LOGARITHMIC DEPTH CIRCUITS FOR ALGEBRAIC FUNCTIONS*

JOHN H. REIF†

Abstract. This paper describes circuits for computation of a large class of algebraic functions on polynomials, power series, and integers, for which, it has been a long standing open problem to compute in depth less than $\Omega(\log n)^2$.

Algebraic circuits assume unit cost for elemental addition and multiplication. This paper describes $O(\log n)$ depth algebraic circuits which given as input the coefficients of n degree polynomials (over an appropriate ring), compute the product of $n^{O(1)}$ polynomials, the symmetric functions, as well as division and interpolation of real polynomials. Also described are $O(\log n)$ depth algebraic circuits which are given as input the first n coefficients of a power series (over an appropriate ring) compute the product of $n^{O(1)}$ power series, as well as division, reciprocal and reversion of real power series.

Furthermore this paper describes boolean circuits of depth $O(\log n(\log\log n))$ which, given n-bit binary numbers, compute the product of n numbers and integer division. As corollaries, we get boolean circuits of the same depth for evaluating, within accuracy 2^{-n}, polynomials, power series, and elementary functions such as (fixed) powers, roots, exponentiations, logarithm, sine and cosine.

All these circuits have constant indegree, polynomial size, and may be uniformly constructed by a deterministic Turing machine with space $O(\log n)$.

Key words. algebraic computation, circuit depth, parallel computation, integer division, powering

1. Introduction. Much research is now done on parallel algorithms, although in fact at this time most current computers contain only a single processor. However, these computers do use parallel circuits to implement the most basic and often repeated operations, such as the arithmetic operations: addition, subtraction, multiplication and division. These operations are generally applied to integers with an n bit binary representation, and to floating point reals with relative accuracy 2^{-n}. Other frequently used repeated operations, which certainly would merit special purpose circuits, are the elementary functions such as sine, cosine, arctangent, exponentation, logarithm, square roots, and fixed powers. For practical reasons we require circuits of constant indegree which can be uniformly constructed within $O(\log n)$ deterministic space (and thus deterministic polynomial time).

The *depth* of a circuit is the time for its parallel execution. What is the minimum depth of boolean circuits for these arithmetic operations and elementary functions?

For integer addition, Ofman [62], Krapchenko [67] and Ladner and Fischer, [80] give boolean circuits of depth $O(\log n)$ and size $O(n)$. Subtraction circuits with the same asymptotic depth and size can easily be gotten from these addition circuits. Also Reif [83] has recently given linear size, constant indegree boolean circuits of depth $O(\log\log n)$ for addition and subtraction of random numbers with error probability at most n^{-c}.

For integer multiplication, Ofman [62] and Wallace [64] give boolean circuits of depth $O(\log n)$, and Schönhage and Strassen [71] also achieve depth $O(\log n)$ with simultaneous size $O(n(\log n)\log\log n)$.

The problem of computing division or the elementary functions in better then depth $\Omega(\log n)^2$ has been open for at least 17 years since S. Cook's Ph.D. thesis (Cook [66]) (also see Borodin and Munro [75], and Savage [76]). Wallace [64] first gave a

---

supported in part by the National Science Foundation under grant NSF-MCS82-00269 and the Office of Naval Research under contract N00014-80-C-0647.

† Aiken Computation Laboratory, Division of Applied Sciences, Harvard University, Cambridge, Massachusetts 02138.

Preprint of paper appearing in *SIAM Journal on Computing*, Vol. 15, No. 1, February 1986, pp. 231-242.

division circuit with depth $\Omega(\log n)^2$. Subsequently, Anderson et al. [67] gave a division circuit of the same depth which was implemented by them on the IBM/360 Model 91 Floating-Point Execution Unit. Knuth [81] and Aho, Hopcroft and Ullman [74] described a division circuit attributed to Steve Cook of depth $(\log n)^2$ and size $O(n \log n \log \log n)$. The best known boolean circuit depth for the elementary functions was $\Omega(\log n)^2$ (Brent [76], Kung [76]). Many of the above mentioned boolean circuits of depth $\Omega(\log n)^2$ for division and elementary functions use a second order Newton iteration with $\Omega(\log n)$ steps, each requiring an n-bit integer multiplication with $\Omega(\log n)$ depth. Alternatively, a reduction is often made to the problem of computing the mth power of a n-bit integer modulo 2^n+1 for $m = O(n)$. This can be computed by $\Omega(\log n)$ steps of repeated squaring, where each square computation requires $\Omega(\log n)$ depth.

By new techniques we achieve depth less than $(\log n)^2$. An essential technique in the construction of our circuits is the use of convolutions, which can be computed in boolean depth $O(\log n)$ by the fast Fourier transforms. This technique was first introduced by Schönhage and Strassen [71] for the multiplication of two integers. Our innovation was to generalize the convolution technique to products of more than two terms.

Section 2 introduces the appropriate mathematical groundwork for the generalized polynomial convolution techniques which we utilize. Also in § 2 we give $O(\log n)$ depth algebraic circuits for various polynomial and power series operations. These algebraic circuits are interesting in the theoretical context of parallel algebraic computation, where arithmetic operations are assumed to be of unit cost.

The last part of this paper is concerned with the possibly more practical construction of boolean circuits, which originally motivated this work. In § 3 we give uniform boolean circuits of nearly logarithmic depth for the problem computing the product of $n^{O(1)}$ integers modulo (2^n+1). In an earlier version of this paper (Reif [83]) we proved our boolean circuits had depth $O(\log n(\log \log n)^2)$. This draft includes an improvement to our construction due to Beame, Cook, and Hoover [84a, b] which reduces the depth by a factor of $\log \log n$ to $O(\log n(\log \log n))$ and gives simultaneously polynomial size. These results imply uniform boolean circuits of depth $O(\log n(\log \log n))$ for the problems of division and computing elementary functions, among others.

