論　文

浮動小数点ユークリッドノルム計算回路

熊澤　文雄[†a)]　高木　直史[†]　武内　大輔[†]　高木　一義[†]

Floating-Point Euclidean Norm Computing Circuit

Fumio KUMAZAWA[†a)], Naofumi TAKAGI[†], Daisuke TAKEUCHI[†], and Kazuyoshi TAKAGI[†]

あらまし　三つの浮動小数点数で与えられる 3 次元ベクトルのユークリッドノルムを計算する回路の設計と評価について述べる．以前提案したハードウェアアルゴリズムの改良を行い，それに基づく回路を設計する．回路の入出力は IEEE754 浮動小数点標準の基本フォーマットとする．回路は，前処理回路，仮数部計算回路，正規化回路で構成する．仮数部計算回路については 3 種類の構成を示す．けた合せ処理を仮数部計算と統合することにより，回路全体のハードウェア量を削減する手法を提案する．回路の評価の結果，提案した手法により，回路全体の面積を約 20%削減することができた．平方算 3 回，加算 2 回，開平算 1 回に相当するユークリッドノルム計算を行う回路が，開平器に比べ，約 1.5 倍の遅延，約 2 倍の面積，約 2 倍の計算時間で実現できることが明らかになった．

キーワード　ユークリッドノルム，算術演算回路，ハードウェアアルゴリズム，3D グラフィックス，VLSI

1. まえがき

マイクロプロセッサや信号処理プロセッサ，グラフィックスエンジンなどにおいて，高い算術演算性能が求められている．高い算術演算性能を実現するためには，基本算術演算の高速化とともに，いくつかの基本算術演算の組合せで実現されている複合演算のための専用回路を構成することも有効であると考えられる．出現頻度の高い複合算術演算の専用回路を付加することにより，その複合演算を高速化するとともに，他の基本算術演算回路，特に，加算器や乗算器をこの複合演算の計算から解放することができ，プロセッサ全体の演算性能を大幅に向上させることができるものと考えられる．

コンピュータグラフィックスなどで頻繁に現れ，現在いくつかの基本算術演算の組合せで実現されている演算として，3 次元ベクトルのユークリッドノルム計算（以下 3D ユークリッドノルム計算）が挙げられる．これは，ベクトル (F_1, F_2, F_3) のユークリッドノルム $\sqrt{F_1^2 + F_2^2 + F_3^2}$ を求める計算であり，頻繁に出現するベクトルの正規化の過程で用いられる．この計算は通常，積和演算器，あるいは乗算器と加算器によってベクトルの要素の 2 乗和 $F_1^2 + F_2^2 + F_3^2$ を計算し，この値の開平を行うことで実現されている．

これまでに，CORDIC 法により 2 次元ベクトルのユークリッドノルムを計算できるということが知られている [1]．しかし，この手法を 3D ユークリッドノルム計算に拡張することは困難である．3D ユークリッドノルムを計算するアルゴリズムの提案や，この計算のための専用回路の実現に関する研究は，我々が知る限りではほかに見当たらない．

我々は 3D ユークリッドノルム計算の専用回路の実現を目指し，VLSI での実現に向いたハードウェアアルゴリズムを提案してきた [2], [3]．このアルゴリズムは減算シフト型のアルゴリズムであり，三つの浮動小数点数に対し，ユークリッドノルム計算の仮数部の計算を行う．

本論文では，以前提案した基数 2 及び基数 4 のアルゴリズムの改良を行い，それらに基づく，浮動小数点 3D ユークリッドノルム計算回路の設計及び評価を行う．入出力は IEEE754 標準の基本フォーマットで表される浮動小数点数とする．回路は，前処理回路，仮数部計算回路，正規化回路で構成する．仮数部計算回路については，1 サイクルで基数 2 の反復 1 回を行う

† 名古屋大学大学院工学研究科情報工学専攻，名古屋市
Department of Information Engineering, Nagoya University, Furo-cho, Chikusa-ku, Nagoya-shi, 464–8603 Japan
a) E-mail: kumazawa@takagi.nuie.nagoya-u.ac.jp

回路，基数 4 の反復 1 回を行う回路，及び，基数 2 の反復 2 回を行う回路（基数 2 の 2 段重ねの回路）の三つの回路構成を示す．また，けた合せと仮数部計算を統合し，回路全体のハードウェア量を削減する手法も提案する．

アルゴリズムの改良により，基数 2 では補正値の選択が簡単になり，基数 4 ではサイクル数を 2 減らすことができた．回路の評価の結果，基数 4 の回路及び基数 2 の 2 段重ねの回路は基数 2 の回路に比べ，計算時間が約 5〜20%短く，約 25% の面積の増加で実現できることがわかった．基数 4 の回路と基数 2 の 2 段重ねの回路を比較すると，面積はほぼ同じであり，計算時間は基数 2 の 2 段重ねの回路が約 15%高速となった．けた合せ処理を仮数部計算に埋め込むことにより，面積を約 20%削減することができた．設計した回路を開平器と比較すると，約 1.5 倍の遅延，約 2 倍の面積，約 2 倍の計算時間で実現できることが判明した．

以降，**2.** では，以前提案したハードウェアアルゴリズムについて述べる．**3.** では，基数 2 及び基数 4 のアルゴリズムの改良を行う．**4.** では，けた合せと仮数部計算を統合し，仮数部計算回路の中でけた合せを行う手法を提案する．**5.** では，設計した回路の構成を示し，**6.** でその評価を行う．

2. ハードウェアアルゴリズム

三つの浮動小数点数 F_1，F_2，F_3 に対し，$F_P = \sqrt{F_1^2 + F_2^2 + F_3^2}$ を仮数部の誤差が 1 ulp（unit in the last place）以内となるように求める．演算数 F_1，F_2，F_3，結果 F_P はともに，IEEE754 浮動小数点標準の基本フォーマットで表されているものとする．F_i の指数部を E_i，仮数部を M_i（$1 \leqq M_i < 2$）とし，$F_i = 2^{E_i} \cdot M_i$ とする．

