として必要となることを

```
LOOP i = 6 TO 1 STEP -1
  ^OF^^_{i} = ^OF_{i}
  ^ON^^_{i} = ^ON_{i} + hat^0t_{i} >< ^OF_{i}
  ^OF^ast_{i} = ^OF^ast_{i+1} + ^OF^^_{i}
  ^ON^ast_{i} = ^ON^ast_{i+1} + ^ON^^_{i}
  tau_{i} = ^0beta_{i} T* ^OF^ast_{i} +
            ^0gamma_{i} T* ^ON^ast_{i} +
            b_{i} * dtheta_{i}
ENDLOOP
```

図5　ソースファイルの記述例

```
tau_2 = ^0beta_2[1] * ^OF^ast_2[1] +
        ^0beta_2[2] * ^OF^ast_2[2] +
        ^0beta_2[3] * ^OF^ast_2[3] +
        ^0gamma_2[1] * ^ON^ast_2[1] +
        ^0gamma_2[2] * ^ON^ast_2[2] +
        ^0gamma_2[3] * ^ON^ast_2[3] +
        b_2 * dtheta_2
^OF^^_1[1] = ^OF_1[1]
^OF^^_1[2] = ^OF_1[2]
^OF^^_1[3] = ^OF_1[3]
^ON^^_1[1] = ^ON_1[1] + hat^0t_1[2] * ^OF_1[3] -
                        hat^0t_1[3] * ^OF_1[2]
^ON^^_1[2] = ^ON_1[2] + hat^0t_1[3] * ^OF_1[1] -
                        hat^0t_1[1] * ^OF_1[3]
^ON^^_1[3] = ^ON_1[3] + hat^0t_1[1] * ^OF_1[2] -
                        hat^0t_1[2] * ^OF_1[1]
^OF^ast_1[1] = ^OF^ast_2[1] + ^OF^^_1[1]
^OF^ast_1[2] = ^OF^ast_2[2] + ^OF^^_1[2]
^OF^ast_1[3] = ^OF^ast_2[3] + ^OF^^_1[3]
^ON^ast_1[1] = ^ON^ast_2[1] + ^ON^^_1[1]
^ON^ast_1[2] = ^ON^ast_2[2] + ^ON^^_1[2]
^ON^ast_1[3] = ^ON^ast_2[3] + ^ON^^_1[3]
tau_1 = ^0beta_1[1] * ^OF^ast_1[1] +
        ^0beta_1[2] * ^OF^ast_1[2] +
        ^0beta_1[3] * ^OF^ast_1[3] +
        ^0gamma_1[1] * ^ON^ast_1[1] +
        ^0gamma_1[2] * ^ON^ast_1[2] +
        ^0gamma_1[3] * ^ON^ast_1[3] +
        b_1 * dtheta_1
```

図6　ループ文の展開後

示している。

このようにデータ依存グラフの深さ毎に並べ替えられた多入力積和演算を，多重バスによる中間結果の転送を考慮しながら各PEに割り当てていくという単純なアロケーションとスケジューリングを行うことにより，各PEのVLIW制御プログラムを生成可能としている。一例として，次式で表される多入力積和演算のアロケーションとスケジューリングを図8に示す。

$$1 \quad c_1 = a_1 b_1 + a_2 b_2 \tag{3}$$

$$1 \quad c_2 = a_1 b_1 + a_2 b_2 + a_3 b_3 \tag{4}$$

$$1 \quad c_3 = a_4 b_4 + a_5 b_5 + a_6 b_6 \tag{5}$$

$$2 \quad f_1 = c_1 d_1 + c_2 d_2 + e_1 \tag{6}$$

$$2 \quad f_2 = c_2 d_3 + c_3 d_4 + d_5 d_6 \tag{7}$$

$$2 \quad f_3 = c_1 c_2 + c_3 \tag{8}$$

```
27  tau_3 = ^0beta_3[1] * ^OF^ast_3[1] +
            ^0beta_3[2] * ^OF^ast_3[2] +
            ^0beta_3[3] * ^OF^ast_3[3] +
            ^0A_2[1][3] * ^ON^ast_3[1] +
            ^0A_2[2][3] * ^ON^ast_3[2] +
            ^0A_2[3][3] * ^ON^ast_3[3] +
             b_3 * dtheta_3
28  ^OF^ast_1[2] = ^OF^ast_2[2] + ^OF_1[2]
28  tau_2 = ^0beta_2[1] * ^OF^ast_2[1] +
            ^0beta_2[2] * ^OF^ast_2[2] +
            ^0beta_2[3] * ^OF^ast_2[3] +
            ^0A_1[1][3] * ^ON^ast_2[1] +
            ^0A_1[2][3] * ^ON^ast_2[2] +
            cos_alpha_1 * ^ON^ast_2[3] +
             b_2 * dtheta_2
29  tau_1 = ^0beta_1[1] * ^OF^ast_1[1] +
              ^0h_0[1] * ^OF^ast_1[2] +
                         ^ON^ast_2[3] +
            sin_alpha_1 * ^1IAdomega_1[2] +
            cos_alpha_1 * ^1IAdomega_1[3] +
            hat^0t_1[1] * ^OF_1[2] -
            hat^0t_1[2] * ^OF_1[1] +
             b_1 * dtheta_1
```

