SIMD 型並列計算機における高速処理方式

正 員 岩瀬 正 (三菱電機 (株))

非会員 磯西徹明 (三菱電機 (株))

非会員 宮田裕行 (三菱電機 (株))

正 員 小泉寿男 (東京電機大学)

High Speed Processing Method for A SIMD Parallel Computer

Akira IWASE (Mitsubishi Electric Corp.)

Tetsuaki ISONISHI (Mitsubishi Electric Corp.)

Hiroyuki MIYATA (Mitsubishi Electric Corp.)

Hisao Koizumi, Member (Tokyo Denki University)

In order to achieve high performance of parallel programs executed on a SIMD parallel computer, it is necessary to speed up the execution of scalar operations as well as array operations. A scalar processor, which is one of the main component of a SIMD parallel computer, has an important role for not only executing scalar operations but also controlling execution of parallel programs and array operations.

In this paper, we propose a way to realize a high-speed scalar processor using general purpose microprocessor and the methodology to execute the parallel programs efficiently. By adapting these methods, we show that we can achieve both high performance of scalar operations and improvement of total performance, and can keep future expandability of a SIMD parallel computer.

キーワード：並列計算機, SIMD, アレイ演算, スカラ演算, 命令セットアーキテクチャ, 接続ネットワーク, 画像処理ライブラリ

Keyword: Parallel Computer, SIMD, Array Operation, Scalar Operation, ISP Architecture, Connection Network, Image Library

1. はじめに

マルチメディア処理の高速化には並列処理方式が有効である。中でも、2次元の画像データ等に対して様々な処理を高速かつ効率的に実行する方式として SIMD (Single Instruction Multiple Data stream) 方式がある。SIMD 方式の実現方法は、その用途により2つの方法に大別される。一つは、独立型の SIMD 型並列計算機として実現することである。大規模な2次元画像データの高速処理やストリーム画像のリアルタイム処理等を行う場合に適用する¹⁾。もう一つは、マイクロプロセッサ (μ P) への組込み型^{2),3)}あるいはコプロセッサ⁴⁾として

実現する。 μ P のマルチメディアエンジンとして機能し、一つの μ P で汎用処理とマルチメディア信号処理を効率よく実行する場合に有効である。SIMD 方式では、並列プログラムのアレイ演算可能な部分に対して、理想的にはアレイ演算部の基本演算要素 PE の数に比例した性能向上が見込まれる^{5),6)}。これによりスケーラブルな構成を取ることが特徴である。一方、並列プログラム中には、アレイ演算の対象となるデータのアドレス計算を行う算術論理演算をはじめとして、多くのスカラ演算が含まれている。このため、並列プログラムの高速化には、スカラ演算の高速実行が必須である^{7),8)}。 μ P 組込み型

の場合、スカラ演算は μP 自身のスカラ命令が実行し、プログラムの実行制御も μP の命令実行制御に従うので、並列プログラムの実行性能は μP の性能に依存する。一方、独立型のSIMD型並列計算機においては、スカラ演算は独立のスカラプロセッサのスカラ命令が実行する。このスカラプロセッサは、スカラ命令の実行だけでなく、並列プログラムの並列実行制御と並列演算部で実行されるアレイ命令の実行制御という実行性能に関わる重要な役割を担っている。

従って、独立型のSIMD型並列計算機における高速処理方式とスカラプロセッサのアーキテクチャの決定に際しては、高性能化と将来への拡張性についての考慮が必要である。

本論文では、大規模データの高速処理を目的とした、独立型のSIMD型並列計算機(以下単にSIMD型並列計算機と記す)の高速処理方式について述べる。スカラプロセッサとして、市販・汎用マイクロプロセッサ(μP)を採用し、SIMD型並列計算機に固有の機械語命令セットを汎用 μP の機械語命令セットに最適になるように変換する方式と、変換された並列プログラムを高速並列処理する方式を提案する。この方式の適用により、スカラ演算の高速化、総合性能の向上と将来への拡張性を確認できたことを報告する。