F_1，F_2，F_3 の中で，指数が最大であるものを F_X とし，残りの二つを F_Y，F_Z とする．$E_X \geqq E_Y$，$E_X \geqq E_Z$ である．ここで，

$$\begin{aligned} X &= M_X \cdot 2^{-1} \\ Y &= M_Y \cdot 2^{E_Y - E_X - 1} \\ Z &= M_Z \cdot 2^{E_Z - E_X - 1} \end{aligned} \tag{1}$$

とし，指数部が 2^{E_X+1} となるように仮数部のけた合せを行うと，$F_P = \sqrt{F_X^2 + F_Y^2 + F_Z^2} = \sqrt{X^2 + Y^2 + Z^2} \cdot 2^{E_X+1}$ となる．

以前提案したハードウェアアルゴリズム [2], [3] は仮数部の計算を行うもので，$\frac{1}{2} \leqq X < 1$，$0 \leqq Y < 1$，$0 \leqq Z < 1$ である r 進小数 X，Y，Z（r は基数，2 のべき）に対し，$P = \sqrt{X^2 + Y^2 + Z^2}$ を計算する．ここに，X は n けたである．

$$-r^{-n} \leqq \sqrt{X^2 + Y^2 + Z^2} - P < r^{-n} \tag{2}$$

を満たす P を r 進で小数点以下 n けたまで求める．

P は，初期値を上位から 1 けたずつ順に補正していくことにより求める．j 回補正を繰り返した後の中間結果を $P[j]$ とすると，$P[j+1]$ は次の漸化式によって得られる．

$$P[j+1] = P[j] + q_{j+1} r^{-j-1} \tag{3}$$

補正値 q_{j+1} は冗長なけた集合 $\{-a, \cdots, -1, 0, 1, \cdots, a\}$ から選択する．ここに，$\frac{r}{2} \leqq a < r$ である．

残余（スケーリングした剰余）$W[j]$ を

$$W[j] = r^j (X^2 + Y \cdot Y_{j+h} + Z \cdot Z_{j+h} - P[j]^2) \tag{4}$$

と定義する．ここに，Y_{j+h}，Z_{j+h} はそれぞれ Y，Z の小数点以下 $j+h$ けた目までの値，すなわち $Y_{j+h} = [.y_1 y_2 \cdots y_{j+h}]$，$Z_{j+h} = [.z_1 z_2 \cdots z_{j+h}]$ である．h は非負の整数であり，補正値の選択が可能となるように定める．残余に関する漸化式は，

$$\begin{aligned} W[j+1] = rW[j] &+ Y y_{j+h+1} r^{-h} + Z z_{j+h+1} r^{-h} \\ &- 2P[j] q_{j+1} - q_{j+1}^2 r^{-j-1} \end{aligned} \tag{5}$$

となる．y_{j+h+1}，z_{j+h+1} はそれぞれ Y，Z の $j+h+1$ けた目である．漸化式中の加減算をけた上げ伝搬なしに高速に行うため，残余 $W[j]$ はけた上げ保存形で表現する．

式 (2) を満たす P を得るための $W[j]$ の範囲は

$$\begin{aligned} -2P[j]\rho + \rho^2 r^{-j} &\leqq W[j] \\ &< 2P[j]\rho + \rho^2 r^{-j} - 2r^{-h} \end{aligned} \tag{6}$$

となる．ここに，$\rho = \frac{a}{r-1}$ は補正値のけた集合の冗長度である．

中間結果及び残余の初期値は，$P[0] = X + 2^{-1}$，$W[0] = Y \cdot Y_h + Z \cdot Z_h - X - 2^{-2}$ とする．$\rho = 1$ のときは，$P[0] = X$，$W[0] = Y \cdot Y_h + Z \cdot Z_h$ とすることもできる．

残余の漸化式 (5) の $rW[j] - 2P[j]q_{j+1} - q_{j+1}^2 r^{-j-1}$ は開平の漸化式 [4] の右辺に一致する．また，$Y y_{j+h+1} r^{-h}$，$Z z_{j+h+1} r^{-h}$ は Y^2，Z^2 の部分積で

ある．すなわち，この漸化式では，開平計算と 2 乗和計算（$+Y^2+Z^2$）を重畳している．中間結果の初期値を $P[0]=X+2^{-1}$（あるいは $P[0]=X$）とするため，X の 2 乗計算を行う必要がない．

補正値 q_{j+1} を冗長なけた集合から選択するため，結果 $P[n]$ を最終的に非冗長表現に変換する必要がある．そこで，除算などで用いられている on-the-fly 変換 [5] を拡張し，中間結果 $P[j]$ を，$P[j]+r^{-j}$ の非冗長表現 $P[j]^+$ と $P[j]-r^{-j}$ の非冗長表現 $P[j]^-$ の二つで保持する．

アルゴリズムは以下のように表される．

[ユークリッドノルム計算アルゴリズム]

Step 1:

$P[0]^+ := X+2^{-1}+1;$

$P[0]^- := X+2^{-1}-1;$

$W[0] := Y\cdot Y_h + Z\cdot Z_h - X - 2^{-2};$

$(Y_h=[.y_1y_2\cdots y_h],\ Z_h=[.z_1z_2\cdots z_h])$

Step 2:

for $j := 0$ **to** $n-1$ **do**

$q_{j+1} := \mathbf{select}(r\hat{W}[j], \hat{P}[j]);$

$W[j+1] := rW[j] + Yy_{j+h+1}r^{-h} + Zz_{j+h+1}r^{-h} - 2P[j]q_{j+1} - q_{j+1}^2 r^{-j-1};$

$P[j+1]^+$ と $P[j+1]^-$ を拡張 on-the-fly 変換の規則に従って計算する；

end for

Step 3:

結果 $P[n]$ を $P[n]^+$ と $P[n]^-$ から得る；

ここに，select() はけた選択関数，$r\hat{W}[j]$ は $rW[j]$ の小数点以下 t ビットまでの値，$\hat{P}[j]$ は $P[j]$ の小数点以下 d ビットまでの値である．冗長なけた集合を用いているので，補正値 q_{j+1} を見積り値 $r\hat{W}[j]$ 及び $\hat{P}[j]$ から定めることができる．

けた選択関数は，選択の境界となる値の集合 $\{m_k(\hat{P}[j])|k=-a+1,-a+2,\cdots,a\}$ で表される．$m_k(\hat{P}[j]) \leqq r\hat{W}[j] < m_{k+1}(\hat{P}[j])$ ならば，q_{j+1} として k が選ばれる．けた選択関数は，

$$\max_{\hat{P}[j]}(L_k[j]) \leqq m_k(\hat{P}[j]) \leqq \min_{\hat{P}[j]}(U_{k-1}[j]) - 2^{-t} - 2(r-1)r^{-h} \qquad (7)$$

を満たす 2^{-t} の倍数となるように定める．

$k>0$ のとき，

$$\max_{\hat{P}[j]}(L_k[j]) = 2(\hat{P}[j]+2^{-d})(k-\rho) + (k-\rho)^2 r^{-j-1}$$

$$\min_{\hat{P}[j]}(U_{k-1}[j]) = 2\hat{P}[j](k-1+\rho) + (k-1+\rho)^2 r^{-j-1} - 2r^{-h} \qquad (8)$$

$k \leqq 0$ のとき，

$$\max_{\hat{P}[j]}(L_k[j]) = 2\hat{P}[j](k-\rho) + (k-\rho)^2 r^{-j-1}$$

$$\min_{\hat{P}[j]}(U_{k-1}[j]) = 2(\hat{P}[j]+2^{-d})(k-1+\rho) + (k-1+\rho)^2 r^{-j-1} - 2r^{-h} \qquad (9)$$

となる．これらの値は j に依存するので，j によってけた選択関数が異なることもある．

Step 1 は，$j=-h$ から -1 まで，$P[j]^+$，$P[j]^-$ を更新せず，$q_{j+1}=0$ として Step 2 の反復を h 回実行することにより実現できる．Step 3 は，$j=n$ で $q_{j+1}=0$ として Step 2 の反復を 1 回実行することにより実現できる．

3. アルゴリズムの改良

3.1 基数 2 のアルゴリズムの改良

基数 r が 2，補正値 q_{j+1} のけた集合が $\{-1,0,1\}$（$a=1$，$\rho=1$）の場合について考える．以前提案したアルゴリズム [2], [3] では，$t=2$，$h=3$ とすることにより，けた選択関数をすべての j について $m_0=-\frac{3}{4}$，$m_1=0$ としてきた．しかし，より詳細な考察により，$t=1$，$h=3$ とすることにより，すべての j について $m_0=-1$，$m_1=0$ とできることが判明した．これにより，補正値 q_{j+1} の選択に必要な $2W[j]$ のビット数が 1 少なくなる．以下に導出過程を示す．

基数 2 の場合，漸化式は

$$P[j+1] = P[j] + 2^{-j-1}q_{j+1}$$

$$W[j+1] = 2W[j] + Yy_{j+h+1}2^{-h} + Zz_{j+h+1}2^{-h} - 2P[j]q_{j+1} - 2^{-j-1}q_{j+1}^2 \qquad (10)$$

となる．$\rho=1$ より，初期値を $P[0]=X$，$W[0]=Y\cdot Y_h+Z\cdot Z_h$ とすることができる．

式 (8)，(9) より，$\min_{\hat{P}[j]}(U_{-1}[j]) = -2^{-h+1}$，$\max_{\hat{P}[j]}(L_0[j]) = -2\hat{P}[j]+2^{-j-1}$，$\min_{\hat{P}[j]}(U_0[j]) = 2\hat{P}[j]+2^{-j-1}-2^{-h+1}$，$\max_{\hat{P}[j]}(L_1[j])=0$ であり，

式 (7) より，すべての j に対して共通のけた選択関数を定めるには，

$$-2\hat{P}[j]+2^{-1} \leqq m_0 \leqq -2^{-h+1}-(2^{-t}+2^{-h+1}) \tag{11}$$

$$0 \leqq m_1 \leqq 2\hat{P}[j] - 2^{-h+1} - (2^{-t}+2^{-h+1}) \tag{12}$$

を満たさなければならない．$\hat{P}[j] \geqq 2^{-1}$ なので，式 (12) より，

$$2^{-t}+2^{-h+2} \leq 1 \tag{13}$$

であれば $m_1 = 0$ と定めることができる．

ここで，$j = 0$ の場合について考える．$W[0] = Y \cdot Y_h + Z \cdot Z_h \geqq 0$ であり，これをけた上げ保存形で表現するとき，常に $2\hat{W}[0] \geqq 0$ となる．よって，$m_1 = 0$ と定めることができると仮定すると，$m_1 \leqq 2\hat{W}[0]$ より，必ず q_1 として 1 が選ばれるので，$j = 0$ のときの m_0 を -2^{-t} 以下の任意の数とすることができる．このとき，$\hat{P}[1] \geqq 2^{-1}+2^{-1}$，$\hat{P}[2] \geqq 2^{-1}+2^{-2}$，$\cdots$，$\hat{P}[j] \geqq 2^{-1}+2^{-j}$，$\cdots$ となるので，$j \geqq 1$ の場合，$\max_{\hat{P}[j]}(L_0[j]) = -2\hat{P}[j] + 2^{-j-1} \leqq -2(2^{-1}+2^{-j})+2^{-j-1} = -1$ となる．したがって，式 (11) より，$t = 1$，$h = 3$ として，$m_0 = -1$ と定めることができる．また，これらの値は式 (13) を満たすので，$m_1 = 0$ と定めることができる．