This also implies sequential space complexity upper bounds for these and related problems. In particular, Borodin [77] proved that if a function f is computed in uniform boolean circuit depth $D(n) \geqq \log n$, then f can be computed by a deterministic Turing machine with space $D(n)$. Thus for example, division, the elementary functions and the first n bits of π can be computed by deterministic space $O(\log n(\log \log n))$.

2. Circuits for polynomial and power series computations. Our basic techniques are best understood first in the simpler context of polynomials and power series. In fact, this context is interesting in itself. We might envision a special purpose computer designed for algebraic computation. Its data are (coefficients of) polynomials and power series. The arithmetic operations including division of polynomials and power series are elementary operations of our "algebraic computer." Also, frequently applied operations are the composition of power series, revision of a power series, computation of elementary functions applied to power series, and interpolation of polynomials. We give in this section circuits of depth $O(\log n)$ for all these polynomial and power series operations, where each gate of the circuits computes an addition, multiplication, or a division of two elements of the domain.

2.0. Circuit definitions. A *circuit* α_N over a commutative ring $\mathcal{R}=(\mathcal{D},+,\cdot,0,1)$ is an acyclic labeled digraph, with

(i) a list of N distinguished *input nodes* that have no entering edges;
(ii) *constant nodes* with indegree 0 and labeled with constants in $\mathcal{D}$;
(iii) *internal nodes* with indegree two and labelled with the symbols in {"+", "·"};
(iv) a list of N' distinguished *output nodes*.

Given an assignment of the input nodes from domain $\mathcal{D}$, the value of the circuit at the output nodes is gotten by evaluation of the gates in topological order. The circuit α_N thus defines a mapping from $\mathcal{D}^N$ to $\mathcal{D}^{N'}$. A circuit α_N over a field is similarly defined, except the internal nodes can also compute division. Since division may yield an undefined value, a circuit over a field defines in general a partial mapping of inputs to outputs.

Let $\mathcal{R}[x]$ be the polynomials over commutative ring $\mathcal{R}$. Let $\mathcal{R}[[x]]$ be the power series over $\mathcal{R}$.

Let f be a partial function of (the coefficients of) m polynomials $p_1(x),\cdots,p_m(x)$ in $\mathcal{R}[x]$ of degree $n-1$. A circuit α_N for f has $N=mn$ inputs, namely the list of N coefficients in $\mathcal{D}$ of the given polynomials. The output nodes of α_N give the list of coefficients of $f(p_1(x),\cdots,p_m(x))$. If on the other hand f is a function of m power series $p_1(x),\cdots,p_m(x)$ in $\mathcal{R}[[x]]$ each with n given low order coefficients, then the circuit α_N for f also has $N=nm$ inputs, and the outputs nodes of α_N only give some prescribed finite number of the coefficients of (the possibly infinite) power series $f(p_1(x),\cdots,p_m(x))$.

The *depth* of circuit α_N is the length of its longest path. A partial function f over polynomials or power series in $\mathcal{R}$ has simultaneous *depth* $O(D(N))$ and *size* $O(S(N))$ if there exists an infinite family of circuits $\alpha_1,\cdots,\alpha_N,\cdots$ and constants $c_1, c_2 \geqq 1$ such that $\forall N \geqq 1$, α_N has depth not more than $c_1 D(N)$ and size not more than $c_2 S(N)$ and given N input coefficients of the input polynomial or power series, α_N computes f within the prescribed number of coefficients.

Let $\alpha_1, \alpha_2, \cdots$ be a family of circuits over $\mathcal{R}=(\mathcal{D},+,\cdot,0,1)$ where $\mathcal{D}$ is countable. Fix some enumeration $c_1, c_2, \cdots$ of the constants in $\mathcal{D}$. We assume each circuit α_N is encoded by a binary string where the binary representation of i is used to represent each constant symbol c_i labeling a node in α_N. (Thus, for example, the Nth root of unity, if it exists, might be represented by a binary string of length $\log N$.) The circuit family $\alpha_1,\cdots,\alpha_N,\cdots$ is *uniform* in the sense of Borodin [77] if there exists a logarithmic space deterministic Turing machine which given any $N>0$ in unary outputs for the binary encoding of α_N. All the circuits considered in this paper are uniform in this sense.

2.1. The discrete Fourier transform. Fix a commutative ring $\mathcal{R}=(\mathcal{D},+,\cdot,0,1)$. We assume ω is a principle Nth root of unity in $\mathcal{R}$ and that N has a multiplicative inverse. (For example, $e^{2\pi\sqrt{-1/N}}$ is a principle Nth root of unity in the complex numbers.) Given a vector $a \in \mathcal{D}^N$, the Discrete Fourier Transform is

$$\mathrm{DFT}_N(a)=Aa$$

where $A_{ij}=\omega^{ij}$ for $0\leqq i, j<N$. Then A^{-1} exists (Aho, Hopcroft and Ullman [74, p. 253]), where $A_{ij}^{-1}=(1/N)\omega^{-ij}$. The inverse Discrete Fourier Transform is $\mathrm{DFT}_N^{-1}(a)=A^{-1}a$ and obviously satisfies $\mathrm{DFT}_N^{-1}(\mathrm{DFT}_N(a))=a$. (*Note*: given a vector $a\in\mathcal{D}^n$, where $n<N$, $\mathrm{DFT}_N(a)$ we be defined to be $\mathrm{DFT}_N(a^+)$ where a^+ is the vector of length N derived by concatenating a with $N-n$ zeros.) Cooley and Tukey [65] gave the Fast Fourier Transform for which

THEOREM 2.1. *DFT_N and DFT_N^{-1} over $\mathscr{R}$ have simultaneous depth $O(\log N)$ and size $O(N \log N)$.*

Note: the assumption of the nth root of unity is not really essential to our techniques, since in general, our techniques will be applicable whenever a $O(\log n)$ depth circuit exist for the Discrete Fourier Transform. For example, Theorem 2.1 obviously applies to the complex numbers, and since the field operations over complex numbers can be simulated over the reals with only a factor of two depth increase, Theorem 2.1 also applies to the reals.

