特定用途向け低ビット複合演算回路の一設計法

大窪啓太[†] 神戸尚志[††]

組み込みシステムに用いる演算回路は，回路規模，処理時間，計算精度の間でトレードオフがあり，どの項目を優先すべきかは対象となるシステムに依存する．本文では，画像処理でよく用いられるテンプレートマッチング法における相関値計算に特化した演算回路について考察する．相関値計算は分母に平方根を持つ除算処理があり，減算シフト型アルゴリズムに「補間テーブル」を加えることによる演算の高速化，ならびに閾値判別のための演算打ち切り法を提案し，処理時間短縮と回路規模削減の効果を示す．

An Application Specific Arithmetic Circuit Design

Keita Ohkubo[†] and Takashi Kambe[††]

In the design of the arithmetic circuits for the embedded systems, there are trade-offs among circuit scale, processing time, and calculation accuracy and the circuits have to be optimized based on different criteria for each embedded system. In this paper, we examine the correlation coefficient calculation circuit which is a part of template matching method frequently used in image processing systems. It is designed based on Digit-Recurrence division algorithm with the square root in the denominator. Accelerating the subtract-shift operation with a table lookup interpolation and finishing the calculation by the threshold value are proposed and their performances are evaluated.

1. はじめに

組み込みシステムで用いられる演算回路の設計目標は，対象となる組み込みシステムによって異なり，回路規模，処理時間，計算精度などの間でトレードオフの関係がある．対象となるシステムごとに用途に特化した演算回路を設計することで，計算精度を確保しつつ，高速化，回路規模の削減が可能となる[2)]．

車や人物といった物体の検出や追跡は，セキュリティ分野や画像解析で多く用いられる画像処理の 1 つである．画像の中から，特定のパターンを探し出すには，「テンプレートマッチング法」が多く用いられている[5),6)]．テンプレートマッチング法では事前に取得したテンプレート画像と，入力画像の一部の領域との相関値を求め，テンプレートと最も類似した画像領域を検出する．これを毎フレーム繰り返すことで，同一物体を追跡できる．テンプレートマッチング法で用いる相関値計算は，分母に平方根を持つ除算処理を含み，その出力が閾値より大きいか小さいかを判別することで，対象物体を識別する．相関値計算の式を式 (1) に示す．

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

式 (1) 中のパラメータは以下の意味を持つ．

n，m：水平，垂直相関領域サイズ

i，j：水平，垂直位置

$I_{i,j}$：元画像の輝度値　$\overline{I}$：元画像の輝度値の平均

$M_{i,j}$：テンプレート画像の輝度値

$\overline{M}$：テンプレート画像の平均値

なお，式 (1) において，

$$\sqrt{\left[\sum_{i|1\le i\le n,\, j|1\le j\le m} (M_{i,j}-\overline{M})^2\right]} \quad (2)$$

はテンプレート画像の輝度の標準偏差であり，一定なので定数とし，式 (1) の演算から除外することができる．相関値計算は複雑な計算であり，たとえば，入力

† NEC エレクトロニクス株式会社
NEC Electronics Corporation
†† 近畿大学理工学部電気電子工学科
Department of Electric and Electronic Engineering, School of Science and Technology, Kinki University

画像が 2,016 ピクセル × 2,036 ピクセルのとき，相関値計算は約 400 万回必要となり，テンプレートマッチング法を用いたシステム全体の処理時間の大半を占め，高速化のボトルネックとなる．相関値計算の処理サイクル数を削減することができれば，システム性能を大きく向上させることができる．

このことから，本文ではテンプレートマッチング法における相関値計算のための演算回路について考察する．まず，必要精度を保ちつつ，処理サイクル数を削減する手法として，漸化式計算と補間テーブルを併用した手法を提案する．

次に，相関値計算において閾値との大小を判別するため，判別がついた時点で計算を終了する手法を提案する．

以下では，これらの手法により，計算の高速化と回路規模の削減効果について比較評価を行う．

2. 演算アルゴリズム

本回路の設計には，複雑で回路が大規模化する乗除算を必要としない漸化式計算により，商の小数上位から 1 ビットずつ求めていく減算シフト型アルゴリズム[1] を用いる．