第2章でSIMD型並列計算機の制御方式と課題について述べ、第3章で機械語命令の変換方式と変換された命令の高速処理方式を提案する。第4章で適用した結果に対する評価を論じる。

2. SIMD型並列計算機の制御方式と課題

2.1 SIMD型並列計算機の構成

典型的なSIMD型並列計算機の全体構成を図1に示す。

図1 SIMD型並列計算機の構成

Fig.1 Structure of SIMD Parallel Computer

SIMD型並列計算機は、アレイ演算部(AU)と並列計算機全体を制御する制御部(CU)から構成されている。CUはスカラプロセッサであり、AUは多数の基本演算要素(PE:Processor Element)とPE間を接続するネットワークから構成されるアレイプロセッサである。AUはPE

数や接続関係を可変にすることで、目的や要求性能にあわせた構成にできることが特徴である。フロントエンドプロセッサ(FE)は、並列プログラムの開発やCUへのプログラムおよびデータの転送、CUやAUのリソース管理等を行うもので、汎用計算機を用いることが多い。

2.2 SIMD型並列計算機の制御方式

並列プログラムの実行制御はCUが行う。

SIMD型並列計算機の機械語命令は機能的に分類すると次のようになる。

(1) アレイ命令

AUの各PE内のアレイデータに対して、全PE一斉にアレイ演算を行う。ロード・ストア、ムーブ、シフト、算術・論理演算、PE間転送などの命令がある。

(2) スカラ命令

並列プログラムの実行順序制御とスカラデータに対する演算を行う。ロード・ストア、ムーブ、シフト、算術・論理演算、分岐などの命令がある。

(3) アレイ→スカラ命令

全PEのアレイデータをスカラデータに変換し、CUに受け渡す命令である。

(4) スカラ→アレイ命令

CU内のデータを全PEに共通にブロードキャストする命令である。

(5) 入出力命令

各ハードウェアブロック間および外部とのデータ転送を指示する命令である。

並列プログラムは、これらの命令の組み合わせであり、実行前にCUに格納される。CUは並列プログラムの実行時に命令を読み出してデコードする。これがアレイ命令かアレイ→スカラ命令、スカラ→アレイ命令、入出力命令であった場合にはデコード信号をAUに送る。AUはこの信号を受けて、各PEがクロックに同期して同時に同じ演算またはデータの入出力を行う。一方、デコードの結果、スカラ命令の場合には、CU自らその命令を実行する。

2.3 CUのアーキテクチャと実装上の課題

スカラプロセッサであるCUのアーキテクチャ選定およびそのハードウェア実装については、スカラ演算の高速化、ハードウェア開発の効率化の点では、高性能化と低コスト化の進む市販・汎用 μP の採用が望ましい。しかし、並列プログラム開発の効率化、プログラムの高速化の点では、CUとAUのISP(Instruction Set Processor)アーキテクチャが異なることは、スカラ命令とそれ以外のアレイ命令等の命令形式が異なることになり、プログラム開発者にとって好ましくない。スカラ命令とアレイ命令等の命令形式を同一にしようとするれば、そのISPアーキテクチャを実現する専用スカラプロセッサを開発する必要がある。これは、ハードウェア開発の負担が増すだけでなく、演算性能の点でも問題がある。開発時点ではある程度の演算性能を発揮しても、VLSI

化に支えられた汎用 μ Pの性能向上の前に、すぐに陳腐化してしまう。汎用 μ Pで固有ISPアーキテクチャのスカラ命令をエミュレーションする方法もあるが、高性能化は難しい。また、ある固有ISPアーキテクチャの命令セットを他の固有ISPアーキテクチャの命令セットに動的に変換する機能を μ Pに内蔵するダイナミックトランスレーションの方法⁹⁾もあるが、これは μ Pの開発そのものであり、専用スカラプロセッサを開発するのと同じ問題がある。