2.2. Products of polynomials. Suppose we are given m vectors $a_i \in \mathscr{D}^n$ for $i = 1, \cdots, m$. Each vector $a_i = (a_{i,0}, \cdots, a_{i,n-1})^T$ gives the coefficients of a $n-1$ degree polynomial $A_i(x) = \sum_{j=0}^{n-1} a_{i,j} x^j$ in $\mathscr{R}[x]$. Let $N = nm$. We wish to compute the product polynomial $B(x) = \sum_{k=0}^{N-1} b_k x^k$, where $B(x) = \prod_{i=1}^m A_i(x)$. (Note that we have $b_k = 0$ for $N - m + 1 \leqq k \leqq N - 1$.)

In the special case $m = 2$ and $N = 2n$, the convolution vector $b = (b_0, \cdots, b_{N-1})^T = a_1 \circledast a_2$ gives the coefficients of $B(x)$. By the Convolution Theorem:

$$a_1 \circledast a_2 = \mathrm{DFT}_N^{-1} (\mathrm{DFT}_N (a_i) \cdot \mathrm{DFT}_N (a_2)) \text{ where } \cdot \text{ denotes pairwise product.}$$

Hence the well-known result:

THEOREM 2.2. *The product of two polynomials in $\mathscr{R}[x]$ of degree $n-1$ has simultaneous depth $O(\log n)$ and size $O(n \log n)$.*

In the case of general $m \geqq 2$, we wish to compute the coefficient vector

$$b = (b_0, \cdots, b_{N-1})^T = a_1 \circledast \cdots \circledast a_m.$$

By repeated application of the Convolution Theorem we get

LEMMA 2.1. $b = \mathrm{DFT}_N^{-1} (\mathrm{DFT}_N (a_1) \cdots \mathrm{DFT}_N (a_m))$.

First in parallel for $i = 1, \cdots, m$ compute $f_i = \mathrm{DFT}_N (a_i)$, where $f_i = (f_{i,0}, \cdots, f_{i,N-1})^T$. Next we compute in parallel for $j = 10, \cdots, N-1$ the elementary products $F_j = \prod_{i=1}^m f_{i,j}$. Finally, we compute $\mathrm{DFT}_N^{-1} ((F_0, \cdots, F_{N-1})^T)$. Since the computation of DFT_N, DFT_N^{-1} and the required products F_j, each have depth $O(\log N)$, we have:

THEOREM 2.3. *The product of m polynomials in $\mathscr{R}[x]$ of degree $n-1$ has depth $O(\log (nm))$.*

Note that in contrast, the naive method of repeated producting by Theorem 2.2 has depth $(\log (m) \log (n))$. Also note that since Theorem 2.1 applies to the real polynomials so do Theorems 2.2 and 2.3.

2.3. Modular products of polynomials. Let $B(x) = \prod_{i=1}^m a_i(x)$ be the product polynomial considered in the previous section. Here we consider the computation of the *modular product* $D(x) = \sum_{i=0}^{n-1} d_i x^i$ where $D(x) \equiv B(x) \bmod (x^n + 1)$.

LEMMA 2.2. *The coefficients of $D(x)$ are $d_i = \sum_{r=0}^{m-1} (-1)^r b_{nr+i}$ for $i = 0, \cdots, n-1$.*

Proof.

$$B(x) = \sum_{j=0}^{N-1} b_j x^j = \sum_{r=0}^{m-1} \sum_{i=0}^{N-1} b_{nr+i} x^{nr+i}$$

$$\equiv \sum_{r=0}^{m-1} (-1)^r b_{nr+i} x^i \bmod (x^n + 1)$$

since $(-1)^r \equiv x^{nr} \bmod (x^n + 1)$.

We assume ω is a principal nth root of unity in $\mathscr{R}$ and n has a multiplicative inverse. We also assume there exists an $\psi \in \mathscr{D}$ such that $\psi^2 = \omega$ and $\psi^n = -1$. Let

$\hat{a}_i = (a_{i,0}, \psi a_{i,1}, \cdots, \psi^{n-1} a_{i,n-1})^T$. The *negatively wrapped convolution* of $a_1, \cdots, a_m$ is

$$\hat{d} = (d_0, \psi d_1, \cdots, \psi^{n-1} d_{n-1})^T.$$

LEMMA 2.3. $\hat{d} = \mathrm{DFT}_n^{-1}(\mathrm{DFT}_n(\hat{a}_1) \cdots \mathrm{DFT}_n(\hat{a}_m))$.

Proof. For $i = 1, \cdots, m$ let $\mathrm{DFT}_n(\hat{a}_i) = (g_{i,0}, \cdots, g_{i,n-1})^T$ where

$$g_{i,k} = \sum_{j=0}^{n-1} a_{i,j} \psi^j \omega^{jk}$$

for $k = 0, \cdots, n-1$. Let

$$e_k = \left(\prod_{i=1}^m g_{i,k}\right) = \sum_{0 \leq j_1, \cdots, j_m < n} \psi^{\Sigma j_i} \omega^{k(\Sigma j_i)} \left(\prod_{i=1}^m a_{i,j_i}\right).$$

Now let $\mathrm{DFT}_n(\hat{d}) = (e'_0, \cdots, e'_{n-1})^T$. Then for $k = 0, \cdots, n-1$ we get

$$\begin{aligned} e'_k &= \sum_{h=0}^{n-1} d_h \psi^h \omega^{kh} \\ &= \sum_{h=0}^{n-1} \sum_{r=0}^{m-1} \psi^h \omega^{kh} (-1)^r b_{nr+h} \quad \text{by Lemma 2.2} \\ &= \sum_{h=0}^{n-1} \sum_{r=0}^{m-1} \psi^h \omega^{kh} (-1)^r \sum_{\substack{0 \leq j_1, \cdots, j_m < n \\ nr+h = \Sigma j_i}} \prod_{i=1}^m a_{i,j_i}. \end{aligned}$$

But if we substitute $h = (\sum_{i=1}^m j_i) - nr$ into the above expansion, we get

$$\psi^h \omega^{kh} (-1)^r = \psi^{\Sigma j_i} \omega^{k(\Sigma j_i)}$$

since $\psi^{nr} = (-1)^r$ and $\omega^n = 1$. Hence $e'_k = e_k$.

