数理解析研究所講究録
第522巻 1984年 251-265

指数・対数関数用ハードウェアアルゴリズムについて

高木 直史　矢島 脩三

Naofumi TAKAGI and Shuzo YAJIMA

京都大学 工学部

Faculty of Engineering, Kyoto University

1. まえがき

近年の集積回路技術の進歩により、大規模な論理回路を比較的容易に構成することが可能となってきており、これに伴い、さまざまな専用回路の研究・開発が行われるようになってきている。特に、計算機等のディジタルシステムにおいて最も重要な役割の一つを担う算術演算回路の研究・開発は盛んである[1]。我々も、既に乗算[2]、除算[3]、開平[4]について、内部計算に冗長2進表現を利用した、VLSI向きの高速ハードウェアアルゴリズムを提案している。本稿では、指数関数と対数関数について、内部計算に冗長2進表現を利用した、高速ハードウェアアルゴリズムを提案する。

指数・対数関数、三角関数等の初等関数の計算法については古くから研究が行われており、べき級数展開等による多項式近似やCORDIC法、STL法等の収束法が知られている[1]。このうち収束法は、同一の操作の繰り返しにより求める値に収束させていくもので、計算機向きの計算法として、従来からいくつかのアルゴリズムが提案されている。CORDIC法は、座標軸の段階的な回転により初等関数を計算するもので、最初Volderによって提案され[5]、後に多くの研究者が改変し、Waltherによって統一型アルゴリズムが示された[6]。STL(Sequential Table Look-up)法は、逐次表を引きながら指数・対数関数を計算するもので、Bemerの対数計算法[7]を基に、Cantorらによって提案され[8]、後にSpecker[9]やChen[10]によって改良、拡張された。また、DeLugishはこの方法に冗長リコード手法を利用した方法を提案してい

る[11]。Waltherの統一CORDIC法やSpeckerの方法、Chenの方法などはいずれも、加減算とシフト、定数の読み出しという単純な操作の繰り返しにより初等関数を計算できる。これらの計算法の実現方法としては、加算器とシフタ、定数テーブル用のメモリからなる回路を用い、マイクロプログラムの制御により同一の操作を繰り返して計算することが考えられており[6][10]、CORDIC法についてはLSIも試作されている[12]。また、田丸らは、近年の集積回路技術の進歩を考慮し、これらの計算法のアレイ構造の回路での実現について考察している[13]。

本稿では、指数･対数関数用のハードウェアアルゴリズムを提案する。本稿のアルゴリズムは、Speckerの方法[9]やChen[10]の方法を改良したもので、内部計算に冗長2進表現[2]を利用することにより、各加(減)算における桁上げ(桁借り)の伝播をなくし[2][14]、計算を高速化している。DeLugishは、冗長リコード手法を利用しているが、内部での計算は通常の2進表現で行っている[11]。Speckerの方法やChenの方法などの従来の方法では、各加(減)算における桁上げ(桁借り)の伝播のため、nビットの指数･対数関数の計算に$O(n^2)$あるいは$O(n \log n)$の計算時間が必要であった。これに対し、本稿のアルゴリズムでは、計算時間は$O(n)$と高速になる。本稿のアルゴリズムに基づく指数計算回路、対数計算回路を組合せ回路で実現すると、素子数はいずれも$O(n^2)$になる。

本稿では、2で準備を行い、3では指数関数用、4では対数関数用のハードウェアアルゴリズムを提案する。

2. 準 備

2.1 計算モデル

本稿では、指数計算回路、対数計算回路の主に組合せ回路による実現について考える。組合せ回路はある与えられた論理素子を用いて構成される、フィードバックループをもたない論理回路である。本稿では、各論理素子の入次数(fan-in)は、ある定数以下に制限されるものとする。出次数(fan-out)は制限しない。遅延は素子内部

にのみ存在するものとし、配線上での遅延は考えない。また、各素子の遅延は一定とする。

組合せ回路の段数とは、回路の入力から出力に至るすべての経路のうち、最も多くの素子をもつ経路上の素子の個数である。上記のような仮定の下では、回路での計算時間は回路の段数に比例すると考えられる。また、組合せ回路の素子数とは、回路全体を構成する素子の個数である。

2.2 冗長2進表現

本稿で利用する冗長2進表現[2]は、Avizienisが提案したSD(Signed-Digit)表現[14]の一種で、基数2の拡張SD表現である。通常の2進表現と同様、小数点以下i桁目は2^{-i}の重みをもつが、各桁は$\{-1, 0, 1\}$の要素である。以降、各桁の-1を$\bar{1}$で表す。冗長2進表現で$[x_0.x_1 \cdots x_n]_{SD2}$ $(x_i \in \{\bar{1}, 0, 1\})$ は、$\Sigma_{i=1}^{n} x_i \cdot 2^{-i}$ という数を表す。

一つの冗長2進表現は一つの数を表すが、一つの数をいくつかの冗長2進表現で表すことができる。例えば、$[0.101]_{SD2}$, $[0.11\bar{1}]_{SD2}$, $[1.0\bar{1}\bar{1}]_{SD2}$, $[1.\bar{1}01]_{SD2}$, $[1.\bar{1}1\bar{1}]_{SD2}$ はいずれも'0.625'を表している。ただし、'0'の表現は一意である。

冗長2進表現では、符号なしで、正負いずれの数も表すことができる。ある冗長2進数の正負の反転は、各桁の正負の反転、すなわち、各桁において1を$\bar{1}$に、$\bar{1}$を1にすることにより行える。従って、ある冗長2進数の正負の反転は、桁数に関係なく一定の計算時間で行うことができる。