このようなハードウェア、プログラム開発者双方の要求を満たしつつ、スカラ演算の高速化、将来への拡張性とハードウェア実装の柔軟性を可能にするスカラプロセッサの実現方法が課題である。

これまでに開発されたSIMD型並列計算機のスカラプロセッサは、ほとんどが固有のISPアーキテクチャの専用スカラプロセッサである^{1),10)}。

3. 汎用 μ Pを用いた高速処理方式の提案

本章では、実ハードウェアのCU即ちスカラプロセッサには汎用 μ Pを採用し、並列プログラム開発者には、目的とする並列処理に最適な固有のISPアーキテクチャを提供するSIMD型並列計算機の高速処理方式を提案する。本方式は、SIMD型並列計算機の固有ISPアーキテクチャの機械語命令で記述されている並列プログラムを、汎用 μ Pの機械語命令の並列プログラムに自動変換する「機械語命令変換方式」と、この変換された並列プログラムの「汎用 μ Pによる高速処理方式」に特徴がある。

本方式での並列プログラムの開発と実行の手順を図2に示す。並列プログラムの開発はFEプロセッサで行う。高位言語または固有ISPアーキテクチャの機械語命令を用いて開発し、ソフトウェアシミュレータ上でデバッグを行う。高位言語コンパイラの出力はこのISPアーキテクチャの機械語命令である。この並列プログラムのソースプログラムを並列計算機での実行前に、変換プログラ

ムを用いて機械語命令の自動変換を行う。こうして得られる、 μ Pの機械語命令に変換された並列プログラムのソースプログラムを、アセンブル/リンクして得られるロードモジュールを汎用 μ Pで高速処理する方式である。

3.1 機械語命令変換方式

本変換方式は、変換プログラムによるスタティック変換方式であり、変換に必要な定義データ部と変換実行部を分けて実装している。これは、変換定義データを変更するだけで、各種タイプの汎用 μ Pに容易に対応できる柔軟性を持たせることと、並列プログラム中のマクロ機能を汎用 μ Pの機械語命令で直接記述することによる並列プログラムの高速化を実現するためである。変換定義データはSIMD型並列計算機の固有ISPアーキテクチャの機械語命令と汎用 μ Pの機械語命令との対応関係を定める。スカラ命令だけでなく、スカラ命令以外、即ち、アレイ命令、アレイ→スカラ命令、スカラ→アレイ命令および入出力命令の変換定義を含んでいる。

機械語命令変換の手順である変換アルゴリズムは、スカラ命令とそれ以外の命令に分けて考える。

・スカラ命令

スカラ命令はスカラプロセッサであるCUで実行するので、SIMD型並列計算機の固有ISPアーキテクチャのスカラ命令を、もとの機能を変えずに、目的とする汎用 μ Pのスカラ命令群に置換する。この際、レジスタやアドレッシング形式、演算ステータスなど両方のアーキテクチャで異なる項目は、その対応関係を変換定義データとして作成し、変換処理実行時に参照できるようにしておく。

・スカラ命令以外

スカラ命令以外、即ち、アレイ命令、アレイ→スカラ命令、スカラ→アレイ命令および入出力命令はAU側で実行される。その実行制御には、CUが各々の命令コードと引数、データをAUに転送し、その実行を指示する方式が有効である。従って、これらの命令に対しては、対応する命令コードと引数、データをAUに転送する汎用 μ Pの機械語命令群と、AUへの実行指示およびAUからの実行結果のステータスの取り込みを行う汎用 μ Pの機械語命令群に置換する。