基数 2 の改良アルゴリズムは以下のようになる．

[基数 2 のアルゴリズム]

Step 1:

$P[0]^+ := [1.1x_2 \cdots x_n]$;

$P[0]^- := [(-1).1x_2 \cdots x_n]$;

$W[0] := Y \cdot Y_3 + Z \cdot Z_3$;

$(Y_3 = [.y_1y_2y_3], Z_3 = [.z_1z_2z_3])$

Step 2:

for $j := 0$ **to** $n-1$ **do**

$$q_{j+1} := \begin{cases} -1 & \text{if} \quad & & 2\hat{W}[j] & \leqq & -\frac{3}{2} \\ 0 & \text{if} \quad -1 & \leqq & 2\hat{W}[j] & \leqq & -\frac{1}{2} \\ 1 & \text{if} \quad 0 & \leqq & 2\hat{W}[j] & & \end{cases};$$

（$2\hat{W}[j]$ は $2W[j]$ の小数点以下 1 ビットまで）

$W[j+1] := 2W[j] + Yy_{j+4}2^{-3} + Zz_{j+4}2^{-3} - 2q_{j+1}(P[j] + 2^{-j-2}q_{j+1})$;

$P[j+1]^+$ と $P[j+1]^-$ を拡張 on-the-fly 変換の規則に従って計算する；

end for

Step 3:

結果 $P[n]$ を $P[n]^+$ と $P[n]^-$ から得る；

3. 2　基数 4 のアルゴリズムの改良

基数 r が 4，補正値 q_{j+1} のけた集合が $\{-2,-1,0,1,2\}$（$a = 2$，$\rho = \frac{2}{3}$）の場合について考える．以前提案したアルゴリズム [2], [3] では，$d = 5$，$t = 5$，$h = 5$ としていた．しかし，より詳細な考察により，$d = 6$，$t = 5$，$h = 3$ や，$d = 5$，$t = 4$，$h = 4$ や，$d = 5$，$t = 7$，$h = 3$ などとできることが判明した．これにより，h を小さくできるので，反復回数が少なくなる．以下に導出過程を示す．

基数 4 の場合，漸化式は

$$\begin{aligned} P[j+1] &= P[j] + 4^{-j-1}q_{j+1} \\ W[j+1] &= 4W[j] + Yy_{j+h+1}4^{-h} + Zz_{j+h+1}4^{-h} \\ &\quad - 2P[j]q_{j+1} - 4^{-j-1}q_{j+1}^2 \end{aligned} \tag{14}$$

となる．初期値は $P[0] = X + 2^{-1}$，$W[0] = Y \cdot Y_h + Z \cdot Z_h - X - 2^{-2}$ とする．

基数 r が 4 の場合，$j \geqq 1$ に対して共通のけた選択関数を定めることができる．$j = 0$ の場合は別個に考える [2], [3]．式 (8)，(9) より $\max_{\hat{P}[j]}(L_k[j])$，$\min_{\hat{P}[j]}(U_{k-1}[j])$ が求まり，更に式 (7) より，$j \geqq 1$ に対して共通のけた選択関数を定めるには，

$$\begin{aligned} &-\frac{10}{3}\hat{P}[j] + \frac{25}{9}\cdot 4^{-2} \leqq m_{-1} \\ &\quad \leqq -\frac{8}{3}\hat{P}[j] - \frac{8}{3}\cdot 2^{-d} - 2\cdot 4^{-h} - (2^{-t}+6\cdot 4^{-h}) \end{aligned} \tag{15}$$

$$\begin{aligned} &-\frac{4}{3}\hat{P}[j] + \frac{4}{9}\cdot 4^{-2} \leqq m_0 \\ &\quad \leqq -\frac{2}{3}\hat{P}[j] - \frac{2}{3}\cdot 2^{-d} - 2\cdot 4^{-h} - (2^{-t}+6\cdot 4^{-h}) \end{aligned} \tag{16}$$

$$\begin{aligned} &\frac{2}{3}\hat{P}[j] + \frac{2}{3}\cdot 2^{-d} + \frac{1}{9}\cdot 4^{-2} \leqq m_1 \\ &\quad \leqq \frac{4}{3}\hat{P}[j] - 2\cdot 4^{-h} - (2^{-t}+6\cdot 4^{-h}) \end{aligned} \tag{17}$$

$$\begin{aligned} &\frac{8}{3}\hat{P}[j] + \frac{8}{3}\cdot 2^{-d} + \frac{16}{9}\cdot 4^{-2} \leqq m_2 \\ &\quad \leqq \frac{10}{3}\hat{P}[j] - 2\cdot 4^{-h} - (2^{-t}+6\cdot 4^{-h}) \end{aligned} \tag{18}$$

を満たされなければならない．$\hat{P}[j] \geqq 2^{-1}$ である．

ここで，$j = 1$ の場合について考える．q_1 として -2 が選ばれたとき，$W[1] = 4W[0] + Yy_{h+1}4^{-h} + Zz_{h+1}4^{-h} - 2P[0]q_1 - 4^{-1}q_1^2 = 4(Y \cdot Y_h + Z \cdot Z_h) +$

$Yy_{h+1}4^{-h}+Zz_{h+1}4^{-h}\geqq 0$ となるので，q_2 として -2 が選ばれることはない．したがって，q_2 として -2 が選ばれるのは，$P[0]\geqq 1$，$q_1\in\{-1,0,1,2\}$ より，$P[1]\geqq\frac{3}{4}$ の場合に限る．よって，$\max_{\hat{P}[j]}(L_{-1}[j])$ を $j=1$ の場合と $j\geqq 2$ の場合で別々に考えることにより，式 (15) を，

$$-\frac{10}{3}\hat{P}[1]+\frac{25}{9}\cdot 4^{-2}\leqq m_{-1}\leqq -\frac{8}{3}\hat{P}[1]-\frac{8}{3}\cdot 2^{-d}-2\cdot 4^{-h}-(2^{-t}+6\cdot 4^{-h}) \tag{19}$$

