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

2005－ＳＬＤＭ－119（15）
2005／3／17

粒子追跡技術のハードウェア化の一手法
—相関値計算部のハードウェア化—

大口　貴裕†　大倉　崇宜‡　小西　徹也‡　上甲　憲市‡　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　31 号館 5F

E-mail: †04870418d@msa.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、小規模化の実現について述べる。

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

Hardware Acceleration for Particle Mask Correlation method

Takahiro OHGUCHI† Takanobu OHKURA‡ and Tetsuya KONISHI‡

and Kenichi JYOUKOU‡ and Takashi KAMBE‡

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

E-mail: †04870418d@msa.kindai.ac.jp, ‡tkambe@ele.kindai.ac.jp

Abstract　Various techniques are proposed so that the particle pursuit technology may measur e the flowfield at continuously . In this paper, a software and hardware system is designed based on the KC method that combines the particle mask correlation meth od with the kalman filter and Chai-square Test, and the processing speed and the area of the system are evaluated.

Keyword　Bach system，Particle Mask Correlation method，Kalman-filter，Chai-square Test

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相関値計算回路の高速化・回路規模の削減を検討する。

マスク相関値計算回路は、4 種類の並列処理により高速化を図った。入力データ(全画面 1008pixel×1018pixel)に対し、水平分割、十字分割、水平パイプライン、さらに1 pixel ずつ処理することで外部メモリやメモリアクセス数を最小限にした1pixel キャッシュ処理などである。また、各々の処理時間で大きな割合を占める平方根計算を VHDL で記述したものを組み込むことにより全体の処理時間の高速化を図った。

本システムは、マスク相関値計算回路が整数24bit 小数20bit の固定小数で実現され、ソフトウェアでの処理時間220ms(Pentium4 2.4GHz)に対し、119ms(90MHz)となり、約 50 倍の高速化を実現した。

2. 粒子追跡技術とハードウェア化

2.1 粒子マスク相関法

粒子マスク相関法(以下 PMC 法)とは、円形に近い粒子画像の抽出を行う手法である。粒子画像は対称な二次元正規分布で近似するとし、これをマスクと呼ばれるテンプレートとして対象画像上に重ね相関係数を計算する。マスクと粒子画像の形状が等しければ、輝度は異なっていても相関

係数はほぼ1になる。その値があるしきい値以上になる部分に粒子が存在するとみなし、粒子の位置を推定する。

PMC 法は"マッチドフィルタ定理"を利用している。"マッチドフィルタ定理"と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とき、これをテンプレートとして画像と相互相関係数をとると、画像情報の中身である信号パワーとノイズパワーの期待値の比を最大にするというものである。通常は、テンプレートのサイズや回転方向等を変えないと画像中の対象物の形状に合わせることができないが、粒子マスクの場合、回転によって形状は変わらない。また、サイズの変化に対して相関係数値の変化はきわめて鈍感である。また、$r \leq 1$に規準化された相関係数値によるので、中心輝度によらず、粒子像中心付近の相関数値は約 1 となる。従ってPMC 法では、テンプレートの回転、サイズ、輝度等の3パラメータを無視し、位置パラメータだけを変数とすることができるので、効果的に粒子像を抽出することができる。

次に、PMC 法のアルゴリズムについて説明する。概要は以下のとおりである。

① マスク相関値の計算:マスク画像を 1 枚の静止画像に置き、輝度分布との相関係数を求める。マスク画像の中心座標をピクセル単位で動かし、全画面上を走査しこの計算を行う。マスク画像と撮影画像との相関係数を求める。式は(2.1)式に示す。

② 2値化処理:相関係数があるしきい値以上の領域とそれ以下の領域に2値化する。二値化は粒子像を抽出するための処理であり、背景の輝度値が0の場合、一つの粒子像は輝度値1をもつ画素の一つの集まりは基本的に 1 個の粒子像に対応する。