3.2 汎用 μ Pによる高速処理方式

独立型のSIMD並列計算機では、CU、AUおよび入出力は機能的に独立している。提案する高速処理方式は、これらを並行動作させることで性能向上を目指すものである。

スカラ命令の実行は、CU自身が変換された汎用 μ Pのスカラ命令をそのまま実行する。汎用 μ Pの機械語命令群に置換されるスカラ命令以外のアレイ命令等の実行制御にはメールボックス方式を採用する。図3にその制御方式の概念を示す。CUである汎用 μ Pのメモリ領域にメモリボックスを設定する。AUには、アレイ演算と入出力を制御するAU制御部を設け、命令コードを受け

図2 並列プログラム開発と実行手順
Fig.2 Development flow and execution of Parallel Program

取る命令キューや AU の状態を示すステータスレジスタを置く。CU は機械語命令の実行により、アレイ命令コードと引数、データおよびヘッダーをメールボックスに書きこむ。AU 制御部は、メールボックスのヘッダーを読み出し、ヘッダー内のコマンドを解釈して、命令コードを命令キューに格納する。命令キューに命令コードが格納されると、それに対するステータスがメールボックスに書き込まれ、CU にはステータス転送終了報告として、割り込み要求が出される。この一連の動作の正常終了で、CU と AU は論理的に切り離されることになり、以降、互いに独立して動作可能となる。

図3 メールボックスによる命令コードの転送
Fig.3 Transfer of Instruction through Mail Box

4. 適用結果の検証と考察

実際の SIMD 並列計算機に、提案した高速処理方式を適用して、スカラ命令の実行性能の評価と考察および開発効率の考察を行う。ここでは、筆者等が開発した SIMD 型の大規模並列プロセッサ CAP^{11),12)}での実行結果をもとに評価を行う。

図4にCAPのハードウェア構成を示す。CAPは並列演算部(CAU)とCAP全体を制御するCAP制御部(GCU)から構成されており、このCAPと共有メモリ(SM)を専用バスで介して結合したSIMD型並列計算機である。CAUは並列演算を実行するセルラアレイ部、その制御を行う演算制御部(ACU)およびデータの入出力を行う入出力制御部(DCU)からなる。GCUはスカラプロセッサであり、CAPでは86系汎用μPのシングルボードコ

図4 CAPの構成
Fig.4 Structure of CAP

ンピュータを実装している。なお、CAU、GCUは図1のSIMD型並列計算機のAU、CUにそれぞれ対応する。また図5にCAPの機械語命令形式を示す。

CAPでは提案した方式を次のように実現している。

4.1 CAPの機械語命令変換方式

並列プログラム開発は、CAP固有ISPアーキテクチャの機械語命令をもとにしたプログラミング言語で行う。その並列プログラムを変換する機械語命令変換プログラムはGCU上で動作し、変換定義データを参照しつつ、CAP固有ISPアーキテクチャの図5の形式で示される機械語命令を86系汎用μPの機械語命令に変換する。

Array Instruction: TYPE-5,7,8,10,11,12,13
Scalar Instruction: TYPE-1,2,3,4,5,6
Array→Scalar Instruction: TYPE-7
Scalar→Array Instruction: TYPE-5,7,9
Input-Output Instruction: TYPE-5

図5:CAP命令形式

Fig5: Instruction Format of CAP

CAPの機械語命令は、実行性能向上を図るために、86系汎用μPの機械語命令親和性を持たせ、双方の機械語同士の対応がとりやすくなっている。

アレイ命令、アレイ→スカラ命令、スカラ→アレイおよび入出力命令はメールボックス方式での実行制御が行えるように、CAPでは次のように変換する。メールボックスはGCU上のメモリ領域に設定し、GCUからCAUへアレイ命令コード等の転送とステータスの受け取りは、このメールボックスとシステムバスを介して行う。CAPではこの一連の動作を汎用μPのランタイムライブラリとして定義し、スカラ命令以外の命令実行時に共通に使えるようにしている。従って、スカラ命令以外の機械語命令変換は、CAUへ転送する命令コードと引数、データを汎用μPのメモリにセットする機械語命令とランタイムライブラリを呼び出すコール命令に置換する。図6に

