数理解析研究所講究録
第666巻 1988年 115-124

Area-Time Efficient Evaluation of Elementary Functions

Yasuo OKABE and Shuzo YAJIMA
Faculty of Engineering, Kyoto University

ABSTRACT

This paper describes area-time efficient VLSI algorithms for evaluating elementary functions. For square rooting, a VLSI implementation of AT^2-optimal, i.e., $AT^2=O(n^2)$, circuits are presented, where A is the chip area and T is the computation time in the range $[\Omega((\log n)^{1+\varepsilon}, O(\sqrt{n})]$ for arbitrary $\varepsilon>0$. For logarithms and exponentials, an upper bound $AT^2=O(n^2(\log n)^2)$ is given for any $T\in[\Omega((\log n)^{2+\varepsilon}, O(\sqrt{n}\log n)]$. VLSI circuits for these functions with minimum computation time $O(\log n)$ and almost optimal AT^2-preformance, i.e., $AT^2=O(n^{2+\varepsilon})$, are also exhibited. These are achieved by using an extension of the Beame-Cook-Hoover Method which we have proposed before.

1. Introduction

Much research has long been conducted on arithmetic operations, such as addition, multiplication, division, square rooting and evaluation of elementary functions. Recent advances in large-scale integration technology of circuits have especially motivated the research on hardware algorithms for arithmetic operations, suitable for VLSI implementations.

In VLSI circuits, the chip area is a more reasonable cost measure than the number of gates. Considering this, VLSI models are proposed as more practical computation models [1][2], and much attention has been paid about area-time tradeoffs for many basic operations such as arithmetic operations.

For multiplication, AT^2-optimal, i.e., $AT^2=O(n^2)$, multipliers are known for all computation times in the range $[\Omega(\log n), O(\sqrt{n})]$ [3]. For division, Mehlhorn and Preparata exhibited an AT^2-optimal design of dividers with computation times in the range $[\Omega((\log n)^{1+\varepsilon}), O(\sqrt{n})]$ for arbitrary constant $\varepsilon>0$ [4], utilizing Beame-Cook-Hoover division technique. Mehlhorn also exhibited AT^2-optimal square rooting circuits for all computation time in the range $[\Omega((\log n)^3), O(\sqrt{n})]$ [5].

In this paper, we present several fast and area-time efficient algorithms for evaluating elementary functions, and give some new upper bounds of the AT^2-complexity of these functions. First we describe AT^2-optimal square rooting circuits for all computation times in the range $[\Omega((\log n)^{1+\varepsilon}), O(\sqrt{n})]$ for arbitrary $\varepsilon>0$. Next we exhibit

circuits evaluating logarithms and exponentials with $AT^2=O(n^2\log^2 n)$ for all computation times in the range $[\Omega((\log n)^{2+\varepsilon}), O(\sqrt{n}\log n)]$ for arbitrary $\varepsilon>0$. These are based on the arithmetic-geometric mean iteration [6], and implemented using our square rooting circuits. We also consider implementations of circuits with minimum computation time, and present circuits computing these functions with time $O(\log n)$ and area $O(n^{2+\varepsilon})$ for arbitrary $\varepsilon>0$.

Throughout this paper, we are concerned with the evaluation of square root $\sqrt{x}$ ($1/4\leq x<1$), exponential function $\exp x$ ($0\leq x<ln2$), logarithmic function $\ln x$ ($1\leq x<2$), etc., for an n-bit unsigned fixed-point binary number x, with absolute error $<2^{-n}$. For any number, the approximate values of these functions can easily be computed from the value for x in the above domains. We adopt the *VLSI model* developed by Brent and Kung [2] as our computation model, and consider *time* and *area* as our cost measures.

2. Area-Time Optimal Square Rooting Circuits for $T=\Omega((\log n)^{1+\varepsilon})$

In this section, we present an area-time optimal square rooting circuit for $T=\Omega((\log n)^{1+\varepsilon})$. Our square rooting algorithm is a modification of Mehlhorn-Preparata's area-time optimal division [4]. In our algorithm, $1/\sqrt{x}$ is first evaluated by the Newton iteration as a root of the equation $u^{-2}-x$, and $\sqrt{x}$ is calculated by the equation $\sqrt{x}=x\cdot(1/\sqrt{x})$. The iteration rule is

$$u_{i+1}=\frac{1}{2}u_i(3-x\,u_i^{\,2}).$$

Since the Newton iteration has a self-correcting property, it is sufficient to compute with 2^{i+1}-bit precision at the i-th stage of the iteration. Using this rule, it is immediately shown that there exists an AT^2-optimal, i.e., $AT^2=O(n^2)$, n-bit square rooting circuit for any $T\in[\Omega((\log n)^2), O(\sqrt{n})]$. (See [5].)

