関数の高速計算法の改良と新提案

太田滋生[†] 後藤英一[†,††]
黄栄輝[†,†††] 吉田宣章[†]

黄，後藤，吉田は浮動小数点数を引き数とする基本初等関数の数値計算を表を用いて高速化する方法として単精度計算用の ATA 法と倍精度計算用の ATA-M 法を提案した．本論文では ATA 法の表の大きさを 3/16 強に縮める改良を示す．また倍精度計算用に，ATA-M 法より索表が簡単なことと，索表後の計算を倍精度演算器を分割使用して並列化することの2点を特徴とする分割並列乗算法を提案する．これらの新計算法は市販の高速浮動小数点演算ユニット程度の規模の集積回路に実装できる．また，同一回路を用いて倍精度除算を高速化すること，倍精度疑似乱数を高速に発生すること，および，単精度複素数の乗算を高速化すること等を提案する．

Improvement and New Proposal on Fast Evaluation of Elementary Functions

Shigemi Ohta,[†] Eiichi Goto,[†,††] Weng Fai Wong[†,†††]
and Nobuaki Yoshida[†]

Wong, Goto, and Yoshida introduced fast methods for numerical evaluation of elementary functions based on table lookup. They are called ATA and ATA-M methods respectively for single- and double-precision calculations. In this paper an improvement of ATA methods that shrinks the size of the table by a factor of about 3/16 is presented. Another method called Split Parallel Multiplication method, which is characterized by simpler table lookup than ATA-M and by split and parallel use of double-precision floating point circuitry, is also introduced. These new methods fit onto integrated circuits of the size comparable with commercially available floating-point accelerators. Methods for accelerating double-precision division, generation of uniform pseudo-random numbers in double-precision, and accelerating multiplication of single-precision complex numbers using the same circuitry are proposed.

1. 序

黄，後藤，吉田[1]は，浮動小数点数を引き数とする基本初等関数の値の計算を表を用いて高速化する手法として 32 ビット（単精度）浮動小数点数を引き数とする場合のために ATA (Add-Table-lookup-Add) 法を，また 64 ビット（倍精度）浮動小数点数を引き数とする場合のために ATA-M (Add-Table-lookup-Add-Multiply) 法を提案した．これらの方法を用いると基本初等関数の値を求めるのに浮動小数点乗算数回分の時間で済む．ただし，ここにいう基本初等関数とは，逆数，平方根，平方根の逆数，指数関数，対数関数，三角関数，および逆三角関数である．本論文では ATA 法で用いる表の大きさを縮める改良と，ATA-M 法にかわる新方法として，必要とする回路資源が少ない分割並列乗算 (Split Parallel Multiplication, SPM) 法を提案する．これらの改良および新方法によって，半導体集積回路への実装が非常に簡単になる．また，さらに幾つかの計算機能を付け加えることも提案する[2]．

論文の構成は次のとおりである．2章では，ATA 法に必要な表の形式を変更しその大きさを 3/16 程度に縮める改良法を示す．これにより，すべての基本初等関数についての表を併せてもその大きさは高々数百

† 理化学研究所後藤特別研究室
Goto Laboratory, The Institute of Physical and Chemical Research (RIKEN)
†† 神奈川大学理学部情報科学科
Department of Information Science, Faculty of Science, Faculty of Science, Kanagawa University
††† シンガポール国立大学 DISCS
Department of Information Systems and Computer Science, Faculty of Science, National University of Singapore

キロバイトとなり，実装が著しく簡単になる．3章では，倍精度浮動小数点数を引き数とする場合のために，ATA-M 法より索表が簡単なことと，索表後の計算を倍精度演算器を分割使用して並列に行うことの二点を特徴とする新方法である分割並列乗算法を提案する．この方法は，市販されている高速浮動小数点計算用チップと同程度の規模の集積回路に実装できる．さらに，このような集積回路を用いて倍精度除算を高速化すること，倍精度疑似乱数を高速に発生すること，および単精度複素数の乗算を高速すること，を各々4，5，6章で提案する．これらの方法によって初等関数の計算が浮動小数点乗算数回分の時間でできるようになると，従来慣用の諸アルゴリズムの再検討が必要になる．7章はこれについての考察を行う．最後に8章において，初等関数の高速計算法を実現する集積回路の機能仕様を提案する．なお，簡単のために，本論文で取り扱う浮動小数点数はすべてIEEE規格に従うものとする．

2. 改良 ATA 法

ATA 法は単精度浮動小数点数を引き数とする基本初等関数の値を表を用いて高速に計算するアルゴリズムである[1]．その実装を簡単にするため，この方法が必要とする表を小さくする改良について述べる．浮動小数点数の符号および指数部の取り扱いは論文[1]と同じなので説明を省略し，仮数部の取り扱いについてだけ説明する．今，IEEE 規格の 32 ビット浮動小数点数の正規化された仮数部 $1+{}_{22}x$ に長さ7ビットのガードビットを加えて得られる 31 ビットの数を次のように展開することを考える．

$$1+{}_{23}x=1+{}_{n_0}X_0+2^{-n_0}{}_{n_1}x_1 +2^{-n_0-n_1}{}_{n_2}x_2+2^{-n_0-n_1-n_2}{}_{n_3}x_3. \quad (1)$$

