SIAM J. COMPUT.
Vol. 16, No. 5, October 1987

COMPUTING POWERS IN PARALLEL*

JOACHIM VON ZUR GATHEN†

Abstract. Fast parallel computations are presented for large powers modulo an element that has only small prime factors. They work for integers and polynomials over small finite fields.

Key words. parallel processing, circuit depth, algebraic computing, symbolic manipulation, powers of integers, powers of polynomials

AMS(MOS) subject classifications. Primary 68C20; secondary 10A30, 12C05

1. Introduction. Consider the problem of computing $a^b \bmod m$ in parallel, where a, b, m are n-bit integers. This problem and related tasks come up as a subroutine in many computational problems, e.g., factoring integers, primality tests, cryptographic protocols, and factoring polynomials over finite fields. The method of "repeated squaring" does not yield fast parallel computations, i.e., of parallel time $(\log n)^{O(1)}$.

We present Boolean circuits for this problem. Provided that the modulus m has only small prime factors $p \leqq n$ ("m is n-smooth"), the depth is $O(\log^2 n \operatorname{loglog} n)$ with polynomial size for log-space uniform families, and the depth is of the optimal order $O(\log n)$ for P-uniform families.

In § 2, we use the analogue of Boolean circuits for parallel computations over fields, called arithmetic circuits. We determine exactly the parallel complexity of the powering problem over infinite fields in this model. In particular, no arithmetic circuits of poly-logarithmic depth exist for this problem. We also present the basic technique used for the fast parallel algorithms of the next sections; of course, it works only in a special case.

Section 3 deals with the case of computing $a^b \bmod m$, where m is a power of a small prime number; this is the core of the algorithm. Section 4 starts with some auxiliary parallel algorithms, for factoring numbers and Chinese remaindering, and then describes the powering algorithm. The fact that factoring can (trivially) be achieved fast in parallel puts the constraint of smoothness into perspective: it is a severe restriction, and for many applications the more interesting case is that of large prime factors. However, the present result is the first one that provides an exponential parallel speed-up of the sequential methods for modular integer powering. In particular, it disproves a conjecture in the first version of Reif [17]. A slightly different problem—computing in parallel an approximation to the high-order bits of a large power of an integer—is considered in Alt [1].

In §§ 5 and 6 we translate the approach into the setting of polynomials over finite fields. Under the corresponding constraint that m have only small irreducible factors, we get shallow Boolean circuits of polynomial size for the powering problem. The depth is $O(\log^2 n \operatorname{loglog} n)$ under log-space uniformity, and $O(\log n)$ under P-uniformity.

Subsequent to the present paper, Fich and Tompa [6] obtained a fast parallel powering algorithm in large finite fields of small characteristic. This leads to the

* Received by the editors January 4, 1985; accepted for publication (in revised form) December 15, 1986. A preliminary version of this work appeared in the Proceedings of the 25th Annual IEEE Symposium on the Foundations of Computer Science, Singer Island, Florida, 1984. Part of this work was done while the author was visiting Universität Zürich, Zürich, Switzerland. This work was supported by Schweizerischer Nationalfonds grant 2175-0.83, and by Natural Sciences and Engineering Research Council of Canada grant 3-650-126-40.

† Department of Computer Science, University of Toronto, Toronto, Ontario M5S 1A4, Canada.

surprising observation that for certain polynomial computations, Boolean circuits are exponentially more powerful than arithmetic circuits (von zur Gathen and Seroussi [10]). The problem of computing large powers discriminates between the two models, also other natural problems like testing for quadratic residuosity.

2. Powers over fields. In this section, we consider the standard model for parallel computations over fields: arithmetic circuits. It is easy to determine the parallel complexity (=depth) of the powering problem exactly over infinite fields, and almost exactly over finite fields (Theorem 2.5). In particular, in this model no computation of depth $(\log n)^{O(1)}$ for powers with an n-bit exponent is possible, except (trivially) when the field is very small, say has at most n elements. The reader interested only in the algorithms of §§ 3 through 6 may skip this section.

It is always interesting to have problems whose computational complexity we can determine exactly. The main motivation here is, however, to provide some justification for the assumption that we make later, namely that the ground field be very small.

Let F be an arbitrary field. The model of computation in this section are Strassen's [19] arithmetic circuits (or "straight-line programs"). We only need to consider special arithmetic circuits, with one input variable and one specially designated output node. (See von zur Gathen [9] for general parallel arithmetic computations.) Such an arithmetic circuit α over $F \cup \{x\}$ takes as inputs x and constants from F, performs arithmetic operations $+, -, *$, and $/$, and produces an output. At each node of the computation graph a rational function from $F(x)$ is computed. $\psi_\alpha \in F(x)$ is the rational function computed at the output node. On substituting some $a \in F$ for x, field elements are computed. We stipulate that in α no division by the rational function zero occurs. However, for certain substitutions $a \in F$, a division by the field element zero may occur.

$$R_\alpha = \{a \in F: \text{no division by zero occurs for } x \leftarrow a\}$$

is the domain of definition of α. $F \setminus R_\alpha$ is finite. If $A \subseteq F$ and $f: A \to F$ is a total function, then α computes f if and only if $\psi_\alpha(a) = f(a)$ for all $a \in A \cap R_\alpha$. If A is infinite and α computes f, then $f = \psi_\alpha \restriction A$ is a rational function. (We ignore issues of "uniformity" for arithmetic circuits, i.e., how the individual circuits α_n of a family $(\alpha_n)_{n \in \mathbb{N}}$ can be computed, given n.)

As usual, the size (=sequential time) $S(\alpha)$ of α is the total number of arithmetic operations in α, and the depth (=parallel time) $D(\alpha)$ is the length of a longest chain of subsequent operations. For a function $f: A \to F$, where $A \subseteq F$, the depth $d(f)$ is the smallest depth of arithmetic circuits α over $F \cup \{x\}$ which compute f. (If there is no α computing f, then $d(f) = \infty$.) We also denote by $d^+(f)$ the smallest depth of arithmetic circuits α over $F \cup \{x\}$ which compute f everywhere, i.e., with $A \subseteq R_\alpha$. Clearly $d \leqq d^+$, and if A is infinite, then $d = d^+$. For $A \subseteq F$ and $b \in \mathbb{N}$, let

$$\pi_A^b: A \to F, \qquad a \mapsto a^b$$

be the powering function. When F is finite and $A \subseteq F$ "large", then a computation for π_A^b is an "approximate" computation for π_F^b. For a finite field $F = GF(p^l)$, with p prime, the functions $\pi_F^{p^e}$ are important for extracting pth roots in F [10] and squarefree decomposition in $F[x]$ [7]. Well-known algorithms give the following upper bounds on the depth. (Throughout the paper "log" means $\log_2$.)

FACT 2.1. *Let F be a field, $A \subseteq F$ finite with s elements, $b, c, k \in \mathbb{N}$, $b \equiv c \bmod k$, $b \geqq 1$, $1 \leq c < k$ and $f: A \to F$ any function. Then*

(i) $d^+(f) \leqq \log s + (2 \log s)^{1/2} + 3$,

(ii) *F algebraically closed* $\Rightarrow d^+(f) \leqq \lceil \log s \rceil + 2$,

(iii) $d^+(\pi_F^b) \leq \lceil \log b \rceil$,

(iv) $A \subseteq \{a \in F: a^{k+1} = a\} \Rightarrow d^+(\pi_A^b) \leqq \lceil \log c \rceil$,
(v) $A \subseteq \{a \in F: a^k = 1\} \Rightarrow d^+(\pi_A^b) \leqq \lceil \log (k-c) \rceil + 1$.