We now describe an area-time optimal square rooting circuit with computation time T in the range $[\Omega((\log n)^{1+\varepsilon}), O((\log n)^2)]$ for any $\varepsilon>0$, which is a modification of Mehlhorn-Preparata's area-time optimal divider with computation time in the same range [4]. It is easily shown that almost all techniques for division are also applicable to square rooting, except for the polynomial approximation of inverses; the approximation, however, essentially depends on the peculiarity of the Taylor expansion of $1/x$.

Instead of the polynomial approximation, we present a new technique for approximation of inverse of square roots, based on the Newton iteration. Consider the algorithm described below:

Algorithm INSQRT2(x)
Given: an integer $s\in[2,l)$ and an integer $k=\lceil \log_4 s \rceil$
Input: an l-bit number $x\in[1/4,1)$
Output: an l+3-bit number $v\in(1,2]$ s.t. v gives the leading (l+3) bits of $1/\sqrt{x}$

```
   function f_x^k (u∈[1,2):real number): real number;
    begin
(f1)   for i:=1 to k do u:=½u(3−x·u²);
(f2)   f_x^k:=u;
    end;
(1) begin i:=0; u_0:=1;
(2)  while 1−x·u_i²<2^−(l+3) do
      begin
(3)     u_{i+1}:=leftmost 2^{i+1} bits of f_x^k(u_i);
(4)     i:=i+1
      end;
(5)  v:=u_i;
   end.
```

In line (3) of Algorithm INSQRT2,

$$f_x^{\,k}(u)=f_x(f_x(\cdots f_x(u)\cdots))$$

is evaluated in one step, instead of k-time evaluation of f_x.

The following theorem tells us the area-time preformance of computing $f_x^{\,k}(u)$ in lines (f1)-(f2) [7]. This theorem is proved by using an extension of Beame-Cook-Hoover's method [8].

PROPOSITION 1. [Okabe, Takagi and Yajima, 1987] There exists a circuit which computes the n-bit approximate value of a polynomial of degree k in an n-bit number, in time $O(\log(nk))$ and with area $O((nk)^2)$ if $k\geq O(\log n)$ or $O(n^2 k\log n)$ if $k< O(\log n)$.

Since $f_x^{\,k}(u)$ is a polynomial of degree 4^k ($\approx s$) in u and x, $f_x^{\,k}(u)$ can be computed in time $O(\log l)$ and with area $O(l^2 s(s+\log l))$. The number of iterations required in lines (2)-(4) is $\lceil \log_2 l \rceil / k$, and thus the total computation time of the square rooting circuit is $O(\log^2 l/\log s)$ and the chip area is $O(l^2 s(s+\log l))$. We now obtain the following lemma:

LEMMA 1. For any $s\in[2,l]$, there exists a circuit which computes the l-bit inverse of the square root of an l-bit number in time $O((\log l)^2/\log s)$ and has area $O((ls)^2)$ if $s\geq O(\log l)$ or $O(l^2 s\log l)$ if $s< O(\log l)$.

We are now ready to describe our square rooting algorithm with optimal AT^2-performance for the computation time $T=O(\log^{1+\varepsilon} n)$ for any $\varepsilon>0$.

The successive refinement technique [4] for square rooting can be written as:

```
Algorithm INSQRT3(x)
Given: an integer sequence l_1<l_2<...<l_J=l
Input: an l-bit number x∈[¼,1)
Output: an l-bit number v∈(1,2] s.t. vx=1+ε, ε<2^{-l}

(1) begin v:=1;
(2)   for i:=1 to J do
(3)     begin t_i:=leftmost (l_i+1) bits of x;
(4)           z_i:=v^2 t_i;
(5)           x_i:=leftmost (l_i+1) bits of z_i;
(6)           v_i:=(leftmost (l_i+1)-bit overinverse of square root of x_i); {i.e., ⌈2^{l_i+1}/√x_i⌉·2^{-(l_i+1)}}
(7)           v:=v·v_i;
      end;
  end.
```

In line (6), Algorithm INSQRT2 presented in the previous subsection is called as a subroutine. To proceed the same discussion as in the case of division, we need the next lemma:

LEMMA 2. If an l-bit number $x\in[¼,1)$ has $(l'-1)$ zeros immediately to the right of the leading 1, the l-bit inverse of the square root of x can be computed in time $T=O(\log (l/l')\cdot\log l/\log s))$ and with area $A=O((ls)^2)$, for any $s\in[O(\log l),l/l']$.