CAP のスカラ命令 LDIC と LDC, 図 7 に CAP のアレイ命令 CLR, MOV, COM および MCN の 86 系汎用 μ P の機械語命令への変換例をそれぞれ示す。EXARRAY はランタイムライブラリである。

また, 表 1 に CAP の機械語命令変換プログラムのソースステップ数と実行性能を示す。この開発には約 2 人月を要したが, 固有 ISP のスカラプロセッサを開発することに比べれば, その負荷は小さく, さらに, 変換定義データを変えることで複数のアーキテクチャにも容易に対応できる利点の方が大きいといえる。

THNG3	LDIC	SP,ASTK_E	CAP
THNG3:	LEA	DX,ASTK_E	
	SHR	DX,1	× 86
	SHR	DX,1	
	MOV	R_SP,O	
	MOV	R_SP+2,DX	
*			
	LDC	A,CAPSX	CAP
	MOV	SI,CAPSX	
	MOV	DI,CAPSX+2	× 86

図6 スカラ命令の変換例

Fig.6 Translation Example of Scalar Instruction

THNG3_CRS	CLR	5RF_WK1	CAP
THNG3_CRS:	MOV	DX,8422H	
	MOV	CX,OFOOOH	86系
	CALL	EXARRAY	
	MOV	1RF_WK3,RF_X1	CAP
	MOV	DX,OC803H	86系
	MOV	CX,9380H	
	CALL	EXARRAY	
	COM	1	CAP
	MOV	DX,OC300H	8086
	MOV	CX,9120H	
	CALL	EXARRAY	
	MCN	16,O	CAP
	MOV	DX,OD07AH	8086
	MOV	CX,OA2AOH	
	MOV	AX,R_IX+2	
	AND	AX,O1FH	
	OR	CX,AX	
	CALL	EXARRAY	

図7 アレイ命令の変換例

Fig.7 Translation Example of Array Instruction

表1 CAPの機械語命令の変換プログラムの性能

Table.1 Performance of Translation Program for CAP

	Number of steps of source code	Size of load module
Translation program (C-Lang.)	7,609	≈ 128kb
Translation definition data	1,447	-
CAP run-time library (assembler)	1,104	-
	Translation time	Note
Source program 2000 steps	≈ 1min.	by PC

4.2 CAP の高速処理方式

変換された並列プログラムは, GCU のメモリに格納され実行される。スカラ命令は GCU 自身が実行し, アレイ命令, アレイ→スカラ命令およびスカラ→アレイ命令に対応したランタイムライブラリは GCU と ACU が, また入出力命令に対応したランタイムライブラリは GCU と DCU が, それぞれシステムバス経由で実行する。具体的には, 86 系 μ P のメモリマップト I/O 機能を

これに当てている, 即ち, 86 系 μ P である GCU のメモリ空間内で入出力領域として割り付けられる領域の一部をアレイ命令等の実行のためのメールボックスとして使用する。GCU は, アレイ命令等の命令コードと引数, データをメールボックスにセットすると同時に ACU または DCU の起動を行う。ACU または DCU はメールボックスの内容を読み込んで, ACU 内の命令キューに命令コードを格納し, GCU に対して所定の動作終了を示す割り込み要求をかける。ランタイムライブラリ実行後は, GCU と ACU および GCU と DCU はそれぞれ論理的に切り離されるので, CAP ではスカラ, アレイおよび入出力の各演算を並行実行可能である。