2.1 漸 化 式

分母 D，分子 N，商 S としたとき，分母に平方根を持つ除算は以下の式で表される．

$$S = \frac{N}{\sqrt{D}} \tag{3}$$

式 (3) に対し，文献 1) より，以下の漸化式が得られる．

$$\begin{aligned} X_{n+1} &= 2X_n - 2 \cdot Y_n \cdot s_{n+1} - D \cdot 2^{-n-1} \\ Y_{n+1} &= Y_n + D \cdot s_{n+1} \cdot 2^{-n-1} \end{aligned} \tag{4}$$

式 (4) 中の X_n は残余，Y_n は分母に中間結果を乗じた値，s_n は商の n 桁目の値を表す．初期値 $X_0 = N^2$，$Y_0 = 0$ としたとき，残余 X_n の正負を判別することで，小数点以下の商を上位から 1 ビットずつ求める．X_n の符号が正のとき，商の小数点以下 n 桁目のビットが 1 となり，負のときは 0 となる．X_n が負の場合，補数表現を用いて漸化式の計算を行う．

2.2 2 乗 計 算

式 (4) の漸化式では，初期値に，分子 N の 2 乗を求める必要があり，乗算回路が必要となる．2 乗計算の特性を生かした 2 乗計算専用回路を用いることで，処理の高速化および回路規模の削減を図った．

2 乗計算では，部分積に以下のような特徴がある．**図 1** で示すように，◯ で囲まれた部分積を線で結び，その線を境界とした上部と下部の部分積は等しくなる．

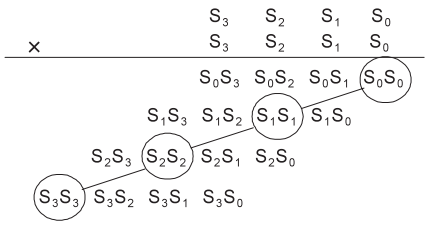

図 1 2 乗計算の部分積
Fig. 1 The partial product of the square calculation.

表 1 乗算回路の比較
Table 1 The comparison of the multiplication circuits.

	回路規模(gate)	処理時間(ns)
Booth 乗算回路	2,576	16.19
2 乗専用回路	1,024	11.67

上部もしくは下部のどちらか片方の部分積の和を 2 倍した値と ◯ で囲んだ部分の和が部分積の総和と等しく，加え合わせを必要とする部分積の個数を削減することができる．

Booth アルゴリズムを用いた乗算回路を例として，2 乗計算専用回路を用いた場合との比較を行った．16 ビットの乗算処理を，CSA の Wallace 木構造と桁上げ先見加算を用いて構成した．**表 1** に論理合成の結果を示す．Booth を用いた乗算回路に比べ，部分積生成に複雑な処理を必要としないことから，2 乗計算専用回路が規模および処理時間において良好な結果が得られた．

2.3 計 算 範 囲

今回採用したアルゴリズムは商の小数点以下のビットを上位から 1 ビットずつ求めていくものであり，$N < \sqrt{D}$ と考えることができる．この条件から入力値は以下の範囲に限定される．

$$\begin{aligned} &0.5 \le N < 1 \\ &1 \le D < 4 \end{aligned} \tag{5}$$

この範囲外の入力値はシフト操作により限定範囲内に収める．

限定範囲内に収めた値はアルゴリズムを用いた計算を行った後に，逆にシフト操作を行い，入力値に対する計算結果を求める（以下，これを復元と呼ぶ）．復元時に行うシフト数および右シフトであるか，左シフトであるかの判別は，範囲限定時の分子のシフト数 nS，分母のシフト数 dS とし，右シフトを $-$，左シフトを $+$ で表したとき，復元時のシフト数 rS は以下の式で求める．

$$rS = \frac{dS}{2} - nS \tag{6}$$

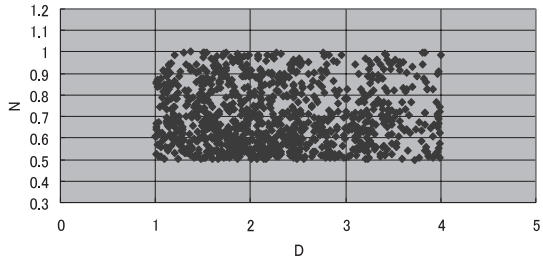

図 2 粒子追跡システムにおける N，D 値の分布
Fig. 2 The distribution of N and D in the particle tracking system.

