社団法人　情報処理学会　研究報告
IPSJ SIG Technical Reports

2005－ＳＬＤＭ－120（１）
2005／5／19

# 粒子追跡におけるKC法のハードウェア化
## —カルマンフィルタ、$x^2$検定処理の最適化—

上甲　憲市† 　大口　貴裕 　小西　徹也 　大倉　崇宜 　神戸　尚志‡

近畿大学大学院　総合理工学研究科　エレクトロニクス系工学専攻　システム設計工学研究室

〒577-8502 大阪府東大阪市小若江 3-4-1　30 号館 5F

E-mail: †0533340409y@kindai.ac.jp, ‡tkambe@ele.kindai.ac.jp

**あらまし**　粒子追跡技術は、流れ場を空間的に連続して測定すること目的とし、様々な手法が提案されている。本論文では粒子マスク相関法(PMC 法:Particle Mask Correlation Method)と、カルマンフィルタ(Kalman-filter)と$x^2$検定(Chai-square Test)を組み合わせた KC 法を用い、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の実現について述べる。

**キーワード**　カルマンフィルタ法，$x^2$検定，粒子マスク相関法，Bach システム

# Hardware Acceleration for KC method

Kenichi　JYOKO†, Takahiro OHGUCHI, Takanori OHKURA,

Tetuya　KONISHI and　Takashi　KAMBE‡

System Design Methodology Laboratory, Kinki University, 3-4-1 kowakae, Higashi-Osaka City, Osaka, 577-8502 Japan

E-mail: †k20arcclemon@hotmail.com, ‡tkambe@ele.kindai.ac.jp

**Abstract**　Various techniques are proposed to track particles in the field continuously . In this paper, a software and hardware system based on the KC method that combines the particl e mask correlation method with the kalman filter and Chai-square Test is designed. And the processing speed and the area of the system are evaluated.

**Keyword**　Bach system , Particle Mask Correlation method , Kalman-filter , Chai-square Test, software and hardwar e codesign, C based design

## 1. はじめに

粒子追跡技術は、ある時間間隔で画像中の各トレーサ粒子の移動を自動的に追跡し、流れ場を計測する。この技術を応用し、物体や流体の微細な変形や運動を画像計測することができる。応用例として流体観測できなかった種々のミクロ・スケールの高速現象、ハードディスク等の高速回転体やそれに付随する高速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等を立体的スローモーションで観察することが可能になると考えられる。

本研究では、まずC言語で記述された粒子マスク相関法とカルマンフィルタ (Kalman-filter) と$x^2$検定 (Chai-square Test) を組み合わせたKC法からなる粒子追跡プログラムを解析し、計算時間を要している部分に対してハードウェア化を行い、ソフトウェア・ハードウェア協調システムを設計すると共に、ハードウェア化したKC法回路の高速化・回路規模の削減を検討する。

本文では、特にKC法回路について、カルマンフィルタと$x^2$検定のパイプライン処理、カルマンフィルタの並列処理やメモリの削減など4種類の高速化について提案し、実装により、その性能を評価する。

## 2. 粒子追跡技術とハードウェア化

### 2.1 KC法の概要

カルマンフィルタ法は、確立・統計学の基本に基づいたカルマンフィルタ理論を粒子追跡に適用した手法である。基本的な考え方として、まず、粒子像座標、速度、加速度などを状態量として選び、それらの動的システムの線型方程式を状態方程式に設定する。また、実際に計測できる粒子像座標、移動距離などを観測量として選び、状態量と観測量を関係付ける観測方程式を設定する。状態方程式、観測方程式が決まれば、カルマンフィルタ理論により逐次最適推定値を求めるこ