図7　冗長な数式等の削除後

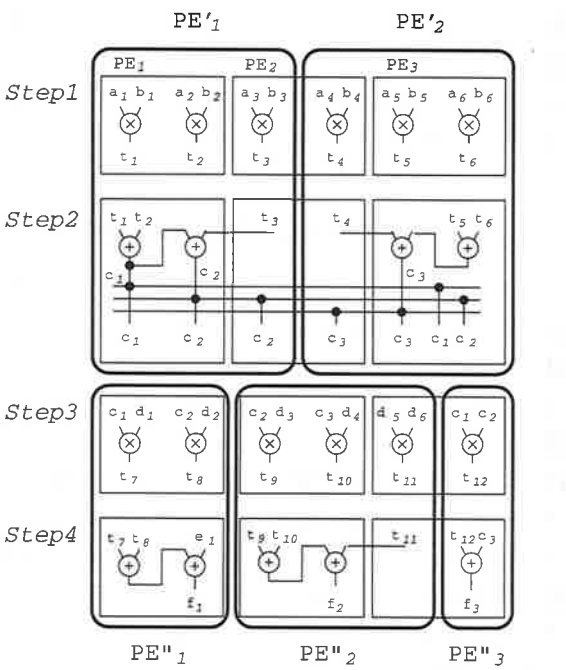

図8　アロケーションとスケジューリング

このように，ステップ1と2にはデータ依存グラフの深さが1である式(1)，(2)，(3)の各積和演算が，利用可能であるPEと多重バスに順に割り当てられており，同様にステップ3と4にはデータ依存グラフの深さが2である式(4)，(5)，(6)の各積和演算が割り当てられている。このように簡易なアロケーションアルゴリズムでも多入力積和演算の空間的並列処理が容易に実現されているため，データ転送回数と演算ステップをできるだけ減少するように個々の多入力演算をDSPにより逐次的

に処理する場合と比較しても，演算遅れ時間を大幅に減少可能である。

4. マニピュレータ動的制御への適用

6自由度マニピュレータの動的制御[10]に本ソフトウェア開発環境を適用した結果を図9から図13に示す。ここで，比較対象である複数個のDSPを用いた並列処理については，データ転送のオーバヘッドをできるだけ減少して演算遅れ時間を減少するため，1個の多入力積和演算を1個のDSPに割り当てて逐次的処理を行うようにしている。図9に示すように，単純なアロケーションとスケジューリングでありながら，複数個のDSPを単一共有バス結合した並列プロセッサと比較すると，再構成可能並列プロセッサにより演算遅れ時間を約1/4に減少できる。また，DSPアーキテクチャをそのまま拡張して仮に複数個のDSPを多重バスで結合したとしても，演算遅れ時間を約1/2に減少可能である。

再構成可能並列プロセッサの場合には多入力積和演算の空間的並列処理が可能であるため，図10に示すように演算器数の増加に伴い演算遅れ時間を減少可能である。ここで，演算器数と演算遅れ時間とが正確に逆比例の関係とならないのは，例題として取り上げた動的制御アルゴリズムの演算の並列性が十分大きくないためと考えられる。これに対し，DSPによる並列処理では，図11に示すように演算器の個数を増加させても演算遅れ時間は減少していない。これは，再構成可能並列プロセッサが多入力積和演算の空間的並列処理により乗算や加算単位で

