2次元ジグザグ走査による相関値計算回路の設計とその評価*

上津 寛和[†]・中川 寛誉[‡]・神戸 尚志[§]

An Acceleration of the Correlation Coefficient Calculation using Two Dimensional Zigzag Tracing*

Hirokazu UETSU[†], Hirotaka NAKAGAWA[‡] and Takashi KAMBE[§]

Particle tracking technology is one of the most promising methods for the measurement of the velocity fluctuation of flows. It consists of the particle mask correlation (PMC) method and the KC method. The correlation coefficient calculation has the largest computational overhead in the PMC method. In this paper, a hardware of the correlation coefficient calculation is designed and accelerated by the zigzag tracing, the modification of the correlation coefficient equation, and parallel processing. The processing speed and circuit size of the design are evaluated.

1. はじめに

現在, 科学技術の分野で流れは様々な場面で出現する. 航空機・船舶・自動車などの交通機関の性能や安全性, あるいは高層ビルの耐風性や風害, 半導体チップの製造や冷却における流れの問題, あるいは医学における血流などの問題など, 流れ現象の解明と制御が科学技術の新たな展開のキーテクノロジーとなっている.

流れを観測するために, 従来は観測点となる場所にセンサを設置しその反応を測定する手法がとられてきた. しかしこの場合観測可能な点が限られ, 必ずしも精密な測定が行えるとは限らなかった. それに変わる手法として, トレーサとよばれるマーカ(粒子)を流れ場に挿入することで可視化を行い, これにデジタル画像処理技術を加え, 流れ場の瞬時・多点の速度情報を抽出するPIV(Particle Image Velocimetry: 粒子画像流速測定法)[2]とよばれる手法がある. これは粒子追跡システムともよばれ, 流れ場を空間的に, 多点同時に連続して測

定することが可能である点で優れている.

粒子追跡システムには様々な手法が提案されているが, 非常に多くの観測ポイントを同時に高精度に観測するには, 画像中の粒子の抽出とその追跡に高度な演算を多用する必要がある, 高速に解析を行うのが困難であった. このような高精度多点同時解析のアルゴリズムに, 粒子マスク相関法(以下PMC法: Particle Mask Correlation Methodとよぶ)およびカルマンフィルタ(Kalman-filter)と χ^2 検定(Chai-square Test)を組み合わせたKC法を用いた粒子追跡技術[1]がある. PMC-KC法は最高精度レベルの抽出・追跡能力を持つ反面, 高度な処理を多用し, 解析に多大な時間を要する. 筆者らはPMC-KC法をもとに, ソフトウェアとハードウェアで構成するシステムを設計し, 処理の高速化, システムの小規模化を進め, 高性能カメラと統合することによりリアルタイム画像解析システムを構築することを目指している.

筆者らが目標とするシステムではごく短い時間間隔で画像中に捕らえられた膨大な数のトレーサ粒子の移動を自動的に追跡し, 高性能カメラが流れ場の撮影を行うと同時に, 流れ場の計測を行うことが可能となる. さらにこれを応用し, 物体や流体の微細な変形や運動を画像計測することで, 従来技術では流体観測できなかった種々のマイクロ・スケールの高速現象やハードディスク等の高速回転体やそれに付随する高速な流れによって生じる境界層の構造の乱れ, 微細掘削加工時に生じる様々な高速現象, 人の運動の様子などを観測するスポーツ工学等,

* 原稿受付 2007年5月2日

[†] 近畿大学 大学院 総合理工学研究科 Graduate School of Science and Technology, Kinki University; 3-4-1, Kowakae, Higashi-Osaka city, Osaka 577-8502, JAPAN

[‡] (株)日立製作所 Hitachi, Ltd.; 6-6, Marunouchi 1-chome, Chiyoda-ku, Tokyo 100-8280, JAPAN

[§] 近畿大学 理工学部 電気電子工学科 Department of Electrical and Electric Engineering, Kinki University; 3-4-1, Kowakae, Higashi-Osaka city, Osaka 577-8502, JAPAN

Key Words: particle tracking, C based design, memory access, acceleration.

様々な科学的分野において従来の観測技術をはるかに凌駕する精密で高速な観測を行うことができるようになることを考える。

本論文では、粒子を抽出するPMC法に注目し、そのソフトウェアを解析、多大な計算をしている部分である相関値計算部に対してハードウェア化を行い、データアクセスの効率化、専用演算回路化、並列処理などにより高速化を図る。とくに、データアクセスでは、全画面を2次元的にジグザグ走査することによりメモリアccess回数を削減する。

ハードウェアの設計はシャープ(株)で開発されたBachシステム[3]を用いた。Bachシステムは現在、主に設計に用いられているハードウェア設計言語HDLよりも抽象度が高い、C言語によるアルゴリズムレベル記述からのハードウェア設計が可能である。この利点については3.に詳しく述べる。

本論文の構成は、2.で粒子抽出技術について、3.でハードウェア化手法について述べる。4.で相関値計算部のハードウェア設計とその高速化、5.で実装結果とその考察を示し、最後に、まとめと今後の課題を述べる。