通常の2進数はそのままで各桁が非負の冗長2進数とみなすことができ、従って、2進表現から冗長2進表現への変換には、特に計算の必要はない。

冗長2進表現から2進表現への変換は、二つの2進数の通常の減算によって行える。特に、正の値をもつn桁の冗長2進数は、nビットの符号なし2進数に変換できる。例えば、$[1.\bar{1}01]_{SD2}$は、$[1.\bar{1}01]_{SD2}=[1.001]_2 - [0.100]_2=[0.101]_2$ という計算により、$[0.101]_2$ に変換される。この変換は、順次桁上げ加算器を用いると計算時間O(n)、桁上げ先見加算器を用いると計算時間O(logn)で行える。

2.3 冗長2進数の加減算

通常の2進数の加算においては、桁上げの伝播のため、組合せ回路を用いても、少なくとも演算数の桁数の対数に比例する計算時間を要する[1]。これに対し、冗長2進数の加算においては、その冗長性を利用し、桁上げが高々1桁しか伝播しないようにすることができ、従って、組合せ回路による並列加算を演算数の桁数に関係なく一定時間で行うことができる[2]。

桁上げが高々1桁しか伝播しない加算は、二つのステップで行われる。ステップ1では、各桁で、$x_i + y_i = 2 \cdot c_i + s_i$ となるように、中間桁上げ c_i と中間和 s_i を決める。このとき、被加数 x_i と加数 y_i のうち一方が1で他方が0のときは、冗長2進表現において、'1' を2桁で $[01]_{SD2}$ と $[1\bar{1}]_{SD2}$ の2通りに表すことができることを利用し、下位から1桁上げの可能性がある場合は、$[c_i, s_i]$ を $[1, \bar{1}]$ とし、下位から $\bar{1}$ 桁上げの可能性がある場合は、$[c_i, s_i]$ を $[0, 1]$ とする。x_i, y_i のうち一方が $\bar{1}$ で他方が0のときも同様にする。下位からの桁上げの可能性は、一つ下位の桁の数 x_{i-1} と y_{i-1} を調べることにより知ることができる。従って、c_i と s_i は、x_i, y_i と x_{i-1}, y_{i-1} の4つの数から定まる。ステップ2では、各桁で、$z_i = s_i + c_{i-1}$ を計算し、中間和 s_i と一つ下位の桁からの中間桁上げ c_{i-1} から和 z_i を求める。z_i は s_i と c_{i-1} のみから定まり、桁上げは生じない。従って、z_i は x_i, y_i, x_{i-1}, y_{i-1} と x_{i-2}, y_{i-2} の6つの数からさだまる。ここに、x_i, y_i, c_i, s_i, z_i はいずれも $\{\bar{1}, 0, 1\}$ の要素である。

図1に冗長2進数の加算の例を示す。ステップ1では、x_i, y_i のうち一方が1で他方が0のときは、x_{i-1}, y_{i-1} がともに非負の場合、$[c_i, s_i]$ を $[1, \bar{1}]$ とし、その他の場合は $[0, 1]$ としている。また、x_i, y_i のうち一方が $\bar{1}$ で他方が0のときは、x_{i-1}, y_{i-1} が

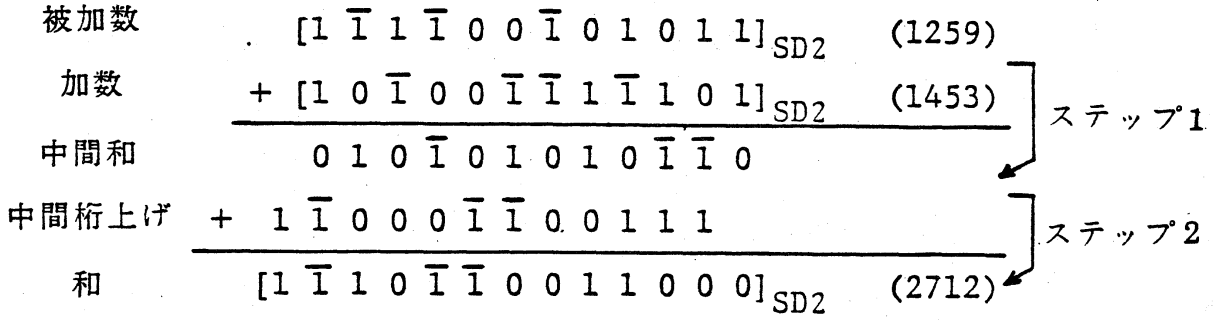

図1. 冗長2進数の加算の例

ともに非負の場合、$[c_i, s_i]$ を$[0, \bar{1}]$ とし、その他の場合は$[\bar{1}, 1]$ としている。最下位では、それより下位は0であると考える。

このように、冗長2進数の加算においては、桁上げが高々1桁しか伝播しないようにすることができ、従って、組合せ回路による並列加算を桁数に関係なく一定時間で行うことができる。

冗長2進数の減算は、減数の正負を反転し被減数に加えることによって行える。2.2で述べたように、ある冗長2進数の正負の反転は、桁数に関係なく一定の計算時間で行うことができるので、並列減算も桁数に関係なく一定時間で行うことができる。

以上、一般の冗長2進数の加減算について述べたが、加数(減数)が各桁が非負の冗長2進数である場合は、ステップ1において、各桁で中間桁上げ (c_i)と中間和(s_i)がその桁の被加数(x_i)と加数(y_i)のみから定まり、加数(減数)が一般の冗長2進数である場合より、計算が簡単になる。