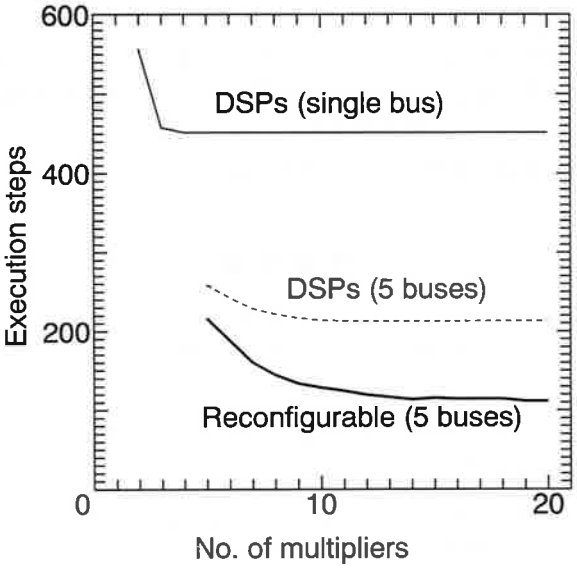

図9　乗算器の個数と演算遅れ時間の関係

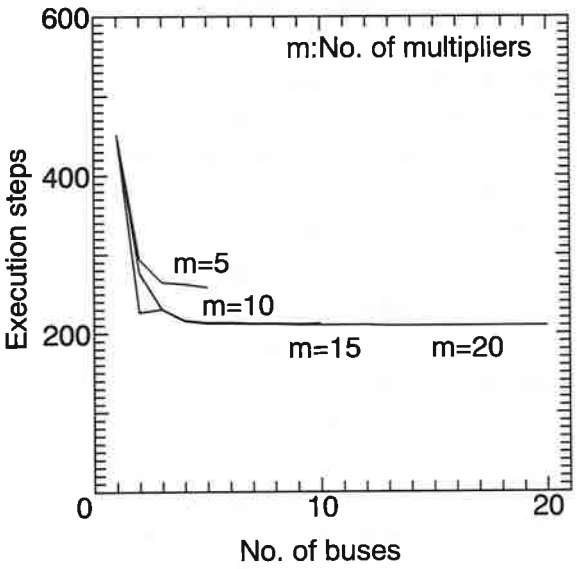

図11　多重バスの本数と演算遅れ時間の関係（複数個のDSPを用いる場合）

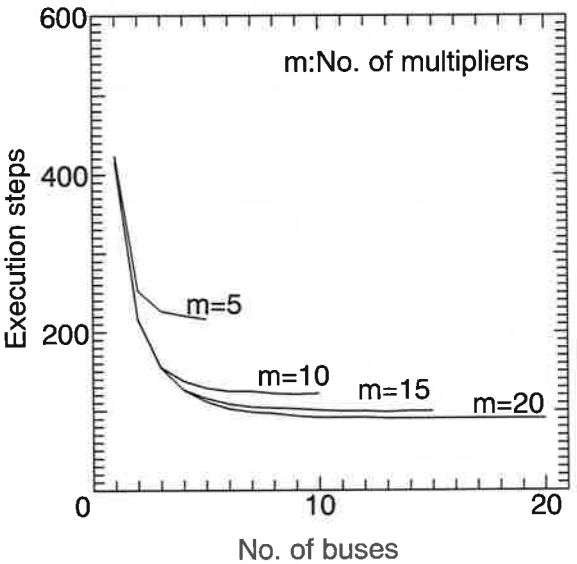

図10　多重バスの本数と演算遅れ時間の関係（再構成可能並列プロセッサの場合）

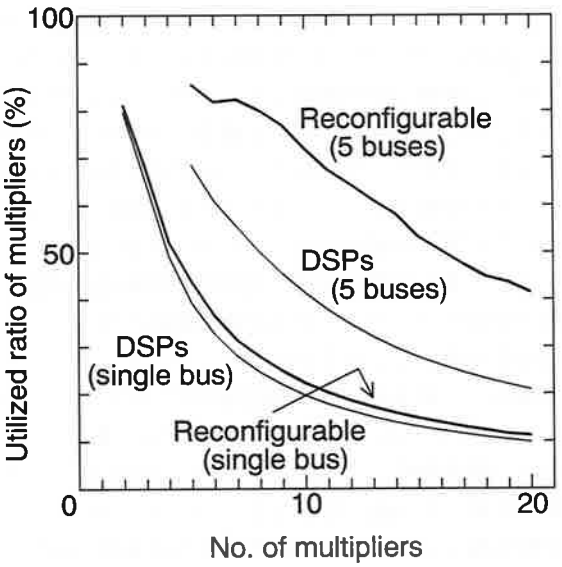

図12　乗算器の稼動効率の比較

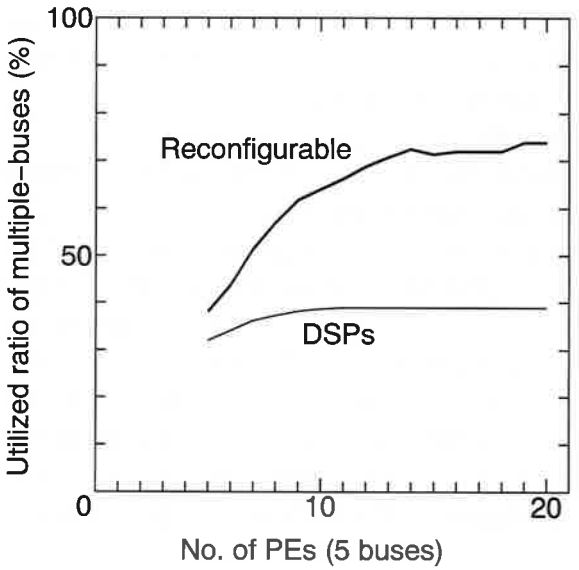

図13　多重バスの稼働効率の比較

の並列性を有効に活用し細粒度高並列処理を実現できるのに対し，DSPを用いた並列処理ではデータ転送回数を減少するために個々の多入力積和演算を単一DSP内部で逐次処理しなければならず，多入力積和演算単位での粒度の粗い並列性しか活用できないためと考えられる。したがって，多入力積話演算を複数個のPEで空間的並列処理が可能な再構成可能並列プロセッサは，本質的に演算遅れ時間の減少に有用であることがわかる。