The above Lemmas 2.2, 2.3 and Theorem 2.1 imply:

THEOREM 2.4. *The modular product $(A_1(x) \cdots A_m(x)) \bmod (x^n + 1)$ of polynomials $A_1(x), \cdots, A_m(x)$ in $\mathcal{R}[x]$ of degree $n-1$ has simultaneous depth $O(\log(nm))$ and size $O(nm \log(nm))$. The modular power $A(x)^m \bmod (x^n + 1)$ of a single polynomial $A(x)$ of degree $n-1$ has simultaneous depth $O(\log(nm))$ and size $O(n \log(nm))$.*

2.4. Elementary functions of power series. An immediate consequence of Theorem 2.3 is

COROLLARY 2.1. *The composition of two power series in $\mathcal{R}[[x]]$ has depth $O(\log n)$.*

The elementary functions $\exp(x)$, $\log(x)$, $\sin(x)$, $\cos(x)$, $\arctan(x)$, and square root (x), etc. all have known Taylor series expansions convergent over given intervals. Thus by Corollary 2.1 we have

COROLLARY 2.2. *The elementary functions on $\mathcal{R}[[x]]$ have depth $O(\log n)$.*

For some given $x_1, \cdots, x_N \in \mathcal{D}^N$ it is frequently useful in algebraic computations to determine the polynomial $\prod_{i=1}^N (y - x_i) = \sum_{j=0}^N (-1)^j p_j y^j$ whose coefficients $p_j = \sum_{i_1 < i_2 < \cdots < i_j} x_{i_1} \cdots x_{i_j}$ are the elementary symmetric functions. It was pointed out to us by Les Valiant that Theorem 2.3 immediately implies

COROLLARY 2.3. *The elementary symmetric functions in $\mathcal{R}[[x]]$ have depth $O(\log N)$.*

2.5. Division, interpolation and reversion. Let $A(z) = \sum_{i=0}^{n-1} a_i z^i$ be a real power series where $a_0 = 1$. The reciprocal of $A(z)$ is the power series $I(z) = \sum_{i=0}^{\infty} r_i z^i$ such that $A(z) \cdot I(z) = 1$. $I(z)$ has the infinite series expansion

$$I(z) = \sum_{i=0}^{\infty} (1 - A(z))^i.$$

We wish to compute the first n coefficients of $I(z)$. Since $I(z)=\sum_{i=0}^{n-1}(1-A(z))^i+O(z^n)$, we have by Theorem 2.3:

COROLLARY 2.4. *The first n terms of the reciprocal of a real power series and the division of two real power series can be computed in depth* $O(\log n)$.

An alternative method using the lemma below results in a circuit of depth $O(\log n)$ with smaller circuit size.

LEMMA 2.4. *If*

$$\tilde{I}(z)=\prod_{i=0}^{\lceil\log(n+1)\rceil-1}(1+(1-A(z))^{2^i})$$

then $I(z)-\tilde{I}(z)=O(z^{n+1})$.

Proof. Let $B(z)=1-A(z)$. Then $A(z)\tilde{I}(z)=(1-B(z))\tilde{I}(z)=1-B(z)^{\tilde{n}}=1-(1-A(z))^{\tilde{n}}$, where $\tilde{n}=2^{\lceil\log(n+1)\rceil}$. So

$$I(z)-\tilde{I}(z)=\frac{(1-A(z))^{\tilde{n}}}{A(z)}=O(z^{\tilde{n}})=O(z^{n+1}). \qquad \square$$

COROLLARY 2.5. *Given real polynomials* $a(x)$, $b(x)$ *of degree at most n, we can compute in depth* $O(\log n)$ *the unique polynomials* $q(x)$, $r(x)$ *such that* $a(x)=q(x)b(x)+r(x)$ *and degree* $(r(x))<$ degree $(b(x))$.

Proof. (Also, see Knuth [81].) Let $n_1=$ degree $(a(x))$ and $n_2=$ degree $(b(x))$. The computation is trivial unless $n_1\geqq n_2\geqq 1$. Then

$$A(z)=Q(z)B(z)+z^{n_1-n_2+1}R(z)$$

where

$$A(z)=z^{n_1}a\left(\frac{1}{z}\right),\quad B(z)=z^{n_2}b\left(\frac{1}{z}\right),\quad Q(z)=z^{n_1-n_2}q\left(\frac{1}{z}\right)$$

and

$$R(z)=z^{n_2-1}r\left(\frac{1}{z}\right).$$

Thus to compute the coefficients of $q(x)$, $r(x)$ we compute the first n_1-n_2+1 coefficients of $A(z)/B(z)=Q(z)+o(z^{n_1-n_2+1})$, then compute the power series $A(z)-B(z)Q(z)=z^{n_1-n_2+1}R(z)$, and finally output the coefficients of $Q(z)$, $R(z)$. $\square$

COROLLARY 2.6. *Interpolation of a real polynomial has depth* $O(\log n)$.

Proof. Suppose we are given real polynomials $p_1(x),\cdots,p_m(x)$ each of degree $n-1$, and real polynomials $q_1(x),\cdots,q_m(x)$ where degree $(q_i(x))<$ degree $(p_i(x))$ for $i=1,\cdots,n$. Let $P(x)=\prod_{i=1}^m p_i(x)$. The Chinese Remainder Theorem states that there is a unique polynomial $Q(x)$ of degree less than that of $P(x)$ such that $Q(x)\equiv q_i(x) \bmod p_i(x)$ for $i=1,\cdots,m$, coprimality of the p_i assumed.