2. 粒子抽出技術

粒子追跡技術の概要を述べるとともに、本論文で着目する粒子抽出のアルゴリズムについて述べる[1]。

2.1 粒子追跡システムの概要

粒子追跡技術は、流れ場を空間的に連続して測定する目的で開発されたものであり、粒子マスク相関法(PMC法)により、撮影画像(以下、粒子画像とよぶ)から粒子を抽出する。さらに、抽出した粒子情報をもとに、カルマンフィルタと χ^2 検定を組み合わせたKC法により、粒子を追跡する。

実際の計測現場では水中の粒子画像を1/15[s]間隔で撮影する。粒子画像はx軸方向1008ピクセル、y軸方向1018ピクセルの8bit輝度である。Fig.1はある時刻の粒子の流れを、粒子追跡システムを用いて追跡した様子である。多点同時に粒子追跡されている様子がわかる。粒子抽出を向上させるため、線形補完を用いて粒子画像を4倍に拡大する。このため、Fig.1のサイズは、2016×2036となる。

2.2 PMC法

PMC法の原理とアルゴリズムについて述べる。

2.2.1 PMC法の原理

PMC法はテンプレートマッチング法をもとにした粒子抽出法である。テンプレートマッチング法は画像から抽出しようとする対象物の形がわかっている場合、これをマスクとして画像との間で相関値を計算するものである。PMC法は円形に近い粒子画像の抽出のために開発された手法であり、粒子画像は対称な2次元正規分布で近似できるものとし、これをマスク(テンプレート)とし

Fig. 1 An example of the particle tracking technology

て粒子画像上に重ね、相関値を計算する。マスクと粒子画像の形状が等しければ、輝度は違ってても相関値はほぼ1になる。Fig.2に示すように、マスクを粒子画像上で走査させ、相関値を計算する。その値がある閾値以上になる部分に粒子が存在するとみなし、粒子位置を割り出す。相関値 r を求める計算は(1)式で表される。

$$r = \frac{\sum_{i,j}^{n \times m} (I_{i,j} - \bar{I})(M_{i,j} - \bar{M})}{\sqrt{\left[\sum_{i,j}^{n \times m} (I_{i,j} - \bar{I})^2 \right] \left[\sum_{i,j}^{n \times m} (M_{i,j} - \bar{M})^2 \right]}} \quad (1)$$

n, m : 水平,鉛直相関領域サイズ

i, j : 水平,鉛直位置

$I_{i,j}$: 粒子画像の輝度値

$\bar{I}$: 粒子画像の輝度値の平均値

$M_{i,j}$: マスク画像の輝度値

$\bar{M}$: マスク画像の輝度値の平均値

Fig. 2 Extraction of particles

2.2.2 PMC 法のアルゴリズム

PMC 法は以下の処理からなる。

- (1) 粒子画像の読み込み
粒子画像の輝度値を読み込む。粒子画像の輝度は 8bit であり、輝度値の範囲は 0~255 をとる。
- (2) 粒子画像を 4 倍
粒子抽出精度の向上のため、線形補完を用いて粒子画像を 4 倍に拡大する。
- (3) 相関値計算
マスク画像と粒子画像の相関値を (1) 式を用いて計算する。相関値計算は、粒子画像の全画面に対して行う。
- (4) 二値化処理
相関値が 0.7 値以上の領域を粒子、それ以下の領域を背景とし、二値化する。
- (5) 連結領域の抽出 (ラベリング)
個々の粒子画像を区別するため、二値化された粒子画像に対してそれぞれの画素の集まりに番号付けを行う。
- (6) 粒子の中心計算
求めた粒子の番号ごとに画素の集まりの中心を求め、その座標をその粒子位置とする。
- (7) 粒子位置情報の出力
粒子の個数, x 座標, y 座標, 大きさの情報を出力し、追跡のために KC 法に渡す。

3. ハードウェア化手法

本章では、Bach システムによるハードウェア化手法について述べる。

3.1 ハードウェアの利点

ソフトウェア処理は基本的に逐次的に行われる。それに対して、ハードウェアは独立した計算を並列に動作させることができる。これにより、ソフトウェアよりも高速に動かすことが可能となる。また、ソフトウェアを実行するには、PC や OS が必要となり、消費電力、システム規模などが大きい。それに対し、ハードウェアは、必要な処理を一つの LSI で実現すると、チップサイズはおおよそ数 mm 角、消費電力も PC を用いたソフトウェアシステムに比べるとはるかに小さくなる。

3.2 ハードウェア設計技術

ハードウェアの設計手法は、人が一つ一つのゲートの接続を記述していくゲートレベルの設計や、VHDL, Verilog-HDL 等のレジスタ転送 (RT) レベルのハードウェア設計言語 (HDL) を用いた設計が挙げられる。半導体技術の進歩により、一つの LSI に数千万ゲートの回路が搭載可能となったため、現在の主流は HDL 記述による RTL 設計である。近年、HDL よりもさらに抽象度の高い記述での回路設計が行われている [4,5]。これらは、動作合成システムといわれ、C 言語やそれに近いアルゴ