Proof. (i) and (ii) follow from Munro and Paterson [16], applied to the polynomial in $F[x]$ interpolating f over A. The method of "repeated squaring" (see [13, § 4.6.3]) proves (iii). In (iv), we have $\pi_A^b = \pi_A^c$, and in (v), $\pi_A^b = 1/\pi_A^{k-c}$. □

An upper bound $d(f) \leqq \lceil \log s \rceil + 2$ holds in (i) for any field, with a rather unproductive circuit α computing $\prod_{a \in A} (x-a)^{-1}$, so that $A \cap R_\alpha = \emptyset$. If we allow additions for free, then Kung [14] has a surprising algorithm that uses depth 2.

To put the subsequent results into perspective, we next note some lower bounds on $d(\pi_A^b)$ and $d^+(\pi_A^b)$. The degree $\deg f$ of a rational function $f \in F(x)$ is $\max\{\deg g, 1 + \deg h\}$, if $f = g/h$ and $g, h \in F[x]$ with $\gcd(g, h) = 1$. Kung [14] proves the following lower bound for any arithmetic circuit α.

FACT 2.2. $\deg \psi_\alpha \leqq 2^{D(\alpha)}$. □

For a polynomial $f \in F[x]$ and $A \subseteq F$, we denote by $f \restriction A: A \to F$ the map induced by f. We use the monus function $a \dot{-} b = \max\{a - b, 0\}$.

LEMMA 2.3. *Let F be an arbitrary field, $A \subseteq F$ be finite with $s \geqq 2$ elements, $f \in F[x] \setminus \{0\}$ of degree $b \geqq 1$, α an arithmetic circuit over $F \cup \{x\}$ computing $f \restriction A$, and $d = D(\alpha)$.*

(i) *If $A \subseteq R_\alpha$ then $d \geqq \min\{\log b, \log (s \dot{-} b + 1)\}$.*
(ii) *If $A \cap R_\alpha = \emptyset$, then $d \geqq \log s - \operatorname{loglog} s$.*

Proof. (i) Suppose that the function computed by α is $\psi_\alpha = g_1/g_2$, with $g_1, g_2 \in F[x]$, $g_2 \neq 0$, $\gcd(g_1, g_2) = 1$, and let $h = g_1 - g_2 f$. Since $A \subseteq R_\alpha$ and $\psi_\alpha(a) = f(a)$ for all $a \in A$, we have $h(a) = 0$ for $a \in A$, and

$$\prod_{a \in A} (x-a) | h.$$

If $h = 0$, then $\deg g_1 \geqq \deg f = b$. If $h \neq 0$, then either $\deg g_1 \geqq s$ or $\deg g_2 \geqq s \dot{-} b$. The claim follows from Fact 2.2.

(ii) For a vertex v of the graph of α, let $D(v)$ denote the depth of v. Thus an input or constant vertex v has $D(v) = 0$, and the output vertex v has $D(v) = d$. For $0 \leqq i \leqq d$, let L_i' consist of the vertices v with $D(v) = i$, and $L_i \subseteq L_i'$ consist of those v at which a division is performed. For $v \in L_i$, let $\eta(v)$ be the numerator of the rational function which is the divisor at v. Then $\deg \eta(v) \leqq 2^{i-1}$ for each $v \in L_i$. We may assume that every vertex is connected to the output vertex, so that

$$\# L_i \leqq \# L_i' \leqq 2^{d-i}, \qquad \sum_{v \in L_i} \deg \eta(v) \leqq 2^{i-1} \# L_i \leqq 2^{d-1}.$$

Since $A \cap R_\alpha = \emptyset$, we have

$$\prod_{a \in A} (x-a) \Big| \prod_{\substack{1 \leqq i \leqq d \\ v \in L_i}} \eta(v), \qquad s = \# A \leqq \sum_{\substack{1 \leqq i \leqq d \\ v \in L_i}} \deg \eta(v) \leqq d 2^{d-1},$$

which implies (ii). □

In view of Fact 2.2, one might hope to improve the lower bound in (ii) to $\log s$. Somewhat surprisingly, this is not possible. The following is an example of an arithmetic circuit α over $\mathbb{C} \cup \{x\}$ of depth d and with $\# A \geqq d 2^{d-3}$, where $A = \mathbb{C} \setminus R_\alpha$.

Example 2.4. Let $d \in \mathbb{N}$, $n = 2^{d-1}$, $F = \mathbb{C}$, and $a_1, \cdots, a_n \in \mathbb{C}$ algebraically independent over $\mathbb{Q}$. Let α be a complete binary tree of depth d, with $x, a_1, x, a_2, \cdots, x, a_n$ at the leaves, and alternating levels of $+$- and $/$-nodes. For example, the following are among the functions computed at levels 1, 2 and 3, respectively:

$$x + a_1, \quad \frac{x+a_1}{x+a_2}, \quad \frac{x+a_1}{x+a_2} + \frac{x+a_3}{x+a_4}.$$

One can show that the new denominator polynomials introduced by division steps have no common zeros with the previous ones, and that the sum of their degrees is $d2^{d-3}$, which equals $\#(\mathbb{C} \setminus R_\alpha)$.

THEOREM 2.5. *Let $b \in \mathbb{N}$, $b \geq 1$, F be a field, and $F^\times = F \setminus \{0\}$.*

(i) *If F is infinite, then*

$$d(\pi_F^b) = d^+(\pi_F^b) = \lceil \log b \rceil.$$

(ii) *If $A \subseteq F$ is finite with $s \geqq 2$ elements, then*

$$\min\left\{\log b, \log\left(\frac{s}{2} \dot{-} b + 1\right), \log s - \operatorname{loglog} s - 1\right\} \leqq d(\pi_A^b) \leqq d^+(\pi_A^b) \leqq \lceil \log b \rceil,$$

$$\min\{\log b, \log(s \dot{-} b + 1)\} \leqq d^+(\pi_A^b).$$

(iii) *If F is finite with q elements, $1 \leqq b < q$ and $m = \min\{\lceil \log b \rceil, \lceil \log(q - b) \rceil\}$, then*

$$\min\left\{\log b, \log\left(\frac{q}{2} \dot{-} b + 1\right), \log q - \operatorname{loglog} q - 1\right\} \leqq d(\pi_F^b) \leqq d^+(\pi_F^b) \leqq \lceil \log b \rceil,$$

$$m \leqq d^+(\pi_{F^\times}^b) \leqq m + 1.$$

If $q - b - 1$ is a power of two or $b \leqq q/2$, then $d^+(\pi_{F^\times}^b) = m$.

Proof. (i) follows from Facts 2.1(iii) and 2.2. For (ii), let α be an arithmetic circuit computing π_A^b, and $S = A \cap R_\alpha$. If $\#S \geqq s/2$, we find from Lemma 2.3(i)

$$D(\alpha) \geqq \min\left\{\log b, \log\left(\frac{s}{2} \dot{-} b + 1\right)\right\}.$$

If $\#S < s/2$, then Lemma 2.3(ii) implies that

$$D(\alpha) \geqq \log(\#(A \setminus S)) - \operatorname{loglog}(\#(A \setminus S)) \geqq \log\frac{s}{2} - \operatorname{loglog} s.$$

In (iii), we use Lemma 2.3(i) and Fact 2.1(iii) and (v). □

Now consider the following four problems, where $u \in F$, and $a, m \in F[x]$ of degree n are inputs, and $b \in \mathbb{N}$ with $2^{n-1} < b \leqq 2^n$.