4.3 機械語命令変換後の実行ステップ数の評価

スカラプロセッサの命令実行能力を評価するために, 計算機の性能評価尺度として用いられる命令 Mix 法の考え方を適用する。命令 Mix 法での命令の使用頻度を重みにして平均実行ステップ数を計算し比較する。ここで, ステップ数とは, CAP の機械語命令が 86 系汎用 μ P の機械語命令の何命令に変換されるかを表す。表 2 に CAP のスカラ命令に対応した 86 系汎用 μ P の実行ステップ数と平均実行ステップ数の計算値を示す。CAP の 1 スカラ命令は汎用 μ P の 2 ~ 15 実行ステップ数で実現できている。平均実行ステップ数をみると, CAP のステップ数を 100 とした場合, 変換後の汎用 μ P のステップ数は 427 となった。これは, 仮に汎用 μ P と同等性能の CAP 固有 ISP アーキテクチャのスカラプロセッサが開発できたなら, スカラ演算性能は, 提案した方式, 即ち, スカラプロセッサとして汎用 μ P を採用した場合に比べて約 4 倍の性能が得られるということである。しかしながら, 汎用 μ P の開発では, 最新の半導体技術, 最新の高速アーキテクチャを駆使して最高性能を追求し, 3 年で 4 倍といわれる早さで性能向上が進んでいる。

表2 スカラ命令の実行ステップ数

Table.2 Number of Execution Steps of Scalar Instruction

Scalar Instruction of CAP	Number of Execution Steps for x 86 μ P	Execution Frequency of CAP Scalar Instruction	Instruction Steps Converted into x 86
Load	2	35	70
Index	5	17	85
Compare	3	8	24
Jump	6	15	90
Addition	4	15	60
Multiplication	15	6	90
Move	2	4	8
Average Execution Steps		100	427

従って, 4 倍程度の性能ギャップも数年で埋まってしまうので, これに対抗して固有 ISP アーキテクチャのスカラプロセッサを開発するよりも汎用 μ P を活用する方が得策である。

4.4 スカラ命令の実行性能評価

表 3 に実際の μ プロセッサによる CAP のスカラ命令の実行時間を示す。この実行時間は、各 μ P の命令実行時間（公表値）をもとに求めた。各ケースの仕様は次の通り。

- ・ ケース 1： μ P：8086(8MHZ)，データ幅 16 ビット
- ・ ケース 2： μ P：80386（16MHZ）
リアルアドレスモード，データ幅 16 ビット
- ・ ケース 3： μ P：80386（16MHZ）
リアルアドレスモード，データ幅 32 ビット

また、表中の数値は実行時間（単位： μ 秒）である。

ケース 1 とケース 2(8086 と 80386)の比較では、Store 命令以外は 2.84～5.78 倍の高速化が達成されている。これはキャッシュメモリとパイプライン等の高速化アーキテクチャの相違に起因するものと考えられる。Store 命令の場合は、ストアバッファの相違により 15 倍もの高速化が達成されたと考えられる。このように、汎用 μ P を高性能版と入れ換えることにより、SIMD 型並列計算機のスカラ演算性能が大幅に向上することが確認できる。しかも、この場合、汎用 μ P は 8 6 系でデータバスも変わらないため、機械語命令変換プログラムの変換定義データの変更は不要である。

表 3 スカラ命令の実行時間比較

Table.3 Comparison of Execution time of

Variety of μ P's (Clock Frequency) (Width of Data-Bus)	Scalar Instruction		
	Case1(8086) 8 MHz 16 bit	Case2(80386) 16 MHz 16 bit	Case3(80386) 16 MHz 32 bit
Scalar Instruction	Execution Time (μ s)		
Load	3.5 μ s	0.5 μ s	0.25 μ s
Index	6.5	1.125	0.75
Compare	6.875	1.8125	0.375
Jump	1.875	0.625	0.625
Addition	6.375	1.5625	0.375
Multiplication	285.625	87.875	2.6875
Move	3.5	0.5	0.125
Divide	214.375	73.75	3.125
Store	3.75	0.25	0.125
Loop	4.625	1.625	1.5
And	6.375	1.5625	0.375
Increment	3.625	1.0625	0.125
Arithmetic	29.25	6.275	0.3125
Jump on Condition	2.375	0.8125	0.8125
Call	2.375	0.625	0.625