とができる。

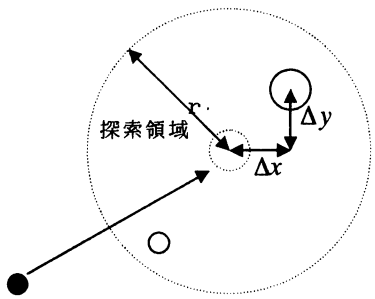
一般的に離散型の線形動的システムは次式で表される。

$$h(t+1) = \Phi(t+1|t)h(t) + w(t) \qquad (2.1)$$

$$m(t) = D(t)h(t) + v(t) \qquad (2.2)$$

ここで、$h(t)$は時刻$t$の状態量ベクトル、$m(t)$は観測量ベクトル、$\Phi(t+1|t)$は時刻$h(t)$から$h(t+1)$への遷移行列、$D(t)$は$m(t)$と$h(t)$を結びつける観測行列、$w(t)$は状態方程式における誤差ベクトル、$v(t)$は観測方程式における誤差ベクトルである。カルマンフィルタでは状態量ベクトル$h(t)$の最小二乗推定値を、時々刻々得られる観測量ベクトル$m(t)$から逐次推定し、次時刻の状態量の最適推定値を予測することができる。

ある時刻tに粒子像座標、速度、加速度などの状態量がわかっていれば、次時刻t+1の最適状態量がカルマンフィルタによって推定される。その最適推定値と観測地から、同一粒子像の同定を行う。同定ができれば、観測データからさらにt+2時刻の状態量の最適推定値が求められ、さらに同様の操作を繰り返し行うことにより同一粒子像を時々刻々追跡していく。同定法として図 2.1 のように2時刻間で$x^2$検定を用いる方法を使用している。




カルマンフィルタによる予測

○ :(t+1)時刻の実測粒子位置

◌ :カルマンフィルタによる(t+1)時刻の実測粒子位置

● :t時刻の粒子位置

図 2.1 カルマンフィルタ・$x^2$検定法の概念図

Fig.2.1 Concept of Kalman-filter・Chai-square Test

具体的には次のような手順となる。

1. カルマンフィルタによりt画面上にある粒子の(t+1)画面上における位置を推定する。
2. (t+1)画面上において推定された粒子位置から半径rの検索領域以内にある粒子すべてについて$x^2$検定を行う。$x^2$値が最小となるものを仮の対応する粒子とする。
3. $x^2$値がある棄却水準以下であれば同一粒子と確定する。棄却水準以上であれば対応する粒子がないものとする。

以上の計算を画面上すべての粒子に対して行う。

## 2.2 ハードウェア設計技術

図 2.2 に示すように、現在の HDL によるハードウェア設計は、C 言語等を用いてアルゴリズム設計・検証を行い、その後、HDL での回路設計、RTL 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 での記述とプログラミング言語との差は大きく、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、おおきな手間が必要となる。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 Bach-C を用い、機能設計・検証を行うことで、HDL による記述・検証期間が削減されるたけでなく、Bach-C 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る。

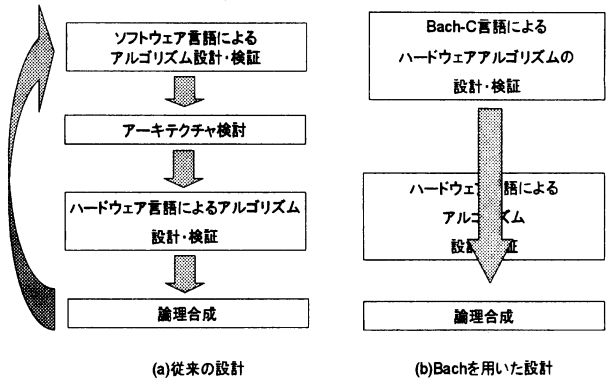



図 2.2 Bach の 設計工程

Fig.2.2 Design process using Bach system

## 3. KC 法のハードウェア化

一般にハードウェアは、高速化、低消費電力化のために用いられ、ソフトウェアは汎用性を持たせて処理を行いたい時に効果的である。ソフトウェアをハードウェア化するには、構造によって適切な部分が異なる。他の関数による影響が少なく処理が独立している関数をハードウェア化することにより、全体の処理を大きく変更することなくハードウェアとソフトウェアを組み合わせてシステムを実現することが出来る。