POWER$_1$: Compute u^b.

POWER$_2$: Compute $a^b \bmod m$.

POWER$_3$: Compute $a^b \bmod x^n$.

POWER$_4$: Compute $a^b \bmod x^n$, if a has constant term 1.

The coefficient vectors of a and m form the input, and one obtains a coefficient vector as output; thus we now have to allow arithmetic circuits with many inputs and outputs.

COROLLARY 2.6. *Let $1 \leqq i \leqq 3$ and α be an arithmetic circuit computing* POWER$_i$ *everywhere.*

(i) *If F is infinite, then $D(\alpha) \geqq n$.*

(ii) *If F is finite with q elements and $1 \leqq b < q$, then $D(\alpha) \geqq \min\{n, \lceil \log(q - b) \rceil\}$.*

Proof. For POWER_1, this is Theorem 2.5(iii). By using input $a \in F$, any circuit for POWER_2 or POWER_3 also solves POWER_1. □

So much for the negative results that show that computing bth powers, where b is an n-bit integer, requires depth $\Omega(n)$ on arithmetic circuits. In only one case can we hope for shallow arithmetic circuits over F, namely when F is small, say $q = \# F \leq n$, so that Corollary 2.6(ii) only gives a lower bound of order $\log n$. Subsequent to this paper, Fich and Tompa [6] gave fast parallel Boolean algorithms over nonprime large finite fields of small characteristic.

The basis for the fast parallel powering algorithms in the next sections is the following binomial expansion, which provides an algorithm for POWER_4 (where $a \equiv 1 \bmod x$):

$$a^b = (1+(a-1))^b = \sum_{0 \leq i \leq b} \binom{b}{i} (a-1)^i \equiv \sum_{0 \leq i < n} \binom{b}{i} (a-1)^i \bmod x^n.$$

PROPOSITION 2.7. POWER_4 *can be computed by an arithmetic circuit of depth* $O(\log^2 n)$ *and size* $n^{O(1)}$.

Proof. The constants $\binom{b}{i}$ can be "hard-wired" into the circuit, and one multiplication of polynomials modulo x^n can be performed in depth $O(\log n)$. □

In characteristic zero, we could use the formula $a^b = \exp(b \log a)$ to obtain the same result.

The algorithms to be discussed below use the Boolean circuit families for elementary arithmetic. Although a priori the individual circuits (one for each relevant input length) of such a family are not required to be related to each other, it is convenient to know that a description of the circuit for input size parameter n can be constructed, say, by a (deterministic) Turing machine M on input n in unary. If M is polynomial-time bounded, then the family is P-uniform, and if the stronger requirement that M be logarithmic-space bounded holds, then the family is log-space uniform. Ruzzo [18] and Cook [4] discuss uniformity and parallel Boolean computations.

We avoid reference to the various notions of uniformity by using $\text{div}(n)$ to denote some function such that there exist Boolean circuits of depth $\text{div}(n)$ and size $n^{O(1)}$ that compute the division with remainder for n-bit integers. "Long division" yields the trivial bound $\text{div}(n) = O(\log^2 n)$. By Beame, Cook and Hoover [2], $\text{div}(n) = O(\log n)$ for P-uniform circuits, and by Reif [17], $\text{div}(n) = O(\log n \log\log n)$ for log-space uniform circuits. The "iterated" product of n n-bit integers can be computed in depth $O(\text{div}(n))$ [2].

Note that the upper bounds in Theorem 2.5 are achieved by log-space uniform families while the lower bounds make no uniformity assumption at all.

The depth bound of Proposition 2.7 can be improved by considering arithmetic networks, a more general model of computation that encompasses both arithmetic circuits and Boolean circuits. The interface between the two types of circuits is provided in the one direction by testing an arithmetic value $a \stackrel{?}{=} 0$ and producing a Boolean value. In the other direction, one can select one of two arithmetic values according to the value of a Boolean variable.

Reif [17] has shown that division with remainder of polynomials of degree at most n can be calculated on arithmetic networks of depth $O(\log n)$, provided the ground field F contains the roots of unity required for a Fast Fourier Transform. Eberly [5] generalized this to arbitrary fields, by constructing "log-space uniform"

families of arithmetic networks of depth $O(\log n \log\log n)$, and "P-uniform" families of depth $O(\log n)$ for division with remainder of polynomials. (These notions of uniformity for arithmetic networks are defined in von zur Gathen [9].) If we use $\mathrm{div}_F(n)$ to denote the maximum of the Boolean $\mathrm{div}(n)$ and these arithmetic depth functions over F, under both notions of uniformity, then also the iterated product of n polynomial with degree at most n can be computed in depth $O(\mathrm{div}_F(n))$. For all families considered, the size is polynomial in n.

The Boolean component of arithmetic networks allows us to solve a slightly stronger version of POWER_4: The binary representation of b can now be considered to be part of the input. In the algorithm, even the constants $\binom{b}{i}$ need not be hard-wired, but can be computed in depth $O(\mathrm{div}_F(n))$ using arithmetic in F. This is clear if $\mathrm{char}\, F = 0$: one simply computes the iterated products for numerator and denominator of the binomial coefficients, and then their quotient. If $\mathrm{char}\, F = p > 0$, we can use Lucas' [15] formula:

$$\binom{b}{i} \equiv \binom{b_0}{i_0}\binom{b_1}{i_1}\cdots \bmod p$$

where

$$b = b_0 + b_1 p + \cdots, \qquad i = i_0 + i_1 p + \cdots$$

are the p-ary expansions of b and i, with $0 \leqq b_j, i_j < p$. Given $j \leqq n$, and p, b, we compute $b_j = \lfloor b/p^j \rfloor - p\lfloor b/p^{j+1} \rfloor$ in depth $O(\mathrm{div}(n))$. The individual binomial coefficients can now be calculated as for characteristic zero; no division by zero occurs.

COROLLARY 2.8. POWER_4 *can be computed on arithmetic networks of depth* $O(\mathrm{div}_F(n))$. □

This statement comprises two results: log-space uniform families of arithmetic networks of depth $O(\log n \log\log n)$ for POWER_4, and P-uniform families of depth $O(\log n)$.

3. Integer powers modulo a prime power. In this section, we study the base case for computing large powers of integers modulo another integer, namely with the modulus being a prime power. In the next section we will use the Chinese remainder algorithm for a more general case.

Throughout this section, n is an input size parameter, a and b in $\mathbb{N}$ are n-bit numbers (i.e. $1 \leqq a, b < 2^n$), $p \leqq n$ is a prime, and $1 \leqq e \leqq n$. We want to compute $a^b \bmod p^e$. The fast parallel algorithm we get for the computation $\bmod p^e$ contrasts with the fact that by Theorem 2.5 no arithmetic circuits over F of poly-logarithmic depth exist for the same computation over a field F with p^e elements. (However, Fich and Tompa [6] present an arithmetic network over the prime field $\mathbb{Z}_p \subseteq F$ of depth $O(\log^2 n)$ that computes a^b.)