ケース 2 とケース 3（80386 のデータ幅が 16 ビットと 32 ビット）の比較では、演算が主である Multiply や Divide 命令では、それぞれ 32.7、23.6 倍となっているのに対して、Jump on Condition や Loop 命令では、それぞれ 1.0、1.08 倍となり、ほとんど高速化されていない。これは、演算を繰り返す場合にはデータ幅による差が顕著に表れるが、演算を伴わない制御系の命令はデータ幅に殆ど依存しないためと考えられる。この場合、データバス幅が異なり、アーキテクチャの一部変更となるので、変換定義データの変更が伴う。しかし、変換定義データの変更をとんでもなく、より高速な汎用 μ P をスカラプロセッサとすることで、SIMD 型並列計算機のスカラ演算性能の高速化を容易に達成できることも示して

いる。

また、CAP のスカラ命令をケース 1 と同じ汎用 μ P でエミュレーションした場合、その実行時間はケース 1 の数十倍から数百倍もかかった。エミュレーション方式に比べ本方式は優れていることが確認された。

4.5 画像処理ライブラリによる評価

表 4 に画像処理ライブラリを CAP で実行させた場合の実行性能を示す。測定時の CAP の仕様は次の通り。

- ・ セルラアレイ部：PE（8 ビット演算器）数=4,096
（64×64 の 2 次元格子接続）
動作周波数=13.3MHZ
- ・ CAP 制御部： μ P=80386（16MHZ）
リアルアドレスモード
データ幅 32 ビット

表では、スカラ、アレイおよび入出力の各演算をオーバーラップさせずに実行した場合と、提案した高速処理方式により、各演算をオーバーラップさせ並行実行した場合とを比較して示している。

表 4 画像処理ライブラリの処理性能

Table.4 Performance of Image Library

Functions	Execution Time(ms)		Efficiency of Parallel Execution(B/A)
	Parallel Execution(A)	Non Parallel Execution(B)	
Affine Transformation	2804.0ms	3171.0ms	1.13
Histograming	994.0	1040.0	1.05
Gray Scale Transformation	88.1	160.8	1.83
Fast Fourier Transformation	2.9	5.0	1.72
Average Filtering	108.8	229.2	2.11
Median Filtering	109.4	229.5	2.10

Parallel Exec. : Overlapping of Input, Output and Array Operation
Non Parallel Exec. : No Overlapping
Data Size
FFT : 4096 Point Complex number
Others : 1024 × 1024 pixel (8bit/pixel)

この結果、並行実行により 1.05 倍から 2.11 倍の性能向上が得られている。ヒストグラム作成と高速フーリエ変換は全体の処理時間に対してアレイ演算の実行時間の占める割合が高いこと、またアフィン変換と濃度変換は入出力の実行時間の占める割合が高いために一様重みフィルタやメジアンフィルタに比較して性能比が低くなっているが、並行実行による効果は一目瞭然である。

以上、4.3～4.5 に示した通り、提案した方式は SIMD 型並列計算機のスカラ演算性能の高速化と総合性能の向上に有効であることが確認された。

5. まとめ

独立型の SIMD 型並列計算機の制御部であるスカラプロセッサは、並列計算機自身の実行制御と並列プログラム中のスカラ演算を実行する重要な要素である。本論文で筆者らは、SIMD 型並列計算機の固有 ISP アーキテクチャの機械語命令で記述されている並列プログラムを汎用 μ P の機械語命令の並列プログラムに変換する方式と、

変換された並列プログラムを汎用 μ Pで高速処理する方式を提案した。本方式を筆者らが開発しているSIMD型並列計算機CAPに適用した結果、スカラ演算性能の汎用 μ Pの最新性能に追従した高速性および画像処理ライブラリの高高速化が得られ、高性能なSIMD型並列計算機を実現できた。また、固有ISPのスカラプロセッサ開発をなくしたこと、汎用 μ Pのソフトウェア開発環境の活用により、機械語命令変換プログラムの開発負荷を補って余りある開発の効率化も達成できた。以上のように、提案した高速処理方式は、SIMD型並列計算機のスカラ演算の高速化、総合性能の向上、将来への拡張性、および、開発効率向上に有効であることが確認できた。