Proof. In Algorithm INSQRT2, the first approximation $u_0=1$ has a precision of at least l' bits. This implies that $O(\log (l/l'))$ iterations are sufficient to compute $1/\sqrt{x}$ to a precision of l bits. □

For $i=2,...,J$, it is obviously verified that x_i in Algorithm INSQRT3 satisfies the condition of x in the above lemma for $l=l_i$ and $l'=l_{i-1}$.

Only a straightforward discussion remains. Choose

$$l_i=\frac{n}{(\log n)^{1+\varepsilon}s_i},\qquad s_i=\Big(\frac{n}{(\log n)^{1+\varepsilon}}\Big)^{1/2^{s(\log n)^{\varepsilon(i-1)}}}\quad (i=1,\ldots,J),$$

where J is a largest value of i for which $s_i>2$. (Indeed $J=\theta(1/\varepsilon)$.) Then it is proved that the successive refinement modules based on Algorithm INSQRT3 can be implemented as a VLSI circuit with $T=O((\log n)^{1+\varepsilon})$, $A=O(\ n^2/((\log n)^{1+\varepsilon})^2\ )$, though we will omit the proof.

The Newton iteration techniques utilized here are completely equal to those of Mehlhorn-Preparata's for their division algorithms [4]. Thus we have:

THEOREM 1. For any fixed $\varepsilon>0$, the n-bit square root of an n-bit number can be evaluated with optimal AT^2-performance $O(n^2)$ for any $T\in[\Omega((\log n)^{1+\varepsilon}),O(\sqrt{n})]$.

Note that similar results can also be derived for the computation of $\sqrt[k]{x}$ (the k-th root of x) for any fixed integer k.

On the other hand, by choosing s as $s=l^{\varepsilon}$ $(1\geq\varepsilon>0)$ in Algorithm INSQRT2, the resulting circuit achieves $T=O((1/\varepsilon)\log l)$ and $A=O(l^{2(1+\varepsilon)})$. Thus we get the following result:

THEOREM 2. For any $\varepsilon>0$, there exists a circuit which computes the n-bit square root of an n-bit number in time $O(\log n)$ and with area $A=O(n^{2+\varepsilon})$.

3. Area-Time Efficient Evaluation of Exponentials and Logarithms

In this section, we will consider the area-time efficient implementation of circuits for exponentials and logarithms.

First we present an algorithm for evaluating n-bit logarithms with $AT^2=O(n^2\log^2 n)$ based on the arithmetic-geometric mean iteration of Gauss [6]. The algorithms is as follows:

Algorithm LOG1(x)
Input: an n-bit number $x\in[1,2)$
Output: an n-bit number $y\in[0,\ln 2)$ s.t. $|y-\ln x|<2^{-n}$

(1) **begin** $a_0:=1$; $b_0:=2^{2-n}\cdot x^{-1}$; $i:=0$;
(2) **while** $a_i-b_i>2^{-n}$ **do**
begin
(3) $a_{i+1}:=(a_i+b_i)/2$;
(4) $b_{i+1}:=\sqrt{a_i\cdot b_i}$;
(5) $i:=i+1$;
end;
(6) $y:=\pi/(2a_i)-n\ln 2$
end.

Using the result on area-time optimal square rooting circuits, the following theorem follows immediately.

THEOREM 3. For any fixed $\varepsilon>0$, the n-bit logarithm of an n-bit number can be calculated with $AT^2=O(n^2(\log n)^2)$ for any computation time $T\in[\Omega((\log n)^{2+\varepsilon}), O(\sqrt{n}\log n)]$.