Let us first consider the case $p \geqq 3$. We use the well-known decomposition $\mathbb{Z}_{p^e}^\times = G \times H$ (see e.g. [11, Chap. 4]), with the injective group homomorphism

$$\alpha : \mathbb{Z}_p^\times \to \mathbb{Z}_{p^e}^\times, \qquad r \bmod p \mapsto r^{p^{e-1}} \bmod p^e,$$

$$G = \mathrm{im}\,\alpha, \qquad H = \{r \bmod p^e : r \equiv 1 \bmod p\}.$$

Thus $G \cong \mathbb{Z}_p^\times$, $\# G = p - 1$, and $\# H = p^{e-1}$. We will split the problem of computing powers in $\mathbb{Z}_{p^e}^\times$ into the corresponding problems in G and H. Our first task is to compute α.

LEMMA 3.1. *Given r with $1 \leqq r < p \leqq n$ and $p \geqq 3$ prime, one can compute s such that*

$$s \equiv \alpha(r) \equiv r^{p^{e-1}} \bmod p^e$$

by a Boolean circuit of depth $O(\log n \cdot \operatorname{div}(n))$ and size $n^{O(1)}$.

Proof. Since $\# \mathbb{Z}_{p^e}^{\times} = (p-1)p^{e-1}$, we have

$$s^{p-1} \equiv 1 \bmod p^e \text{ and } s \equiv r \bmod p$$

for $s = r^{p^{e-1}}$. A quadratic Newton method for solving the first equation, with the second equation providing an initial value, would yield the following recursion:

$$s'_0 = r,$$

$$s'_i = s'_{i-1} - \frac{(s'_{i-1})^{p-1} - 1}{(p-1)(s'_{i-1})^{p-2}} \bmod p^{2^i},$$

$$1 \leqq s'_i < p^{2^i}.$$

In order to avoid divisions, we consider the following division-free variant of this Newton iteration:

$$s_0 = r,$$

$$s_i \equiv s_{i-1} + (s_{i-1}^{p-1} - 1)(1 + p + p^2 + \cdots + p^{2^i-1}) s_{i-1}^{(p-2)^2} \bmod p^{2^i},$$

$$1 \leqq s_i < p^{2^i}.$$

For completeness, we check that the formula is correct, i.e., that

$$s_i^{p-1} \equiv 1 \bmod p^{2^i},$$

by induction on i. The case $i = 0$ being clear, we set $t = s_{i-1}^{p-1}$, so that

$$\begin{aligned} s_i^{p-1} - 1 &\equiv t + (p-1) s_{i-1}^{p-2} (t-1)(1 + \cdots + p^{2^i-1}) s_{i-1}^{(p-2)^2} - 1 \\ &= (t-1)(1 + (-1 + p^{2^i}) t^{p-2}) \\ &\equiv (t-1)(1 - t^{p-2}) = (t-1)^2 (-t^{p-3} - \cdots - 1) \equiv 0 \bmod p^{2^i}. \end{aligned}$$

Set $l = \lceil \log e \rceil$. Then

$$s_l^{p-1} \equiv 1 \bmod p^e,$$

and the uniqueness of Newton iteration guarantees that

$$\alpha(r) \equiv s_l \bmod p^e.$$

The computation of s_i from s_{i-1} can be done in depth $O(\operatorname{div}(n))$, and the total depth of the resulting circuit is $O(\log n \operatorname{div}(n))$. □

Algorithm INTEGER POWERS MODULO A PRIME POWER.

Input: An input size parameter n, integers $a, b, p, e \in \mathbb{N}$ such that $p, e \leqq n$; $a, b < 2^n$, and $p \geqq 3$ is prime.

Output: $c \in \mathbb{N}$ such that $a^b \equiv c \bmod p^e$.

1. If $a = 0$, then return $c = 0$ and stop. Compute $l \in \mathbb{N}$ such that $p^l \nmid a$ and $p^{l+1} \nmid a$. If $lb \geqq e$, return $c = 0$ and stop. Otherwise replace a by a/p^l. [We can now assume $a \not\equiv 0 \bmod p$.]
2. Compute r such that

$$a \equiv r \bmod p, \qquad 1 \leqq r < p.$$

3. Compute s such that

$$s \equiv \alpha(r) \equiv r^{p^{e-1}} \bmod p^e, \qquad 1 \leqq s < p^e.$$

4. Compute h and u such that

$$h \equiv b \bmod p-1, \qquad 0 \leqq h < p-1,$$
$$u \equiv s^h \bmod p^e, \qquad 1 \leqq u < p^e.$$

5. Compute the inverse t of s such that

$$st \equiv 1 \bmod p^e, \qquad 1 \leqq t < p^e.$$

6. Compute v and w such that

$$v \equiv at \bmod p^e,$$
$$w \equiv \sum_{0 \leqq i < e} \binom{b}{i} (v-1)^i \bmod p^e,$$
$$1 \leqq v, w < p^e.$$

7. Return $c = p^{lb} uw$.

THEOREM 3.2. *Algorithm* INTEGER POWERS MODULO A PRIME POWER *works correctly as described. For input size parameter n, it can be implemented on a Boolean circuit of depth $O(\log n \operatorname{div}(n))$ and size $n^{O(1)}$.*

Proof. We can assume that $a \not\equiv 0 \bmod p$, i.e., $l = 0$. Note that $s \equiv r \equiv a \bmod p$ and $v \equiv 1 \bmod p$, and therefore

$$v^b = (1+(v-1))^b \equiv \sum_{0 \leqq i < e} \binom{b}{i} (v-1)^i \equiv w \bmod p^e.$$

Since $\# \mathbb{Z}_p^\times = p-1$, we also have

$$uw \equiv s^h v^b \equiv \alpha(r)^h v^b \equiv \alpha(r^h) v^b \equiv \alpha(r^b) v^b \equiv \alpha(r)^b v^b \equiv a^b \bmod p^e.$$

Thus the algorithm works correctly. We use a quadratic Newton iteration to compute t in step 5:

$$t_0 = r, \quad t_i \equiv t_{i-1} - (-t_{i-1} + t_{i-1}^2 s) \bmod p^{2^i}, \quad 1 \leqq t_i < p^{2^i}.$$

Each iteration step can be performed in depth $O(\operatorname{div}(n))$. This depth is also sufficient for the iterated products required for the binomial coefficients (see end of § 2) and powers of $v-1$ in step 6. Thus the depths are:

steps 1, 2, 4, 6, 7: $O(\operatorname{div}(n))$,

steps 3, 5: $O(\log n \operatorname{div}(n))$. □

By exploiting the power of $\dot{P}$-uniformity, we can actually get Boolean circuits of optimal (up to constant factors) depth.

COROLLARY 3.3. *There exists a P-uniform family of Boolean circuits of depth $O(\log n)$, whose nth circuit computes $a^b \bmod p^e$ for all $a, b, p, e \in \mathbb{N}$ with $a, b < 2^n$; $e, p \leqq n$, and $p \geqq 3$ prime.*

Proof. Since $\operatorname{div}(n) = O(\log n)$ for P-uniformity, it is sufficient to implement steps 3 and 5 of INTEGER POWERS MODULO A PRIME POWER in depth $O(\log n)$. In the circuit for input size parameter n, we hard-wire for every (r, p, e) with $1 \leqq r < p \leqq n$, $e \leqq n$, and p prime, a table of constants s_{rpe} and t_{rpe} such that

$$s_{rpe} \equiv r^{p^{e-1}} \bmod p^e, \qquad t_{rpe} \cdot s_{rpe} \equiv 1 \bmod p^e, \quad 1 \leqq s_{rpe}, t_{rpe} < p^e.$$

These are less than n^3 constants, and they can be computed in (sequential) polynomial time. Then steps 3 and 5 can be replaced by a table look-up in depth $O(\log n)$. □

Remark 3.4. For $r \equiv a \not\equiv 0 \bmod p$, we have

$$a^b \equiv r^{b-e+1} \sum_{0 \le i < e} \binom{b}{i} r^{e-i-1} (a-r)^i \bmod p^e.$$

By hard-wiring r^{2^j} for all $r < p$ and $0 \le j < n$, this formula yields another P-uniform family of Boolean circuits for our powering problem, with depth $O(\log n)$. It also works when p is 2 or not prime, so that Corollary 3.3 also holds for $p = 2$. For log-space uniformity, we obtain the following improvement.