③ 連結領域の抽出(ラベリング):粒子画像中には粒子が多数存在する。個々の粒子を区別するためそれぞれに番号付けを行う。

④ 粒子像中心計算:番号ごとに粒子の中心を求め、その座標をその粒子位置とする。

式(2.1)に相関値を求める式を示す。

$$r = \frac{\sum_{i,j=1,1}^{n\times m}\left(I_{i,j} - \bar{I}\right)\left(M_{i,j} - \bar{M}\right)}{\sqrt{\left[\sum_{i,j=1,1}^{n\times m}\left(I_{i,j} - \bar{I}\right)^2\right]\left[\sum_{i,j=1,1}^{n\times m}\left(M_{i,j} - \bar{M}\right)^2\right]}} \quad (2.1)$$

n,m:水平,鉛直相関領域サイズ　i,j:水平,鉛直位置

I :元画像の輝度値　$\bar{I}$:元画像の輝度値の平均値

M :元画像の輝度値の平均 $\bar{M}$:マスク画像の平均値

2.2 ハードウェア技術

図 2.1 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だけでなく、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。

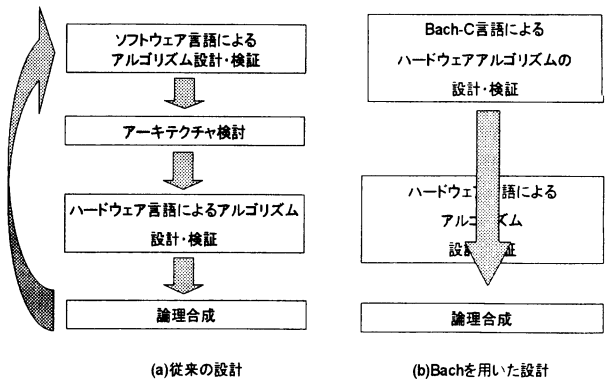

図 2.1　Bach の 設計工程

Fig.2.1 Design process using Bach system

3. PMC 法のハードウェア化

3.1 PMC 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。

本研究では、PMC 法に対し、

- 計算量が多い
- メモリアクセスや処理を分割にすることで効率が上がる

部分をハードウェア化し、ソフトウェアとの通信を最小限とし、全体の効率向上を図る。

我々は、PMC 法のプログラム構造の解析、処理時間の分析により相関値計算部に着目し、その中で相関値計算を行っている部分を PMC から独立させ、ハードウェア設計を行った。

このシステムは、主にソフトウェア部分、通信部、ハードウェア部からなり、通信手段は、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。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（PMC）の起動

② 初期化（配列の初期化）

③ 画像入力（画像数値化）の後、画像4倍

④ 数値をファイル（datain.dat）へ出力

————ハードウェア部・通信部の処理————

⑤ 入力データファイル（datain.dat）から読み込み、外部メモリ（RAM）に格納

⑥ マスク相関値計算

⑦ 計算結果をファイル（dataout.dat）へ出力

————ソフトウェア部・通信部の処理————

⑧ dataout.dat から計算結果を読み込み、計算結果格納用配列に入力

⑨ 二値化、ラベリング（粒子の並び替え）

⑩ 粒子数、位置情報を出力

3.1.1 ハードウェア化箇所の決定

我々は、以下の手順で PMC 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と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、その役割とつながりを調べる。

(b) clock 関数を用いた実処理時間の測定　処理時間を gprof で測定すると標準関数の実行時間は含まれない。clock 関数を使用することで実際の処理時間を測定することができる。5種類の画像データを実行し平均処理時間の結果を表 3.1 に示す。

表 3.1clock 関数での分析結果（単位：%）

Table 3.1Execution time of some functions in PMC

関数名＼画像データ	平均
Avesd	52.22
Imask	39.59
Crosscoll	5.62
Expand	0.35

プログラム構造の解析と処理時間の分析により、マスク相関値計算部が、全体の処理時間に対する割合が97%と高い。マスク相関値計算部は、粒子マスクの正規分布（Maskdata 関数）を計算し、対応する画面上の領域の平均値（Avesd 関数）を求め、粒子マスクと対応する範囲を比較（Imask 関数）計算し、相関値（Crosscoll 関数）を求めている。ソースファイルの解析の結果、各関数で処理を全画面分計算しており、処理全体に占める割合が多いことから、PMC 法のマスク相関値計算部に注目し、ハードウェア化を検討する。図 3.1 に設計するシステムを示す。