$j\geqq 2$ の場合，

$$-\frac{10}{3}\hat{P}[j]+\frac{25}{9}\cdot 4^{-3}\leqq m_{-1}\leqq -\frac{8}{3}\hat{P}[j]-\frac{8}{3}\cdot 2^{-d}-2\cdot 4^{-h}-(2^{-t}+6\cdot 4^{-h}) \tag{20}$$

と置き換えることができる．ここに $\hat{P}[1]\geqq\frac{3}{4}$ である．式 (16)，(17)，(18)，(19)，(20) より，$d=6$，$t=5$，$h=3$ や，$d=5$，$t=4$，$h=4$ や，$d=5$，$t=7$，$h=3$ などとすることができる．

ここでは Step 1 のための反復回数に影響する h の値をできるだけ小さくするため，$d=5$，$t=7$，$h=3$ を採用する．けた選択関数を，$m_{-1}=-3\hat{P}[j]-2^{-4}$，$m_0=-\hat{P}[j]$，$m_1=\hat{P}[j]$，$m_2=3\hat{P}[j]+2^{-5}$ と定めることができる．

$j=0$ の場合を考える．$j=0$ のときのけた選択関数を $\{m_k^0|k=-1,0,1,2\}$ とすると，$\hat{P}[0]\geqq 1$ より，$m_{-1}^0=-3\hat{P}[0]-2^{-4}+2^{-1}=m_{-1}+2^{-1}$，$m_0^0=-\hat{P}[0]=m_0$，$m_1^0=\hat{P}[0]=m_1$，$m_2^0=3\hat{P}[0]+2^{-5}+2^{-1}=m_2+2^{-1}$ と定めることができる．

基数 4 の改良アルゴリズムは以下のようになる．

[基数 4 のアルゴリズム]

Step 1:

$P[0]^+:=[2.(x_1-2)x_2\cdots x_n]$;

$P[0]^-:=[0.(x_1-2)x_2\cdots x_n]$;

$W[0]:=Y\cdot Y_3+Z\cdot Z_3-X-2^{-2}$;

$(Y_3=[.y_1y_2y_3], Z_3=[.z_1z_2z_3])$

Step 2:

for $j:=0$ **to** $n-1$ **do**

if $j=0$ **then** $e:=2^{-1}$; **else** $e:=0$;

$q_{j+1}:=$

$$\begin{cases} -2 \text{ if } & 4\hat{W}[j]<-3\hat{P}[j]+e-2^{-4} \\ -1 \text{ if } -3\hat{P}[j]+e-2^{-4}\leqq & 4\hat{W}[j]<-\hat{P}[j] \\ 0 \text{ if } & -\hat{P}[j]\leqq 4\hat{W}[j]<\hat{P}[j] \\ 1 \text{ if } & \hat{P}[j]\leqq 4\hat{W}[j]<3\hat{P}[j]+e+2^{-5} \\ 2 \text{ if } & 3\hat{P}[j]+e+2^{-5}\leqq 4\hat{W}[j] \end{cases};$$

（$\hat{P}[j]$ は $P[j]$ の小数点以下 5 ビットまで）

（$4\hat{W}[j]$ は $4W[j]$ の小数点以下 7 ビットまで）

$W[j+1]:=4W[j]+Yy_{j+4}4^{-3}+Zz_{j+4}4^{-3}-2q_{j+1}(P[j]+\frac{1}{2}q_{j+1}4^{-j-1})$;

$P[j+1]^+$ と $P[j+1]^-$ を拡張 on-the-fly 変換の規則に従って計算する；

end for

Step 3:

結果 $P[n]$ を $P[n]^+$ と $P[n]^-$ から得る；

4. けた合せと仮数部計算の統合

浮動小数点数に対する 3D ユークリッドノルム計算では，仮数部計算の前に，指数部を比較して仮数部のけた合せを行う処理が必要となる．このけた合せを行う回路と仮数部計算回路を別個に実現する場合，けた合せ処理回路は仮数部計算回路への入力である X，Y，Z を出力する．F_Y の仮数部 M_Y と F_Z の仮数部 M_Z のけた合せを行うために，最大 n けた右シフトするバレルシフタが二つ必要となる．よって，回路の面積が大きくなり，また，けた合せ処理の計算時間が回路全体のクロックサイクルを大きくする場合も考えられる．そこで，けた合せ処理を仮数部計算に埋め込むことを考える．以下では，仮数部計算回路が基数 2 のアルゴリズムに基づく場合について述べる．

けた合せされた仮数 $Y=M_Y\cdot 2^{E_Y-E_X-1}$ について考える．$1\leqq M_Y<2$ より，Y の範囲は，

$$2^{E_Y-E_X-1}\leqq Y<2^{E_Y-E_X} \tag{21}$$

となる．これは，Y の小数点以下 E_X-E_Y ビット目までは 0 が連続する，すなわち，

$$Y=[.\underbrace{0\cdots 0}_{E_X-E_Y}1y_{E_X-E_Y+2}\cdots y_{E_X-E_Y+n}] \tag{22}$$

であることを示している．

仮数部計算回路において Y は，Y^2 の部分積 $Yy_{j+4}2^{-3}$ を計算するために用いられる．Step 1 と Step 2 で，反復回数 j の値は -3（$=-h$）から $n-1$ まで変化するので，1 反復ごとに，$Yy_1 2^{-3}$ か

ら $Yy_{n+3}2^{-3}$ までの部分積が計算される．式 (22) より，最初の $E_X - E_Y$ 回の反復の間，部分積の値は 0 であり，Y の値が必要になるのは $E_X - E_Y + 1$ 回目からである．したがって，M_Y のけた合せと Y^2 の部分積の計算を次のような手法で行うことができる．