3. 指数関数用ハードウェアアルゴリズム

3.1 指数関数の計算

実数 $R=E\cdot\ln 2 + X$ $(0 \leqq X < \ln 2)$ に対して、$\exp R = \exp(E\cdot\ln 2 + X) = 2^E\cdot\exp X$ であるから、2進数 R に対して $\exp R$ を計算する場合、まず R を $\ln 2$ で割り、商 E と剰余 X $(0 \leqq X < \ln 2)$ を求め、次に $\exp X$ を計算し、これを E ビットシフトすればよい。$\exp R$ を2進正規化浮動小数点表示で表すときは、$0 \leqq X < \ln 2$ より、$1 \leqq \exp X < 2$ であるから、指数部は $E + 1$、仮数部は $\exp X$ を1ビット右シフトした数にすればよい。$\ln 2$は定数であり、定数による除算は比較的容易に行えるので、本稿では、$\exp X$ $(0 \leqq X < \ln 2)$ の計算について考える。 X は小数点以下nビットまでで、$\exp X$ を小数点以下nビットまで計算するものとする。

3.2 アルゴリズムの原理

任意の$X^{(i)}$, $Y^{(i)}(>0)$, $A^{(i)}(>0)$に対して、$X^{(i)}+\ln Y^{(i)}=(X^{(i)}-\ln A^{(i)})+\ln (Y^{(i)}\cdot A^{(i)})$ が成り立つ。 $X^{(i+1)} = X^{(i)} - \ln A^{(i)}$, $Y^{(i+1)} = Y^{(i)}\cdot A^{(i)}$ とおくと、 $X^{(i)} + \ln Y^{(i)} =$

$X^{(i+1)}+\ln Y^{(i+1)}$となる。そこで、$X^{(0)}+\ln Y^{(0)}=X^{(1)}+\ln Y^{(1)}=\cdots=X^{(n+1)}+\ln Y^{(n+1)}$という変換を考え、$X^{(0)}=X$, $Y^{(0)}=1$とし、$X^{(n+1)}\simeq 0$とすると、 $X+\ln 1\simeq 0+\ln Y^{(n+1)}$より、$X\simeq \ln Y^{(n+1)}$ すなわち、$Y^{(n+1)}\simeq \exp X$となる。つまり、$X^{(i)}$を0に近付けていくと、$Y^{(i)}$は$\exp X$に近付いていく。

3.3 指数関数用ハードウェアアルゴリズム

本稿の指数関数用アルゴリズムは、次の漸化式に従う。

$$\begin{cases} X^{(i+1)} = X^{(i)} - \ln A^{(i)} \\ Y^{(i+1)} = Y^{(i)} \cdot A^{(i)} \end{cases} \qquad (i = 0, 1, \cdots, n)$$

すべてのiに対し、$-2^{-i} < X^{(i)} < 2^{-i}$ となるようにし、$X^{(i)}$を小数点以下i-1桁が0の冗長2進表現で表す。そして、各iについて、$X^{(i)}$の小数点以下i, i+1, i+2桁目の3つの桁の数 $x^{(i)}_i, x^{(i)}_{i+1}, x^{(i)}_{i+2}$ ($\in \{\bar{1}, 0, 1\}$) を調べ、$[x^{(i)}_i\ x^{(i)}_{i+1}\ x^{(i)}_{i+2}]_{SD2}$の大きさ$|X^{(i)}|_3$により、$A^{(i)}$を $\{1-2^{-i-1}, 1, 1+2^{-i-1}, 1+2^{-i}\}$ の4つの数の中から選び、次に、漸化式の計算を冗長2進数体系内で行う。$\ln A^{(i)}$は定数であるから、予め用意しておく。2n個の定数が必要である。以下にアルゴリズムを示す。

<u>指数関数用アルゴリズム</u>

*入力　$X(0 \leqq X < \ln 2)$：nビット2進小数

*出力　$Y = \exp X (1 \leqq Y < 2)$：整数部1ビット 小数部nビットの2進数

*アルゴリズム

ステップ1：Xを冗長2進数$X^{(0)}$に変換する(そのまま)。 $Y^{(0)} := 1$とする。

ステップ2：for i := 0 to n do

begin

次の規則により$A^{(i)}$を定める。

$$\begin{cases} (-4\leqq)\ |X^{(i)}|_3 \leqq -2 & \text{のとき} \quad A^{(i)} := 1-2^{-i-1} \\ -1 \leqq |X^{(i)}|_3 \leqq 1 & \text{のとき} \quad A^{(i)} := 1 \\ |X^{(i)}|_3 = 2 & \text{のとき} \quad A^{(i)} := 1+2^{-i-1} \\ 3 \leqq |X^{(i)}|_3\ (\leqq 4) & \text{のとき} \quad A^{(i)} := 1+2^{-i} \end{cases}$$

次の計算を冗長2進数体系内で行う。

$$X^{(i+1)} := X^{(i)} - \ln A^{(i)}$$

$$Y^{(i+1)} := Y^{(i)} \cdot A^{(i)}$$

end

ステップ3: 冗長2進数$Y^{(n+1)}$を等価な2進数Yに変換する。

$\ln A^{(i)}$ を 通常の2進数、すなわち各桁が非負の冗長2進数として記憶しておくと、$X^{(i+1)} := X^{(i)} - \ln A^{(i)}$ の計算は、2.3の最後で述べたように、 加数(減数)が一般の冗長2進数である場合より簡単になる。 $Y^{(i+1)} := Y^{(i)} \cdot A^{(i)}$ の計算は、シフトと一般の冗長2進数の加(減)算によって行える。