以上に述べたように，シフト操作を多く使用することから，アンダーフローによる計算精度の低下が考えられるが，式 (4) は小数部の計算のみとなるので，処理時間を短縮できる．

本手法をテンプレートマッチング法による粒子追跡システム[8] に適用し，その性能を評価する．この粒子追跡システムの場合，式 (5) における値 N，D がとる値について 1,000 個の分布を**図 2** に示す．範囲に偏りなく広く分布しており，粒子追跡システムのデータを用いて本手法の評価を行うことについて問題ないことが確認できる．

3. 補間テーブル

式 (4) の漸化式計算は，商の上位ビットから順に 1 ビットずつ求めているため，計算回数を多く行えば，得られる計算精度が向上する．しかし，回数が多くなると，処理時間が増加してしまうという問題がある．

本文では漸化式の計算回数を削減し，なおかつ精度を保つ手法として，テーブルを用いた手法を提案する．以下，これを「補間テーブル」と呼ぶ．商の全ビットを補間テーブルに持つと，分母分子に変数を持つ除算では，演算に用いるビット数を n としたとき，2^{2n} 通りの値を記憶しなければならず，メモリ規模が膨大になる．そこで，補間テーブルを用いて求める部分を商の小数下位のビットとし，小数上位のビットは漸化式により求める手法を考える．

3.1 補間テーブルの作成方法

式 (4) の漸化式において，計算回数 $n \to \infty$ とすると，$D \cdot 2^{-n-1}$ は 0 に収束する．これを利用すると，以下のようになる．

$$X_{n+1} = 2X_n - 2 \cdot Y_n \cdot s_{n+1} \tag{7}$$

式 (4) の Y_n は，$D \cdot 2^{-n-1} = 0$ であると考えると，計算回数 $n \to \infty$ とすると値が変化しない．

式 (7) が与えられたとき，X_n および Y_n の値から，その後の漸化式計算をあらかじめ計算し，その値を格納した補間テーブルから下位の値を読み出す手法を考える．ただし，X_n，Y_n のすべてのビットをインデックス情報とした補間テーブルを用意するとテーブルサイズが大きくなるため，X_n，Y_n をいくつかの範囲に区切り，インデックス情報を少ないビット数で表すことで，補間テーブルのサイズを抑えることができる．

Y_n は，文献 1) より，分母 D と中間結果 Sn の乗算であることから，$D \times (N/\sqrt{D})$ に収束する．範囲限定を行っているため，Y_n のとりうる値の下限と上限は以下のようになる．

$$\begin{aligned} Y_{\min} &= 1 \times \frac{0.5}{\sqrt{1}} = 0.5 \\ Y_{\max} &= 4 \times \frac{1}{\sqrt{4}} = 2 \end{aligned} \tag{8}$$

$0.5 \le Y_n < 2$ の区間を分割し，Y_n の値が存在する区間のインデックス情報 Y_{index} を求める．

X_n の値が Y_n の 2 倍以上の場合，式 (7) の計算を行っても X_n の符号に変化はない．このことから，X_n の値は 0〜4 の範囲とし，$0 \le X_n < 4$ の区間を分割する．X_n の値が存在する区間のインデックス情報 X_{index} を求める．X_{index} と Y_{index} を連結し，補間テーブルのインデックス情報とする．

漸化式計算によって得られた Y_n および X_n からの区間選択および補間テーブル読み出しの手法を以下に示す．

1) Y_n のインデックス情報は，0.5〜2 を K_y 間隔で分割したとき，Y_n が以下の式を満たす値である場合，インデックス情報 Y_{index} を I_y とする．ただし I_y は，正の整数値である．

$$0.5 + K_y \times I_y \le Y_n < 0.5 + K_y \times (I_y + 1) \tag{9}$$

たとえば，Y_n が 0.8 で，0.5〜2 を 16 分割したとする．0.5〜2 を 16 分割すると，$K_y = 0.09375$ となる．この K_y および Y_n を式 (9) に代入すると，

$$\begin{aligned} &0.5 + 0.09375 \times I_y \\ &\quad \le 0.8 < 0.5 + 0.09375 \times (I_y + 1) \end{aligned} \tag{9.1}$$