ただし，${}_{23}x$は区間 $[0,1[$ に値をとる 23 ビットの2進小数，n_i $(i=0,1,2,3)$ は $\sum_i n_i=30$ なる整数，${}_{n_0}X_0$ および ${}_{n_i}x_i$ $(i=1,2,3)$ は区間 $[0,1[$ に値をとる各々 n_i ビットの2進小数とする．本論文では以下このように2進数を表す記号の左下付き添え字によってその2進数のビット幅を表すことにする．

論文[1]の ATA 法は記法がやや異なるが，$n_0=12$, $n_1=n_2=n_3=6$ に対応する．この場合，${}_{n_3}x_3={}_6x_3=0$ であるから，区間 $[1+{}_{n_0}X_0, 1+{}_{n_0}X_0+2^{-12}[$ における任意の基本初等関数 F の値が次の ATA 公式で与えられる．

$$\begin{aligned}P&=F(1+{}_{n_0}X_0)\\&\quad+2^{-n_1-1}\{F(1+{}_{n_0}X_0+2^{-n_0+n_1}{}_{n_1}x_1)\\&\quad-F(1+{}_{n_0}X_0-2^{-n_0+n_1}{}_{n_1}x_1)\}\\&\quad+2^{-n_1-n_2-1}\{F(1+{}_{n_0}X_0+2^{-n_0+n_2}{}_{n_2}x_2)\\&\quad-F(1+{}_{n_0}X_0-2^{-n_0+n_2}{}_{n_2}x_2)\}\\&\quad+T_{0,1}({}_{n_0}{}^6X_0, {}_{n_1}{}^6x_1) \qquad (2)\end{aligned}$$

$$\begin{aligned}&T_{0,1}({}_{n_0}{}^6X_0, {}_{n_1}{}^6x_1)\\&=\frac{1}{2}F^{(2)}(1+{}_{n_0}{}^6X_0)2^{-2n_0}({}_{n_1}{}^6x_1)^2\\&\quad-\frac{1}{6}F^{(3)}(1+{}_{n_0}{}^6X_0)2^{-3n_0+2n_1}({}_{n_1}{}^6x_1)^3. \qquad (3)\end{aligned}$$

精度は 30 ビットである．この多項式の初項は 12 ビットのアドレスで指定できる．第2項と第3項は高々 13 ビットのアドレス $({}_{n_0}X_0+2^{-n_0+n_1}{}_{n_1}x_1)$ で指定できる．第4項と第5項も高々 13 ビットのアドレス $({}_{n_0}X_0+2^{-n_0+n_2}{}_{n_2}x_2)$ で指定できる．終項は ${}_{n_0}X_0$ の上位6ビット（これを左上付き添え字を使って ${}_{n_0}{}^6X_0$ と表す．以下他の変数についてもおなじく左上付き添え字を使って上位ビットを採ることを示すこととする．）と ${}_{n_1}x_1$ の上位6ビット ${}_{n_1}{}^6x_1$ の計 12 ビットのアドレスで指定できる．そこで，これらの表をあらかじめ準備しておけば，基本初等関数 F の計算はアドレス計算のための加算（Add），索表（Table-lookup），および加算（Add）だけですむ．これは高々浮動小数点演算2回分の時間で実行される．しかし，このような表をすべての基本初等関数について準備すると，全体で数メガバイトとなり実装に不便である．そこで，以下に表を小さくする改良法を提案する．

まず，量的に最も大きい 13 ビットのアドレス $({}_{n_0}X_0+2^{-n_0+n_i}{}_{n_i}x_i)$ の諸表を 1/2 強に縮められることがわかる．なぜなら，この 13 ビットアドレスの最上位ビットが立つのは，12 ビット小数同士の和が桁あふれを起こす例外時だけなので，その場合を扱う小さな表を独立させればよいからである．

次にビット分割を変更して，$n_0=10, n_1=4, n_2=n_3=5$ とする．ただし ${}_{n_3}x_3$ のうしろにはガードビット6ビットを付け加える．さらに，${}_{n_1}x_1$ と ${}_{n_2}x_2$ に符号を導入し，その絶対値を 1/2 以下とする次のような工夫を加える．

$$\begin{aligned}&1+{}_{23}x=1+{}_{n_0}X_0'+2^{-n_0}{}_{n_1}x_1' +2^{-n_0-n_1}{}_{n_2}x_2'+2^{-n_0-n_1-n_2}{}_{n_3}x_3,\\&{}_{n_0}X_0'={}_{n_0}X_0+2^{-n_0-1},\\&{}_{n_1}x_1'={}_{n_1}x_1-2^{-1}+2^{-n_1-1},\\&{}_{n_2}x_2'={}_{n_2}x_2-2^{-1}. \qquad (4)\end{aligned}$$