（1） シフト前の仮数 $M_Y \cdot 2^{-1}$ を仮数部計算回路に入力する．

（2） 最初の $E_X - E_Y$ 回の反復の間は，仮数を 1 ビット右シフトし，部分積の値を 0 とする．

（3） $E_X - E_Y$ 回の反復後の仮数の値は Y となるので，以降は仮数をシフトさせず，部分積の値を $Yy_{j+4}2^{-3}$ とする．

M_Z のけた合せと Z^2 の部分積の計算も同様にして行うことができる．このようにして，バレルシフタを用いずに，けた合せを仮数部計算回路の中で行うことができる．

5. 回路の構成

4. で示した手法を用いた浮動小数点 3D ユークリッドノルム計算回路の構成を示す．ただし，例外処理などは扱わないものとする．回路全体は，前処理回路，仮数部計算回路，正規化回路で構成する．Y，Z の代わりに，シフト前の仮数 $M_Y \cdot 2^{-1}$，$M_Z \cdot 2^{-1}$ とシフト量 $E_X - E_Y$，$E_X - E_Z$ を仮数部計算回路に入力する．仮数部計算回路については，基数 2 による実現，基数 4 による実現，基数 2 の 2 段重ねによる実現の 3 種類の構成を示す．

5.1 前処理回路

前処理回路の構成を図 1 に示す．前処理回路では，F_1，F_2，F_3 の指数部を比較し，F_X，F_Y，F_Z を決定する．けた合せは行わない．

回路は主に以下のモジュールで構成される．

- **comp-E12，comp-E13，comp-E23：**二つの指数を入力し，どちらが大きい指数であるかと，指数の差の絶対値を出力する回路．
- **max-dec：**comp-E12, comp-E13, comp-E23 からの出力により，三つの指数のうちどれが最大であるかを決定する回路．
- **EX-selector，EYZ-selector，M-selector：** E_X，$E_X - E_Y$，$E_X - E_Z$，$M_X \cdot 2^{-1}$，$M_Y \cdot 2^{-1}$，$M_Z \cdot 2^{-1}$ を決定するためのセレクタ．

5.2 仮数部計算回路

5.2.1 基数 2 及び基数 4 による実現

基数 2 のアルゴリズムに基づく仮数部計算回路の構成を図 2 に示す．太線で示されたモジュールは，**4.** で示した手法を用いた箇所である．基数 4 の仮数部計算回路の構成は，モジュール CD selector への入力として REG-PP 及び REG-PM からの出力の上位ビットがあり，REG-WS への入力として初期値設定のための X があることのみが，基数 2 の場合と異なる．基数 2 の回路，基数 4 の回路のどちらも，仮数部の計算は $n+4$ サイクルで終了する．P を小数点以下 N ビットまで求める場合，基数 2 の回路では $n = N$，基数 4 の回路では $n = \lceil \frac{N}{2} \rceil$ となる．

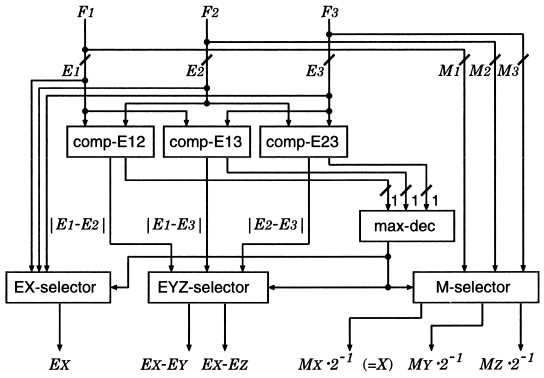

図 1 前処理回路の構成
Fig. 1 Preprocessing circuit.

回路は主に以下のモジュールで構成される．

- **REG-PP，REG-PM：**$P[j]^+$，$P[j]^-$ を保持するレジスタ．
- **FF-PJ：**拡張 on-the-fly 変換や加算器入力の生成のために必要な $p[j]_j^+$ の値を保持するフリップフロップ．
- **REG-WS，REG-WC：**けた上げ保存形で表現された残余 $W[j]$ を保持するレジスタ．REG-WS は中間和を，REG-WC は中間けた上げを保持する．
- **SREG-J：**反復回数 j を保持するシフトレジスタ．
- **SREG-X：**拡張 on-the-fly 変換や加算器入力の生成のために必要な $p[0]_{j+1}$ 及び $p[0]_{j+2}$ を参照するためのシフトレジスタ．
- **COUNT-Y，COUNT-Z：**基数 2 の回路では $E_X - E_Y$，$E_X - E_Z$ を，基数 4 の回路では $\lfloor \frac{E_X - E_Y}{2} \rfloor$，$\lfloor \frac{E_X - E_Z}{2} \rfloor$ を初期値とするダウンカウンタ．1 サイクルごとに，保持している値から 1 を減ずる．出力は，保持している値が 1 以上ならば 0 とし，保持している値が 0 になったら，以降は常に 1 とする．

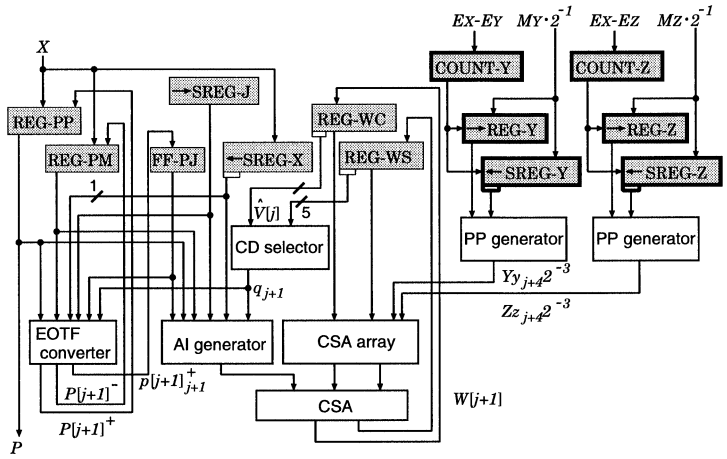

図 2　基数 2 の仮数部計算回路の構成
Fig. 2　Mantissa processing circuit with radix-2.