となり，$I_y = 3$ のとき，式 (9.1) は以下のようになる．

$$0.78125 \le 0.8 < 0.875 \tag{9.2}$$

よって，Y_{index} を 3 とする．

2) X_n のインデックス情報は 0〜4 を K_x 間隔で分割したとき，X_n が以下の式を満たす値である場合，インデックス情報 X_{index} を I_x とする．

$$K_x \times I_x \le X_n < K_x \times (I_x + 1) \tag{10}$$

式 (9) および (10) に示す各区間の代表値を $0.5 + K_y \times I_y + (K_y/2)$ と $K_x \times I_x + (K_x/2)$ とする．補間テーブルには，Y_{index} と X_{index} を連結したインデックス情報が指し示す箇所に，区間の代表値を用いて式

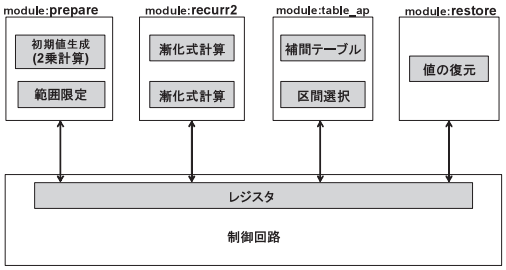

図 3 補間テーブルを用いた回路構成
Fig. 3 The circuit diagram using the table reference.

(7) より計算した値を保持する．読み出した値を，漸化式計算で求めた商に連結し，計算結果とする．漸化式で求めるビット数と補間テーブルで求めるビット数の振り分けは，対象となるシステム用途に応じて最適なものを選ぶ．

3.2 計算精度

補間テーブルを用いた場合の計算精度の評価を行った．本回路を適用する粒子追跡システム[8] は整数部 24 ビット，小数部 20 ビットの固定小数点方式を用いており，演算回路の出力を小数点以下 20 ビットとする．演算回路のビット幅については，6 章で述べる．2.3 節で述べた範囲限定を行っていることから，商は 0.25〜1 の値となり，閾値 0.121421（相関値計算式 (1) の閾値 0.7 にマスク輝度の標準偏差を求める式 (2) を乗じた値）付近になる可能性があるものは，値の復元を行う際に，2 または 3 ビットの右シフトを必要とする．このことから，小数点以下 18 ビットまで求める．

粒子追跡システムで使用しているデータを用いて，漸化式のみで計算を行った場合と，漸化式計算回数 14 回と 4 ビット補間テーブルを組み合わせた場合との精度比較を行った．4 ビット補間テーブルを用いた場合のインデックス情報の生成に用いる K_y および K_x の値は，$K_y = 0.046875$（32 分割）と $K_x = 0.0625$（64 分割）である．インデックス情報に必要なビット数は，Y_{index} に 5 ビット，X_{index} に 6 ビットの計 11 ビットとなる．

全体の回路構成を**図 3** に示す．演算処理を行う 4 つのモジュールと，それを制御する回路で構成した．初期値生成の 2 乗計算には 2.2 節で述べた回路を用いた．各モジュールは 1 サイクルで処理を行う．動作周波数は 100 MHz で合成を行った．

漸化式計算 14 回と 4 ビット補間テーブルを組み合わせた場合の回路規模，処理時間，および計算精度を**表 2** に示す．補間テーブルを使用せずに計算精度が同等となる漸化式の計算回数（18 回）を行う回路と比較した．誤差は，粒子追跡システムで使用しているデータを 10 万個処理したときの，相対誤差の平均値をとった．

表 2 4 ビット補間テーブルを用いた回路性能
Table 2 The circuit performance (using 4 bit table).

	回路規模 (gate)	メモリサイズ (bit)	処理時間 (ns)	誤差 ($\times 10^{-5}$)
4 ビット補間テーブル使用（計算回数 14 回）	9,327	8,192	100	1.480
漸化式の計算回数 18 回(補間テーブルなし)	7,451	-	110	1.467

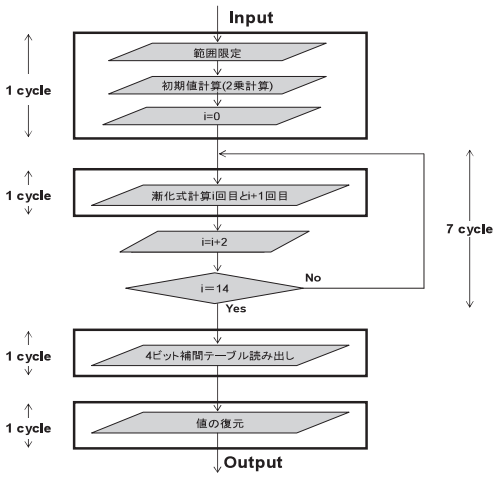

図 4 補間テーブル使用時のフローチャート
Fig. 4 The flowchart of the correlation coefficient calculation circuit using the table reference.