${}_{n_1}x_1'$ は符号1ビット，有効3ビットの小数，${}_{n_2}x_2'$ は

符号1ビット，有効4ビットの小数となる．これに応じて ATA 公式を次のように変更する．

$$
\begin{aligned}
P=&F(1+{}_{n_0}X_0')(1-2^{-4n_1})\\
&+2^{-4n_1-1}\{F(1+{}_{n_0}X_0'+2^{-n_0+n_1}{}_{n_1}x_1')\\
&+F(1+{}_{n_0}X_0'-2^{-n_0+n_1}{}_{n_1}x_1')\}\\
&+2^{-n_1-1}\{F(1+{}_{n_0}X_0'+2^{-n_0+n_1}{}_{n_1}x_1')\\
&-F(1+{}_{n_0}X_0'-2^{-n_0+n_1}{}_{n_1}x_1')\}\\
&+2^{-n_1-n_2-1}\{F(1+{}_{n_0}X_0'+2^{-n_0+n_2}{}_{n_2}x_2')\\
&-F(1+{}_{n_0}X_0'-2^{-n_0+n_2}{}_{n_2}x_2')\}\\
&+2^{-n_1-n_2-n_3-1}\{F(1+{}_{n_0}X_0'+2^{-n_0+n_3}{}_{n_3}x_3)\\
&-F(1+{}_{n_0}X_0'-2^{-n_0+n_3}{}_{n_3}x_3)\}\\
&+T_{0,1,2}({}_{n_0}{}^4X_0',\ {}_{n_1}{}^4x_1',\ {}_{n_2}{}^4x_2'), \qquad (5)
\end{aligned}
$$

$$
\begin{aligned}
&T_{0,1,2}({}_{n_0}{}^4X_0',\ {}_{n_1}{}^4x_1',\ {}_{n_2}{}^4x_2')\\
&=\frac{1}{2}F^{(2)}(1+{}_{n_0}{}^4X_0')2^{-2n_0-2n_1-n_2+1}({}_{n_1}{}^4x_1')({}_{n_2}{}^4x_2')\\
&\quad-\frac{1}{6}F^{(3)}(1+{}_{n_0}X_0')\{2^{-3n_0+2n_1}({}_{n_1}{}^4x_1')^3\\
&\quad+2^{-3n_0-n_1+2n_2}({}_{n_2}{}^4x_2')^3\}. \qquad (6)
\end{aligned}
$$

精度はやはり30ビットである．これを改良 ATA 公式と呼ぶ．この多項式の初項は 10 ビットのアドレス ${}_{n_0}X_0'$ の表で索ける．ただし表の値は新しい補正のファクター $(1-2^{-4n_1})$ を含む．第2項から第5項までは各々 10 ビットのアドレス $({}_{n_0}X_0'+2^{-n_0+n_1}{}_{n_1}x_1')$ の表と桁あふれ例外の小さな表で索ける．4項あるが，索表は2回で済む．第6，7項と第8，9項は各々 10 ビットのアドレス $({}_{n_0}X_0'+2^{-n_0+n_2}{}_{n_2}x_2')$ と $({}_{n_0}X_0'+2^{-n_0+n_3}{}_{n_3}x_3)$ の表と桁あふれ例外用の小さな表で索ける．また終項は ${}_{n_0}X_0'$ の上位4ビット ${}_{n_0}{}^4X_0'$ と ${}_{n_1}x_1'$ の上位4ビット ${}_{n_1}{}^4x_1'$ と ${}_{n_2}x_2'$ の上位4ビット ${}_{n_2}{}^4x_2'$ からなる計 12 ビットのアドレスで必要な有効数字5桁を指定できる．

以上の改良をすべて併せると，表の大きさは元の 3/16 強になることがわかる．また，アドレス計算もわずかだが速くなる．ただし，一つの引き数から作られる索表のためのアドレスは不規則に分布するので，表が普通のメモリにおかれキャッシュレジスタを介して読まれる場合はほとんど必ずミスヒットすることになると考えられる．したがって表は高速読み出しメモリに置くことが望ましい．

3. 分割並列乗算法

倍精度浮動小数点数を引き数とする基本初等関数の値を高速で計算する新しいアルゴリズムを提案する．

このアルゴリズムは，論文[1]で提案された ATA-M 法と比べると，浮動小数点乗算1回分ほど遅いが，索表とその後の計算に要する回路資源を大幅に節約し，市販の高速浮動小数点計算チップと同程度の規模の集積回路で実現できる．IEEE 規格の 64 ビット浮動小数点数は，1ビットの符号部，11 ビットの指数部，および 52 ビットの正規化された仮数部を持つ．符号部と指数部の取り扱いはほとんどの場合は通常と異ならないので説明を省略し，必要な場合についてのみ後述する．

まず正規化された仮数部を $1+{}_{10}X+2^{-10}{}_{42}x$ のように上位 10 ビットと下位 42 ビットに分割する．ここに，${}_{10}X$ は区間 $[0, 1-2^{-10}]$ に値を持つ 10 ビットの小数，${}_{42}x$ は区間 $[0, 1-2^{-42}]$ に値を持つ 42 ビットの小数である．今，$X'=1+{}_{10}X$ の近傍の区間 $[X', X'+2^{-10}[$ において基本初等関数 $F(X'+2^{-10}{}_{42}x)$ を最適近似する多項式

$$C(x)=C_0+C_1x+C_2x^2+C_3x^3+C_4x^4 \qquad (7)$$

が知られ，係数 C_i が ${}_{10}X$ をアドレスとする表として与えられたものとする．目的の関数値も IEEE 規格の 64 ビット浮動小数点数として表示するので，その仮数部を与えるこの多項式の計算に求められる精度は 53 ビットである．一方，すでに論文[1]で示したようにこの多項式近似の精度は 59 ビットなので十分である．以下では，4ビットのガードビットを付け加え 57 ビットの仮数部を計算するものとする．

この多項式の計算を市販の高速浮動小数点演算器程度の規模の集積回路を使って高速に行う方法を提案する．以下では，そのような回路で IEEE 規格の倍精度浮動小数点数の乗算を行うのに要する時間 τ_* を単位時間とする．さらに簡単のために IEEE 規格の倍精度浮動小数点数の加算を行うのに要する時間 τ_+ は τ_* と等しいものと仮定する．