図2に、上記のアルゴリズムに基づく指数関数の計算の例をしめす。

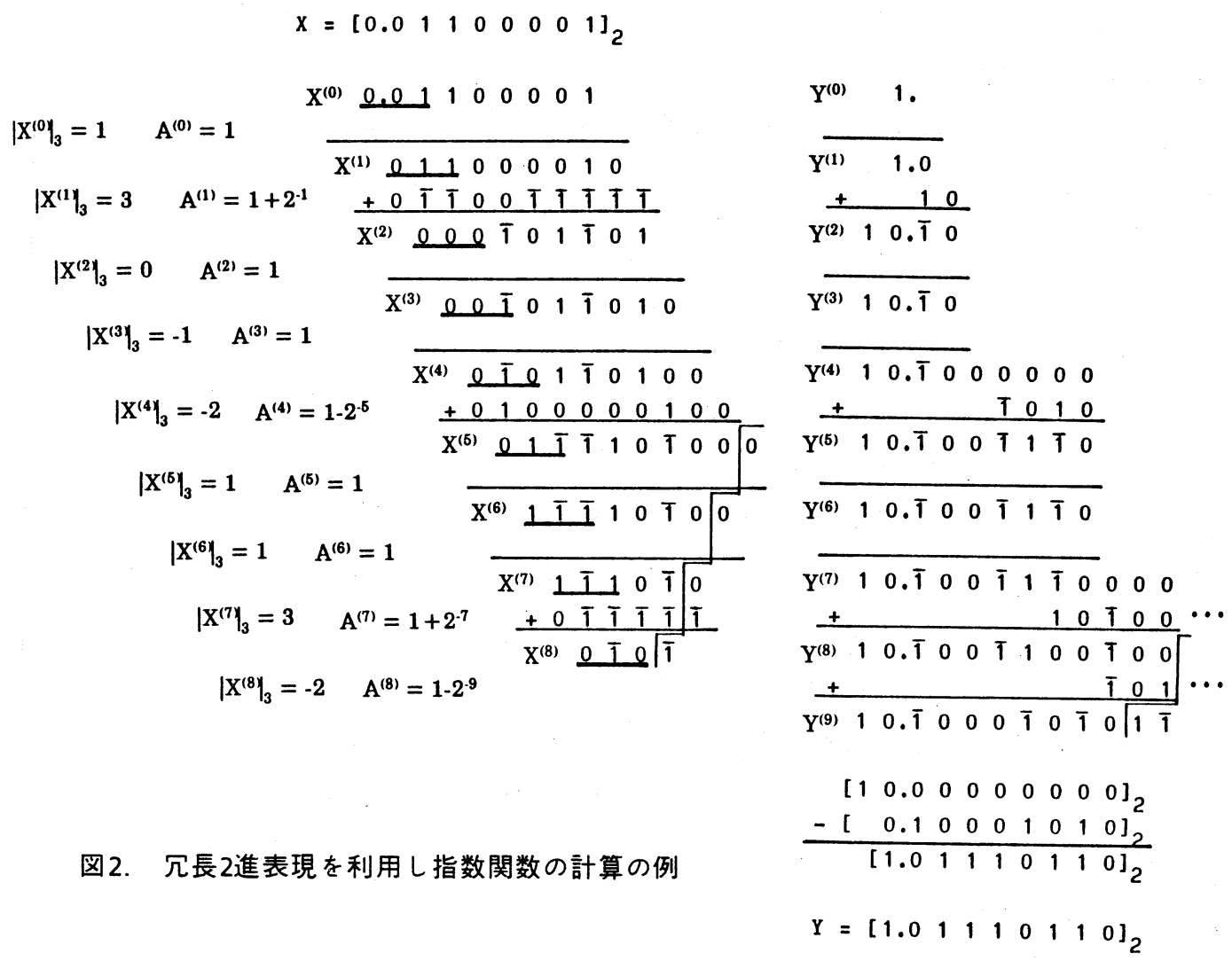

図2. 冗長2進表現を利用し指数関数の計算の例

3.4 アルゴリズムの正当性

前節で示したアルゴリズムでは、すべての i について、$-2^{-i} < X^{(i)} < 2^{-i}$ が成り立つ。これは、iに関する数学的帰納法によって証明できる。

証明

(I) $i=0$ のとき、

$-2^0=-1<0\leqq X^{(0)}=X<\ln 2<1=2^0$ であるから、$-2^0 <X^{(0)} < 2^0$ が成り立つ。

(II) $i=k$ のとき、

$-2^{-k} <X^{(k)} < 2^{-k}$ が成り立つと仮定すると、

$i=k+1$ のとき、ステップ2の計算規則より、

(1) $(-4\leqq)\ |X^{(k)}|_3 \leqq -2$ のとき、

$(-2^{-k}<)X^{(k)} <-2^{-k-2}$ であり、$A^{(k)} := 1-2^{-k-1}$ で、べき級数展開より[15]、

$\ln(1-2^{-k-1}) = -2^{-k-1}-2^{-2k-3}-\cdots$ であるから、

$-2^{-k-1} < -2^{-k-1}+2^{-2k-3}+\cdots <X^{(k+1)} < 2^{-k-2}+2^{-2k-3}+\cdots <2^{-k-1}$