補間テーブルを用いることによりメモリが新たに必要となったが，処理時間は 1 サイクル削減できた．4 ビット補間テーブルを使用したときの 1 データの処理に必要なサイクル数（10 サイクル）の内訳を**図 4** に示した．範囲限定と初期値計算（2 乗計算）を 1 サイクルで行う．漸化式計算は，1 サイクルで 2 回分行うため，7 サイクルとなる．テーブル読み出しと値の復元にはそれぞれ 1 サイクルなので，合計 10 サイクルとなる．

図 5 には，補間テーブルを使用しなかった場合のサイクル数（11 サイクル）の内訳を示した．補間テーブルを用いた場合と比較すると，漸化式計算の回数が 14 回から 18 回に増加したことで，2 サイクル増加する．ただし，テーブル読み出しがないため，合計 11 サイクルとなる．

また，補間テーブルが 4 ビット，6 ビット，8 ビット

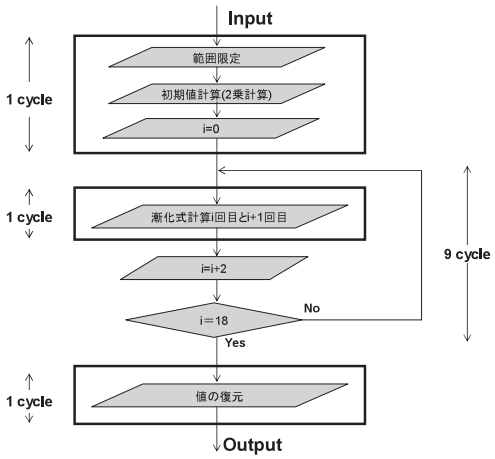

図 5 補間テーブル未使用時のフローチャート
Fig. 5 The flowchart of the correlation coefficient calculation circuit without the table reference.

表 3 補間テーブルにおけるビットサイズ比較
Table 3 The comparison of the bit length.

テーブルビット数	方向分割数		誤差 ($\times 10^{-5}$)	メモリサイズ (bit)
	Y	X		
-	-	-	1.467	-
4	16	32	1.495	2,048
	32	64	1.480	8,192
	64	128	1.479	32,768
6	16	32	1.856	3,072
	32	64	1.561	12,288
	64	128	1.492	49,152
8	16	32	5.140	4,096
	32	64	2.812	16,384
	64	128	1.954	65,536

の 3 つの場合についての比較を行った（**表 3**）．この比較から，4 ビット補間テーブルが 6 ビット，8 ビットの場合に比べ，少ないメモリ量で同等の精度を得ているので，以下では，4 ビット補間テーブルを用いる．

4. 回路規模の削減

補間テーブルを用いた場合，インデックス情報の生成部と補間テーブルが付加されることから回路規模が増加してしまう．特に，分割した区間からの選択に用いるセレクタにより回路規模が増加する．そこで，回路規模縮小のため，セレクタを用いないインデックス情報の生成手法を考える．

分割した区間の中から X_n および Y_n がどの区間に存在するかの選択において，X_n は 0〜4 の値を 2^α 分割している．4 を 2^α で割ると，式 (10) 中の K_x の値は，$2^{-(\alpha-2)}$ となり，以下のようになる．

$$2^{-(\alpha-2)} \times I_x \leq X_n < 2^{-(\alpha-2)} \times (I_x + 1) \quad (11)$$

X_n が式 (11) を満たすとき，X_n の $2^{-(\alpha-2)}$ 未満のビットをいずれの値に変化させても，式 (11) を満たす．$2^{-(\alpha-2)}$ 未満のビットを切り捨てると，式 (11) は以下のようになる．

$$2^{-(\alpha-2)} \times I_x = X_n \quad (12)$$

まず，X_n の値が 4 以上であるか 4 以下であるかを判別する．4 以下なら X_n の整数部 2 ビット，小数部は $(\alpha - 2)$ ビットで十分であり（$2^{-(\alpha-2)}$ より），これらを直接 X_{index} とすることができる．これによりセレクタを用いずにインデックス情報を得ることができる．なお，X_n が 4 以上のときは，Y_n が 0.5〜2 であるため，式 (7) で計算しても，X_n の符号変化が起こらないため，答えはすべて 1 となり，1111 を漸化式の解に連結して計算を終了する．