すでに述べたように，うえの多項式 $C(x)$ の計算に求められる精度は高々 57 ビットである．したがって，この多項式中に現れる加算は 57 ビットの桁上げ保存固定小数点方式で計算すればよいので，これに要する時間も回路資源も浮動小数点乗算に比べれば無視できるほど小さい．そこで以下ではこの所要時間を無視する．一方，すでに加算が固定小数点方式で計算される以上は，x の高次項の計算も必要な桁数だけの乗算を行えばよい．演算器には1単位時間 τ_* に少なくとも1回の 53 ビット×53 ビット乗算を行うだけの回路資源が備わっているはずだから，これを分割使用すれ

ば，このような桁数の少ない乗算のいくつかは同時に並列して行える．係数 C_i の表は外付けのメモリに格納する．また以下に述べるようにこの表の読み出しには高々 64 ビットの幅しか必要としないので演算器の入力用データ線で十分である．したがって，多項式 $C(x)$ の計算は有効桁数の多い低次部の計算と桁数の少ない高次部の計算とを並列化した次のような順序で行うのが効率的である．これを分割並列乗算（Split Parallel Multiplication, SPM）法と名付ける．

I. x^2 を計算する．結果の上位高々 27 ビットしか必要としないので，初めから x の上位 27 ビット同士の乗算として行う．この乗算と同時に係数 C_2 および C_3 を読み込む．これらの読み込みも各々上位 37 ビットと 27 ビットのみ行えば良いので，読み幅は 64 ビットである．

II. $x^4=(x^2)^2$ を計算する．これは結果の上位高々 17 ビットしか必要としないので，初めから x^2 の上位 17 ビット同士の乗算として行う．また並行して C_3*x^2 の 27 ビット乗算と，C_2*x の 37 ビット乗算を行う．係数 C_1 の 47 ビット読み込みと C_4 の 17 ビット読み込みを行う．ここでも読み幅は 64 ビットである．三項加算 $C_3x^2+C_2x+C_1$ を桁上げ保存固定小数点で行う．

III. C_4*x^4 の 17 ビット乗算を行う．$(C_3x^2+C_2x+C_1)*x$ の 47 ビット乗算を行う．係数 C_0 の 57 ビット読み込みを行う．三項加算 $C_4x^4+(C_3x^2+C_2x+C_1)x+C_0$ を桁上げ保存固定小数点で行う．

以上の操作の流れを**図 1** に示す．また次のように略記する．

$$
\begin{aligned}
&[{}^{47}([{}^{27}({}^{27}(x^2)_I * {}^{27}\{C_3\}_I)_{II}\\
&\quad + {}^{37}({}^{37}x * {}^{37}\{C_2\}_I)_{II} + {}^{47}\{C_1\}_{II})]_{II} * {}^{47}x)_{III}\\
&\quad + {}^{57}\{C_0\}_{III}\\
&\quad + {}^{17}({}^{17}({}^{27}({}^{27}x^2)_I{}^2)_{II} * {}^{17}\{C_4\}_{II})_{III}]_{III}. \qquad (8)
\end{aligned}
$$

ここで，括弧の左上付の整数はその括弧に括られた乗算の結果のビット数を，また右下付のローマ数字はその乗算のタイミングを表す．中括弧は係数の読み込みを表し，乗算の場合と同様に左上付の整数は係数の有効桁数を，また右下付のローマ数字は読み込みのタイミングを表す．大括弧は固定小数点加算を括り，その右下付のローマ数字は加算のタイミングを表す．