(2) $-1 \leqq |X^{(k)}|_3 \leqq 1$ のとき、

$-2^{-k-1} < X^{(k)} < 2^{-k-1}$ であり、$A^{(k)} := 1$ で、$\ln 1 = 0$ であるから、

$-2^{-k-1} < X^{(k+1)} < 2^{-k-1}$

(3) $|X^{(k)}|_3 = 2$ のとき、

$2^{-k-2} <X^{(k)}< 2^{-k-1}+2^{-k-2}$ であり、$A^{(k)} := 1+2^{-k-1}$ で、べき級数展開より、

$\ln(1+2^{-k-1}) = 2^{-k-1}-2^{-2k-3}+\cdots$ であるから、

$-2^{-k-1} < -2^{-k-2}+2^{-2k-3}-\cdots <X^{(k+1)} < 2^{-k-2}+2^{-2k-3}-\cdots <2^{-k-1}$

(4) $3 \leqq |X^{(k)}|_3 (\leqq 4)$ のとき、

$2^{-k-1} < X^{(k)} (< 2^{-k})$ であり、$A^{(k)} := 1+2^{-k}$ で、べき級数展開より、

$\ln(1+2^{-k}) = 2^{-k}-2^{-2k-1}+\cdots$ であるから、

$-2^{-k-1} < -2^{-k-1}+2^{-2k-1}-\cdots <X^{(k+1)} < 2^{-2k-1}-\cdots <2^{-k-1}$

となり、いずれの場合も $-2^{-k-1} < X^{(k+1)} < 2^{-k-1}$ が成り立つ。

(I)、(II)より、すべての $i\ (=0, 1, \cdots, n+1)$ について、$-2^{-i} < X^{(i)} < 2^{-i}$ が成り立つ。

また、ここでは詳しく述べないが、すべてのi について、$X^{(i)}$ が小数点以下i-1桁目までが0の冗長2進表現で表せることも、iに関する数学的帰納法によって証明できる。このとき、$X^{(i-1)} := X^{(i)} - \ln A^{(i)}$ の計算において、$X^{(i)}$ が小数点以下 i-1桁目までが0のとき$X^{(i+1)}$が小数点以下i桁目までが0となるように、小数点以下i桁目およびi+1桁目については特別の計算規則を与える必要があるが、実際にこのような計算規則が存在する。

前節で示したアルゴリズムに従って計算すると、得られる値Yの $\exp X$ からの誤差は、丸めの誤差を考慮しなければ、2^{-n+1}以下である。

証明

$X = X^{(n+1)} + \ln Y$より、$Y = \exp(X - X^{(n+1)}) = \exp X \cdot \exp(-X^{(n+1)})$ である。上記の証明より、$-2^{-n-1} < X^{(n+1)} < 2^{-n-1}$ であるから、べき級数展開より[15]、$1 - 2^{-n-1} + 2^{-2n-3} - \cdots < \exp(-X^{(n+1)}) < 1 + 2^{-n-1} + 2^{-2n-3} + \cdots$ すなわち、$-2^{-n-1} - 2^{-2n-3} - \cdots < 1 - \exp(-X^{(n+1)}) < 2^{-n-1} - 2^{-2n-3} + \cdots$ となる。従って、$|1 - \exp(-X^{(n+1)})| < 2^{-n}$ となり、$|Y - \exp X| < \exp X \cdot 2^{-n} < 2^{-n+1}$ となる。

3.5 評価

ステップ1では、計算の必要がない。

ステップ2では、各 i について、$A^{(i)}$ は$X^{(i)}$ の中の3桁のみから定まり、漸化式の計算は冗長2進数体系内で行うので、ループ1回分の計算はnに関係なく一定の計算時間で行える。$X^{(i+1)}$ は小数点以下 i+1 桁目以降だけ計算すればよい。また、$X^{(i+1)}$の計算では、加数 (減数)は各桁が非負なので回路が簡単になる。$X^{(i+1)}$の計算に約2n桁分、$Y^{(i+1)}$の計算に約3n桁分の冗長2進加減算器を用意すれば、丸めの誤差も2^{-n}程度に押えることができる。ただし、入力X自体に丸めの誤差が含まれているので、余り下位まで計算しても意味がない。いずれにしても、ループ1回分の計算にnに比例する素子数があればよい。ステップ2ではループをn+1回繰り返すので、計算時間はO(n)、素子数は$O(n^2)$必要である。

ステップ3の変換は、2.2 で述べたように2つの2進数の通常の減算によって行え、順次桁上げ加算器を用いれば計算時間O(n)、素子数O(n)、桁上げ先見加算器を用いれば計算時間O(log n)、素子数O(n)あるいはO(n log n)必要である。

従って、全体として、nビットの指数計算が、計算時間O(n)、素子数$O(n^2)$で行える。

回路は規則正しいセル配列構造になるが、$Y^{(i+1)}$の計算においてi桁あるいはi+1桁のシフトが必要なので、配線は少し複雑になり、VLSI に埋め込んだ場合[16]、面積は$O(n^3)$になる。

4. 対数関数用ハードウェアアルゴリズム

4.1 対数関数の計算

2進正規化浮動小数点数 $R = X \cdot 2^E$ $(2^{-1} \leqq X < 1)$ に対して、$\ln R = \ln(X \cdot 2^E) = \ln X + E \cdot \ln 2$ であるから、自然対数$\ln R$を計算する場合、仮数Xの自然対数$\ln X$を求め、これに指数Eの$\ln 2$倍を加えればよい。$\ln 2$は定数であり、定数倍や加算は比較的容易に行えるので、本稿では$\ln X$ $(2^{-1} \leqq X < 1)$ の計算について考える。Xは小数点以下nビットまでで、 $\ln X$ を小数点以下nビットまで計算するものとする。