COROLLARY 3.5. *For $p = 2$ and $p = 3$, there exist log-space uniform families of Boolean circuits of depth $O(\log n \log\log n)$ and size $n^{O(1)}$ that compute $a^b \bmod p^e$ for all $a, b < 2^n$ and $e \le n$.*

Proof. For $p = 2$, we have $r = s = t = 1$ in the algorithm. For $p = 3$, we use the residue system $\{1, -1\}$ for $\mathbb{Z}_3^\times$, so that $r = s = t \in \{1, -1\}$. □

In the first version of Reif [17], it was conjectured that no Boolean circuits of depth $(\log n)\, O(1)$ exist to compute $a^b \bmod 2^n$; this conjecture is now disproven.

4. Chinese remaindering. In order to generalize the fast parallel algorithm for powering with prime power moduli p^e to more general moduli, we use, of course, a Chinese remainder algorithm. It turns out that we first have to present Boolean circuits for some other problems, which may be of independent interest. Inputs are the various n-bit numbers denoted as $a, b, a_1, \cdots, a_k, m, m_1, \cdots, m_k$ with $k \le n$, and output an n-bit number c as below:

POWER: $c \equiv a^b \bmod m$.

GCD: $c = \gcd(a, m) \ge 1$.

MODINV: $\begin{cases} ac \equiv 1 \bmod m & \text{if } \gcd(a, m) = 1, \\ c = 0 & \text{otherwise.} \end{cases}$
("modular inverse")

LAGRANGE: $c \equiv 0 \bmod m_2 \cdots m_k$,
$c \equiv \begin{cases} 1 \bmod m_1 & \text{if } \gcd(m_1, m_2 \cdots m_k) = 1, \\ 0 \bmod m_1 & \text{otherwise.} \end{cases}$
("Lagrange interpolation coefficient"),

CHINREM: If $\gcd(m_i, m_j) = 1$ for all $i \ne j$,
then $c \equiv a_i \bmod m_i$ for all i.
Otherwise $c = 0$.
("Chinese remainder algorithm")

FACTOR: compute the prime factors of m (with multiplicities).

Using DIV for the problem of integer division with remainder, we have NC^1-reductions (see Cook [4] for terminology):

$$\text{CHINREM} \le \text{LAGRANGE} + \text{GCD} \quad \text{and} \quad \text{LAGRANGE} \le \text{MODINV} + \text{DIV}.$$

The second reduction follows by using an inverse d of $m_2 \cdots m_k \bmod m_1$; then $c = d \cdot (m_2 \cdots m_k)$ is sufficient. The iterated product $m_2 \cdots m_k$ is reducible to division (Beame, Cook and Hoover [2]).

At the present time, none of these problems is known to be in NC—i.e., solvable in depth $(\log n)^{O(1)}$—or log-space complete for P. All problems, except possibly FACTOR, are in P.

We need fast parallel algorithms for these problems in special cases. We use "SMOOTH" to denote the condition that the moduli $m, m_1, \cdots, m_k$ are n-smooth, i.e., that $p \leqq n$ for every prime factor p of m or m_i. Note that the condition depends on the parameter n; where this parameter is clear, we will not mention it explicitly. The results would also hold if we used $p \leqq n^c$ for some fixed c. The number of smooth integers is a well-studied topic in analytic number theory (see e.g. Hildebrand [12]).

THEOREM 4.1. *The following problems can be computed by Boolean circuits of polynomial size and the stated depth:*

(i) SMOOTH-FACTOR *and* SMOOTH-GCD *in depth* $O(\operatorname{div}(n))$,

(ii) SMOOTH-MODINV, SMOOTH-LAGRANGE, *and* SMOOTH-CHINREM *in depth* $O(\log n \cdot \operatorname{div}(n))$.

Proof. We leave away the "SMOOTH-". First consider the trivial algorithm for FACTOR: For each number $p \leqq n$, test whether p is prime (by a sieve method) and determine its multiplicity in m. This can be performed in depth $\operatorname{div}(n)$, by computing $p, p^2, p^3, \cdots, p^n$ and testing for each i whether p^i divides m. Now GCD is trivial.

Now consider MODINV in the special case where $m = p^e$, p prime. We have the following algorithm:

1. Test if $p|a$, and return $c = 0$ if "yes."
2. Find by exhaustive search $c_0 \in \mathbb{N}$ such that
$$1 \leqq c_0 < p, \qquad ac_0 \equiv 1 \bmod p.$$
3. Compute $c \equiv a^{-1} \bmod p^e$ by a quadratic Newton iteration as in step 5 of INTEGER POWERS MODULO A PRIME POWER.

The circuit depth for this algorithm is dominated by step 3:

$$O(\log e \cdot \operatorname{div}(n)) = O(\log n \operatorname{div}(n)).$$

Next we consider LAGRANGE. If $m_1 = p^e$ for a small prime p, then we can solve LAGRANGE in depth $O(\log n \operatorname{div}(n))$ by the reduction to MODINV+DIV. For the general case $m_1 = p_1^{e_1} \cdots p_r^{e_r}$, with small pairwise distinct primes $p_1, \cdots, p_r$ (and no p_i dividing $m_2 \cdots m_k$), we compute $c_1, \cdots, c_r$ such that

$$c_i \equiv \begin{cases} 1 \bmod p_{p_i}^{e_i}, \\ 0 \bmod p_{p_1}^{e_1} \cdots p_{i-1}^{e_{i-1}} p_{i+1}^{e_{i+1}} \cdots p_r^{e_r} m_2 \cdots m_k. \end{cases}$$

The power products can be computed in depth $O(\operatorname{div}(n))$. Each c_i is given by one of the special LAGRANGE problems already solved, and

$$c = c_1 + \cdots + c_r$$

solves the current problem. Therefore LAGRANGE and CHINREM can be solved in depth $O(\log n \operatorname{div}(n))$.

Now for the general case of MODINV, where $m = p_1^{e_1} \cdots p_r^{e_r}$ with small primes p_i, we have the algorithm:

1. For all i, $1 \leqq i \leqq r$, compute c_i such that $ac_i \equiv 1 \bmod p_i^{e_i}$,
2. Compute c such that for all $i \leqq r$, $c \equiv c_i \bmod p_i^{e_i}$.

Then

$$\forall i \leqq r \quad ac \equiv ac_i \equiv 1 \bmod p_i^{e_i} \Rightarrow ac \equiv 1 \bmod m. \qquad \square$$

We now have a fast parallel algorithm for powering.