リズム記述からのハードウェアの設計を行う。本研究で使用した Bach システムもこの動作合成を実現したシステムの一つである。

3.3 Bach システムを用いた設計

Fig. 3 に示すように、従来の HDL によるハードウェア設計は、トップダウン式に行われているが、アルゴリズムの作成に C 言語等を用いて行い、その後記述したソフトウェアを実行することによる検証、HDL での回路設計、論理合成、シミュレーションによる検証という手順で行われる。しかし、HDL による記述とソフトウェア言語との抽象度の差は大きく、HDL で回路設計を行った後、不具合や変更によるアルゴリズムの修正には、大きな手間が必要となる。

Fig. 3 Design flow with Bach system

Bach システムを用いた設計では、トップレベルの設計から抽象度の高い BachC 言語を用い、機能設計・検証を行うことで、HDL による機能検証の手間が削減されるだけでなく、BachC 言語による記述から RTL の VHDL が自動生成され、ハードウェアの設計が大幅に効率化され、かつ各種のアーキテクチャを短期間に探索できる。BachC 言語はアルゴリズム設計でよく使用されている C 言語をもとに、ハードウェアアルゴリズムの設計に適するように、独自に以下の拡張を行ったものである [6]。

(1) ビット幅

変数宣言時に変数のビット幅を指定することができる。ビット幅は # を前置した数字を型名と変数名の間に記述して指定する。また、型変換演算子も同様の書式で指定する。

(2) 並列記述

BachC 言語では par 構文により、並列動作を明示的に記述できる。BachC 言語では、処理が par 構文に移ると、par 構文内の各ブロックまたは文を並列に実行し、全ブロックが終了した後、その par 構文の次の処理に移る。

(3) データ通信

また、同期チャネルまたは非同期チャネルを用いて並列動作するブロック間でデータ通信が可能である。同期チャネルや非同期チャネルは par 構文の外で chan や achan を用いて宣言する。同期チャネ

ルは、par 構文内の異なるブロックにある send 関数と receive 関数間でデータを転送する．同期チャンネルは 1 対 1 の片方通信であり，データ転送が完了するまで次の処理に移らない．非同期チャンネルは，このような制限がなく，いつでも読み書き可能であり，並列動作する複数のブロックから読み書き可能な共有変数として使用する．

4. 相関値計算部のハードウェア設計

PMC 法の高速化を検討するために，まず，Table 1 に示す粒子画像データ 2 枚程度を用いて，PMC 法を実装したソフトウェアによる処理時間を分析した．計算時間の大きい関数上位三つの計算時間の全体に占める割合を Table 2 に示す．粒子数は，ソフトウェア処理 (Pentium4：動作周波数 2.4GHz) における PMC 法全体で求めた数字であり，平均処理時間は 22 秒である．ここで，Avesd, Imask, Crosscoll は粒子追跡プログラム中の関数名である．相関値計算部は，(1) 式をもとに，粒子マスクの正規分布を計算し，対応する範囲での平均値を求め (関数 Avesd)，粒子マスクと対応する範囲を比較計算し (関数 Imask)，それぞれの結果をもとに相関値を求めている (関数 Crosscoll)．

Table 1 Input data

粒子画像	画像サイズ	粒子数
撮影 1	1008 × 1018	14231
撮影 2	1008 × 1018	16239

Table 2 Processing time of PMC method

関数名	処理時間の平均 [%]
Avesd	52.22
Imask	39.59
Crosscoll	5.62

この解析結果から，相関値計算部が，全体の処理時間に対する割合が 97% と高い．相関値計算部がボトルネックとなっているため，そのハードウェア化を行う．

ハードウェア化では，データアクセスの効率化，専用演算回路化，並列化等が重要である [8–12]．相関値計算部に対し，おのおのの点について以下の設計を行った．

4.1 データアクセスの効率化

システム LSI では，各種アプリケーションを高速に処理するため，データ記憶およびそれをアクセスするアーキテクチャの設計が重要である．たとえば，近年のオンチップメモリの増大傾向に見られるように，各種データの記憶法およびアクセス法が，コスト・性能などの点でシステム LSI 設計に大きな影響を与えている．

データアクセスアーキテクチャにおいて重要な点は，データアクセス時間がシステム LSI 性能のボトルネックとならないよう，オンチップにデータを保持しつつ，チッ

プサイズを増大させないようオンチップのメモリサイズを最小化することである．本論文では，ジグザグ走査によるメモリアccessの効率化を行った．

相関値処理の際，ソフトウェアでは，関数 Avesd, Imask, Crosscoll は，各処理を全画面分について行い，結果をメモリに記憶するという方法をとっていた．1 マスクごとにおのおのの計算を行い，最後にその結果をメモリに記憶するよう回路構成を変更し，メモリアccess数を削減する．以下，これを 1 マスク処理とよぶ．Fig. 4 に 1 マスク処理と逐次処理の比較を図示する．