4.2 アルゴリズムの原理

任意の $X^{(i)}(>0)$, $Y^{(i)}$, $A^{(i)}(>0)$ に対して、$Y^{(i)} + \ln X^{(i)} = (Y^{(i)} - \ln A^{(i)}) + \ln (X^{(i)} \cdot A^{(i)})$ が成り立つ。$X^{(i+1)} = X^{(i)} \cdot A^{(i)}$, $Y^{(i+1)} = Y^{(i)} - \ln A^{(i)}$ とおくと。$Y^{(i)} + \ln X^{(i)} = Y^{(i+1)} + \ln X^{(i+1)}$となる。そこで、$Y^{(1)} + \ln X^{(1)} = Y^{(2)} + \ln X^{(2)} = \cdots = Y^{(n+1)} + \ln X^{(n+1)}$という変換を考え、$X^{(1)} = X$, $Y^{(1)} = 0$ とし、$X^{(n+1)} \simeq 1$ とすると、 $0 + \ln X \simeq Y^{(n+1)} + \ln 1$ より、$Y^{(n+1)} \simeq \ln X$ となる。つまり、$X^{(i)}$を1に近付けていくと、$Y^{(i)}$は$\ln X$に近付いていく。

4.3 対数関数用ハードウェアアルゴリズム

本稿の対数関数用アルゴリズムは、次の漸化式に従う。

$$\begin{cases} X^{(i+1)} = X^{(i)} \cdot A^{(i)} \\ Y^{(i+1)} = Y^{(i)} - \ln A^{(i)} \end{cases} \qquad (i = 1, 2, \cdots, n)$$

すべての i に対し、$1-2^{-i} \leqq X^{(i)} < 1+2^{-i}$ となるようにし、$X^{(i)}$ を整数部が 1 で、小数点以下 i-1 桁が 0 の冗長2進表現で表す。そして、各 i について、$X^{(i)}$ の小数点以下 i, i+1, i+2 桁目の3つの 桁の数 $x^{(i)}_i, x^{(i)}_{i+1}, x^{(i)}_{i+2}$ ($\in \{\bar{1}, 0, 1\}$) を調べ、$[x^{(i)}_i \; x^{(i)}_{i+1} \; x^{(i)}_{i+2}]_{SD2}$ の大きさ $|X^{(i)}|_3$ により、$A^{(i)}$ を $\{1-2^{-i-1}, 1, 1+2^{-i-1}, 1+2^{-i}\}$ の4つの数の中から選び、次に、漸化式の計算を冗長2進数体系内で行う。$\ln A^{(i)}$ は定数であるから、予め用意しておく。2n個の定数が必要である。以下にアルゴリズムを示す。

対数関数用アルゴリズム

*入力　$X\,(2^{-1} \leqq X < 1)$: n ビット2進小数

*出力　$Y = \ln X\,(-1 \leqq Y < 0)$: 負の n ビットの2進小数

*アルゴリズム

ステップ1 : $X = [0.1x_2 \cdots x_n]$ を冗長2進数 $X^{(1)} = [1.\bar{1}x_2 \cdots x_n]_{SD2}$ に変換する。

$Y^{(1)} := 0$ とする。

ステップ2 : for i := 1 to n do

begin

次の規則により $A^{(i)}$ を定める。

$$\begin{cases} (-4 \leqq)\, |X^{(i)}|_3 \leqq -3 & \text{のとき} \quad A^{(i)} := 1+2^{-i} \\ |X^{(i)}|_3 = -2 & \text{のとき} \quad A^{(i)} := 1+2^{-i-1} \\ -1 \leqq |X^{(i)}|_3 \leqq 1 & \text{のとき} \quad A^{(i)} := 1 \\ 2 \leqq |X^{(i)}|_3 \;(\leqq 4) & \text{のとき} \quad A^{(i)} := 1-2^{-i-1} \end{cases}$$