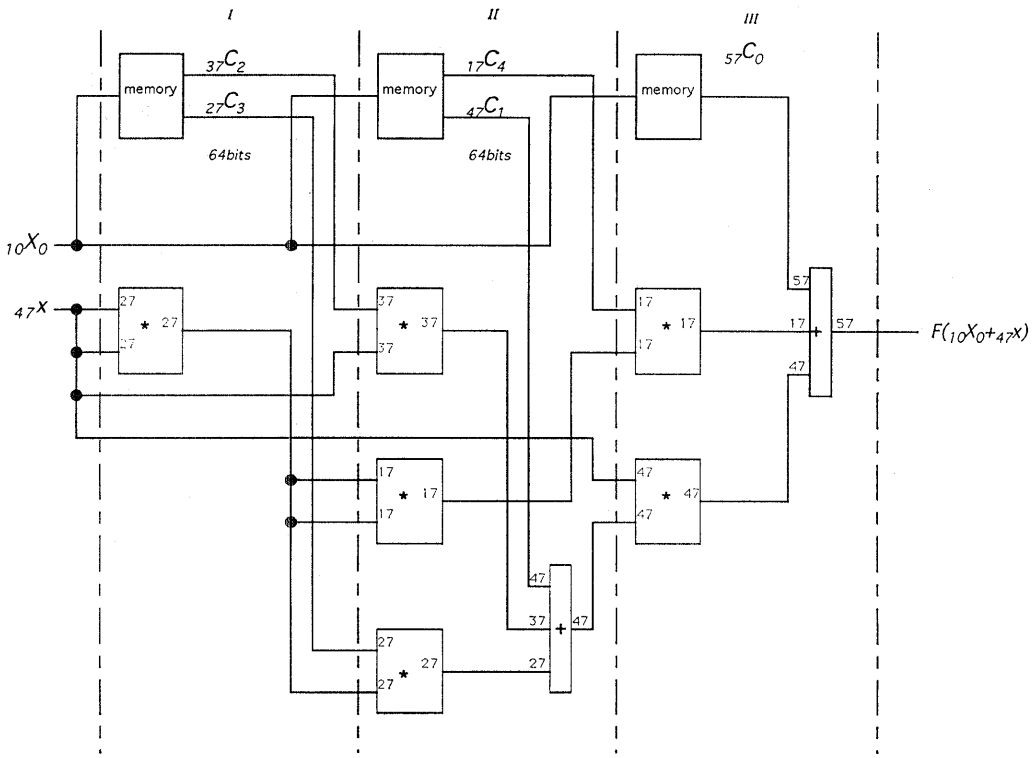


図 1 分割並列乗算 (Split Parallel Multiplication, SPM) 法による基本初等関数計算の手順を示す模式図
Fig. 1 Split Parallel Multiplication (SPM) scheme for fast numerical evaluation of elementary function.

n ビット×m ビットの乗算に必要なゲートの数は $n\times m$ に比例するものとすれば，うえの各段階で必要となる演算回路資源は，この単位で，各々 $27\times27=729$, $17\times17+27\times27+37\times37=2387$, および $17\times17+47\times47=2498$ に比例する．これらはいずれも，IEEE 規格 64 ビット浮動小数点数同士の乗算に必要と考えられる $53\times53=2809$ 単位より少ない．したがって，分割並列乗算法は市販されている 64 ビット浮動小数点高速演算ユニットと同規模の集積回路上に実装可能と考えられる．

すでに述べたように係数 C_i の表は外付けのメモリに格納するが，ATA 法の場合とは異なって，その 10 ビットのアドレス ${}_{10}X$ はすべて共通なので，普通のメモリからキャッシュを介して読みだすとしてもそれほど非能率とはならない．もちろん高速読みだしメモリに格納することが望ましい．

以上から，分割並列乗算法による基本初等関数の数値計算に要する時間は，仮数部に限れば 64 ビット浮動小数点数乗算 3 回分程度と見積もられる．論文[1] の補遺にあるエミッタ結合回路の方法に従ってより細かく検討すると，段階 I の所要時間は 64ビット浮動小数点乗算 1 回の所要時間の 9/16，段階 II の所要時間は 12/16，段階 III の所要時間は 13/16 である．これを合計すれば，全所要時間は 64 ビット浮動小数点乗算 1 回の所要時間の 34/16, すなわち 2 回強であることがわかる．仮に Horner 法を用いたとすれば 64 ビット浮動小数点数乗算・加算各 4 回を要するから，計算時間で 4 分の 1 に迫る大幅な節約である．これに，符号部と指数部の処理にかかる時間を加えれば基本初等関数の計算にかかる時間の見積もりが得られるが，それは以下に述べるように初等関数の種類ごとに異なる．

まず逆数の場合はこれらの取り扱いは自明で，その所要時間も無視できる．平方根，および平方根の逆数については，符号部の取り扱いは自明であり，また指数部はその偶奇に応じて引き数を適当に変換すればよく，やはりその所要時間を無視できる．引き数 x の $\pi/2$ 倍の正弦 $\mathrm{sinp}\,x=\sin(\pi x/2)$ と余弦 $\mathrm{cosp}\,x=\cos(\pi x/2)$ の場合も符号部と指数部の処理は自明で所要時間を無視してよい．したがって，これらの関数の計算はすべて 3 単位時間でできる．

つぎに対数関数 $\ln x$ の場合は，11 ビットの指数部に 57 ビットの定数 $\ln 2$ をかけて，仮数部の対数に加える必要がある．この定数 $\ln 2$ を外部から読み込まなくてよいように演算器内部にワイヤインしておけば，掛け算は上に述べた仮数部の SPM 計算の第 I 段階で並列実行できる．したがって全体の所要時間はやはり 3 単位時間である．

指数関数 $\exp x$ の計算では，符号部と指数部の処理にどうしても 1 単位時間を余計にとられる．普通の正弦 $\sin x$ または余弦 $\cos x$ の計算は，引き数 x の $1/\pi$ 倍をとってから sinp または cosp をよぶことになる．対数の場合と同様に定数 $1/\pi$ を演算器内部にワイヤインしておけば，掛け算のために 1 単位時間を余計にとられるだけですむ．したがってこれらの関数には 4 単位時間かかる．

逆正接関数 $\mathrm{atan}\,x$ は，引き数の絶対値が 1 より小さければ，符号部と指数部の処理は自明で余計な時間はとらない．しかし，引き数の絶対値が 1 より大きくなると，まずその逆数を計算するために 3 単位時間余計にかかる．

4. 倍精度除算の高速化

SPM 法と同様の手法で，二つの IEEE 規格 64 ビット浮動小数点数 X と Y の商 $Q=Y/X$ を高速に，かつ回路資源を節約して計算できる．符号および指数の取り扱いは通常と変わらないので説明を省略する．以下本章では X, Y および Q は各々の仮数部にガードビットを加えた 57 ビットの数を表す．次の手順により商 Q が求められる．必要とする時間は IEEE 規格 64 ビット浮動小数点乗算高々 3 回分である．