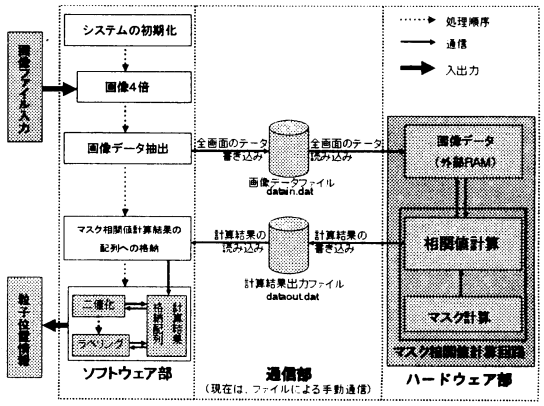

図 3.1　システムの概要

Fig.3.1 Block diagram of target system

4. ハードウェアの最適化

高速化を目指すにあたって、処理する画像を分割し、並列化処理することにより方法(1)～(6)により高速化を検討する。

方法(1)　水平分割処理

方法(2)　十字分割処理

方法(3)　水平メモリパイプライン

方法(4)　VHDL 記述のコール

方法(5)　1pixel 処理

方法(6)　1pixel キャッシュ処理

4.1.1 水平分割並列処理

全画像データを水平分割にし、分割の数だけ回路を用意（本研究では4分割）し、それぞれを並列に処理を行う。

並列処理では以下の3つの関数で構成される。

(a) 水平4分割並列処理関数(水平4段に並列計算を行う)

(b) マスク相関値計算を行う

(c) 関数のオーバーラップ処理

画像を分割処理する場合、境界上の粒子に対して精度低下が懸念される。ここでオーバーラップを持たせ精度を確保した。図 4.1 に水平分割処理でのオーバーラップ処理について示す。

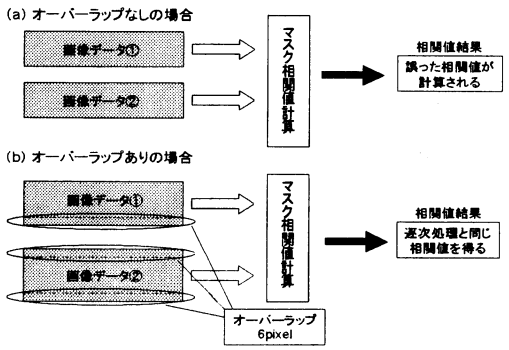

図 4.1　水平 4 分割並列でのオーバーラップ処理

Fig.4.1 Overlap processing by block d iagram of horizontal level 4 parallel

4.1.2 十字分割並列処理

全画面を十字分割にし、分割の数だけ回路を用意（本研究では 4 分割）し、それぞれを並列に処理を行う。

並列処理では以下の 3 つの関数で構成される。

(a) 十字 4 分割並列処理関数(十字に分け回路を並列にマスク相関値計算を行う)

(b) 相関値計算を行う

(c) 各関数のオーバーラップ処理

4.1.3 水平パイプライン処理

逐次処理や水平分割並列処理は、外部メモリを 1 画面分ごとに用意しているため、メモリ消費量が多くなる。水平パイプライン処理では、画像を分割し、ある関数まで処理がくると次の段の処理に必要なメモリを読み出すという流れをとっている。

4.1.4 VHDL 記述のコール

Bach-C 記述では、平方根計算をニュートン法で行っていたが、このニュートン法での処理時間が 1 つの入力に対して 3μs かかっているため、VHDL 記述による平方根回路を用いることによって高速化を行う。PMC では、固定少数点として、整数部を 24 ビット、小数部を 20 ビットの全体で 44 ビットの 1 入力、1 出力の平方根計算関数を必要とする。

使用する VHDL 記述の平方根計算のアルゴリズム[6]は次のようになる。まず、平方根の値は加減算シフト及び、定数読み出しの繰り返しにより近似することができる。$\sqrt{r}$ を求めるための漸化式を(4.3)式に示す。

$$\left.\begin{aligned} X_{n+1} &= X_n + d_n \cdot Y_n \cdot 2^{-\sigma(n)} \\ Y_{n+1} &= Y_n + d_n \cdot X_n \cdot 2^{-\sigma(n)} \\ &\begin{cases} Y_n < 0 \rightarrow d_n = 1 \\ Y_n \geq 0 \rightarrow d_n = -1 \end{cases} \end{aligned}\right\} \quad (4.3)$$