### 3.1 ハードウェア化箇所の決定

ハードウェアとソフトウェアを組み合わせた効率の良いシステムを構築するために、元となるソフトウェアの理解、分析を行った。我々は、以下の手順で KC 法の各関数の役割を調べ、構造を解析し、関数の処理時間を計測し、ハードウェア化に適切な箇所を決定した。

(a) プログラム構造の解析　全体の処理を把握し、

ハードウェア化に適した部分を決定するため、ソースファイルの解析にプログラムのプロファイリングツール「gprof」を用いて関数のコール関係を調べ、その役割と繋がりを調べる。

(b) 処理時間の分析　ソフトウェアで計算時間を多く必要としている部分を求めるため、(a)で用いたgprof から関数処理時間を計測し、処理時間の割合が大きい関数を抽出する。しかし、gprof の出力結果はANSI-C が用意している標準関数の処理時間はカウントされない。そのためプログラム中に clock 関数を使用した測定方法も行う。

KC 法で 6000 個の粒子データを実行し、gprof と clock 関数で各関数の処理時間を分析した。KC 法計算部はカルマンフィルタを計算(Kalman 関数)し、$\chi^2$ 検定(Chai 関数)を行い、粒子を複数の粒子と対応付けていないかチェック(Check 関数)を行い、対応付かなかった粒子への粒子情報の内挿のため、対応付いた粒子を使った平均空間を求めている(Spav 関数)。プログラム構造の解析と処理時間の分析により、Chai と Spav が全体の処理時間に対する割合が高い。このことより Chai、Check、Spav をハード化することが有効と考えられる。また、各関数が粒子情報を用いて計算するため、独立性が低い。そのため KC 法全体に注目し、ハードウェア化を検討する。

### 3.2 システム構造の設計

我々は図 3.1 のようなシステムを設計した。このシステムは、主にソフトウェア部、通信部、ハードウェア部からなり、通信は、ハードウェア部設計を独立して行うため、現在はファイルを介して行っている。実際のシステムはソフトと実行するプロセッサと実用ハードウェアとの間でデータ通信を実現する。本システムの処理は以下のとおりである。

————ソフトウェア部・通信部の処理————

① ソフト（KC 法）の起動
② 初期化
③ 粒子情報の入力
④ 粒子情報をビットシフトしてファイル(metohard.dat)へ出力

————ハードウェア部・通信部の処理————

⑤ ビットシフトされた粒子情報をファイル(metohard.dat)から読み込み、外部メモリ(RAM)に格納
⑥ カルマンフィルタ
⑦ $\chi^2$ 検定
⑧ 同一粒子を 2 つの粒子に対応付けていないかをチェック
⑨ 粒子情報（流速情報）の内挿のため、対応付けができなかった粒子へ対応付いた粒子を使った空間平均を代入
⑩ 計算結果をファイルへ出力

————ソフトウェア部・通信部の処理————

⑪ ファイルからビットシフトされた計算結果を入力
⑫ ビットシフトされた値を元に戻してファイルに出力

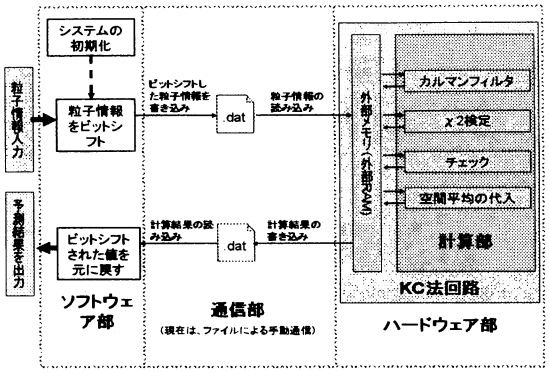



図 3.1　システムの概要

Fig.3.1 Block diagram of target system

### 3.3 ハードウェアとソフトウェアとのデータ通信方法

KC 法の処理内部では配列を使いデータを共有していたが、ソフトウェアと回路を一つのシステムとして実現するには、メモリ (RAM) を用意し、回路とソフトウェアの通信の同期を取ってメモリへアクセスしつつ並列処理する方法が高速化に有効である。