Once an efficient evaluation of logarithms is established, the exponential function $y=\exp x$ can be evaluated as the root of the equation,

$$f(y)=\ln y-x=0$$

for any x, by applying the Newton method. The algorithm can be written as:

```
Algorithm EXP1(x)
Input: an n-bit number x∈[0,ln 2)
Output: an n-bit number y∈[1,2) s.t. |y−exp x|<2^{−n}

   begin
(1)  y_0:=(l-bit approximation of exp x); i:=0;
(2)  while |ln y−x|>2^{−n} do
       begin
(3)      y_{i+1}:=y_i(1+x−ln y);
(4)      i:=i+1;
       end;
(5)  y:=y_i
   end.
```

We now consider the efficient implementation of this algorithm for each computation time T in the range $T\in[\Omega((\log n)^{2+\varepsilon}), O(\sqrt{n}\log n)]$.

THEOREM 4. For any fixed $\varepsilon>0$, the n-bit exponential of an n-bit number can be evaluated with $AT^2=O(n^2(\log n)^2)$ for any computation time $T\in[\Omega((\log n)^{2+\varepsilon}), O(\sqrt{n}\log n)]$.

The theorem is easily proved for T in the range $[\Omega((\log n)^4), O(\sqrt{n}\log n)]$, by choosing $y_0:=1$ as an initial approximate value in line (1), utilizing the efficient Newton iteration technique in [5] and evaluating logarithms in line (3) by Algorithm LOG1. Thus we consider the implementation of Algorithm EXP1 for $T\in[\Omega((\log n)^{2+\varepsilon}), O((\log n)^4)]$.

First we propose a new algorithm for fast evaluation of a good initial l-bit approximate value of exp x in line(1). Let x be an l-bit binary number, and suppose $l=s^k$. Let x_0 be the leftmost bit of x, and x_i be the $(s^{i-1}+1)$-th to s^i-th bits of x $(i=1,\ldots,k)$. Then $x=x_0+x_1+x_2+\cdots+x_k$ and therefore

$$\exp x=\exp(x_0+x_1+x_2+\cdots+x_k)=(\exp x_0)(\exp x_1)(\exp x_2)\cdots(\exp x_k).$$

This leads to the algorithm shown below:

Algorithm EXP2(x)
Given: integers s,k s.t. $s\leq l$ and $k=\lceil \log_s l \rceil$
Input: an l-bit number $x\in[0,\ln 2)$,
Output: an l-bit number $y\in[1,2)$, s.t. $|y-\exp x|<2^{-l}$

(1) **begin**
(2) **for** $i:=1$ **to** k {in parallel} **do**
begin
(3) $x_i:=$the $(s^{i-1}+1)$-th to s^i-th bits of x;
(4) $y_i:=2l$-bit approximation of exp x_i;
end;
(5) $y:=\prod_{i=1}^{k} y_i$
end.

Consider the Taylor expansion of exp x_i,

$$\exp x_i=1+x_i+(1/2!)x_i^2+\cdots+(1/n!)x_i^n+\cdots$$

where $x_i<2^{-s^{i-1}}$.

This means that the sum of the first $2l/s^{i-1}$ terms is an approximate value of exp x_i with the precision of $2l$ bits, since $x_i^{2l/s^{i-1}}<2^{-2l}$. Thus we may assume that y_i in line (4) is a polynomial of x_i of degree $2l/s^{i-1}$.

Since x_i is an (s^i-s^{i-1})-bit number, Step (4) is carried out in time $O(\log l)$ and with area $O(l^2 s(s+\log l))$ from Proposition 1. (Note that $(s^i-s^{i-1})\cdot(2l/s^{i-1})=O(ls)$.) All x_i's can be calculated in parallel (lines (2)-(4)), and multiplied up pairwise in a binary-tree-form (line (5)). Thus computation time required in line (2)-(4) is $O(\log l)$ and area $O(k\cdot l^2 s(s+\log l))$. Using $(2k-1)$ multipliers with time $O(\log l)$ and area $O(l^2)$ in line (5), their product can be obtained in time $O((\log k)\cdot(\log l))$ and with area $O(k\cdot l^2)$. Hence the computation time $O((\log k)\cdot(\log l))$ and the chip area $O(k\cdot l^2 s(s+\log l))$ in line (5). Thus we have the following lemma:

LEMMA 3. For any $s\in[2,l]$, there exists a circuit which computes the l-bit exponential of an l-bit number in time $O((\log l)\log(\log l/\log s))$ and has area $O((ls)^2\log l/\log s)$ if $s\geq O(\log l)$ or $O(l^2 s\log^2 l/\log s)$ if $s<O(\log l)$.