次の計算を冗長2進数体系内で行う。

$X^{(i+1)} := X^{(i)} \cdot A^{(i)}$

$Y^{(i+1)} := Y^{(i)} - \ln A^{(i)}$

end

ステップ3 : 冗長2進数 $Y^{(n+1)}$ を等価な2進数 Y に変換する。

$X = [0.1\ 0\ 1\ 1\ 1\ 0\ 0\ 0]_2$

$X^{(1)} = [1.\bar{1}\ 0\ 1\ 1\ 1\ 0\ 0\ 0]_{SD2}$

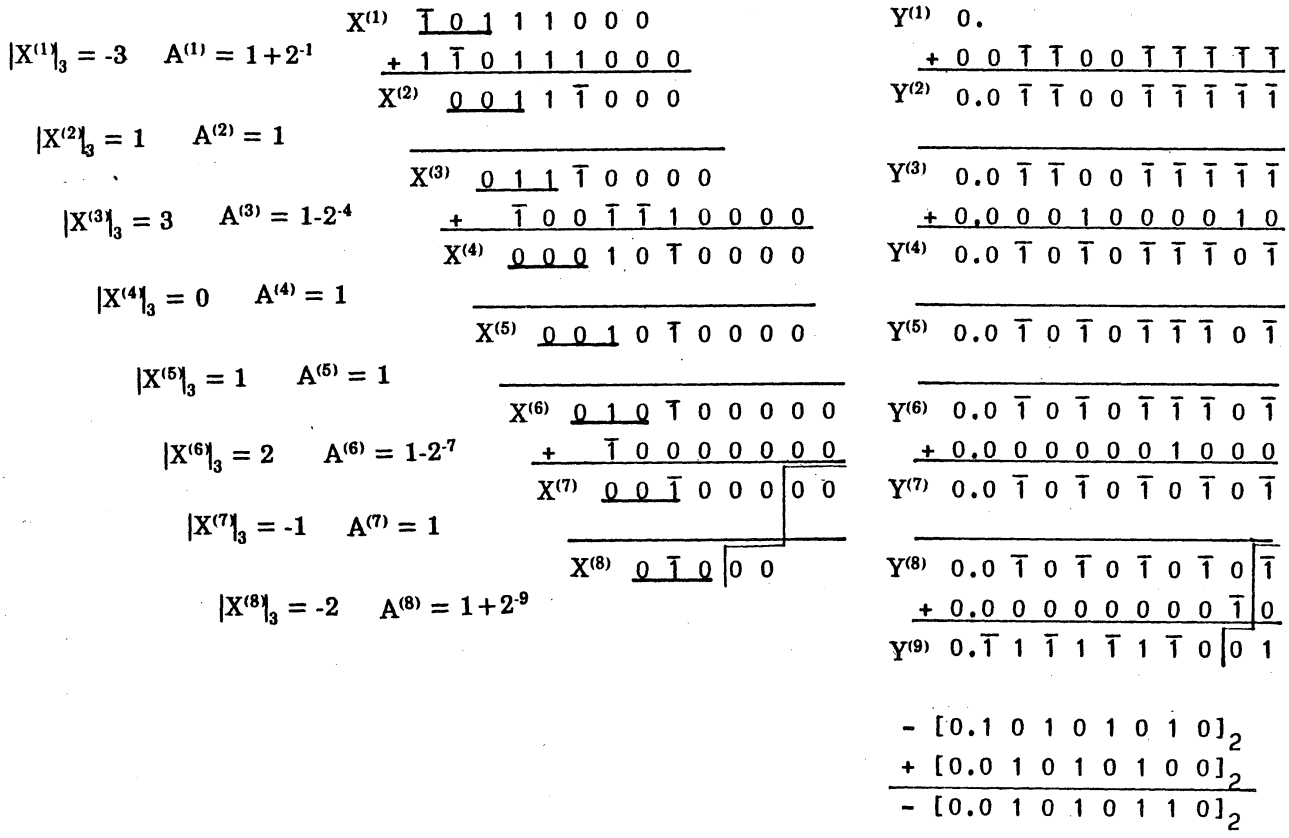

図3. 冗長2進表現を利用した対数関数の計算の例　　$Y = -[0.0\ 1\ 0\ 1\ 0\ 1\ 1\ 0]_2$

$X^{(i+1)} := X^{(i)} \cdot A^{(i)}$ の計算は、シフトと一般の冗長2進数の加(減)算によって行える。また、$\ln A^{(i)}$ を通常の2進数、すなわち各桁が非負の冗長2進数として記憶しておくと、$Y^{(i+1)} := Y^{(i)} - \ln A^{(i)}$ の計算は、2.3の最後で述べたように、加数(減数)が一般の冗長2進数である場合より簡単になる。

図3に、上記のアルゴリズムに基づく対数関数の計算の例をしめす。

4.4 アルゴリズムの正当性

前節で示したアルゴリズムでは、すべての i について、$1-2^{-i} \leqq X^{(i)} < 1+2^{-i}$ (=は i=1 のときのみ) が成り立つ。これは、3.4の証明と同様、iに関する数学的帰納法によって証明できる。また、すべてのiについて、 $X^{(i)}$ が整数部が1で小数点

以下 i-1 桁目までが 0 の冗長2進表現で表せることも、i に関する数学的帰納法によって証明できる。このとき、$X^{(i-1)} := X^{(i)} \cdot A^{(i)}$ の計算において、$X^{(i)}$ が整数部が 1 で小数点以下 i-1桁目までが0のとき $X^{(i+1)}$ が整数部が1で小数点以下i桁目までが0となるように、小数点以下i桁目およびi+1桁目については特別の計算規則を与える必要があるが、実際にこのような計算規則が存在する。

前節で示したアルゴリズムに従って計算すると、得られる値Yの $\ln X$ からの誤差は、丸めの誤差を考慮しなければ、2^{-n}以下である。

証明

$\ln X = Y + \ln X^{(n+1)}$ より、$Y = \ln X - \ln X^{(n+1)}$ である。$1-2^{-n-1} < X^{(n+1)} < 1+2^{-n-1}$ であるから、べき級数展開より[15]、$-2^{-n-1}-2^{-2n-3}- \cdots < \ln X^{(n+1)} < 2^{-n-1}-2^{-2n-3}+\cdots$ となる。従って、$|\ln X^{(n+1)}| < 2^{-n}$ となり、$|Y - \ln X| < 2^{-n}$ となる。