• **REG-Y, REG-Z:** COUNT-Y（COUNT-Z）からの出力が 0 ならば，1 サイクルごとに，基数 2 の回路では 1 ビット，基数 4 の回路では 2 ビット右シフトさせる．COUNT-Y（COUNT-Z）からの出力が 1 ならば，保持している値は Y（Z）となるので，シフトを行わず，値を保持する．

• **SREG-Y, SREG-Z:** COUNT-Y（COUNT-Z）からの出力が 0 ならば，値を保持する．ただし，出力は 0 とする．COUNT-Y（COUNT-Z）からの出力が 1 ならば，1 サイクルごとに，基数 2 の回路では 1 ビット，基数 4 の回路では 2 ビット左シフトさせ，y_{j+4}（z_{j+4}）を出力する．

• **CD selector：** 基数 2 の回路では $2W[j]$ の上位 5 ビット，基数 4 の回路では $4W[j]$ の上位 12 ビットと $P[j]$ の上位 7 ビットから補正値 q_{j+1} を選択する回路．

• **EOTF converter：** 拡張 on-the-fly 変換により $P[j+1]^+$ と $P[j+1]^-$ を求める回路．

• **PP generator：** Y^2，Z^2 の部分積 $Yy_{j+4}r^{-3}$，$Zz_{j+4}r^{-3}$ を計算する回路．

• **CSA array：** $rW[j]+Yy_{j+4}r^{-3}+Zz_{j+4}r^{-3}$ を計算し，けた上げ保存形で出力する回路．

• **AI generator：** 加算器入力 $-2q_{j+1}(P[j]+\frac{1}{2}q_{j+1}r^{-j-1}$ を計算する回路．

• **CSA：** $W[j+1]$ を計算するけた上げ保存加算器．

5.2.2　基数 2 の 2 段重ねによる実現

計算に必要なサイクル数を少なくするには，高基数で回路を実現するほかに，小さい基数の反復を 1 サイクルに数回計算する方法がある．そこで，1 サイクルで [基数 2 のアルゴリズム] の Step 2 の for ループの反復の 2 回分の計算を行う順序回路としての実現を考える．基数 2 の 2 段重ねによる実現の回路構成を図 3 に示す．仮数部の計算は $\lceil \frac{n}{2} \rceil + 3$ サイクルで終了する．P を小数点以下 N ビットまで求める場合，$n = N$ となる．

基数 2 による実現の組合せ回路部を単純に 2 段にしたのでは，サイクル時間，ハードウェア量とも 2 倍になってしまう．そこで，補正値 q_{j+2} の選択を補正値 q_{j+1} の選択と重畳させて行うことで高速化する．

5.3　正規化回路

正規化回路では，仮数を 1 以上 2 未満に正規化する．仮数部計算回路での計算結果 P は，$\frac{1}{2} \leqq P < \sqrt{3}$ である．よって，P の整数部が 0 のときは，P を 1 ビット左シフトしたものを仮数，E_X を指数として出力する．P の整数部が 1 のときは，P を仮数，$E_X + 1$ を指数として出力する．また，符号ビットの出力は常に 0 とする．

6.　回路の評価

IEEE754 標準の浮動小数点基本フォーマットを入出力とする 3D ユークリッドノルム計算回路を設計した．単精度と倍精度について設計を行った．回路は Verilog-HDL で記述し，Synopsys 社の Design Compiler を用いて論理合成を行い，ネットリストを作成した．セルライブラリにはローム社 0.35 μm プロセス用

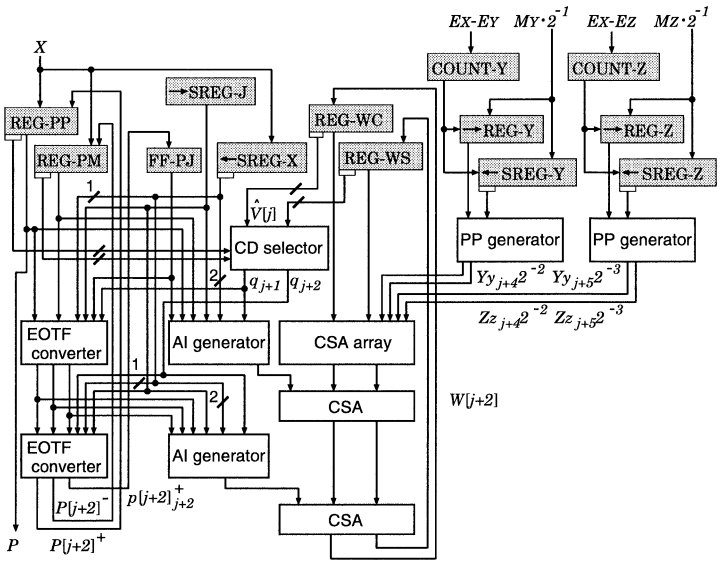

図 3　基数 2 の 2 段重ねの仮数部計算回路の構成
Fig. 3　Mantissa processing circuit with two radix-2 (overlapped).

表 1　回路の面積及び計算時間
Table 1　Area and computation time of the circuits.