Let us continue the proof of the theorem for $T\in[\Omega(\log^{2+\varepsilon} n), O(\log^4 n)]$. Let $l=n/T$, and consider the Algorithm EXP1 for this l. From the above lemma for $s=2$, it follows that there is a circuit, say F_E, which computes the initial l-bit approximation of exp x in line (1) in time

$$T_E=O((\log l)\log\log l)=O((\log n)\log\log n)$$

and has area

$$A_E = O(l^2 \log^2 l) = O(n^2 \log^2 n / T^2).$$

For this initial approximate value x_0, the number of required iterations of lines (2)-(4) is $m = \lceil \log_2(n/l) \rceil = \lceil \log_2 T \rceil$. Suppose $c \geq 0$ be a number which satisfies $T > \log^{2+c} n$. Let $i_0 = \lceil \log_2(\log^{2+c} n) \rceil$. We realize the first i_0 iterations on a single circuit F using feedback, and the later $m - i_0$ iterations on their own circuits $F_{i_0+1}, \ldots, F_m$ respectively.

At each iteration step, F computes at most $2^{i_0} = \lceil n \log^3 n / T \rceil$-bit logarithm and multiplication. From Theorem 3, there exists a circuit F which computes each step of the former i_0 iterations in time $T_F = O(\log^{2+c} n)$ and has area

$$A_F = O((l \cdot 2^{i_0})^2 \cdot (i_0 + \log_2 l)^2 / (\log^{2+c} n)^2) = O(n^2 \log^2 n / T^2) .$$

The circuit F_{m-j} computes the 2^{m-j}-bit logarithm and 2^{m-j}-bit multiplication. We choose the computation time of F_{m-j} as

$$T_{m-j} = T / 2^{j/2}$$

for each $j = 0, \ldots, m - i_0 - 1$. Note that

$$T_{i_0+1} = T / l^{(m-i_0+1)/2} = O((T \log^{2+c} n)^{1/2}) > O(\log^{2+c} n).$$

From the result of Theorem 3, there exists a circuit F_j which has area

$$A_{m-j} = O((2^{m-j})^2 (m-j)^2 / T_j^2) = O(((2^m)^2 \ m^2 / T^2) \ 2^{-j}) = O((n^2 \log^2 n / T^2) \ 2^{-j})$$

for any $j = 0, \ldots, m - i_0 - 1$.

These imply that our circuit has area

$$A_r = A_E + A_F + \sum_j A_j = O(n^2 \log^2 n / T^2) + \sum_j O((n^2 \log^2 n / T^2) \ 2^{-j}) = O(n^2 \log^2 n / T^2)$$

and computes the n-bit logarithm in time

$$T_r = T_E + T_F \cdot i_0 + \sum_j T_j = O((\log n) \log \log n) + O(\log^{1+c} n) \cdot O(\log n) + \sum_j T / 2^{-j/2} = O(T),$$

since $T \geq O(\log^{2+c} n)$. Thus the theorem follows.

Next, let us consider the time-optimal, i.e. $T = O(\log n)$, circuits for exponentials and logarithms, with suboptimal AT^2-performance.

Consider Algorithm EXP2 in the previous subsection, and choose $s = l^{\varepsilon/3}$ for any $1 \geq \varepsilon > 0$. The Circuit in Lemma 3 operates in time $T = O(\log l)$ and has area $A = O(l^{2+(2/3)\varepsilon})$. Thus we have:

THEOREM 5. For any $\varepsilon>0$, there exists a circuit which computes the n-bit exponential of an n-bit number in time $O(\log n)$ and with area $O(n^{2+\varepsilon})$.

We will show the corresponding result for logarithms. Let x be an l-bit binary number and suppose $l=s^k$. It is easily verified that x can be written as $x\simeq\eta_1\cdot\eta_2\cdot\cdots\cdot\eta_k$, where η_i is an s^i-bit number which has at least s^{i-1} consecutive 0's immediately to the right of the leading 1. Since

$$\ln x=\ln(\eta_1\cdot\eta_2\cdot\cdots\cdot\eta_k)=\ln\eta_1+\ln\eta_2+\cdots+\ln\eta_k,$$

$\ln x$ is obtained as:

Algorithm LOG2(x)
Given: integers s,k s.t. $s\leq l$ and $k=\lceil\log_s l\rceil$
Input: an l-bit number $x\in[1,2)$
Output: an l-bit number $y\in[0,\ln 2)$, s.t. $|y-\ln x|<2^{-l}$

(1) **begin** $x_0:=x$; $y:=0$;
(2) **for** $i:=1$ **to** k **do**
begin
(3) $\eta_i:=$leftmost s^i bits of x_{i-1};
(4) $x_i:=x/\eta_i$;
(5) $y_i:=l$-bit approximation of $\ln\eta_i$;
(6) $y:=y+y_i$;
end
end.