Fig. 4 One mask processing and serial processing

マスク画像のサイズが 7×7 画素であるため，相関値計算を行うには $7 \times 7 = 49$ 画素の粒子画像データをメモリから読み込む．つぎに，対象領域を x 軸方向右に 1 画素分ずらしてこの計算を繰り返す．このとき，新たに粒子画像内対象領域となった画素データをメモリから読み出す必要がある．しかし k 回目と $k+1$ 回目では， k 回目に使用した粒子画像データは左端 1 列以外 $k+1$ 回目でも使用可能である． 7×7 画素のデータを横 7 画素 1 行分 $\times$ 7 本のシフトレジスタに記憶することによって，メモリアccess回数を $1/7$ とすることができる．これをシフトレジスタ利用とよぶ．この走査方法を y 軸方向にも適用する．すなわち， y 軸方向に i 画素分走査した後， x 軸方向に 1 列シフトする． y 軸方向に走査するとき用いるシフトレジスタを増設することで，メモリアccess回数を削減できる．以下，これを i ジグザグ走査とよぶ． 7×7 画素単位で相関値計算を行うため，上下端 3 行，左右端 3 列は計算対象から除く．このため， 7×7 画素の中心位置を (y 行目, x 列目) とすると， y は 4～2033， x は 4～2013 の範囲となる．

i ジグザグ走査のアルゴリズムを以下に示す．

7 画素分のシフトレジスタを $7 + (i-1)$ 行分用意し， m について 0 から $\lfloor 2030/i \rfloor - 1$ まで，(1)，(2) を繰り返す．ここで， m はジグザグ走査をカウントする変数．

- (1) $(m \times i + 1)$ 行から $\{(m \times i + 1) + 6 + (i-1)\}$ 行まで，1 列目から 7 列目までの画素データをシフトレ

ジスタに読み込む。

- (2) n について 1 から 2010 まで (a), (b) を繰り返す。
 - (a) j について 1 から i まで, $j \sim (j+6)$ 行のシフトレジスタに対し, 座標 $\{n+3, (m \times i + 1) + 3 + j - 1\}$ に対する相関値計算を行う。
 - (b) シフトレジスタのデータを 1 画素分左シフトし ($n = n + 1$), 各シフトレジスタの最右列に 1 列分だけ画素データを読み込む。

3 ジグザグ走査 ($i = 3$) における $m = 4, 5$ の場合の動作を Fig. 5 に示し, $m = 4$ の場合の処理手順を上記のアルゴリズムの各ステップに沿って説明する。

(0) x 軸方向 7 画素分のシフトレジスタを $7 + (3 - 1) = 9$ 本用意する。

(1) シフトレジスタに 1 列目から 7 列目までの画素データを読み込む。 $m = 4$ のとき, 読み込むデータは粒子画像中の $(4 \times 3 + 1) = 13$ 行から $((4 \times 3 + 1) + 6 + (3 - 1)) = 21$ 行の x 軸方向 7 画素分である。

(2)(a) y 軸方向 1 回目の相関値計算は, シフトレジスタの 1 から $(1 + 6) = 7$ 行目の画素データを用いる (Fig. 5, $j = 1$ の図)。このときの対象領域の中心座標は, $(1 + 3), ((4 \times 3 + 1) + 3 + 1 - 1) = 4, 16$ となり, この座標についての相関値が求まる。

(2)(a) y 軸方向 2 回目の相関値計算は, 2 から $(2 + 6) = 8$ 行目の画素データを用い (Fig. 5, $j = 2$ の図), 座標は $(1 + 3), ((4 \times 3 + 1) + 3 + 2 - 1) = 4, 17$ 。

(2)(a) y 軸方向 3 回目の相関値計算は, 3 から $(3 + 6) = 9$ 行目の画素データを用い (Fig. 5, $j = 3$ の図), 座標は $(1 + 3), ((4 \times 3 + 1) + 3 + 3 - 1) = 4, 18$ 。

(2)(b) y 軸方向の走査が終了すると, 9 本の各シフトレジスタの画素データを左に 1 画素分シフトさせ, 各シフトレジスタの最右画素にそれぞれ新規画素データを読み込む (Fig. 5, 右下の図)。

(2) マスク画像が粒子画像の右端に達するまで (a), (b) を繰り返す。

以上のように, 3 ジグザグ走査では 1 回のレジスタ記憶によって 3 回の相関値計算を行い, 右端 1 列分の画素データ追加だけで, さらに 3 回の計算を行うことができる。

2030/ i が割り切れないとき, 粒子画像下端の余った行について, 別途, ジグザグ走査を制御する回路を用意する。3(4) ジグザグ走査に, 共通の相関値計算回路に対し, 3(4) ジグザグ処理と下端用 2 ジグザグ処理との二つの制御回路を設計した。一つの制御回路の中で切り替えを行う方法もあるが, 毎回, 切り替えの判定が行われるため, 処理時間への影響が無視できない。このため, 本論文では二つの制御回路を設計した。

i ジグザグ走査回路による処理時間の見積もりは, (2) 式のようになる。

$$T = (Py/i)(Px(R + C \times i)) - Td \quad (2)$$

Fig. 5 Three zigzags scanning

T : 粒子画像 1 枚の処理時間

Px : 粒子画像の x 軸方向の画素数

Py : 粒子画像の y 軸方向の画素数

R : step(2-b) におけるメモリアクセス時間

C : 相関値計算 1 回にかかる処理時間

Td : 二次微分による処理削減時間

ここで, 二次微分については次節で述べる。

Td は, 対象となる粒子画像に依存するが, i には依存しない。今回の実験に用いた粒子画像では Td の平均が 77.7[ms] となった。 R は厳密には i に依存するが, その影響は少ないので, ここでは定数とする。

i の増加に伴い処理時間は減少していくが, その変化量は徐々に減少する。 i の増加に伴い, 必要となるシフトレジスタ数が増加するため, 回路規模はほぼ線形に増加すると予測できる。