	統合なし	統合あり					
	単精度	単精度			倍精度		
基数	2	2	4	2 の 2 段	2	4	2 の 2 段
面積 [mm^2]	0.4350	0.3503	0.4494	0.4517	0.7304	0.9238	0.9054
ゲート数（2 入力 NAND 換算）	5859	4718	6053	6084	9837	12442	12194
遅延 [ns]	5.55	5.32	8.32	7.32	5.43	9.32	7.92
サイクル数	30	30	18	17	59	33	32
計算時間（遅延 × サイクル数）[ns]	166.50	159.60	149.76	124.44	320.37	307.56	253.44

の東京大学大規模集積システム設計教育研究センター（VDEC）版 EXD 社ライブラリを用いた．ネットリストの作成では，適当な遅延を制約条件として与え，その条件下で面積が最小となるように論理合成を行った．シミュレーションには Cadence 社の Verilog-XL を用いた．

6.1　実現方法の比較

計算時間や回路全体の面積を，仮数部計算回路の実現方法の違いにより比較する．また，**4.** で述べた手法を用いなかった回路（単精度，基数 2）も設計し，けた合せと仮数部計算の統合による効果を調べる．設計した回路の面積及び計算時間を表 1 に示す．

単精度と倍精度のどちらも，基数 4 の回路及び基数 2 の 2 段重ねの回路は基数 2 の回路に比べ，計算時間が約 5〜20%短く，約 25%の面積の増加で実現できることがわかった．基数 4 の回路と基数 2 の 2 段重ねの回路を比較すると，面積はほぼ同じであり，計算時間は基数 2 の 2 段重ねの回路が約 15%高速となった．このことから，サイクル数を小さくしスループットを上げるには基数 2 を複数段重ねる実現法が有効であるといえる．また，けた合せ処理を仮数部計算に埋め込むことにより，面積を約 20%削減することができた．

6.2　開平器との比較

3D ユークリッドノルム計算回路は，3 回の平方算，2 回の加算，1 回の開平算を 1 度に行う回路である．漸化式 (5) からわかるように，開平算に 2 乗和計算を取り込んだ回路となっている．そこで，開平器に対してどのくらいの面積，計算時間で実現できるかを調べる．

基数 2 の開平計算アルゴリズム [4] に基づく浮動小数点開平器（単精度，基数 2）を設計した．ユークリッドノルム計算回路との比較結果を表 2 に示す．ユークリッドノルム計算回路は開平器に比べ，約 1.5 倍の遅延，約 2 倍の面積，約 2 倍の計算時間で実現できることが判明した．

表 2　開平器との比較
Table 2　Comparison with the square-root circuit.

	ユークリッドノルム計算回路	開平器
面積 [mm^2]	0.3503	0.1832
ゲート数（2 入力 NAND 換算）	4718	2467
遅延 [ns]	5.32	3.57
サイクル数	30	25
計算時間（遅延 × サイクル数）[ns]	159.60	89.25

7. むすび

以前提案した基数 2 及び基数 4 のユークリッドノルム計算アルゴリズムの改良を行い，それらに基づく浮動小数点 3D ユークリッドノルム計算回路の設計及び評価を行った．けた合せを仮数部計算と統合することにより，バレルシフタを用いずにけた合せを行う手法を提案した．

回路の評価の結果，けた合せ処理を仮数部計算に埋め込むことにより，面積を約 20%削減することができた．また，平方算 3 回，加算 2 回，開平算 1 回に相当する 3D ユークリッドノルム計算を行う回路が，開平器に比べ，約 1.5 倍の遅延，約 2 倍の面積，約 2 倍の計算時間で実現できることが明らかになった．この回路を導入することにより，3D ユークリッドノルム計算から加算器や乗算器を解放することができるため，プロセッサの性能を大きく向上させることができるものと考えられる．

謝辞　回路の設計には，VDEC を通し提供されている，Cadence 社及び Synopsys 社の CAD ツール，及び，ローム（株）の協力によるセルライブラリを用いた．本研究は一部（株）半導体理工学研究センター（STARC）との共同研究による．

文　献

[1] J.S. Walther, “A unified algorithm for elementary functions,” Proc. Spring Joint Computer Conf., pp.379–385, 1971.
[2] N. Takagi and S. Kuwahara, “Digit-recurrence algorithm for computing Euclidean norm of a 3-D vector,” Proc. 14th IEEE Symposium on Computer Arithmetic, pp.86–93, April 1999.
[3] N. Takagi and S. Kuwahara, “A VLSI algorithm for computing the Euclidean norm of a 3D vector,” IEEE Trans. Comput., vol.49, no.10, pp.1074–1082, Oct. 2000.
[4] M.D. Ercegovac and T. Lang, Division and square root: Digit-recurrence algorithms and implementations, Kluwer Academic, 1994.
[5] M.D. Ercegovac and T. Lang, “On-the-fly rounding,” IEEE Trans. Comput., vol.41, no.12, pp.1497–1503, Dec. 1992.

（平成 14 年 6 月 5 日受付，11 月 20 日再受付，
12 月 20 日最終原稿受付）

熊澤　文雄　（学生員）

平 13 名大・工・電気電子情報工卒．現在同大大学院修士課程在学中．算術演算回路，ハードウェアアルゴリズムに関する研究に従事．

高木　直史　（正員）

昭 56 京大・工・情報工卒．昭 58 同大大学院修士課程了．昭 63 同大工博．昭 59 同大・工・情報工助手．平 3 同助教授．平 6 名大・工・情報工助教授．平 10 同教授．算術演算回路，ハードウェアアルゴリズム，論理設計などの研究に従事．平 7 日本 IBM 科学賞，情報処理学会坂井記念特別賞など受賞．

武内　大輔

平 11 名大・工・電気電子情報工卒．平 13 同大大学院修士課程了．現在，松下電器産業（株）勤務．在学中，算術演算回路，ハードウェアアルゴリズムに関する研究に従事．

高木　一義　（正員）

平 3 京大・工・情報工卒．平 5 同大大学院修士課程了．平 7 奈良先端大助手．平 11 名大・工・情報工助手．平 12 同講師．工博．現在，計算理論，VLSI 設計，VLSI CAD アルゴリズムの研究に従事．