$X_n = 0.21875$，区間を 64 分割したときの例を示す．$\alpha = 6$ となる．式 (11) に α および X_n を代入すると

$$0.0625 \times I_x \leq 0.21875 < 0.0625 \times (I_x + 1) \quad (11.1)$$

となる．$I_x = 3$ のとき以下の式となり，これは式 (11.1) を満たす．

$$0.1875 \leq 0.21875 < 0.25 \quad (11.2)$$

式 (11.2) 中の値を小数点以下 4 ビットの 2 進数で表すと以下のようになる．

$$0011 \leq 0011 < 0100 \quad (11.3)$$

X_n と比較する値は，$0.0625 \times (2^{-4})$ 間隔で変化しているため，小数点以下 4 ビット未満のビットは不必要となる．$X_n = 0.21875$ であるため，X_n の整数部 2 ビット (00) + 小数部 4 ビット (0011) から $X_{\mathrm{index}} = (000011)_2$ となり，式 (11.1) を満たす $I_x = 3$ と等しくなることが分かる．

このように，2 のべき乗間隔で分割することで，セレクタは不必要となる．

Y_n は 0.5〜2 を 2^β で分割すると，2 のべき乗間隔の分割にならないため，セレクタが必要になる．ここで，Y_n を 0〜2 に変更し，2^β 分割することで，式 (9) を以下に変更する．

$$2^{-(\beta-1)} \times I_y \leq Y_n < 2^{-(\beta-1)} \times (I_y + 1) \quad (13)$$

Y_n のインデックス情報の生成は，Y_n の整数部 1 ビッ

ト + 小数部 $(\beta-1)$ ビットから直接 Y_{index} を生成する．

セレクタ削減後の回路構成は，**図 6** に示すように補間テーブル処理と値の復元を 1 サイクルで行うように変更した．**図 7** に変更後のサイクル数（9 サイクル）の内訳を示す．漸化式計算までは図 4 と同様であるが，1 サイクルでテーブル読み出しと値の復元を行うようにしたため，合計 9 サイクルとなる．

4 ビット補間テーブルを使用した場合のセレクタ削除効果を**表 4** に示す．Y_n の区間を大きくしたことから，若干の精度低下が見られたが，約 18%の規模削減と 1 サイクルの高速化が得られた．

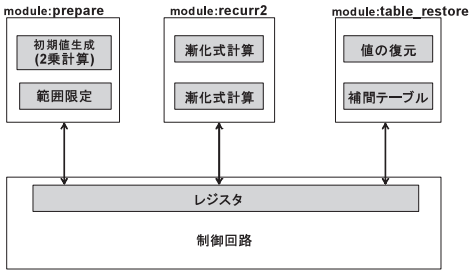

図 6 セレクタ削減後の回路構成
Fig. 6 The selectors' circuit reduction.

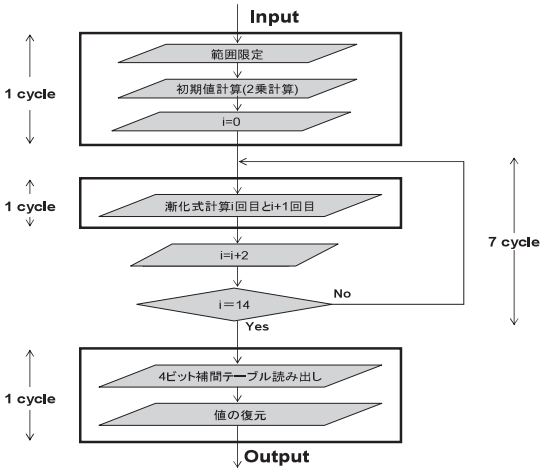

図 7 セレクタ削減後のフローチャート
Fig. 7 The flowchart after the selectors' circuit reduction.

表 4 セレクタ削減の効果
Table 4 The circuit size reduction.