初期値 $X_1 = r + 0.3645129$ ，$Y_1 = r - 0.3645129$ とすると、$X_n \rightarrow \sqrt{r}$ となる。$\sigma(n) = n - k$ であり、k は $3^{k+1} + 2k - 1 \leq 2n$ を満たす最大の整数である。このアルゴリズムで近似可能な範囲は $0.04 < r < 3.4$ である。このアルゴリズムで範囲を拡張した平方根計算を使用している。

4.1.5 1pixel 処理

逐次・水平分割並列・十字分割並列・水平パイプライン処理は、処理の際平均値やマスク計算、相関値計算は、全画面分について行い、結果を配列に記憶するという方法をとっている。1pixel 処理は、1pixel ごとに計算を行い、結果を記憶する。これにより外部メモリの容量やメモリアクセスが削減でき、処理を高速化することができる。図 4.2 に 1pixel 処理と逐次処理を示す。

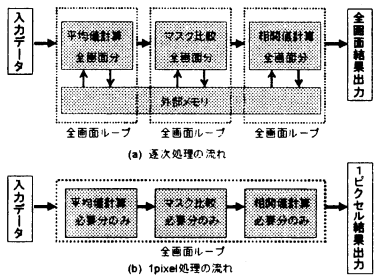

図 4.2　1pixel 処理と逐次処理の比較

Fig.4.2 Comparison of 1pixel processing and serial

4.1.6 1pixel キャッシュ処理

1pixel 処理をさらに高速化するためにキャッシュメモリを用いる。この方法は、平均やマスクとの比較計算の際、マスクの範囲(11×11pixel)で計算を行うのだが、ループで一つ一つ値を読み出しているため、メモリアクセス時間がボトルネックとなる。マスクのデータをキャッシュメモリに入れることでメモリアクセスの時間を短縮させることができ、さらにこの部分を 11 段並列処理にすることにより高速化を目指す。図 4.3 に 1pixel キャッシュ処理を示す。

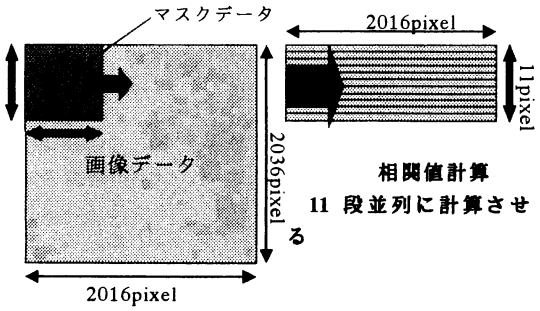

図 4.3 1pixel キャッシュ処理
Fig.4.3 1pixel cash processing

4.2 実験結果と考察

実験は、画素数 40,000 個のデータを用いる。マスク相関値計算部の最適化で設計した回路は、

(a) 逐次処理
(b) 水平パイプライン処理
(c) 水平分割並列処理
(d) 十字分割並列処理
(e) 1pixel 逐次処理
(f) 1pixel キャッシュ処理

の 6 種類である。回路のクロック周波数は 90MHz, メモリアクセス時間は 5ns とする。表 4.1 に(a)-(f)の回路規模と処理時間について、表 4.2 に VHDL 記述による平方根回路を組み込んだ場合の回路規模と処理時間を示す。

表 4.1 処理時間と回路規模の関係
(ニュートン法平方根計算)
Table 4.1 Experimental results for square root function of Newton Approximation

	処理時間(ms)	回路規模
逐次処理	1,660	481,344
水平パイプライン処理	1,803	1,167,216
水平分割並列処理	475	1,911,732
十字分割並列処理	449	1,928,496
1pixel逐次処理	630	461,775
1pixelキャッシュ処理	164	722,616

表 4.2 処理時間と回路規模の関係
(VHDL 記述平方根計算)
Table 4.2Experimental results for square root function of VHDL description

	処理時間(ms)	回路規模(ゲート数)
逐次処理	1,548	516,048
水平パイプライン処理	1,690	1,248,924
水平分割並列処理	449	2,050,788
十字分割並列処理	416	2,066,448
1pixel逐次処理	605	510,775
1pixelキャッシュ処理	86	740,448
ソフト(2.4GHz)	220	

表 4.2 と表 4.3 より、VHDL 記述平方根計算を用いた回路はニュートン法平方根計算を用いた回路に比べて回路規模は 2.46%から最大 7.21%増加する。しかし、処理時間は 6.45%から最大 47.56%削減した。