The Lagrangian interpolation formula gives

$$Q(x)\equiv\sum_{i=0}^{m}q_i(x)r_i(x)s_i(x) \bmod P(x)$$

where $s_i(x)=P(x)/p_i(x)$ and $r_i(x)$ is the multiplicative inverse of $s_i(x) \bmod p_i(x)$.

Theorem 2.2 and Corollary 2.5 imply that preconditioned Chinese remaindering, with the $r_1(x),\cdots,r_m(x)$ also given, has depth $O(\log n)$.

However, in the special case $p_i(x)=x-a_i$ for $i=1,\cdots,m$, where the a_i are distinct then each $r_i(x)=1/s_i(x)$ can be computed in parallel by Theorem 3.3 and Corollary

2.5 in depth $O(\log n)$. In this case the $q_i(x) = b_i$ are constants, since they must have degree less than the $p_i(x)$.

Further note that in this case $Q(x)$ is the unique polynomial such that $Q(a_i) = b_i$ for $i = 1, \cdots, m$. Thus we have proved Corollary 2.6. □

We now show that Theorem 2.3 and Corollary 2.4 imply:

COROLLARY 2.7. *The reversion of a real power series has depth* $O(\log n)$.

Proof. Let $A(x) = \sum_{i=0}^{\infty} a_i x^i$ be a real power series where $a_0 = 0$ and $a_1 = 1$. The *reversion* of $A(x)$ is the power series $R(z) = \sum_{k=0}^{\infty} r_k z^k$ where $z = A(x)$ iff $x = R(z)$. Note that $r_0 = 0$ and $r_1 = 1$. For the kth coefficient, we first compute

$$B(x) = \frac{1}{A(x)^k} = \sum_{i=0}^{\infty} b_{i+1} x^i,$$

and then apply Lagrange's reversion formula (Lagrange, [1768]) $r_k = b_{k-1}/k$ for $k \geqq 2$. Thus Theorem 3.3 implies Corollary 2.7. □

3. Integer computations.

3.0. Boolean circuits. We consider computations over integers given as n bit binary numbers, and reals over $[0, 1]$ given within accuracy 2^{-n}. Our computational model in this section is the *boolean circuit*, defined as usual. The ith input node of α_n takes the ith bit of the encoding of the input integer or real. Each gate of α_n computes a boolean operation $\vee$, $\wedge$, or $\neg$. Each output node provides a bit of the encoding of the computed integer or real. (In the case of reals with floating point representation, we only provide the input and output bits up to some finite prescribed accuracy.)

3.1. The DFT over an integer ring. We assume n and ω are positive powers of two. Let $p = \omega^{n/2} + 1$ and let $\mathbb{Z}_p$ be the ring of integers modulo p.

PROPOSITION 3.1. *In* $\mathbb{Z}_p$, ω *is a principal nth root of unity and n has a multiplicative inverse* modulo p.

Proposition 3.1 implies that DFT_n and DFT_n^{-1} are well defined.

The fast Fourier transform computation of Cooley and Tukey [65] yields an arithmetic circuit α_n of depth $O(\log n)$ and size $O(n \log n)$ computing DFT_n whose elements require:

(i) addition of two $\lceil \log (p) \rceil$-bit integers.

(ii) multiplication of a $\lceil \log (p) \rceil$-bit integer by a power of ω.

We wish to expand α_n into a boolean circuit. Since ω is a power of two, the multiplications can be implemented by the appropriate bit shifts (i.e., the gate connections are shifted by the appropriate amount). The additions can be implemented by Carry-Save Add circuitry of Ofman [62] and Wallace [64] (also see Savage [76]) yielding a boolean circuit of depth $O(\log (n \log p))$ and size $O(n \log p \log (n \log p))$. Thus we have

THEOREM 3.1. DFT_n *and* DFT_n^{-1} *over the ring* $\mathbb{Z}_p$ *have simultaneous boolean depth* $O(\log (n \log p)) = O(\log n)$ *and size* $O(n \log p \log (n \log p))$.

3.2. Products of integers. Schönhage, Strassen [71] have shown:

THEOREM 3.2. *The product of two N-bit integers has simultaneous boolean depth* $O(\log N)$ *and size* $O(N \log N \log \log N)$.

We now prove that for N a power of two, the modulo $2^N + 1$ product of m integers, each of N-bits has boolean depth $O(\log (Nm) \log \log N)$. (Note that the naive method of repeated squaring by Theorem 3.2 results in a boolean circuit of depth $\Omega(\log (m) \log N)$.) We begin with a key lemma which reduces the number of bits of the integers to be produced.

LEMMA 3.1. (DFT reduction). *For N a power of two, mN sufficiently large, and any $m < N^{1/2}$ the product modulo (2^N+1) of m integers each N bits long can be computed in $O(\log mN)$ additional boolean depth and $(mN)^{O(1)}$ additional gates after computing $n = O(mN)^{1/2}$ products modulo (2^n+1) each of m integers each n bits long.* □

Proof. Let $a_1, \cdots, a_m$ be a list of N-bit numbers. We wish to compute $b = \prod_{i=1}^m a_i \bmod (2^N+1)$.

(1) Since $N = 2^u$ for some integer u, we can block each N-bit a_i into n (n a power of 2) chunks $a_{i,0}, \cdots, a_{i,n-1}$ of $h = N/n$ bits each so that

$$a_i = \sum_{j=0}^{n-1} a_{i,j} 2^{hj}$$

where $0 \leqq a_{i,j} < 2^h$. Define the associated polynomial

$$A_i(x) = \sum_{j=0}^{n-1} a_{i,j} x^j$$

and observe $a_i = A_i(2^h)$.