セレクタ	回路規模 (gate)	メモリサイズ (bit)	処理時間 (ns)	誤差 ($\times 10^{5}$)
削減前	9,327	8,192	100	1.480
削減後	7,697	8,192	90	1.484

5. 閾値判別

閾値との大小を判別することが目的であるので，計算途中で判別がついた場合は，計算を終了することができる．閾値判別は，計算後の値の復元時に用いるビットシフト数が求まった段階と，漸化式計算で求まる途中結果を用いて行う．以下に判別手順を示す．

ステップ 1 復元時のシフト数で判別

2.3 節で述べた範囲に値を限定していることから，復元前の値は，0.25〜1 までの値となる．この値が閾値付近の値になるには，2 および 3 ビットの右シフトをする必要がある（3.2 節参照）．よって，復元時のシフト数が 2 および 3 ビットの右シフト以外の場合には，漸化式の計算を省略しても，閾値との大小を判別できる．

ステップ 2 漸化式計算ごとで求まる途中結果で判別

漸化式計算の各段階で以下の 4 種類の条件判別を行い，条件を満たす場合，閾値以上もしくは以下として計算を終了する．以下では，閾値を S とし，中間結果を M_n として表す．ビットシフト数を i とし，$i=2$, 3 である．

1) 復元時に i ビットシフトで，閾値以上となる条件
復元時に i ビットシフトするため，$(M_n/2^i) > S$ となれば，閾値以上と判別できる．本文で採用した漸化式は，小数点以下 1 桁目から順に解を求めるため，$M_n \leq M_{n+1}$ となる．このことから，計算途中で得られた M_n が $2^i \times S$ を超えた時点で閾値以上と判別できる．

2) 復元時に i ビットシフトで，閾値以下となる条件
M_n が得られたとき，その後の計算で求まる解の最大値（M_n 以下のビットがすべて 1 の場合）を $M_{\max}$ とすると，復元時に i ビットシフトするため，$M_{\max}$ が $(M_{\max}/2^i) < S$ を満たす値であれば，閾値以下と判別できる．よって，$M_{\max}$ の値が $2^i \times S$ より小さくなった時点で閾値以下と判別できる．

閾値判別機能を追加した回路の規模および処理時間を**表 5** に，閾値判別の各段階ごとに判別されたデー

表 5 閾値判別機能の効果
Table 5 The effect of the threshold value judgment.

	処理時間 (ns)	回路規模 (gate)	メモリサイズ (bit)
閾値判別機能なし	90.00	7,697	8,192
閾値判別機能あり	20.71	8,192	8,192

表 6 閾値判別の分布
Table 6 The distribution of the calculation finishing process.

第1ステップ		第2ステップ		ループ数							
			しきい値	1	2	3	4	5	6		計
シフト数による判別		復元時に2ビットシフト	以下	-	17,723	2,595	-	35	8	最後まで計算	20,361
			以上	712	-	-	362	64	15		1,153
		復元時に3ビットシフト	以下	97,422	3,188	58	-	4	-		100,672
			以上	-	-	3	8	1	-		12
回数	277,703			98,134	20,911	2,656	370	104	23	99	400,000
割合	69.43%			24.53%	5.28%	0.66%	0.09%	0.03%	0.00%	0.02%	100.00%

表 7 小数部のビット幅による性能比較
Table 7 The comparison of the bit size.

手法	小数部ビット数	回路規模(gate)	処理時間(ns)	メモリサイズ(bit)	誤差（$\times 10^{-5}$）
4ビット補間テーブル使用	20ビット	9,690	90.00	8,192	0.759
	16ビット	7,697	90.00	8,192	1.484
閾値判別機能追加	20ビット	10,125	20.71	8,192	-
	16ビット	8,192	20.71	8,192	-

表 8 設計結果
Table 8 Design results.

	回路規模(gate)	メモリサイズ(bit)	処理時間(ns)	誤差（$\times 10^{-5}$）
Booth乗算＋補間テーブルなし	10,823	-	110.00	1.467
2乗専用回路＋補間テーブルなし	7,451	-	110.00	1.467
2乗専用回路＋補間テーブル使用	7,697	8,192	90.00	1.484
2乗専用回路＋補間テーブル使用＋閾値判別	8,192	8,192	20.71	

タ個数を**表 6** に示す．表中の処理時間は，粒子追跡システムで使用しているデータを 40 万個処理した時間を示した．閾値判別機能ありの場合は，値により処理時間が異なるため，粒子追跡システムで使用しているデータを 40 万個処理したときの平均時間を示した．閾値判別機能は，40 万個中の約 70%をステップ 1，残りの大半もステップ 2 で判別できており，処理を大幅に削減できている．