4.2 専用演算回路化

数値演算などを繰り返し行う場合, その処理時間が全体のボトルネックとなることが多い。ハードウェアによる乗除算, 各種関数, 複合演算などは, 特に回路規模や処理時間に影響が大きい。C 言語設計における専用回路化の手順を以下に示す。

- (1) プログラムを解析し, 処理時間を要する演算部分を抽出する。
- (2) 浮動小数点演算回路は大規模になるため, 固定小数点演算回路を用い, 各変数について浮動小数点と同等な精度を得るビットを決定する。
- (3) 計算式の変形を行う。除算, 平方根, 平均値計算などを含む複雑な計算式に対しては, 数学的に等価な計算に書き換え, 簡単化を行う。
- (4) 減算シフト法など各種ハードウェアアルゴリズム [7] を Bach C で関数記述し, これをコールする。
- (5) 処理性能を上げる必要がある場合には, 以下の三

つの方法を適用する．

(a) 計算範囲の限定

演算途中の結果や演算する対象データに対する識別などの前処理を追加することにより，演算量を削減する．

(b) preserve プラグマの指定

Bach システムは，デフォルトではスレッド内で演算器の共有化を行う．高速化したい演算に対し，preserve プラグマを指定し，演算器を専用化する．

(c) 演算にループがあれば，throughput プラグマを用いて自動パイプライン化を行う．

本論文では，二次微分による計算範囲の限定，相関値判別式の変形と preserve プラグマの指定を適用する．

4.2.1 二次微分

粒子画像は粒子と背景からなり，二次微分によって背景と識別された画素を相関値計算を省くことでメモリアクセスを削減する．二次微分は，注目画素を中心に前後の画素との輝度値から (3) 式によって求める．二次微分値は，粒子のような輝度値が大きい画素では負の大きな値をとる．注目する画素に対し，その二次微分値が閾値（実データによる実験から -4 とした）より大きい場合，背景と判断し，相関値計算をスキップする．

$$g = (f(x+1) - f(x)) - (f(x) - f(x-1)) \quad (3)$$

x : 注目画素の粒子画像上の x 軸位置

$f(x)$: 位置 x の輝度値

また，マスクの幅が 7×7 のため，幅を 7 に設定し計算式を (三つ後の画素 - 中心画素) - (中心画素 - 三つ前の画素) とすることで，より粒子の大きさを反映した二次微分値になり，正確に正負の判別が可能となる．Fig. 6 に二次微分の簡単な例を示す．また， x 軸方向だけでなく y 軸方向に対しても二次微分を行う．

4.2.2 相関値判別式の変形と preserve プラグマの指定

2.2.2 節 (4) で述べたように相関値が 0.7 以上のとき，粒子と判定する．これに基づき，計算式 (1) を以下のように変形し，平方根計算，除算をなくすことで，処理時間，回路規模の削減を行う．式変形の詳細は付録 1. に示す．

$$\begin{aligned} & (0.7)^2 \times (n \times m \sum_{i,j} I_{i,j}^2 - (\sum_{i,j} I_{i,j})^2) \\ & \times (n \times m \sum_{i,j} M_{i,j}^2 - (\sum_{i,j} M_{i,j})^2) \\ & < (n \times m \sum_{i,j} I_{i,j} M_{i,j} - (\sum_{i,j} I_{i,j})(\sum_{i,j} M_{i,j}))^2 \quad (4) \end{aligned}$$

Fig. 6 Brightness distribution of particles

(4) 式は平均値計算を展開し，for ループを削減した．また，この変形により除算と平方根計算がなくなったため値を整数のみで表すことができる．(1) 式を用いる場合， Σ の乗算部は浮動小数点演算と同等の精度を得るために，Table 1 の粒子画像による実験によって求めたところ，40 ビットの固定小数点精度が必要となった．しかし，(4) 式は整数計算の精度問題となり，同様の実験の結果，unsigned int 型の 8 ビットの乗算で計算可能となり，計算時間と演算器の回路規模を大幅に削減できる．ただし，本論文ではオーバーフローを防ぐため unsigned int 型の 16 ビットで計算する．相関値判別式で最も使用される乗算器の各ビット幅での処理時間と回路規模の比較を Table 3 に示す．