ここでは、まずハードウェアを独立して設計するため、データのファイル化を行う。初期設定情報、粒子情報をそれぞれ初期設定情報ファイル(kctohard.dat)、粒子情報ファイル(metohard.dat)としてファイルに書き出し、予測した粒子の結果を結果ファイル(yy.dat)として出力する。これにより、KC 法の計算はソフトウェア部から完全に独立して計算を行う。

### 3.4 ハードウェア設計

KC 法は ANSI-C を用いて記述されている。本研究で用いる Bach-C はハードウェア設計を行うための C 言語の一種であるが、その用途の違いから ANSI-C とは幾つかの点で文法が異なる。ハードウェアでは、一般に並列動作が多く用いられ、Bach-C では並列動作と、並列動作する回路の同期を取る通信の記述が用意されている。

ハードウェア化を行う KC 法について、以下の手順でハードウェア化を行う。

- 独立に動く回路を制御し、通信されるデータの正しさを保証するための同期通信の実現
- 小規模で効率の良い回路実現を目指した最低限必

要な固定小数点の決定と検証

・ 回路をテストするためのインターフェースの作成

本節では、上記の回路実現手法について述べる。

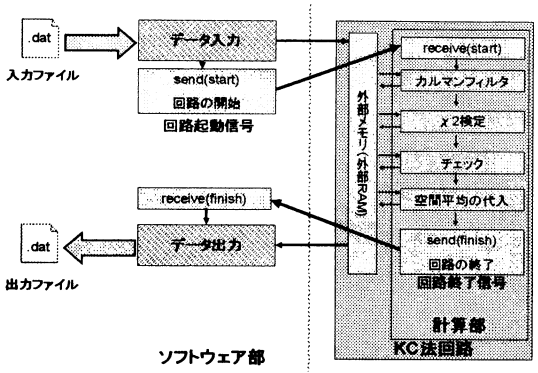



図 3.2 ソフト・ハード構成、通信方法

Fig.3.2 Communication of software and hardware

(a) 各計算回路の同期 ハードウェアは演算回路を並列動作させることにより高速化できる。並列動作では、回路内部で同期をとる通信が必要であるが、回路を逐次的に動かす場合も外部との通信により同期を取り、値の受け渡しを保障する必要がある。Bach-Cでは、通信の同期を行うために chan宣言子によりチャネル通信用の変数を宣言し、send、receive関数で 1 対 1 の単一方向の通信を行う。これらの関数を用いた通信は、送り側と受け取り側が互いに準備出来るのを待ち、通信が完了するまで次の処理に移らない。

先に用意した入力ファイルのデータをメモリに読み込み、KC 法の計算を行った後、結果がファイルに出力される。図 3.2 に考案したソフト・ハード構成、通信方法を示す。

ソフト部と回路の同期信号は、計算開始のための回路起動信号(start)と計算終了の回路終了信号(finish)で行っている。ソフト部から入力データをメモリに格納が終わると信号“start”を送信(send)してKC 法回路を起動する。回路のすべての計算がおわると信号“finish”を回路側から送信(send)し、ソフト部が受信(receive)する。この“finish”の通信が終了するとソフト部がメモリから計算結果を出力して全状態が終了する。

(b) 浮動小数点の固定小数点化 浮動小数演算をハードウェアに実装するためには、大きな演算回路とデータパス部が必要となる。必要とする精度の範囲で計算を行う小規模な回路とするため固定小数点を用いる。KC 法の計算部の出力値は画像の座標であり、画像サイズが縦 2036、横 2016 であることから最大値は 2036 をとる。これから整数値は 11bit で表すことができる。しかし、内部で乗算除算が行われているので、桁あふれを考慮する必要がある。精度を高めるために必要なビット数について実験的に確認し、整数・小数ビットを変更して KC 法の計算を行いソフトの結果を比較し精度を調べた。