Algorithm INTEGER POWERS.
Input: an integer n (the input size parameter), and n-bit integers a, b, m, where $b \in \mathbb{N}$, and $m \in \mathbb{N}$ is n-smooth.
Output: $c \in \mathbb{Z}$ such that $c \equiv a^b \bmod m$.
1. Call FACTOR with input m to compute $p_1, \cdots, p_r, e_1, \cdots, e_r$ such that $m = p_1^{e_1} \cdots p_r^{e_r}$ is the prime factorization of m.
2. For all i, $1 \leqq i \leqq r$, call algorithm INTEGER POWERS MODULO A PRIME POWER with input $n, a, b, p = p_i$, and $e = e_i$ to obtain output $c_i \in \mathbb{Z}$. [Then $c_i \equiv a^b \bmod p_i^{e_i}$.]
3. Call CHINREM with input $a_1, \cdots, a_r, m_1, \cdots, m_r$, where $m_i = p_i^{e_i}$, $a_i \equiv c_i \bmod m_i$, $0 \leqq a_i < m_i$, and return the output c.

THEOREM 4.2. *Algorithm* INTEGER POWERS *can be performed by a family of Boolean circuits of depth* $O(\log n \operatorname{div}(n))$ *and size* $n^{O(1)}$.

Proof. Correctness of the algorithm is clear. By Theorem 4.1, steps 1 and 3 can be performed in depth $O(\log n \operatorname{div}(n))$ and size $n^{O(1)}$, and similarly for step 2 by Theorem 3.2. □

Note that on input of arbitrary n-bit numbers a, b, m, step 1 can be used to determine whether m is n-smooth.

COROLLARY 4.3. *There exists a P-uniform family of Boolean circuits of depth* $O(\log n)$, *whose nth circuit computes* $a^b \bmod m$ *for all n-bit integers a, b, m, where m is n-smooth.*

Proof. In view of Theorem 4.1(i) and Corollary 3.3 it is sufficient to implement step 3 of algorithm INTEGER POWERS in depth $O(\log n)$. For any p, e, q with $1 \leqq p, e, q \leqq n$ and $p \neq q$ prime, we hard-wire the constant u_{peq} such that

$$u_{peq} \cdot q \equiv 1 \bmod p^e, \qquad 1 \leqq u_{peq} < p^e,$$

in the circuit for input size parameter n. This is a table of less than n^3 constants, and any u_{peq} can be looked up in depth $O(\log n)$. At a call of CHINREM with moduli $m_1 = p_1^{e_1}, \cdots, m_r = p_r^{e_r}$, with $2 \leqq p_i \leqq n$, we compute the inverses

$$v_{ij} = u_{p_i, e_i, p_j}^{e_j},$$

for $i \neq j$, so that

$$v_{ij} p_j^{e_j} \equiv 1 \bmod p_i^{e_i}.$$

This can be done in depth $O(\log n)$, and using the reduction CHINREM $\leqq$ MODINV + DIV, step 3 can be performed in depth $O(\log n)$. □

5. Polynomial powers modulo a prime power. In this and the next section, we translate the development of §§ 3 and 4 for integers into the setting of polynomials over a finite field F. The main result is a Boolean circuit of depth $O(\log^2 n \operatorname{loglog} n)$ that computes $a^b \bmod f^e$, where $a, f \in F[x]$, a has degree at most n, f is irreducible and such that $(\#F)^{\deg f} \leqq n$, b is an n-bit number, and $e \leqq n$. We also get an arithmetic circuit over F of the same depth. The above circuits are log-space uniform; we also obtain P-uniform circuit families of optimal depth $O(\log n)$.

So let F be a finite field of characteristic p and with $q = p^l$ elements. We assume that F is represented as (p, g), where $g \in \mathbb{Z}_p[t]$ is an irreducible polynomial of degree l such that $F = \mathbb{Z}_p[t]/(g)$. Elements of F are represented by polynomials from $\mathbb{Z}_p[t]$ of degree less than l, and an element of $\mathbb{Z}_p$ is given by the binary representation of an integer between 0 and $p-1$.

Now let $f \in F[x]$ be monic irreducible of degree d, $e \geqq 1$, and $R = F[x]/(f^e)$. We first consider a decomposition $R^\times = G \times H$ of the group of units of R, analogous to that for $\mathbb{Z}_{p^e}^\times$. Note that $\# R = q^{de}$ and $\# R^\times = (q^d - 1)q^{d(e-1)}$, so that we expect the "small" group G to have $q^d - 1$ elements, and the "large" H to have $q^{d(e-1)}$ elements. We consider the field $K = F[x]/(f)$, and the mapping

$$\alpha : K^\times \to R^\times, \qquad r \bmod f \mapsto r^{q^{d(e-1)}} \bmod f^e.$$

We abbreviate the exponent as $\varepsilon = q^{d(e-1)}$. If $r \in F[x]$ and $r \not\equiv 0 \bmod f$, then r^ε is a unit mod f^e. α is well defined, since $r^\varepsilon - s^\varepsilon = (r-s)^\varepsilon$ for every $r, s \in F[x]$, and $e \leqq 2^{e-1} \leqq 2^{d(e-1)} \leqq q^{d(e-1)} = \varepsilon$; thus $r \equiv s \bmod f$ implies that $r^\varepsilon \equiv s^\varepsilon \bmod f^e$. Obviously, α is a (multiplicative) group homomorphism. (It is also additive where defined.) Since

$$r^\varepsilon \equiv r \bmod f$$

for all $r \in F[x]$, α is injective. We now let

$$G = \operatorname{im} \alpha, \qquad H = \{g \bmod f^e : g \equiv 1 \bmod f\}.$$

These are two subgroups of $R^\times$, with $\# G = q^d - 1$ and $\# H = q^{d(e-1)}$. Since $G \cap H = \{1\}$, we have the desired decomposition $R^\times = G \times H$. The two groups can also be characterized as

$$G = \{u \in R^\times : u^{q^d - 1} = 1\}, \qquad H = \{u \in R^\times : u^{q^{d(e-1)}} = 1\}.$$

Just as in Lemma 3.1, we first describe a fast parallel computation of α. We again denote by n the input size parameter, and assume that $q^d \leqq n$, $de \leqq n$. The second constraint is quite natural, since the usual representatives for R have degree up to $de - 1$. We assume that there exist Boolean circuits of depth $\operatorname{div}(n)$ and size $n^{O(1)}$ that compute the division with remainder, iterated product, and inverse mod x^n for polynomials of degree at most n. Then $\operatorname{div}(n) = O(\log n)$ for P-uniform circuits, and $\operatorname{div}(n) = O(\log n \operatorname{loglog} n)$ for log-space uniform circuits, by Eberly [5]. As of now, no Boolean circuits of depth $(\log n)^{O(1)}$ are known to perform arithmetic in $\mathbb{Z}_p$ using some standard representatives if p has about n bits; inversion mod p is the problem. This can be avoided by using a redundant representation u/v for elements of $\mathbb{Z}_p$, with $u, v \in \mathbb{Z}$, $0 \leqq u, v < p$ and $v \neq 0$; the upper bounds quoted above hold for this representation. But since in our case $p \leqq n$, we can actually find the inverse of $v \bmod p$ by exhaustive search in depth $O(\log n)$.

LEMMA 5.1. *Given $r \in F[x]$ with $0 \leqq \deg r < \deg f$, one can compute $s \in F[x]$ such that*

$$s \equiv r^{q^{d(e-1)}} \bmod f^e$$

by Boolean circuits of depth $O(\log n \operatorname{div}(n))$ and size $n^{O(1)}$.