表 6 中の各段階ごとに示した判別個数の中に，判別なしの箇所がある．ここでは，判別回路は動作せず，次のサイクルに判別を行う．これらの箇所は，以下に示す例についての理由と同様に判別できない．

たとえば，ループ 1 で，復元時に 3 ビットシフトの場合，小数点以下 1 ビット目および 2 ビット目の解を求める．得られる解の最大値（小数点以下 2 ビットがともに 1）は 0.75 となり，$8 \times S$ を超えず，上記の条件を満足しない．よって，ループ 1 終了時点では，復元時に 3 ビットシフトを行う場合，閾値以上と判別できない．

6. 設計結果

まず，演算回路の規模を削減するため，回路のビット幅について検討した．粒子追跡システムで使用している小数部のビット幅である 20 ビット（実際に求めるビット数 18 ビットから 2 ビットの余裕を持つ）で計算した場合と，漸化式で求めるビット数（14 ビット）から 2 ビットの余裕を持たせた 16 ビットで計算を行った場合との比較結果を**表 7** に示す．16 ビットの場合は，アンダーフローの影響があったが，精度低下が 0.5 桁程度であった．漸化式の計算は 1 ビット単位であり，かつ補間テーブルを用いていたためと考えられる．4 ビット分の回路規模削減効果も考え，小数部 16 ビット幅で回路を構成した．

本研究の設計結果を示す．漸化式の初期値生成に

2 乗専用回路を用いたことによる回路規模削減効果と，小数点以下 18 ビットの解を求めるとき，漸化式の計算のみで行う場合と比較し，補間テーブルを用いた場合および閾値判別機能を追加したことによる処理時間短縮効果を**表 8** に示す．閾値判別機能追加により，約 4.5 倍の高速化を実現できた（表 5）．

本文に示した結果は Synopsys 社の Design Compiler を使用し，ライブラリは HITACHI 0.18 μm を用いて合成を行った．

7. ま と め

分母に平方根を持つ除算を同時処理するアルゴリズムに 2 乗専用回路を用いることで，回路規模を削減した．また，漸化式計算と補間テーブルを併用することで，補間テーブルを使用しなかった場合と比較するとメモリ規模は増加していたものの，精度の低下を抑えつつ，処理時間を 2 サイクル短縮した．さらに，計算途中で閾値を判別する機能を追加することで，約 4.5 倍の高速化を達成した．

謝辞 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，シノプシス株式会社の協力で行われたものである．

参 考 文 献

1) 熊澤文雄，高木直史：算術演算のための減算シフト型ハードウェアアルゴリズムの自動合成，情報処理学会研究報告，SLDM-117, pp.245–248 (2004).
2) 高木直史：算術演算の VLSI アルゴリズム，コロナ社，東京 (2005).
3) Koren, I.: *Computer Arithmetic Algorithms*, A K Peters (2002).
4) Muller, J.M.: *Elementary Functions*, Birkhäuser (1997).
5) 可視化情報学会（編）：PIV ハンドブック，森北出版，東京 (2002).
6) 末松良一，山田宏尚：画像処理工学，コロナ社，東京 (2002).
7) Takehara, K. and Eto, T.: A study on Particle Identification in PTV, *Journal of Visualization*, Vol.1, No.3 (1999).
8) Jyoko, K., Ohguchi, T., Uetsu, H., Sakai, K., Ohkura, T. and Kambe, T.: C-based Design of a Particle Tracking System, *Proc. SASIMI2006* (Apr. 2006).
9) 大窪啓太，朝利壮吾，矢野智則，神戸尚志：特定用途向け低ビット複合演算回路設計，デザインガイア (Dec. 2005).

(平成 18 年 9 月 1 日受付)
(平成 19 年 2 月 1 日採録)

大窪 啓太

平成 16 年近畿大学理工学部電気電子工学科卒業，平成 18 年同大学院エレクトロニクス専攻博士前期課程修了．同年 NEC エレクトロニクスに入社．システム LSI 設計に従事．

神戸 尚志（正会員）

昭和 53 年大阪大学大学院工学研究科博士前期課程修了．同年シャープ（株）に入社．平成 4 年 2 月博士（工学）号を取得（大阪大学）．平成 15 年近畿大学理工学部電気電子工学科教授，現在に至る．システム LSI 設計技術，ソフトウェア・ハードウェア協調設計，C 言語設計手法等の研究に従事．IEEE，ACM，電子情報通信学会，システム制御情報学会各会員．