4.5 評価

ステップ1での変換は、nに関係なく一定の計算時間で行える。必要な素子数もnに関係なく一定である。

ステップ2では、各 i について、$A^{(i)}$ は $X^{(i)}$ の中の3桁のみから定まり、漸化式の計算は冗長2進数体系内で行うので、ループ1回分の計算はnに関係なく一定の計算時間で行える。$X^{(i+1)}$ は小数点以下i+1桁目以降だけ計算すればよい。また、$Y^{(i+1)}$ の計算では加数(減数)は各桁が非負であるから回路が簡単になる。$X^{(i+1)}$ の計算に約3n桁分、$Y^{(i+1)}$ の計算に約2n桁分の冗長2進加減算器を用意すれば、丸めの誤差も 2^{-n} 程度に押えることができる。ただし、入力 X 自体に丸めの誤差が含まれているので、余り下位まで計算しても意味がない。いずれにしても、ループ1回分の計算にnに比例する素子数があればよい。ステップ2ではループを n 回繰り返すので、計算時間は $O(n)$、素子数は $O(n^2)$ 必要である。

ステップ3の変換は、2.2 で述べたように2つの2進数の通常の減算によって行え、順次桁上げ加算器を用いれば計算時間 $O(n)$、素子数 $O(n)$、桁上げ先見加算器を用いれば計算時間 $O(\log n)$、素子数 $O(n)$ あるいは $O(n \log n)$ 必要である。

従って、全体として、nビットの対数計算が、計算時間O(n)、素子数$O(n^2)$で行える。

回路は規則正しいセル配列構造になるが、$X^{(i+1)}$の計算においてi桁あるいはi+1桁のシフトが必要なので、配線は少し複雑になり、VLSIに埋め込んだ場合[16]、面積は$O(n^3)$になる。

5. むすび

内部計算に冗長2進表現を利用した指数・対数関数用ハードウェアアルゴリズムを提案した。Speckerの方法やChenの方法などの従来の計算法では、漸化式の計算に、順次桁上げ加算器を用いると、計算時間は$O(n^2)$で素子数も$O(n^2)$であり、桁上げ先見加算器を用いると、計算時間は$O(n \log n)$で素子数は$O(n^2)$あるいは$O(n^2 \log n)$であった。これに対し、本稿のアルゴリズムでは、素子数は$O(n^2)$のままで、計算時間がO(n)と高速になる。

本稿で考察した指数関数計算回路と対数関数計算回路は共通部分が多いので、一つにまとめて、指数・対数関数計算回路とすることもできる。また、それぞれ、冗長2進加減算器とシフタおよび定数テーブル用のメモリからなる回路としても容易に実現でき、マイクロプログラムの制御により、効率よく計算をおこなえる。

謝辞

御討論頂いた本学安浦寛人博士はじめ矢島研究室の諸氏に感謝致します。なお、本研究は一部文部省科学研究費補助金による。

参考文献

[1] K.Hwang, "Computer Arithmetic / Principles, Architecture, and Design," John Wiley & Sons, 1979.

[2] 高木、安浦、矢島、「冗長2進加算木を用いたVLSI向き高速乗算器」、信学論(D)、vol.J66-D、no.6、pp.683-690、昭和58年6月

[3] 高木、安浦、矢島、「冗長2進表現を利用したVLSI向き高速除算器」、第26回情処全大、6P-7、pp.299-300、昭和58年3月

[4] 高木、安浦、矢島、「冗長2進表現を利用したVLSI向き高速開平法」、第27回情処全大、7J-4、pp.41-42、昭和58年10月

[5] J.E.Volder, 'The CORDIC Trigonometic Computing Technique,' IRE Trans. Elec. Comput., vol.EC-8, no.3, pp.330-334, Mar. 1959.

[6] J.S.Walther, 'A Unified Algorithm for Elementary Functions,' AFIPS 1971 SJCC, pp.379-385, 1971.

[7] R.W.Bemer, ' A Subroutine Method for Calculating Logarithms,' CACM, vol.1, no.5, pp.5-7, May. 1958

[8] D.Cantor, G.Estrin and R.Turn, 'Logarithmic and Exponential Function Evaluation in a Variable Structure Digital Computer,' IRE Trans. Elec. Comput., vol.EC-11, no.4, pp.155-164, Apr. 1962.

[9] W.H.Specker, 'A Class of Algorithms for Ln x, Exp x, Sin x, Cos x, $Tan^{-1}x$ and $Cot^{-1}x$,' IEEE Trans. Elec. Comput., vol.EC-14, no.1, pp.85-86, Jan. 1965.

[10]T.C.Chen, 'Automatic Computation of Exponentials, Logarithms, Ratios and Square Roots,' IBM J. Res. & Dev., vol.16, no.4, pp.380-388, July 1972.

[11]B.G.Delugish, 'A Class of Algorithms for Automatic Evaluation of Certain Elementary Functions in a Binary Computer,' Technical Report, no.399, Dept. of Computer Science, Univ. of Illinois, Urbana, Illinois, 1970.

[12]G.L.Haviland and AL A.Tuszynsky, 'A CORDIC Arithmetic Processor Chip,' IEEE Trans. Comput., vol.C-29, no.2, pp.68-79, Feb. 1980.

[13]田丸、金原、「VLSIプロセッサのためのアレイ構造初等関数演算回路」、信学論(D)、vol.J66-D、no.3、pp.309-315、昭和58年3月

[14]A.Avizienis, 'Signed-Digit Namuber Representations for Fast Parallel Arithmetic,' IRE Trans. Elec. Comput., vol.EC-10, no.9, pp.389-400, Sep. 1961.

[15]森口、宇田川、一松、「数学公式II / 級数, フーリエ解析」、岩波全書、1957

[16]安浦、矢島、「論理回路のVLSI上での面積について」、信学論(D)、vol.J65-D、no.8、pp.1080-1087、昭和57年8月