(c) テストベンチ、入力ファイルの用意 設計したＫＣ法計算部は、外部から入力値を得て計算結果を出力するようにしている。これを単体でテストするには図 3.2 に示すソフトウェア部の処理を行う部分が必要となる。この回路との入出力を行う部分をテストベンチと言い、回路の検証と性能評価に使われる。

我々が用意したテストベンチは、初期設定データ(kctohard.dat)、粒子情報データ(metohard.dat)をファイルから入力し、KC 法の処理を行い、回路から出てきた出力データを(yy.dat)というファイルに書き込みを行う。

## 4. ハードウェアの最適化

ハードウェアの最適化は、処理が独立し、全体の 50%の実行時間を占めいていることから前半の Kalman,Chai 関数について行った。

高速化を目指すにあたって、次の３つの方法により高速化を検討する。

方法 1 $\chi^2$検定の処理範囲の限定

方法 2 パイプライン処理

方法 3 並列処理

### 4.1 $\chi^2$検定の処理範囲の限定

$\chi^2$検定では予測された位置から、次画面上でその粒子と対応する可能性のある粒子を選択するため、次画面上のすべての粒子に対して、状態変数値と、実測地の差をとり$\chi^2$検定の検定範囲内ならば状態変数値と、実測値の差の二乗和を計算する。粒子の実測値の情報は外部メモリに格納しているため粒子数が増えるとメモリアクセスに膨大な時間が必要になってくる。そのため予測粒子座標との距離計算と$\chi^2$検定処理をカルマンフィルタでの粒子予測座標の周りの粒子のみに行うことにした。

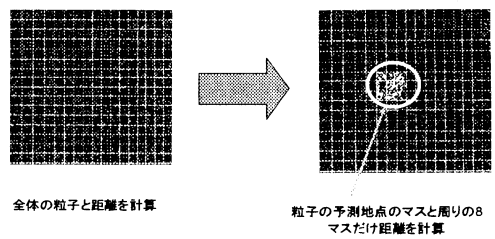



図 4.1 マス分けした粒子画像

Fig.4.1 Division of particle image

画像を縦 20pixel、横 20pixel の正方形で区切り各マスに番号付けをする。画像サイズは横が 2016pixel、縦が 2036pixel なので、マスの個数は横に 101 個、縦に 102 個の全部で 10,302 個になる。

まず各マスに番号づけを行い、粒子マスク相関法で画像から抽出した粒子がどのマスに入っているかを判定する。次にカルマンフィルタで予測した粒子位置がどのマスに入っているのかを検索し、そのマスの周りの 8 個のマスを選択する。オリジナルの $\chi^2$ 検定における検定範囲は 15pixel なので、$\chi^2$ 検定は粒子予測地点と周りの 8 個との計 9 マスだけを検索すれば十分である。

### 4.2 パイプライン処理

カルマンフィルタと $\chi^2$ 検定は処理が独立しているためのパイプライン処理を行う。

パイプライン処理は以下の 3 つの関数で構成される。

(a) パイプライン処理関数
(b) カルマンフィルタ
(c) $\chi^2$ 検定

パイプライン処理を行うためにカルマンフィルタ関数と $\chi^2$ 検定関数を並列に実行する必要がある。またカルマンフィルタ関数と $\chi^2$ 検定関数の同期をとる必要がある。

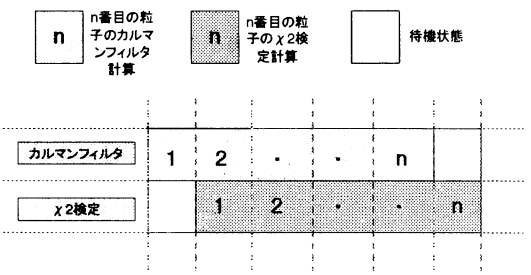



図 4.2 パイプライン処理
Fig.4.2 Pipeline processing

カルマンフィルタと $\chi^2$ 検定のパイプライン処理を図 4.2 に示す。カルマンフィルタと $\chi^2$ 検定はそれぞれ 1 個の粒子ずつ処理を行う。1 個の粒子ごとにカルマンフィルタ計算部が終わると $\chi^2$ 検定計算部に $\chi^2$ 検定起動信号を送信し、計算を開始する。