稼働効率という観点から比較を行なった結果を図12と図13に示す。単一共有バス結合の場合には，図12に示すように再構成可能並列プロセッサとDSPを用いた並列プロセッサとでは演算器の稼働効率に大きな差がない。これは，中間結果の転送を行なうためのバスが1本しかなく，このバスによるデータ転送のための待ち時間が演算性能を支配しているためと考えられる。これに対し，バスの本数を5本に増加させると，再構成可能並列プロセッサの場合は乗算器数が5個程度で演算器の稼働効率が80%に達している。乗算器数をそれ以上増加させると徐々に稼働効率が減少していくが，これは例題として取り上げた動的制御の有する演算の並列性がそれほど高くなく，またデータ依存グラフの各深さ毎に並列性のばらつきが存在するためと考えられる。DSPによる並列処理の方も同一の傾向が見られるが，グラフから明らかなように，多入力積和演算の空間的並列処理が可能な再構成可能並列プロセッサの方が演算器数が多くかつ高い稼働効率を保つことができる。

同様のことが多重バスの稼働効率についてもいえる。図13に示すように，乗算器の個数が十分に多ければ，多重バスの稼働効率も70%以上となり，DSPによる並列処理と比較すると約2倍となる。これは，DSPなどによる並列処理では単純に演算器数やバスの本数を増加させても演算遅れ時間減少という観点での性能向上が困難であるのに対し，再構成可能並列プロセッサはこの問題点の解決に非常に有用であることを意味している。

5. むすび

PE間のデータ転送時間が極めて小さい再構成可能並列プロセッサの特長を活用して，簡易なアロケーションとスケジューリングでありながら演算遅れ時間の大幅な減少を可能とするソフトウェア開発環境を構築するとともに，マニピュレータ動的制御に適用し，並列プログラミングに関する種々の知見を得た。ロボット制御に関するソフトウェア開発環境としては，種々の数式処理ソフトウェアを応用したものが提案されているが，それらから出力される多入力積和演算の数式は若干のフォーマット変更を行なうことにより今回提案した並列プログラム開発環境に入力可能である。今後の課題として，処理の緊急性に応じた多入力積和演算のタスク分割の概念導入により，PEの稼動効率をより向上可能なソフトウェア開発環境の構築が重要と考えられる。

参考文献

1) 亀山，樋口："ロボットとVLSIコンピュータ"，日本ロボット学会誌，6, 4, pp. 332-338 (1988).
2) 樋口："ロボットエレクトロニクス"，信学誌，71, 5, pp. 502-504 (1988).
3) 今井："ASIC技術の基礎と応用"，電子情報通信学会，pp. 118-140 (1994).
4) M. Kameyama, T. Matsumoto, H. Egami and T. Higuchi: "A special-purpose LSI for inverse kinematics computation," Trans. IEICE, E 74, 11, pp. 3829-3837 (1991).
5) S. Kittichaikoonkit, M. Kameyama and T. Higuchi: "Design of a matrix multiply-addition VLSI processor for robot inverse dynamics computation," Trans. IEICE, E 74, 11, pp. 3819-3828 (1991).
6) B. Kim, M. Kameyama and T. Higuchi: "Parallel VLSI processors for robotics using multiple bus interconnection Networks," Trans. IEICE, E 75-A, 6, pp. 712-719 (1992).
7) P. Sadayappan, Y.L.C. Ling, K.W. Olson and D.E. Orin: "A restructurable VLSI robotics vector processor architecture for real-time control," IEEE Trans. on Robotics and Automation, 5, 5, pp. 583-599 (1989).
8) 藤岡，亀山，樋口："冗長マニピュレータ制御用座標変換VLSIプロセッサ"，信学論，J 75-D-I, 10, pp. 909-916 (1992).
9) 藤岡，亀山，苫米地："再構成可能並列プロセッサと知能ロボット制御への応用"，日本ロボット学会誌，13, 6, pp. 112-119 (1995).
10) 中村，横小路，花房，吉川："ロボットマニピュレータの運動学と動力学の統合化計算"，計測自動制御学会論文集，23, 5, pp. 491-498 (1987).