(2) We intend to take DFT_n with $\omega = 4$, $\psi = 2$ and $p = \omega^{n/2} + 1 = 2^n + 1$. Associate with each a_i a coefficient vector $\hat{a}_i$ defined by

$$\hat{a}_i \equiv (a_{i,0}, \psi a_{i,1}, \cdots, \psi^{n-1} a_{i,n-1})^T.$$

(3) Compute in parallel

$$\mathrm{DFT}_n(\hat{a}_i) = \hat{g}_i \equiv (g_{i,0}, \cdots, g_{i,n-1})^T.$$

(4) Compute product

$$e_k \equiv \prod_{i=1}^m g_{i,k} \bmod p.$$

(5) Compute

$$\mathrm{DFT}_n^{-1}((e_0, \cdots, e_{n-1})^T) = \hat{b} \equiv (b_0, \psi b_1, \cdots, \psi^{n-1} b_{n-1})^T$$

to obtain the coefficients of the product polynomial

$$B(x) = \sum_{j=0}^{n-1} b_j x^j$$

where by Lemma 2.3, $b = B(2^h)$.

(6) Evaluate $B(2^h)$ to get b.

Since ψ is a power of two, we can easily extract each b_j from $\psi^{j-1} b_j$ by bit shifting. By Theorem 3.1, the DFT_n and DFT_n^{-1} computations have depth $O(\log n)$. Thus all of these computations have depth $O(\log N + \log h + \log n) = O(\log mN)$ except possibly computing the e_k modular product in step (4). Note that we can use the identity $2^n x \equiv (2^n + 1 - x) \bmod 2^n + 1$ to simplify the computation of e_k to the product of at most m numbers, each of n bits. Thus the depth $D(m, N)$ of the resulting circuit satisfies

$$D(m, N) = D(m, n) + O(\log mN).$$

The reduction is correct if n is a power of two and furthermore the coefficients of $B(x)$ are small, that is if $|b_i| < p/2$. Applying Proposition 3.2, we can ensure $|b_i| < 2^{n-1}$ by having $N \geqq 16$, $m \leqq N^{1/2}$ and choosing n to be the largest power of 2 less than $16(mN)^{1/2}$. □

PROPOSITION 3.2. *For each $j = 0, \cdots, n-1$, the magnitude of the coefficients of $B(x)$ is given by $|b_i| < 2^{2m(h+1+\log n)}$.*

Proof. Let $f(i)$ be the maximum magnitude of any coefficient of a polynomial resulting from a product of 2^i of the $A_j(x)$ polynomials taken mod (x^n+1). Clearly $f(0)\leqq 2^h$ and $f(i)\leqq 2nf(i-1)^2$ for $i>0$. The general solution of the recurrence $S_{i+1}=cS_i^2$ is $S_i=c^{2^i-1}S_0^{2^i}$. Setting $S_0=2^h$ and $c=2n$, we have $f(i)\leqq(2n)^{2^i-1}2^{2^ih}\leqq 2^{h2^i+2^i-1+(2^i+1)\log n}$. Hence,

$$f(\lceil \log m\rceil)\leqq s^{2m(h+1)-1+(2m-1)\log n}\leqq 2^{2m(h+1+\log n)}. \qquad \square$$

The key idea of the Theorem 3.3 is that when $m>N^{1/8}$, the a_i are grouped into blocks of size $<m$ and the product circuit is applied to these smaller blocks, thus reducing m relative to N. When $m\leqq N^{1/8}$ our DFT reduction of Lemma 3.1 is applied to decrease N relative to m. In our original construction (Reif [83]) we required $O(\log\log N)$ applications of Lemma 3.1 to accomplish this decrease of N. Beame, Cook, and Hoover [84a] suggested an improvement which requires only a constant number of applications of our DFT reduction to appropriately reduce N. We give this improved version below, with their kind permission.

THEOREM 3.3. *For N a power of two, the product of m N-bit integers* mod 2^N+1 *has boolean depth $O(\log(m)\log\log N+\log(N))$ and size $(mN)^{O(1)}$.*

Proof. Given a list of N-bit integers $a_1,\cdots,a_m$ we compute the product $\prod_{i=1}^m a_i \bmod (2^N+1)$. Let the boolean depth and size required to compute this product be $D(m,N)$ and $S(m,N)$ respectively. Let $t(x)$ be the largest power of two less than x. Using this notation, Lemma 3.1 leads to the following recurrences

$$\text{(i)}\qquad D(m,N)\leqq D(m,t((mN)^{3/5}))+O(\log mN),$$

$$S(m,N)\leqq (mN)^{3/5}S(m,t((mN)^{3/5}))+(mN)^{O(1)}.$$

(Note: slightly tighter recurrences can be obtained from Lemma 3.1, but this does not significantly affect the asymptotic analysis.)

Reduction of m: When $m>N^{1/8}$ group the m input integers into blocks of size at most $\lceil N^{1/8}\rceil$ and compute the products for each block. Then compute the product of all the $\lceil m/N^{1/8}\rceil$ blocks. To avoid worrying about the ceiling function in describing the number of integers in each of these products, first perform a single multiplication of two integers mod 2^N+1 to reduce this number by one. Thus,

$$\text{(ii)}\qquad D(m,N)\leqq D(N^{1/8},N)+D\left(\frac{m}{N^{1/8}},N\right)+O(\log mN),$$

$$S(m,N)\leqq\left\lceil\frac{m}{N^{1/8}}\right\rceil S(N^{1/8},N)+S\left(\frac{m}{N^{1/8}},N\right)+(mN)^{O(1)}.$$

Continuing this process recursively results in an $N^{1/8}$-ary tree of multiplication nodes; so the desired product may be computed using sub-circuits which compute products of only $N^{1/8}$ integers. This tree has depth of at most $\lceil 8\log m/\log N\rceil$ and certainly has fewer than m nodes. It follows that

$$\text{(iii)}\qquad D(m,N)\leqq\left\lceil\frac{8\log m}{\log N}\right\rceil D(N^{1/8},N)+O(\log mN),$$

$$S(m,N)\leqq mS(N^{1/8},N)+(mN)^{O(1)}.$$

It is now possible to consider the problem for $m\leqq N^{1/8}$. The solution for arbitrary m clearly follows from this solution via a single application of reduction (ii). The method of attack is to use reductions (i) and (ii) alternatively to reduce the problem to a smaller one of the same type.