I. X の上位 29 ビット部分 ${}^{29}X$ の逆数を ATA 法により求める．これを ${}^{29}(I_{\mathrm{ATA}})_I$ と表す．

II. 28 ビットの小数 ${}^{28}(x)_{II}$ を $1-{}^{28}(x)_{II}={}^{57}({}^{29}(I_{\mathrm{ATA}})_I*X)_{II}$ により求める．この操作は，引算を実行する必要はなく，57 ビット乗算の結果の上位 29 ビットを捨ててしまえばできるので，倍精度乗算器の資源を節約できる．この操作を記号 $*^{\#}$ を使って，${}^{28}(x)_{II}={}^{28}({}^{29}(I_{\mathrm{ATA}})_I*^{\#}X)_{II}$ と表す．さらに，倍精度乗算器を分割並行使用して，57 ビットの小数 ${}^{57}({}^{29}(I_{\mathrm{ATA}})_I*Y)_{II}$ も同時に計算する．

III. 積 ${}^{57}(Q_{\mathrm{approx}})_{III}={}^{57}({}^{57}({}^{29}(I_{\mathrm{ATA}})_I*Y)_{II}*(1+{}^{28}(x)_{II}))_{III}$ は 57 ビットの範囲で求める商と一致する．

なぜなら，恒等式

$$\begin{aligned}Q&=YX^{-1}\\&=(IY)(IX)^{-1}\\&=(IY)(1-x)^{-1}\\&=(IY)(1+x)(1-x^2)^{-1}\end{aligned}\qquad(9)$$

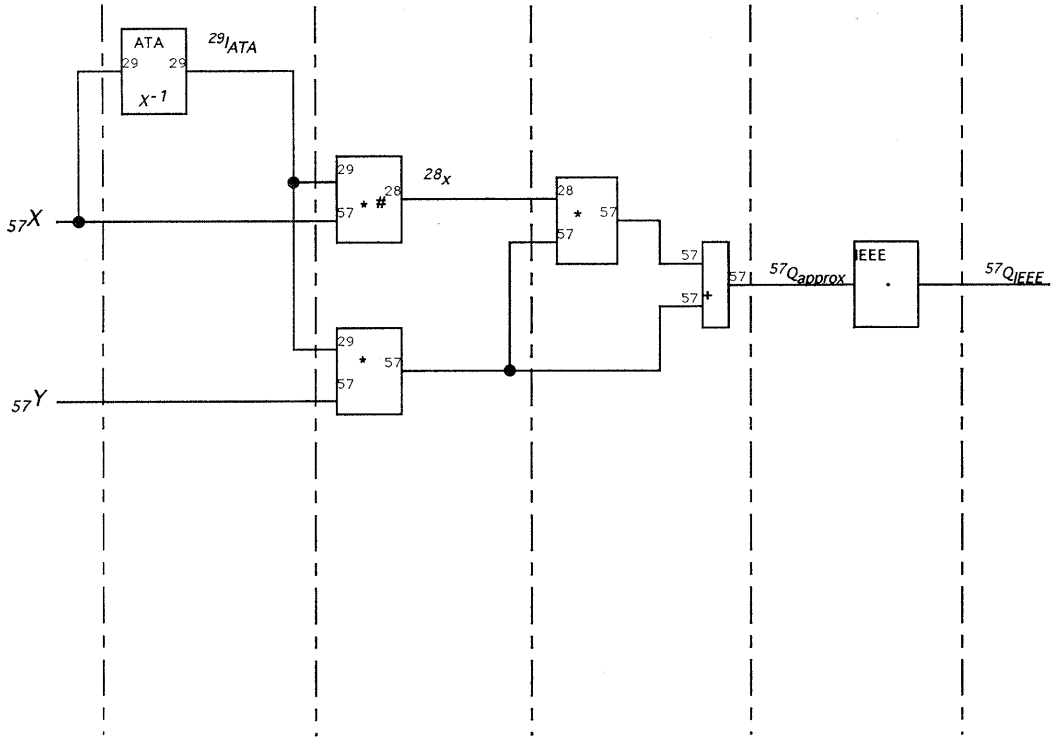


図 2 SPM 法に類似の手法で倍精度除算を高速化する手順を示す模式図
Fig. 2 Fast numerical calculation scheme for division.

において，その最右辺の $(1-x^2)^{-1}$ は 57 ビットの精度では 1 に等しいからである．この手順を**図 2** に示す．この場合も文献 1) の補遺に述べられた方法にしたがって計算すると各段階の所要時間は 64 ビット浮動小数点乗算 1 回の所要時間の 14/16, 9/16, および 11/16 の合計 34/16 であることがわかる．

IEEE 規格に従う剰余補正をするためには実はもう 1 単位程度の余計な時間が必要である．しかし，実用上は大半の計算でこの補正を必要としないと考えられるので，これを行うか否かはユーザが決めるオプションとし，コンパイル時に選択可能とすることを提案する．

また，この方法はニュートン法を 1 ステップだけ行うことに相当する．同様の方法を平方根の逆数の計算にも適用できるが，この場合は前章に述べた分割並列乗算法でもたかだか 3 単位時間でできるので，必要がない．

5. 倍精度乱数の高速発生

科学計算の重要な分野にモンテカルロ法による数値積分などがある．その高速化のためには，区間 [0, 1[に値を持つ一様疑似乱数の高速発生が必要である．上に述べた高速計算回路の資源を用いれば，IEEE 64 ビット浮動小数点規格に従うそのような一様疑似乱数の合同乗算法による発生が 1 単位時間につき 1 乱数のバーストモードで可能になる[3]．