Table 3 Processing time and gate size of multipliers with different bit length

ビット幅	処理時間 [ns]	回路規模 [gates]
16	5.25	690
32	9.75	3,088
40	14.21	9,466

乗算回路規模の削減と高速化により，for ループの展開を行い，処理の並列化が可能となった．また，preserve プラグマの指定による高速化も適用した．

4.3 粒子画像分割による並列化

処理の並列化には，データ分割による並列化，パイプライン化などがある．データ分割による並列化は，大規模なデータに対して有効な場合があるが，並列化数だけ回路規模が増大する．また，分割境界線上で計算が正しく行われるようデータにオーバーラップを持たせるなど付加的な処理が必要になる場合がある．パイプライン化は回路規模への影響が少なく，性能向上に有効であるが，各ステージの処理時間の平均化やパイプライン段数の決定が問題である．

本研究では、粒子画像を分割してそれぞれ別の外部メモリに格納し、分割数分の相関値計算回路を用意することで、並列処理を行う。粒子画像を分割する際に生じる境界部分においてオーバーラップ部（重複）分を、マスク画像の中心と最下位の行との距離が3ピクセルとなることから、3行分設け、精度を保つ。粒子画像分割による並列化（2分割の場合）の様子を Fig. 7 に示す。

このとき、処理結果を格納する RAM にメモリアクセスの競合が起こる危険がある。それを防ぐため、この RAM も同時に分割する。

Fig. 7 Parallel processing of image plane

5. 実装結果と考察

相関値計算部について前章で述べた手法を回路に実装し、処理速度と回路面積を求めた。処理時間は Model Sim によるシミュレーションを行った結果であり、回路面積は Design Compiler により合成した結果である。回路の動作周波数は 100MHz、ライブラリは HITACHI0.18 μ m を用いた。

5.1 ジグザグ数の評価

ジグザグ走査回路を実装し、最も適したジグザグ数を求める。(2) 式の i が 2, 3, 4, 5, 7, 10 の場合を実装した結果を Table 4 に示す。各回路は、前節で述べた手法中、並列処理以外すべてを適用した。

Table 4 における処理時間に関して、(2) 式による理論値と実装結果の比較は、Fig. 8 に示すように、ほぼ一致

Table 4 Processing time and gate size of zigzags scanning circuits

ジグザグ数	処理時間 [ms]	回路規模 [gates]
1	372	98,734
2	228	106,277
3	180	133,913
4	155	145,149
5	142	120,769
7	126	133,335
10	113	149,124

Fig. 8 Processing time of zigzags tracing

Fig. 9 Gate size of zigzags tracing

している。 $i=1$ の回路に比べ、10 ジグザグ処理回路は、処理時間が約 30% に削減され、回路規模は約 1.5 倍に留まっている。

回路規模に関する理論値と実装結果の比較は、Fig. 9 に示すように、ジグザグ数 i の線形に比例している。これは、 i の増加に伴い、レジスタと相関値計算回路の実行などを担う制御回路の複雑化により回路規模が増加するためと考えられる。ただし、 $i=3, 4$ の回路規模は、近似線が示す値よりやや大きくなった。4.1 節で述べたように、粒子画像最下端部のジグザグ走査を行うために別途、制御回路が必要となったためである。また、Fig. 8 に示した 66ms のラインは、リアルタイム処理における処理時間である。

ジグザグ数 7, 10 の回路は、4.3 節で述べた粒子画像分割による並列化と組み合わせると処理時間を半分とすることができ、Fig. 8 に示した 66ms のラインを切ることができる。粒子画像分割による並列化を用いずに、処理時間が 66ms 以下となるまでジグザグ数を増加させる場合、4.1 節で述べたようにジグザグ数の増加に対し処理時間の減少量は小さくなり、Fig. 9 のように回路規模は線形に増加していくことから、ジグザグ数は 10 以下が実用的である。

これらの実験と考察から、ジグザグ数 $i=7$ および 10 が処理時間・回路規模の両面から相関値計算回路の最も有力な候補であるとした。次節では、この二つの回路を用いて、各種高速化手法を実装した結果を示す。

5.2 高速化手法の評価

つぎに、以下に9種類の回路の実装結果を示す。

- A) 逐次処理回路
- B) 1 マスク処理
- C) 相関値判別式の変形とシフトレジスタ利用
- D) C) + 相関値判別式の専用回路化
- E) D) + 二次微分による粒子判別
- F) E) + 7 ジグザグ処理
- G) E) + 10 ジグザグ処理
- H) F) + 粒子画像の水平2分割による並列化
- I) G) + 粒子画像の水平2分割による並列化

Table 5に、二つの粒子画像 (Table 1) に対する A)～I) の回路における処理時間と回路面積を示す。撮影条件への依存性を排除するため、粒子画像と同じサイズの画像に対し、粒子数、粒子の輝度および大きさをランダムに発生するソフトウェアを用いて五つの粒子画像データ (Table 6) を作成し、同様に実験を行った (Table 7)。ソフトウェアによって抽出した粒子数と本回路により求めた粒子数に、数個の差がある。これは、ソフトウェアによる相関値計算が浮動小数点であるのに対し、回路は固定小数点演算であるため、閾値に非常に近い値の判定に差が生じたためである。その差は0.01%以下であり、問題ない範囲である。2次微分は粒子数によって計算時間が異なるので、回路EからIについては処理時間がややばらつくが、ほぼ同じ傾向であることを確認した。