Reduction of N: Apply the DFT reduction (i) twice and then reduction (ii). Thus

$$\text{(iv)} \qquad D(N^{1/8}, N) \leqq D(N^{1/8}, t(N^{1/2})) + O(\log N)$$
$$\leqq 2D((N^{1/2})^{1/8}, t(N^{1/2})) + O(\log N)$$

and

$$S(N^{1/8}, N) \leqq N^{5/4} S(N^{1/8}, t(N^{1/2})) + N^{O(1)}$$
$$\leqq N^{3/2} S((N^{1/2})^{1/8}, t(N^{1/2})) + N^{O(1)}.$$

So for sufficiently large N and some fixed c, d

$$\text{(v)} \qquad D(N^{1/8}, N) \leqq 2D((N^{1/2})^{1/8}, t(N^{1/2})) + c \log N,$$
$$S(N^{1/8}, N) \leqq N^{3/2} S((N^{1/2})^{1/8}, t(N^{1/2})) + N^d.$$

The original problem of size N has been reduced to problems of size $N^{1/2}$. These reductions must be applied $\leqq \log \log N$ times until the problems are of constant size. Analysing (v) carefully by expanding out terms, we get

$$\text{(vi)} \qquad D(N^{1/8}, N) \leqq c \log N + 2c \log N^{1/2} + 2^2 c \log N^{1/4} + \cdots$$
$$+ 2^{\log \log N} c \log N^{2^{-\log \log N}},$$
$$S(N^{1/8}, N) \leqq N^d + N^{3/2+d} + N^{3/2+3/2(1/2)+d} + \cdots$$
$$+ N^{3/2+3/2(1/2)+\cdots+3/2(2^{-\log \log N})+d},$$

where the last term in each expression is the cost of the depth $\log N$ in constant size problems. These last terms are bounded by $O(\log n)$ and N^{3+d}, respectively. Summing the $\log \log N$ terms in each expression of (vi) we get

$$\text{(vii)} \qquad D(N^{1/8}, N) \leqq (c+1) \log N \log \log N,$$
$$S(N^{1/8}, N) \leqq N^{3+d} \log \log N.$$

Substituting (vii) into (iii), we get

$$\text{(viii)} \qquad D(m, N) \leqq O(\log (m) \log \log N + \log mN),$$
$$S(m, N) = (mN)^{O(1)},$$

and the theorem is proven. □

3.3. Multiprecision evaluation of polynomials and power series. Let $p(x)$ be a real polynomial or a real power series with $n-1$ given rational coefficients of magnitude $<2^n$. We wish to evaluate $p(x)$ at a floating point real x_0 within accuracy $o(2^{-n})$. Theorem 3.3 implies

COROLLARY 3.1. *The evaluation of $p(x)$ at a given x_0 to accuracy $o(2^{-n})$ has boolean depth $O(\log n(\log \log n))$ and size $n^{O(1)}$.*

The elementary functions $\exp(x)$, $\log(x)$, $\sin(x)$, $\cos(x)$, $\arctan(x)$, square root (x), etc. have Taylor series expansions convergent within accuracy $o(2^{-n})$ over fixed intervals.

COROLLARY 3.2. *The evaluation of an elementary function over a fixed interval with a Taylor series expansion convergent to accuracy $o(2^{-n})$ has boolean depth $O(\log n(\log\text{-}\log n))$ and size $n^{O(1)}$.*

COROLLARY 3.3. *The elementary symmetric functions (see § 2.4) over the reals have boolean depth $O(\log n(\log \log n))$ and size $n^{O(1)}$.*

3.4. Reciprocals and division of integers. Let a be an integer within bounds $2^{n-1} \leqq a < 2^n$. Then a has a binary representation $\sum_{i=0}^{n-1} a_i 2^i$ where $a_{n-1} = 1$. The reciprocal of a is $2^{-(n-1)} r$, where $r = \sum_{i=0}^{\infty} r_i 2^{-i}$. We wish to compute the first n bits $r_0, \cdots, r_{n-1}$. For this, we can use the product form of Anderson et al. [67] and Savage [76, p. 256].

LEMMA 3.3. *If*

$$\tilde{r} = \prod_{i=0}^{\lceil \log(n+1)-1 \rceil} (1 + (1 - 2^{-n} a)^{2^i})$$

then $|r - \tilde{r}| = o(2^{-n})$.

By Theorem 3.3 and the above lemma,

COROLLARY 3.4. *The reciprocal can be computed within accuracy $o(2^{-n})$ by a boolean circuit of depth $O(\log n(\log\log n))$ and size $n^{O(1)}$.*

COROLLARY 3.5. *Given integers a, b with binary respresentation containing n bits, we can compute in boolean depth $O(\log n(\log\log n))$ the division quotient q and remainder r integers such that $a = qb + r$ and $0 \leqq r < b$.*

Further work and open problems. A subsequent paper of Beame, Cook, and Hoover [84b] gives $O(\log n)$ depth boolean circuits for taking the product of n integers and integer division. These circuits are nonuniform, in the sense of Borodin [77] since their construction requires more than logarithmic space.

It remains an open problem to find a uniform circuit of $O(\log n)$ depth for integer division.

Also, the circuit depth complexity of the following problems remain open: given integers a, b, p such that $0 < a, b < p < 2^n$,

(1) compute $a^b \bmod p$;

(2) compute the greatest common divisor of a and b;

(3) compute the multiplicative inverse of a, for a relatively prime to p.