一様疑似乱数の発生法には合同乗算法[4] の他にも，例えば Galois 体 $GF(2^n)$ の理論に基づく方法（別名シフトレジスター法）[5] など多くの方法が知られている．しかし最近，理論物理学におけるモンテカルロ法利用の最も典型的な例であるイジング模型の数値計算においてこのシフトレジスター法による疑似乱数発生が誤った結果に導く例があたかも大発見であるかのごとく報告された[6]．したがって，このような方法とくらべて信頼度がより高い合同乗算法を高速化することは重要である．

また，合同乗算法では，乗算の結果の下位ビットを正規化して，一様疑似乱数とするが，むしろ中位ビットを用いる方が，乱数としての性質が向上する可能性がある．これを中央合同乗算（mid-congruent prod-

uct) 法と名付け，オプショナルに使用可能とすることも提案する．ただし，次の疑似乱数の生成には普通の合同乗算法と同じく下位ビットを用いる．またこれをわずかに変更するだけでシフトレジスター法についても同様の効果をあげることができる．これらの疑似乱数生成法の詳細はいずれ報告する[7]．

6. 単精度複素数乗算

上と同じ回路資源を同様に分割して使用することにより，32 ビット浮動小数点数からなる実部と虚部を持つ複素数の乗算を高速に行うことを提案する．この乗算には高々 32 ビット×32 ビットの乗算4回と桁上げ保存 32 ビット固定小数点加算2回が現れるので，高々1単位時間で処理できるはずである．

7. 考　　察

従来慣用されてきた数値計算アルゴリズムの多くにおいては，初等関数の計算に非常に長い時間がかかるためこれをなるべく回避するような工夫がしばしばなされている．本論文で述べた初等関数の高速計算法はそのような慣用アルゴリズムの再検討を迫るものである．詳しい検討は多くの実例を集めたあとに譲らねばならないが，今幾つかの簡単な事例について考察する．

まず，2次方程式 $ax^2+bx+c=0$ の解の公式 $x=(-b\pm\sqrt{b^2-4ac})/(2a)$ に現れる，-1, 2, あるいは4による乗算にかかる時間は，平方根や逆数の計算にかかる時間に比べて無視できるものとされてきた．しかし，本論文の高速計算法を用いる場合にはもはや無視できず，これらの乗算のいっそうの高速化が望ましい．実際，これらの単純な乗算は，演算器の命令セットに次のような簡単な変更を加えるだけで，ただちに実現できるものと考えられる．すなわち，load, store, add, subtract, multiply, および divide の各命令に，引き数を -1 倍，$2^{\pm1}$ 倍，$2^{\pm2}$ 倍，および $2^{\pm3}$ 倍するオプションを加えればよい．

また，3次方程式の解を与えるカルダノ法では，立方根と逆正接の計算が律速段階となっている．本論文の計算法によって，指数，対数の両関数を用いて立方根を求めるとすれば8単位時間が必要である．立方根関数の計算表を付け加えれば4単位時間でできる．さらに3実根の場合に現れる合成関数 $\sin(\mathrm{atan}(\sqrt{x})/3)$ および $\cos(\mathrm{atan}(\sqrt{x})/3)$ も表に加えることが望ましい．これらの表によって，カルダノ法が高速化すると，たとえば，3×3 複素(反)エルミート行列の対角化やその指数関数の計算も高速化し，それによって格子 QCD のような大規模数値計算（例えば文献8）など）にも資するところ大である．

8. 結

本論文に述べた浮動小数点数を引数とする初等関数の高速計算法は，市販の高速浮動小数点計算チップ程度の規模の集積回路に納まる．より具体的には，例えば WEITEK 社の WTL 3364 型チップなどとの間に上位互換性を保ちながら，以下の機能を実装することが可能であると考える．

0. IEEE 64 ビット浮動小数点規格に従う加減算，乗算．以下ではこれに要する時間を1単位とする．および IEEE 32 ビット浮動小数点規格に従う加減算，乗算．改良 ATA 法による単精度初等関数の計算．
1. 区間 $[0,1[$ に値を持ち IEEE 64 ビット浮動小数点規格の一様疑似乱数を，1単位時間当たり1個発生する．
2. 4章に述べた高速除算，三角関数のうち $\mathrm{sinp}\,x=\sin(\pi x/2)$, $\mathrm{cosp}\,x=\cos(\pi x/2)$, 平方根とその逆数，対数関数，引き数の絶対値が1より小さい場合の逆正接関数 $\mathrm{atan}\,x$, $\mathrm{patan}\,x=(\mathrm{atan}\,x)/\pi$ 等は，3単位時間で計算できる．
3. IEEE 64 ビット浮動小数点規格に従う除算および逆数計算，指数関数，三角関数のうち $\sin x$, $\cos x$ 等は4単位時間で計算できる．
4. 引き数の絶対値が1より大きい場合の逆正接関数 $\mathrm{atan}\,x$, $\mathrm{patan}\,x=(\mathrm{atan}\,x)/\pi$, および二つの引き数 x と y の比の逆正接をとる関数 $\mathrm{atan2}\,(x,y)$ は6単位時間で計算できる．
5. IEEE 32 ビット浮動小数点数を成分とする複素数の乗算を1単位時間で行う．