Proof. We solve the equation

$$s^{q^d - 1} \equiv 1 \bmod f^e, \qquad s \equiv r \bmod f,$$

by a quadratic Newton iteration. The depth of the iteration is $\lceil \log e \rceil$. Each step

$$s_i \equiv s_{i-1} + (s_{i-1}^{q^d - 1} - 1) s_{i-1}^{(q^d - 2)^2} \bmod f^{2^i}$$

can be performed in depth $O(\operatorname{div}(n))$. Comparing with Lemma 3.1, one factor $q^d - 1 = -1$ in the denominator becomes trivial. □

We use P-POWER to denote the polynomial analogue of POWER_1: with input size parameter $n \in \mathbb{N}$, we are given p, g, a, b, m such that $p, b \in \mathbb{N}$, p is prime, $b < 2^n$, $g \in \mathbb{Z}_p[t]$ is irreducible of degree $l \leqq n$, $a, m \in \mathbb{Z}_p[t, x]$, $\deg_t(a), \deg_t(m) < l$, $\deg_x(a)$,

$\deg_x(m) \leqq n$, and have to compute some $c \in \mathbb{Z}_p[t, x]$ such that $c \equiv a^b \bmod m$ in $F[x]$, where $F = \mathbb{Z}_p[t]/(g)$. Let $q = p^l$. We say that m is n-smooth if there exist $f_1, \cdots, f_r \in F[x]$ irreducible of degree at most d, and $e_1, \cdots, e_r \in \mathbb{N}$ such that $f = f_1^{e_1} \cdots f_r^{e_r}$ and $q^d \leqq n$.

The following is the translation of algorithm INTEGER POWERS MODULO A PRIME POWER to the setting of polynomials.

Algorithm POLYNOMIAL POWERS MODULO A POWER OF AN IRREDUCIBLE.

Input: An input size parameter $n \in \mathbb{N}$, and p, g, a, b, m as in P-POWER, where $m = f^e$ is a power of an irreducible $f \in F[x]$.

Output: $c \in F[x]$ such that $c \equiv a^b \bmod f^e$.

1. If $a = 0$, then return $c = 0$ and stop. Compute $l \in \mathbb{N}$ such that f^l / a and $f^{l+1} \nmid a$. If $lb \geqq e$, return $c = 0$ and stop. Otherwise replace a by a/f^l. (We can now assume $a \not\equiv 0 \bmod f$.)
2. Set $d = \deg f$, and compute $r \in F[x]$ such that
$$a \equiv r \bmod f, \qquad 0 \leqq \deg r < d.$$
3. Compute $s \in F[x]$ such that
$$s \equiv \alpha(r \bmod f) \equiv r^{q^{d(e-1)}} \bmod f^e,$$
$$0 \leqq \deg s < de.$$
4. Compute $h \in \mathbb{N}$ and $u \in F[x]$ such that
$$h \equiv b \bmod q^d - 1, \qquad 0 \leqq h < q^d - 1,$$
$$u \equiv s^h \bmod f^e, \qquad 0 \leqq \deg u < de.$$
5. Compute the inverse $t \in F[x]$ of s such that
$$st \equiv 1 \bmod f^e, \qquad 0 \leqq \deg t < de.$$
6. Compute $v, w \in F[x]$ such that
$$v \equiv at \bmod f^e,$$
$$w \equiv \sum_{0 \leqq i < e} \binom{b}{i} (v-1)^i \bmod f^e,$$
$$0 \leqq \deg v, \deg w < de.$$
7. Return $c = f^{lb} uw$.

THEOREM 5.2. *The above algorithm solves* P-POWER. *If m is a power of an irreducible polynomial f with $q^{\deg f} \leqq n$, it can be implemented on a Boolean circuit of depth $O(\log n \operatorname{div}(n))$ and size $n^{O(1)}$.*

Proof. Everything follows just as for Theorem 3.2. The inverse t in step 5 is calculated by a quadratic Newton iteration. The initial value t_0 such that $st_0 \equiv 1 \bmod f$ can be found by exhaustive search in depth $O(\log n)$, since $q^d \leqq n$; alternatively, one can solve the corresponding $d \times d$-system of linear equations over F, in depth $O(\log^2 d \operatorname{div}(n))$ or $O((\operatorname{loglog} n)^2 \operatorname{div}(n))$. All steps except steps 3 and 5 require only depth $O(\operatorname{div}(n))$. □

COROLLARY 5.3. *There exists a P-uniform family of Boolean circuits of depth $O(\log n)$ that computes* P-POWER *for inputs with m a power of a small irreducible polynomial.*

Proof. Given n, for every p, g, f, e as in P-POWER with input size parameter n and $p^{l \deg f} \leqq n$, and for every polynomial $r \in (\mathbb{Z}_p[t]/(g))[x] \setminus \{0\}$ of degree less than d, let $s_{rfeg}, t_{rfeg} \in (\mathbb{Z}_p[t]/(g))[x]$ be the unique polynomials such that

$$s_{rfeg} \equiv r^{p^{ld(e-1)}} \bmod f^e, \quad s_{rfeg} t_{rfeg} \equiv 1 \bmod f^e, \quad 0 \leqq \deg s_{rfeg}, \deg t_{rfeg} < de.$$

There are less than $n^5 \log n$ possible choices of (p, l, g, d, f, e, r), and therefore all s_{rfeg} and t_{rfeg} can be hard-wired into the nth circuit of a P-uniform circuit. Given these constants, steps 3 and 5 and therefore the whole of algorithm of POLYNOMIAL POWERS MODULO A POWER OF AN IRREDUCIBLE can be performed in depth $O(\log n)$. □

The following is an improvement analogous to Corollary 3.5.

COROLLARY 5.4. *There exists a log-space uniform family of Boolean circuits of depth* $O(\log n \log\log n)$ *and polynomial size, which computes* P-POWER *for all* $m = f^e$ *with* f *linear and* $p^l \leqq n$.

Proof. On input $a, b, f = x - z, e$ with $z \in F$, we can assume that $a(z) \neq 0$, and then the algorithm simplifies to

$$c \equiv a(z)^b \sum_{0 \leqq i < e} \binom{b}{i} \left(\frac{a}{a(z)} - 1\right)^i$$

$$\equiv a(z)^{b-e+1} \sum_{0 \leqq i < e} \binom{b}{i} a(z)^{e-i-1} (a - a(z))^i \bmod f^e. \qquad \square$$

For arithmetic circuits, the field F and the exponents b and e have to be fixed. The appropriate modification of the algorithm above yields the following result.

COROLLARY 5.5. *For any finite field* F, *any* $b < 2^n$ *and* $e \leqq n$ *there exists an arithmetic circuit over* F *of depth* $O(\log^2 n \log\log n)$ *and polynomial size that computes* $a^b \bmod f^e$ *for all inputs* $a, f \in F[x]$ *of degree at most* n *such that* f *is irreducible and* $(\# F)^{\deg f} \leqq n$.

This algorithm can be implemented on a log-space uniform family of arithmetic circuits. For P-uniform families of arithmetic networks, one obtains optimal depth $O(\log n)$.

6. Polynomial powers over finite fields. In this section, we make a rather obvious adaptation of the algorithm from § 5 to more general moduli, via Chinese remaindering. A fast parallel powering algorithm results, for smooth moduli. In particular, we obtain a P-uniform family of Boolean circuits of optimal depth $O(\log n)$.

As usual, F is a finite field with $\operatorname{char} F = p$ and $\# F = q = p^l$, and n is an input size parameter. We consider problems P-CHINREM, P-FACTOR defined by the same conditions as the corresponding problems in § 4, except that now $a_1, \cdots, a_k, m_1, \cdots, m_k, c \in F[x]$ are polynomials of degree at most n; b is an n-bit integer. Polynomials are represented by their coefficient sequences.