逐次処理はオリジナルプログラムのアルゴリズムをそ

Table 7 Processing time of generated input data[ms]

回路	自動1	自動2	自動3	自動4	自動5
A	39,913	39,913	39,913	39,913	39,913
B	2,556	2,556	2,556	2,556	2,556
C	820	820	820	820	820
D	450	450	450	450	450
E	376	376	376	376	382
F	131	131	131	131	131
G	118	118	118	119	119
H	65	65	65	65	66
I	60	60	60	60	60

のまま使用し、ハードウェア化を行った。逐次処理回路Aでは1枚の粒子画像の相関値計算に、約40[s]の処理時間を要した。これは膨大なメモリアクセスがボトルネックとなったためである。相関値判別式の変形とシフトレジスタ利用を用いた回路Cは、1マスク処理のみを実装した回路Bに比べ回路規模は約半以下となり、処理速度は約3倍となった。相関値判別式の専用回路化を実装した回路Dは、回路Cに対し、約50%の高速化、2次微分による粒子判別を実装した回路Eは、回路Dに対し、約17%高速化された。ジグザグ走査によって、7ジグザグ処理回路Fと10ジグザグ処理回路Gは、回路Eを約3倍に高速化し、ともにリアルタイム処理に必要な処理時間である66msに対して、2倍以下の処理時間を実現した。両回路に対して粒子画像2分割による並列化を実装した回路HおよびIは、回路規模は約2倍となったが、相関値計算部のリアルタイム処理を実現できた。

6. まとめと今後の課題

本論文は、ソフトウェアのみで構成された相関値計算部を、C言語設計によりハードウェア化を行い、設計した回路の処理時間と規模を測定した。ハードウェア化した相関値計算部による粒子抽出はソフトウェアのみで行ったときと、ほぼ同等の粒子抽出精度を確保している。相関値計算回路について、ジグザグ走査によるデータアクセス効率化、専用演算回路化、並列化等について、8種類的高速化手法を実装し、目標とするリアルタイム処理を実現する回路を設計することができた。

また、ジグザグ走査回路の設計・検証は約2人月で完了した。抽象度の高い言語を用い、機能設計・検証を行うことで、ハードウェアの設計が大幅に効率化され、各種のアーキテクチャを短期間で探索し、設計の最適化を図ることができた。

今後は、PMC法の相関値計算以外の部分とKC法のハードウェア設計を進め、リアルタイム動作可能な粒子追跡システムを実現する予定である。

Table 5 Experimental results

回路	画像1 処理時間 [ms]	画像2 処理時間 [ms]	回路規模 [gates]
A	39,913	39,913	59,097
B	2,556	2,556	218,345
C	820	820	93,456
D	450	450	109,667
E	377	376	98,734
F	132	136	133,335
G	120	124	149,124
H	66	70	270,273
I	60	64	288,784

Table 6 Input data generation

粒子画像番号	粒子数 (ソフト)	粒子数 (回路)
自動1	14424	14422
自動2	14552	14548
自動3	14540	14536
自動4	14682	14681
自動5	14956	14953

謝 辞

Bach を用いたハードウェア設計に対し、ご支援を頂いたシャープ(株)電子デバイス開発本部先端技術開発研究所の BACH 開発グループの皆様にご心から御礼申し上げます。本研究は東京大学大規模集積システム設計教育研究センターを通し、シノプシス(株)の協力で行われたものです。

参 考 文 献

- [1] 江藤ほか：PTV のための粒子画像抽出法に関する検討－粒子マスク相関法について－；水工学論文集，Vol. 40，pp. 1051–1057 (1996)
- [2] 可視化情報学会編：PIV ハンドブック，森北出版 (2002)
- [3] K. Okada, et al.: Hardware algorithm optimization using Bach C; *IEICE Trans. Fundamentals*, Vol. E85-A, No. 4, pp. 835–841 (2002)
- [4] 藤田編著：システム LSI 設計工学，オーム社 (2006)
- [5] H. Kurokawa, et al.: Study and analysis of system LSI design methodologies using C-based behavioral synthesis; *IEICE Transactions on Fundamentals*, Vol. E86-A, No. 4, pp. 787–798 (2003)
- [6] Bach システムマニュアル，シャープ(株)提供 (2003)
- [7] 高木：算術演算の VLSI アルゴリズム，コロナ社 (2005)
- [8] T. Kambe, et al.: A C-based synthesis system, Bach, and its application; *Proc. of ASP-DAC* (2001)
- [9] T. Kambe, et al.: C-based design of a real time speech recognition system; *ISCAS2006* (2006)
- [10] K. Jyoko, et al.: C-based design of a particle tracking system; *SASIMI2006* (2006)
- [11] 小田島ほか：C 言語設計によるリードソロモン符号復号化回路の最適化；情報処理学会システム LSI 設計技術研究会 (2006)
- [12] 大窪ほか：特定用途向け低ビット複合演算回路設計；電子情報通信学会 VLSI 設計技術研究会 (2005)