以上の時間計算には，$\ln 2$, π, $1/\pi$, $\sqrt{3}/2$ 等の定数が乗算器内にワイヤインされていることを仮定する．データ入出力ライン数は通常と同じである．

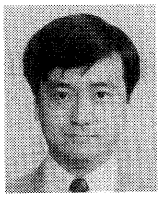
改良 ATA 法および分割並列乗算法に必要な表は外付けのメモリーに格納する．このメモリーは読み出しが速いことを要するが，書き込みはそれほど速くなくともよい．初等関数のほかに，7章で述べたカルダノ法に現れる立方根や合成関数を計算するための表を付け加えることも有用である．アドレス計算やバッファリングの効率を考慮して，近似多項式の係数の格納順序を最適化しておく必要がある．またコンパイラに

よる解析で書き換えのないことが判明した定数の格納にも使えるだろう．

参考文献

1) 黄 栄輝，後藤英一，吉田宣章：初等関数の高速計算法，情報処理学会論文誌，Vol. 34, No. 7, pp. 1570-1579 (1993)．
2) 特許願 平 5-242007号．
3) Oyanagi, Y., Goto, E. and Yoshida, N.: Supercomputing Pseudo Random Number—Proposals Hardware and Software, *Proceedings of the International Symposium on Supercomputing*, Fukuoka, Japan, November 6-8 (1991)．
4) Lehmer, D. H.: *Proc. 2nd Symp. on Large-Scale Digital Computing Machinery*, p. 141, Cambridge, 1948, Harvard University Press (1951)．
5) Tausworthe, R. C.: *Math. Comp.*, Vol. 19, No. 90, pp. 201-209 (1965); Fushimi, M. and Tezuka, S.: The K-Distribution of Generalized Feed-Back Shift-Register Pseudorandom Numbers, *Comm. ACM*, Vol. 26, No. 7, pp. 516-523 (1983)．
6) Ferrenberg, A. M., Landau, D. P. and Wong, Y. J.: Monte Carlo Simulations: Hidden Errors from 'Good' Random Number Generators, *Phys. Rev. Lett.*, Vol. 69, No. 23, pp. 3382-3384 (1992)．
7) 後藤英一，太田滋生，鶴岡信彦：擬似乱数ビット列の生成法と安全性（準備中）．
8) Ohta, S. and Kim, S.: Finite Temperature Phase Structure of Lattice QCD for 8 and 17 Flavors, *Phys. Rev.*, Vol. D 44, No. 2, pp. 504-512 (1991); Kim, S. and Ohta, S.: QCD Thermodynamics with Eight Staggered Quark Flavors on a $16^3 \times 6$ Lattice, *Phys. Rev.*, Vol. D 46, No. 8, pp. 3607-3617 (1992); Ohta, S.: Toward Lattice QCD Simulation on AP 1000, *Proceedings of the International Symposium on Lattice Field Theory "Lattice 91"*, Tsukuba, Japan (1991)．

(平成5年11月29日受付)
(平成6年1月13日採録)

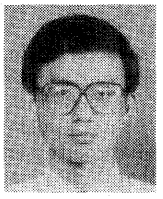


太田 滋生

1958年生．1980年東京大学理学部物理学科卒業．1985年同大学大学院理学系研究科物理学専攻博士課程卒業．理学博士．同年日本学術振興会特別研究員（PD，高エネルギー物理学研究所）．1987年コロンビア大学研究員．1990年理化学研究所基礎科学特別研究員．1993年同研究所後藤特別研究室研究員．格子上の場の量子論，とくに量子色力学（QCD）の数値的研究，およびこれに用いる専用並列・超並列計算機の開発・研究に従事．日本物理学会，American Physical Society 各会員．

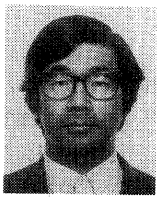
後藤 英一（正会員）

1931年生．1953年東京大学理学部物理学科卒業．1958年同大大学院修了．1958年同大理学部助手，1959年同助教授．1962年理学博士．1968年理化学研究所情報科学研究室主任研究員（非常勤）．1970年東京大学理学部教授．1986年新技術事業団創造科学技術推進事業後藤磁束量子情報プロジェクトプロジェクトリーダー（兼務）．1991年4月より神奈川大学理学部教授．同年5月より理化学研究所後藤特別研究室特別招聘研究員（兼務）．現在，熱機関，量子限界感度磁束計，完全磁気遮蔽，計算機アーキテクチャ，無発熱計算，計算理工学等の研究に従事．



黄 栄輝

1964年生．1988年シンガポール国立大学情報システム計算機科学科卒業．1991年同学科修士．1989年同学科助手．1993年筑波大学工学博士．1990年新技術事業団創造科学技術推進事業後藤磁束量子情報プロジェクト研究員．1991年理化学研究所後藤特別研究室研究員．1993年8月よりシンガポール国立大学情報システム計算機科学科講師．主に計算機アーキテクチャの研究に従事．ACM，IEEE 各会員．



吉田 宣章（正会員）

1955年生．1978年東京大学理学部物理学科卒業．1983年同大学院博士課程修了．理学博士．同年理化学研究所サイクロトロン研究室特別研究生．1985年東京大学理学部情報科学科助手．1991年理化学研究所後藤特別研究室研究員．1994年4月より関西大学総合情報学部教授．原子核物理学，数値解析，計算機アーキテクチャの研究に従事．日本物理学会会員．