For any field F, P-CHINREM can be solved by arithmetic networks over F of depth $O(\log^2 n)$ (von zur Gathen [8]). For finite F, this can also be performed by Boolean circuits of depth $O(\log^2 n)$, using redundant notation for elements of $\mathbb{Z}_p$. (See Borodin, Cook and Pippenger [3] for Boolean circuits for linear algebra; also Eberly [5].) Clearly P-FACTOR can be computed on Boolean circuits of depth $O(\operatorname{div}(n))$, if m is n-smooth.

THEOREM 6.1. P-POWER *can be computed on Boolean circuits of depth* $O(\log n \operatorname{div}(n))$ *and size* $n^{O(1)}$, *for inputs with* m *being* n-*smooth.*

Proof. In order to compute $a^b \bmod m$ for $a, m \in F[x]$, we first factor $m = f_1^{e_1} \cdots f_r^{e_r}$ where $f_1, \cdots, f_r$ are the irreducible factors of m. By Theorem 5.2, we can compute

$c_i \in F[x]$ such that $c_i \equiv a^b \bmod f_i^{e_i}$ for each i. Then we interpolate $c_1, \cdots, c_r$ by P-CHINREM to get the desired $c \in F[x]$ with $c \equiv a^b \bmod m$. □

COROLLARY 6.2. *There exists a P-uniform family of Boolean circuits of depth* $O(\log n)$, *whose nth circuit computes P-*POWER *if m is n-smooth.*

Proof. By Corollary 5.3 and the fact that P-FACTOR can now be solved in depth $O(\log n)$, it is sufficient to consider P-CHINREM. As for Corollary 4.3, we simply hard-wire the required modular inverses $u_{fef'} \in (\mathbb{Z}_p[t]/(g))[x]$ such that

$$u_{fef'} \cdot f' \equiv 1 \bmod f^e, \qquad 0 \leqq \deg u_{fef'} < e \deg f,$$

where p is prime, $g \in \mathbb{Z}_p[t]$ is monic irreducible of degree $l, f \neq f' \in (\mathbb{Z}_p[t]/(g))[x]$ are irreducible monic of degree at most $d, p^{ld} \leqq n$, and $1 \leqq e \leqq n$. This yields a table of polynomial size, which can be constructed in polynomial time. □

7. Conclusion. We have shown that large powers of integers or polynomials modulo an element can be computed fast in parallel, provided the modulus is smooth, i.e., has only small prime factors. The basic question, however, remains open:

Open Question 7.1. Can $a^b \bmod m$, for n-bit integers a, b, m, be computed on Boolean circuits of depth $(\log n)^{O(1)}$ (and size $n^{O(1)}$)?

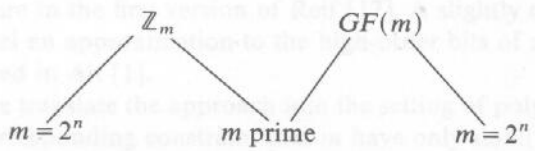
One can hope for further positive results in special cases; a candidate seems to be the case where m is prime and $m-1$ is smooth. No successful algorithm exists that makes use only of arithmetic mod m for prime m (Theorem 2.5).

The number a^b is too long to be computed even in polynomial time, but Allan Borodin has asked the following.

Open Question 7.2. Given n-bit integers a, b, i, can the ith bit of a^b be computed in depth $(\log n)^{O(1)}$ (and size $n^{O(1)}$)?

It is not even clear how to do this in (sequential) polynomial size. The corresponding questions can be asked for polynomials over finite fields. Fich and Tompa [6] give positive answers to both questions for polynomials over finite fields of small characteristic. For computations by arithmetic circuits over large finite fields, the answer is negative to both questions.

The two most basic finite arithmetic structures with m elements (where m is a "large" integer) are $\mathbb{Z}_m$ and $GF(m)$. The extreme cases are when m is a power of 2 (or a power of a small prime, or smooth for $\mathbb{Z}_m$), and when m is prime, in which case $\mathbb{Z}_m = GF(m)$.



The parallel powering problem is solved in both cases with $m = 2^n$.

REFERENCES

[1] H. ALT, *Comparison of arithmetic functions with respect to Boolean circuit depth*, Proceedings 16th Annual ACM Symposium on the Theory of Computing, Washington, DC, 1984, pp. 466–470.

[2] P. W. BEAME, S. A. COOK AND H. J. HOOVER, *Log depth circuits for division and related problems*, this Journal, 15 (1986), pp. 994–1003.

[3] A. BORODIN, S. COOK AND N. PIPPENGER, *Parallel computation for well-endowed rings and space-bounded probabilistic machines*, Inform. and Control, 58 (1983), pp. 113–136.

[4] S. A. COOK, *A taxonomy of problems with fast parallel algorithms*, Inform. and Control, 64 (1985), pp. 2–22.

[5] W. EBERLY, *Very fast parallel matrix and polynomial arithmetic*, Proceedings 25th Annual IEEE Symposium on the Foundations of Computer Science, Singer Island, FL, 1984, pp. 21–30.

[6] F. FICH AND M. TOMPA, *The parallel complexity of exponentiating polynomials over finite fields*, Proceedings 17th Annual ACM Symposium on the Theory of Computing, Providence, RI, 1985, pp. 38–47; J. Assoc. Comput. Mach., to appear.

[7] J. VON ZUR GATHEN, *Parallel algorithms for algebraic problems*, this Journal, 13 (1984), pp. 802–824.

[8] ———, *Representations and parallel computations for rational functions*, this Journal, 15 (1986), pp. 432–452.

[9] ———, *Parallel arithmetic computations: a survey*, Proceedings 12th Internat. Symposium on the Math. Foundations of Computer Science, Bratislava, Springer Lecture Notes in Computer Science, 233, 1986, pp. 93–112.

[10] J. VON ZUR GATHEN AND G. SEROUSSI, *Boolean circuits versus arithmetic circuits*, Proceedings 6th Internat. Conf. Computer Science, Santiago, Chile, 1986, pp. 171–184.

[11] H. HASSE, *Number Theory*, Grundlehren der math. Wiss., 229, Springer-Verlag, New York–Berlin–Heidelberg, 1980.

[12] A. HILDEBRAND, *On the number of positive integers $\leq x$ and free of prime factors $> y$*, J. Number Theory, 22 (1986), pp. 289–307.

[13] D. E. KNUTH, *The Art of Computer Programming*, Vol. 2, 2nd ed., Addison-Wesley, Reading, MA, 1981.

[14] 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.

[15] E. LUCAS, *Sur les congruences des nombres eulériens et des coefficients différentiels des fonctions trigonométriques, suivant un module premier*, Bull. Soc. Math. France, 6 (1877/78), pp. 49–54.

[16] I. MUNRO AND M. PATERSON, *Optimal algorithms for parallel polynomial evaluation*, J. Comput. System Sci., 7 (1973), pp. 189–198.

[17] J. REIF, *Logarithmic depth circuits for algebraic functions*, this Journal, 15 (1986), pp. 231–242. Extended Abstract in Proceedings 24th Annual IEEE Symposium on the Foundations of Computer Science, Tucson, AZ, 1983, pp. 138–145.

[18] W. L. RUZZO, *On uniform circuit complexity*, J. Comput. System Sci., 22 (1981), pp. 365–383.

[19] V. STRASSEN, *Berechnung und Programm. I*, Acta Inform., 1 (1972), pp. 320–335.