今後、SIMD型並列計算機の並列プログラムの実行性能を一層高性能化するために、下記の検討を進める。

- ・スカラ命令以外の命令実行時に使用するランタイムライブラリの高高速化。
- ・並列プログラム中のマクロ機能を汎用 μ Pの機械語命令化。
- ・スカラ演算、アレイ演算およびデータ入出力の並列動作アルゴリズム高度化。

(平成13年4月16日受付, 同13年12月21日再受付)

参考文献

- 1) 富田眞治: 並列コンピュータ工学, 昭晃堂, 73, 1996
- 2) Kunio Uchiyama et al: SH5 The 64-Bit SuperH Architecture, IEEE Micro, July/Aug, 28/39, 2000
- 3) Srinivas K. Raman et al: Implementing Streaming SIMD Extensions on the Pentium III Processor, IEEE Micro, July/Aug, 47/57, 2000
- 4) 松田, 湯浅: SIMD型超並列計算機SM-1の概要, 情報処理学会, 計算機アーキテクチャ研究会, 92-ARC-95, 127/134, Aug.1992
- 5) G.M.Amdahl: Validity of the single processor approach to achieving large scale computing capabilities, AFIPS Spring Joint Computer Conf., Proceeding 30, 483/485, 1967
- 6) 馬場: SIMD型並列計算機, シミュレーション, Vol.11, No.1, 4/15, Mar.1992
- 7) 本郷: スーパーコンピュータの最近の進歩と進捗動向, 日本金属学会会報, Vol.31, No.7, 589/591, 1992
- 8) 清水, 内田: FUJITSU VP2000 シリーズのシステム概要, Fujitsu, Vol.41, No.1, 3/11, Jan.1990
- 9) Alexander Klaidier: The Technology behind Crusoe Processors, Transmeta Corporation, Jan, 2000
- 10) ヒリス, 喜連川 (著訳): コネクションマシン, パーソナルメディア, 94, 151, 1990
- 11) 磯西, 宮田, 岩瀬: セルラアレイ型大規模並列プロセッサのアーキテクチャ, 情報処理学会, 計算機アーキテクチャ研究会, 88-ARC-73, 61/68, Oct.1988
- 12) 宮田, 磯西, 菅, 岩瀬: Distributed Parallel Processor

for Satellite Image Processing (1), 電子情報通信学会, CPSY90-8, 25/32, 1990

岩瀬 正

(正員) 1942年生。1965年東北大学工学部電子工学科卒業。同年三菱電機(株)に入社。以来、汎用計算機、マイクロプロセッサの開発に従事。コンピュータアーキテクチャ、マイクロプロセッサに興味を持つ。

磯西徹明

(非会員) 1958年生。1981年新潟大学工学部情報工学科卒業。1983年同大学院工学研究科情報工学専攻修士課程修了。同年三菱電機(株)に入社。以来、並列計算機の開発に従事。1992~1994年には、MITメディアラボでマルチメディアの研究に従事。並列計算機アーキテクチャ、マルチメディアシステム、コンテンツに興味を持つ。

宮田裕行

(非会員) 1957年生。1980年京都大学工学部情報工学科卒業。1982年同大学院工学研究科情報工学専攻修士課程終了。同年三菱電機(株)に入社。以来、並列計算機、画像処理の開発に従事。1992年京都大学工学博士。並列計算機アーキテクチャ、並列アルゴリズムに興味を持つ。

小泉寿男

(正員) 1938年生。1961年東北大学工学部通信工学科卒業。同年三菱電機(株)に入社。以来、情報通信ソフトウェア、システム構築、システム評価の研究、開発に従事。1993年東北大学工学博士。1997年より東京電機大学理工学部教授。