### 4.3 並列処理

カルマンフィルタ計算では 1 個の粒子に x 座標、y 座標、粒子の大きさ、加速度の 4 つの情報がある。この 4 つの粒子情報は独立して計算されるため、並列に動作させることができる。4 つの粒子情報を並列に処理した場合について図 4.3 に示す。

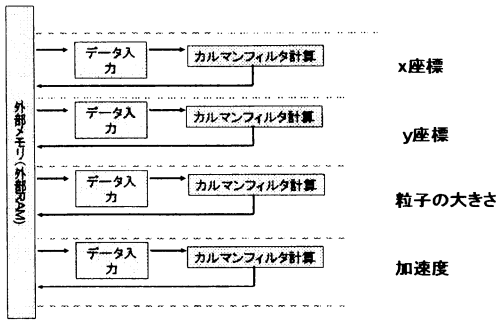



図 4.3 並列処理の回路
Fig.4.3 parallel processing states

## 5. 実験結果と考察

実験を行う上で、用いる粒子情報は粒子数 5906 個と 5896 個のデータを用いる。KC 法計算部の最適化で設計した回路は、

(a) 逐次処理
(b) カルマンフィルタと $\chi^2$ 検定のパイプライン処理
(c) カルマンフィルタ部の並列処理
(d) パイプライン処理と並列処理を組み合わせたもの
(e) $\chi^2$ 検定の処理範囲を限定していない逐次処理

の 5 種類である。(e)の $\chi^2$ 検定の処理範囲を限定していない逐次処理回路以外は $\chi^2$ 検定処理範囲を限定する記述を組み込んだ。以下の実験ではソフトウェアと回路実行の出力結果との一致を確認し、bachc での回路面積の見積もり、RTL でのシミュレーションでカルマンフィルタ、$\chi^2$ 検定にかかる処理時間を測定する。

### 5.1 $\chi^2$ 検定の処理範囲の限定

$\chi^2$ 検定の処理範囲を限定した回路と $\chi^2$ 検定の処理範囲を限定していない逐次処理回路について、回路面積と処理時間を測定した。その結果を表 5.1 に示す。

表 5.1 エリア限定の実行時間と回路面積
Table 5.1 Experimental results (speed and area) of area limitation

| | $\chi$2 検定実行時間 ($\mu$s) | 回路面積（ゲート） |
|---|---|---|
| 処理範囲の限定なし | 11,146,523 | 652,030 |
| $\chi$2 検定の範囲限定 | 24,505 | 682,705 |

表 5.1 から $\chi^2$ 検定の処理範囲を限定した回路は元の回路より回路面積が 4%増加しただけだが $\chi^2$ 検定の実行時間は約 455 倍の高速化ができた。

### 5.2 カルマンフィルタ部の並列処理

カルマンフィルタ部の並列処理を行ったときの回路と逐次処理の回路について回路面積と処理時間を測定した。その結果を表 5.2 に示す。

表 5.2　並列化に対する実行時間と回路面積
Table 5.2 Experimental results (speed and area) of parallel

| | カルマンフィルタ実行時間($\mu$s) | 回路面積（ゲート） |
|---|---|---|
| 逐次処理の回路 | 79,728 | 682,705.00 |
| カルマンフィルタ部の並列処理 | 50,788 | 1,847,414.00 |

カルマンフィルタ処理回路を 4 並列にしているので処理時間も 1/4 にならなければならないが実際の処理時間は 1/4 になっていない。その理由として、同一メモリには同時にアクセスができないということがある。回路は x 座標、y 座標、粒子の大きさ、加速度の処理を同時に実行しようとするため同時に同一メモリにアクセスしようとする。しかし、そこで仲裁回路が働き 1 つのアクセス以外を現在のアクセスが終わるまで待機状態にしてしまう。そのため処理は図 5.1 のように処理にずれができ、すべてを同時に実行することはできなくなっている。