Consider the Taylor expansion of $\ln\eta_i$,

$$\ln\eta_i=\ln(1+h_i)=h_i-(1/2)h_i^2+\cdots+(-1)^n(1/n)h_i^n+\cdots$$

As is obvious in line (3)-(4), $h_i<2^{s^i-1}$. This means that sum of the first $2l/s^{i-1}$ terms is an approximate value of $\ln\eta_i$ with the precision of $2l$ bits. Thus we may assume that y_i is a polynomial of h_i of degree $2l/s^{i-1}$. Since h_i is an (s^i-s^{i-1})-bit number, l-bit approximation of $\ln\eta_i$ can be computed in time $O(\log(s^i\cdot(2l/s^{i-1})))=O(\log(ls))$ and with area $O((s^i\cdot(2l/s^{i-1}))^2)=O((ls)^2)$ (Proposition 1).

Choosing $s=l^{\varepsilon/3}$ for any $1\geq\varepsilon>0$, this satisfies $k=\theta(1/\varepsilon)$. Step (4) can be carried out in time $O(\log l)$ using a divider with area $O(l^{2+(2/3)\varepsilon})$ (see [4]). For each $i=1,\ldots,k$, y_i can be evaluated (line (5)) and added up (line (6)) in time $O(\log(ls))=O(\log l)$ and with area $O((ls)^2)=O(l^2)$ Thus the total computation time of our circuit is $O(k\log l)=O(\log l)$ and the chip area is $O(l^{2+(2/3)\varepsilon})$. We get:

THEOREM 6. For any $\varepsilon>0$, there exists a circuit which computes the n-bit logarithms of an n-bit number in time $O(\log n)$ and with area $O(n^{2+\varepsilon})$.

4. Considerations

Trigonometric functions such as sines, cosines and arctangents can be evaluated from exponentials and logarithms with complex arguments, since

$$\log (v+iw)=\log |v+iw|+i\cdot\arctan(w/v) ,$$

$$\exp i\theta=\cos \theta+i\cdot\sin \theta$$

[6]. It is not difficult to modify our algorithms to accept complex arguments. Thus the same upper bounds as for exponentials and logarithms can be obtained for these functions.

Acknowledgements

We would like to express sincere appreciation to Dr. H. Hiraishi, Dr. N. Takagi, N. Ishiura, and all the members of the Yajima Laboratory at Kyoto University, for their helpful discussions. We would also like to thank Dr. K. Iwama in Kyoto Sangyo University and Dr. H. Yasuura in Kyoto University, for their comments and criticism on parallel and VLSI algorithms.

References

[1] C. D. Thompson: "A Complexity Theory for VLSI", Tech. Rep. CMU-CS-80-140, Dept. of Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, Pa (Jan. 1979).

[2] R. P. Brent and H. T. Kung: "The Area-Time Complexity of Binary Multiplication", *JACM,* **28-3,** 521-534 (1981).

[3] K. Mehlhorn and F. P. Preparata: "Area-Time Optimal VLSI Integer Multipiler with Minimum Computation Time", *Inform. and Control,* **58,** 137-156 (1983).

[4] K. Mehlhorn and F. P. Preparata: "Area-Time Optimal Division for $T=\Omega((\log n)^{1+\epsilon})$", *Inform. and Comput.* **72,** 270-282 (1987).

[5] K. Mehlhorn: "AT^2-Optimal VLSI Integer Division and Integer Square Rooting", *Integration,* **2,** 163-167 (1984).

[6] R. P. Brent: "Multiple-Precision Zero-Finding Methods and the Complexity of Elementary Function Evaluation", *Proc. Symp. on Analytic Computational Complexity,* J.F.Traub, Ed., Academic Press, New York, 151-176 (1976).

[7] Y. Okabe, N. Takagi and S. Yajima: "Log Depth Circuits for Elementary Functions Using Residue Number System", *1987 LA Symposium in Summer,* (July 1987), in Japanese.

[8] P. W. Beame, S. A. Cook. and H. J. Hoover: "Log Depth Circuits for Division and Related Problems", *SIAM J. Comput.,* **15-4,** 994-1003 (1986).