付 録

付録 1. (1) 式から (4) 式への変形

相関値 r を 0.7 以上とし，(1) 式の両辺を自乗すると以下の式になる。

$$(0.7)^2 < \frac{\left[\sum_{i,j}^{n \times m} (I_{i,j} - \bar{I})(M_{i,j} - \bar{M}) \right]^2}{\left[\sum_{i,j}^{n \times m} (I_{i,j} - \bar{I})^2 \right] \left[\sum_{i,j}^{n \times m} (M_{i,j} - \bar{M})^2 \right]} \quad (\text{A1})$$

(A1) 式の分母について， $\frac{1}{n} \sum_j^n x_j = \bar{x} \leftrightarrow \sum_j^n x_j = n\bar{x}$

より

$$\begin{aligned} & \sum_{i,j}^{n \times m} (I_{i,j} - \bar{I})^2 \\ &= \sum_{i,j}^{n \times m} (I_{i,j}^2 - 2I_{i,j}\bar{I} + \bar{I}^2) \\ &= \sum_{i,j}^{n \times m} I_{i,j}^2 - 2\bar{I} \sum_{i,j}^{n \times m} I_{i,j} + \sum_{i,j}^{n \times m} \bar{I}^2 \\ &= n \times m \bar{I}^2 - 2n \times m \bar{I}^2 + n \times m \bar{I}^2 \\ &= n \times m (\bar{I}^2 - \bar{I}^2) \\ &= n \times m \left(\frac{1}{n \times m} \sum_{i,j}^{n \times m} I_{i,j}^2 - \left(\frac{1}{n \times m} \sum_{i,j}^{n \times m} I_{i,j} \right)^2 \right) \\ &= \sum_{i,j}^{n \times m} I_{i,j}^2 - \frac{1}{n \times m} \left(\sum_{i,j}^{n \times m} I_{i,j} \right)^2 \quad (\text{A2}) \end{aligned}$$

同様に，

$$\sum_{i,j}^{n \times m} (M_{i,j} - \bar{M})^2 = \sum_{i,j}^{n \times m} M_{i,j}^2 - \frac{1}{n \times m} \left(\sum_{i,j}^{n \times m} M_{i,j} \right)^2 \quad (\text{A3})$$

また，(A1) 式の分子について，

$$\frac{1}{n} \sum_j^n x_j = \bar{x} \leftrightarrow \sum_j^n x_j = n\bar{x},$$

$$\overline{xy} = \frac{1}{n} \sum_j^n x_j y_j \quad \text{よ り}$$

$$\begin{aligned} & \sum_{i,j}^{n \times m} (I_{i,j} - \bar{I})(M_{i,j} - \bar{M}) \\ &= \sum_{i,j}^{n \times m} (I_{i,j}M_{i,j} - \bar{I}M_{i,j} - I_{i,j}\bar{M} + \bar{I}\bar{M}) \\ &= \sum_{i,j}^{n \times m} (I_{i,j}M_{i,j}) - \bar{I} \sum_{i,j}^{n \times m} M_{i,j} - \bar{M} \sum_{i,j}^{n \times m} I_{i,j} + \sum_{i,j}^{n \times m} \bar{I}\bar{M} \\ &= n \times m \bar{I}\bar{M} - n \times m \bar{I}\bar{M} - n \times m \bar{I}\bar{M} + n \times m \bar{I}\bar{M} \\ &= n \times m (\bar{I}\bar{M} - \bar{I}\bar{M}) \\ &= n \times m \left(\frac{1}{n \times m} \sum_{i,j}^{n \times m} I_{i,j}M_{i,j} \right. \\ & \quad \left. - \left(\frac{1}{n \times m} \sum_{i,j}^{n \times m} I_{i,j} \right) \left(\frac{1}{n \times m} \sum_{i,j}^{n \times m} M_{i,j} \right) \right) \\ &= \sum_{i,j}^{n \times m} I_{i,j}M_{i,j} - \frac{1}{n \times m} \left(\sum_{i,j}^{n \times m} I_{i,j} \right) \left(\sum_{i,j}^{n \times m} M_{i,j} \right) \quad (\text{A4}) \end{aligned}$$

(A2),(A3),(A4) 式を (A1) 式に代入し，分母分子に $n \times m$ をかけ，右辺の分母を左辺に移動する。

$$(0.7)^2 \times (n \times m \sum_{i,j}^{n \times m} I_{i,j}^2 - \left(\sum_{i,j}^{n \times m} I_{i,j} \right)^2)$$

$$\begin{aligned}
& \times (n \times m \sum_{i,j}^{n \times m} M_{i,j}^2 - (\sum_{i,j}^{n \times m} M_{i,j})^2) \\
& < (n \times m \sum_{i,j}^{n \times m} I_{i,j} M_{i,j} - (\sum_{i,j}^{n \times m} I_{i,j})(\sum_{i,j}^{n \times m} M_{i,j}))^2
\end{aligned}
\tag{A5}$$

以上の変形により，(4) 式が求められる．