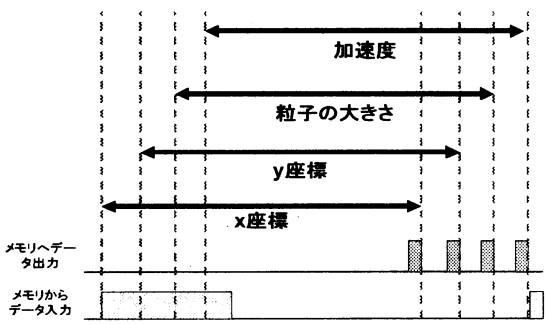



図 5.1　粒子 1 個の並列処理回路の処理動作
Fig.5.1 Simulation result of parallel processi ng of kalman filter for one particle

### 5.3　ＫＣ法回路の最適化

設計した 4 種類の回路についての回路面積と処理時間を測定した。その結果を表 5.3 に示す。これらの高速化処理による計算結果への影響はなく、ソフトウェアでの結果と比較して小数 5 桁まで同じ値を出力している。

## 6. 結論

本研究では、ソフトウェアのみで構成されたカルマンフィルタ(Kalman-filter)・$\chi^2$ 検定(Chi-square Test)法、ハードウェアとソフトウェアを組み合わせたシステムを設計することによりソフトウェアのみでの実行時間より大幅に高速化できた。回路の高速化方法としては、$\chi^2$ 検定の処理範囲の限定のようにハードウェアに適するアルゴリズムに変更することにより大幅な高速化が可能となる。

今後の課題としてはメモリを分割してメモリに同時アクセスができるようにしなければならない。メモリを分割することによってカルマンフィルタ処理回路を 4 並列にしたときの実行時間が逐次処理の回路より 1/4 になると考えられる。また、キャッシュメモリの利用、必要ビット精度の理論的推定、ハードウェアに適するようにアルゴリズムの見直しが挙げられる。その他に、今回行わなかった後半の Check, Spav 関数の高速化も検討する予定である。

表 5.3　並列化とパイプライン化に対する実行時間と回路面積
Table 5.3 Experimental results (speed and area) of parallel and pipeline processing

| | 実行時間($\mu$s) | 回路面積(ゲート数) |
|---|---|---|
| 逐次処理 | 104,232 | 682,705 |
| パイプライン処理 | 80,795 | 799,821 |
| 並列処理 | 75,412 | 1,802,201 |
| パイプライン＋並列 | 51,620 | 1,775,380 |
| ソフト(2.6GHz) | 5,000,000 | |


### 謝辞

粒子追跡技術を使用した研究を行なうにあたり、粒子追跡法のソースとして粒子マスク相関法を提供し、ソフトウェアの解析にあたって直接ご指導いただいた近畿大学理工学部社会環境工学科江藤剛治教授、竹原幸生助教授に深くお礼申し上げます。

Bach を用いたハードウェア設計を実現するに当たり、多大なるご指導を頂いたシャープ株式会社山田晃久様をはじめ、BACH 開発グループの皆様に厚く御礼申し上げます。


## 文　献


[1] 可視化情報学会編：“PIV ハンドブック”、森北出版株式会社、東京、1～4 (2002)

[2] 江藤剛治、竹原幸生、道奥康治、久野悟志：“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、**40**、1051～1057 (1996)

[3] 江藤剛治、竹原幸生、岡本孝司：“標準画像を用いた粒子マスク相関法と KC 法の性能評価”、日本機械学会論文集、**65**、184～191 (1999)

[4] 吉田たけお、尾知博：“ＶＨＤＬで学ぶディジタル回路設計”、ＣＱ出版、東京、(2002)

[5] “Bach システムマニュアル”、シャープ株式会社提供(2003).