The obvious circuits for these problems have $\Omega(n \log n)$ depth. If we use our improved techniques for integer products described in this paper, this depth bound is reduced by a factor of $O((\log\log n)/\log n)$.

NC circuits (see Cook [81]) are uniform boolean circuits of constant degree, $n^{O(1)}$ size and $(\log n)^{O(1)}$ depth. RNC circuits are NC circuits with, in addition, a source of truly random bits. We conjecture that no RNC circuits exist for the above problems (1)-(3). Reif and Tygar [84] show that this conjecture for problem (3) would have an interesting surprising consequence, namely an efficient method for parallel pseudo random number generation. In particular, they give for any $\varepsilon > 0$ and $c \geqq 1$ a NC circuit of depth $O(\log n \log\log n)$ for generating n^c pseudo random bits from only n^ε truly random bits. They show these pseudo random bits cannot be distinguished from truly random bits by any RNC circuit, assuming there is no RNC circuit for problem (3) for infinitely many n.

Acknowledgments. I am grateful to F. Bragdon who first taught me to divide and encouraged me to experiment with faster methods for long division.

The division algorithm of S. A. Cook's Ph.D. thesis stimulated this research L. Valiant gave some useful criticism of preliminary attempts to develop my integer division circuit.

I would like to thank the referees for their useful comments which significantly improved the presentation of this paper.

Also, we would like to thank P. W. Beame, S. A. Cook, and H. J. Hoover for permission to describe their improvement to our boolean circuit for integer product.

REFERENCES

A. V. AHO, J. E. HOPCROFT AND J. D. ULLMAN, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading, MA, 1974.

S. F. ANDERSON, J. G. EARLE, R. E. GOLDSCHMIDT AND D. M. POWERS, *The IBM system 360 Model 91: floating point multiplication unit*, IBM J. Res. Dev., 11 (1967), pp. 34–53.

P. W. BEAME, S. A. COOK AND H. J. HOOVER, personal communication, February, 1984. [84a].

———, *Small depth circuits of integer products, powers, and division*, 25th Annual Symposium on Foundations of Computer Science, Singer Island, FL, 1984, pp. 1-11. [84b].

A. BORODIN, J. VON ZUR GATHEN AND J. HOPCROFT, *Fast parallel matrix and GCD computations*, 23rd Annual Symposium on Foundations of Computer Sciences, Chicago, IL, 1982, pp. 65–71.

A. BORODIN, *On relating time and space to size and depth*, this Journal, 6 (1977), pp. 733–744.

A. BORODIN AND I. MUNRO, *The Computation Complexity of Algebraic and Numeric Problems*, American Elsevier, New York, 1975, pp. 77–147.

R. P. BRENT, *Fast multiple-precision evaluation of elementary functions*, J. Assoc. Comput. Mach., 23 (1976), pp. 242–251.

S. A. COOK, Ph.D. thesis, Harvard Univ., Cambridge, MA, 1966.

———, *Towards a complexity theory of synchronous parallel computation*, Extrait de l'enseignement mathématique, T XXVII, fase 1-2, 1981.

J. W. COOLEY AND J. TUKEY, *An algorithm for the machine calculation of complex Fourier series*, Math. Comp., 19 (1965), pp. 297-301.

A. N. KRAPCHENKO, *Asymptotic estimation of addition time of a parallel adder*, Mat. Zametki, 9 (1967), pp. 35–40; Syst. Theory Res., 19 (1970), pp. 105–122.

D. E. KNUTH, *The Art of Computer Programming: Vol. II, Seminumerical Algorithms*, 2nd edition, Addison-Wesley, Reading, MA, 1981.

H. T. KUNG, *New algorithms and lower bounds for the parallel evaluation of certain rational expressions and recurrences*, J. Assoc. Comput. Mach., 23 (1976), pp. 252–261.

R. E. LADNER AND M. J. FISCHER, *Parallel prefix computation*, J. Assoc. Comput. Mach., 27 (1980), pp. 831–838.

LAGRANGE, Mémoires Acad. Royale des Sciences et Belles-Lettres de Berlin, 24 (1768), pp. 251–326.

J. D. LIPSON, *Chinese remainder and interpolation algorithms*, Proc. 2nd Symposium on Symbolic and Algebraic Manipulation, Association for Computing Machinery, New York, 1971, pp. 372–391.

Y. OFMAN, *On the algorithmic complexity of discrete functions*, Dokl. Akad. Nauk SSSR, 195 (1962), pp. 48–51; Sov. Phys. Dokl., 7 (1963), pp. 589–591.

J. POLLARD, *The fast Fourier transform in a finite field*, Math. Comp., 25 (1963), pp. 365–374.

J. H. REIF, *Probabilistic parallel prefix computation*, TR-08-83, Aiken Computation Laboratory, Harvard Univ., Cambridge, MA, 1983.

———, *Logarithmic depth circuits for algebraic functions*, 24th Annual Symposium on Foundations of Computer Science, Tucson, AZ, 1983, pp. 138–145.

J. H. REIF AND J. D. TYGAR, *Efficient parallel pseudo-random number generation*, TR-07-84, Aiken Computation Laboratory, Harvard Univ., Cambridge, MA, 1984.

J. E. SAVAGE, *The Complexity of Computing*, John Wiley, New York, 1976, pp. 237–260.

A. SCHÖNHAGE AND V. STRASSEN, *Schnelle Multiplikation grosser Zahlen*, Computing, 7 (1971), pp. 281–292.

C. S. WALLACE, *A suggestion for a fast multiplier*, IEEE Trans. Electronic Computing, EC-13 (1964), pp. 14–17.

S. WINOGRAD, *On the time to perform multiplication*, J. Assoc. Comput. Mach., 14 (1967), pp. 793–802.