逐次処理・水平 4 分割処理の処理に時間がかかる理由は、各関数での計算後、値をメモリへ代入しているためと考えられる。パイプライン処理が、逐次処理よりも遅くなった原因は、メモリを読み込みおよび書き込みの際、通信の同期をとっているため、逐次に比べ処理時間が遅くなったと考えられる。メモリへのアクセスをなくし、最適化を図った 1pixel 処理・1pixel キャッシュ処理については、逐次に比べて処理時間が半分以下となった。

最も高速な 1pixel キャッシュ処理回路ではソフト(Pentium4 2.4GHz)よりも高速である。ソフトウェアでは OS、CPU、メモリ、ハードディスクなので構成されるのに対し、本システムはシステム LSI として構成される。

4.3 各周波数での処理時間

逐次処理、水平 4 分割並列処理、水平 4 分割パイプライン処理、1pixel 処理、1pixel キャッシュ処理について、各々の周波数での処理時間の測定結果を図 4.4、に示す。各周波数でのメモリアクセス時間は周波数の 1/2 としている。

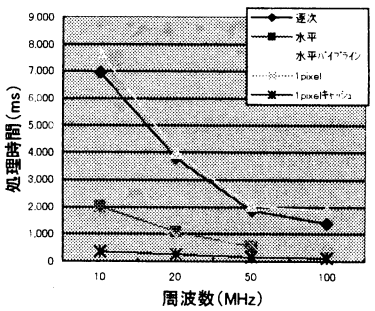

図 4.4 各周波数での処理時間 (ms)
Fig.4.4 Processing time and speed

図 4.4 より処理時間と周波数は反比例することが分かる。1pixel キャッシュ処理では 20MHz でソフトとほぼ同等の処理速度となり、100MHz では約 50 倍程度の高速化を実現した。

4.4 関数化による回路規模削減

記述で多く使われる計算式を関数化することにより、回路規模を削減、処理時間を短縮する。

#pragma preserve 関数名;

という記述を追加することにより、関数全体を1つの演算器のように扱うことができる。関数化の例を図4.5に示す。

```
void  func(){
int;
float;
                :
  sum=a+b+c+d+e;
  dsum=f+g+h+i+j;
  num=k+l+m+n+o;
                :
}
```

```
void  func(){
int;
float;
                :

sum=adder(a,b,c,d,e);
    dsum=adder(f,g,h,i,j);
    num=adder(k,l,m,n,o)
;
                :
}
float  adder(t,w,x,y,z){
float  q;
  q=t+w+x+y+z;
return q;
}
#pragma preserve adder
```

図 4.5　関数化の例
Fig.4.5 Example of　function

関数化することで、回路規模の削減が確認でき有効な手段であるといえる。

5. まとめ

本研究では PMC 法の計算時間を多く要している部分をハードウェア化することにより、システム全体を効率化する一手法を提案した。

ハードウェア化は、画像データからマスク計算、相関値計算を行う部分に対して行い、内部の演算を固定小数演算で実現した。回路による相関値計算を使った PMC はソフトウェアのみで構成された時とほぼ同等の粒子抽出精度を確保している。

また、処理速度については、1pixel キャッシュ処理回路ではソフトウェア(Pentium4 2.4GHz)よりも高速である。動作周波数に関しては、ソフトウェアが 2.4GHz であるのに対して、90MHz のハードウェアでソフトウェア以上の高速化を実現した。動作周波数を下げることにより、消費電力と回路規模の両方を削減できるので小規模化、低消費電力化により多種電子機器への応用が可能となる。

今後はさらに設計の最適化を進め、近畿大学開発の高速カメラと統合し、リアルタイム粒子追跡システムの開発を進める。

6. 課題

粒子マスク相関法をリアルタイムで行うには、さらに高速化する必要がある。ここでは、その検討の1つとして、マスク相関値計算部の性能向上について、実験、考察により考えられる内容を列挙する。

(a)　指数計算回路の VHDL 記述を適用する。

(b)　除算回路は、回路規模、処理時間ともに大きい。本研究で作成した回路は、多数の除算を使用しているため、回路全体での除算の削減の工夫が必要である。

(c)　ソフト・ハード協調検証を行い、システム全体の評価を行う。

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)

[6] 大窪啓太：“開平・除算回路の設計”、近畿大学理工学部電気工学科、東大阪（2005）