J. Inf. Process. Cybern. EIK 25 (1989) 10, 543–547
(formerly Elektron. Inf.verarb. Kybern.)

Improved Parallel Computations in the Ring Z/p^α 1)

By *Thomas Zeugmann*

Abstract: Improved parallel algorithms computing the inverse or a large power modulo an element that has only small prime factors are presented.

1. Introduction

Within the last years parallel algorithms have attracted much attention of computer scientists (cf. e.g. [4], [7]). Thereby it has been shown that, even using a feasible amount of hardware, many problems can be solved substantially faster in parallel than by a sequential algorithm (cf. e.g. [3]). However, despite the enormous progress made within the last decade some problems seem to be very hard parallelizable. Among them we find the following problems concerning parallel computations over the integers:

(1) GCD: Let n-bit integers a, b be given. Compute the greatest common divisor of a and b.

(2) INV: Let n-bit integers a, m be given. Compute the inverse x of a modulo m, i.e., compute the number x satisfying $ax \equiv 1 \bmod m$ if it exists, otherwise return failure.

(3) POW: Let n-bit integers a, b, and m be given, whereas $a \not\equiv 0 \bmod m$. Compute the number $y \in \{1, \ldots, m-1\}$ fulfilling $y \equiv a^b \bmod m$.

It is an open question whether or not GCD, INV, and POW are in NC, i.e., whether they can be solved simultaneously in polylog time using only polynomially many processors.

In a beautiful paper *v. z. Gathen* ([5]) has pointed out that INV and POW are both contained in NC provided the modulus m is of the form $m = p_1^{\alpha_1} \ldots p_k^{\alpha_k}$, where p_i, $\alpha_i \leqq n$ for $i = 1, \ldots, k$.

In the present paper we shall describe improved algorithms computing INV and POW, again under the assumption that the modulus has only small prime factors.

The used model of parallel computation is that of uniform families of Boolean circuits (cf. [2]). Please note that we have to distinguish between log-space uniformity and polynomial time uniformity (abbr. P-uniformity) whenever division is involved.

Thus we denote by $\operatorname{div}(n)$ the amount of parallel time needed to divide two n-bit integers. *Beame/Cook/Hoover* [1] have shown that $\operatorname{div}(n) = \mathcal{O}(\log n \log\log n)$ for log-space uniformity, and that $\operatorname{div}(n) = \mathcal{O}(\log n)$ for P-uniformity.

1) Extended version of a lecture presented at the Colloquium on Computation Theory (CCT '88), Berlin, GDR, September 5–9, 1988.

2. Computing the inverse modulo a number m having only small prime factors

Using Chinese remaindering we can restrict ourselves to exclusively consider the case $m = p^\alpha$, where $p, \alpha \leqq n$ (n is the number of bits). The classical algorithm computing the inverse is mainly based on Euclid's algorithm. Thus it works extremely sequential. Exploiting the special properties of moduli having only small prime factors, *v. z. Gathen* ([5]) has been introduced the following iteration technique:

(i) Compute x_0 such that $ax_0 \equiv \bmod p$

(ii) Compute $x_i := (x_i - (x_{i-1}^2 a - x_{i-1})) \bmod p^{2^i}$ for $i = 1, \ldots, \lceil \log \alpha \rceil$

Thereby the first step can be performed even by exhaustive search since $p \leqq n$.

However at this point we make the observation that quite a few is done in parallel. Consequently we ask whether or not we can do something more in parallel. A closer look to the above iteration technique leads to the following observation: If x is the inverse of a modulo p^e then performing one iteration step (i.e., computing $x' = x - (x^2a - x) \bmod p^{2e}$) augments the exponent only by the factor 2. Hence we are interested in augmenting the exponent somehow faster, thereby doing more and more in parallel, e.g. computing

$$x' = 4x + 4a^2x^3 - 6ax^2 - a^3x^4 \bmod p^{4e}.$$

The latter iteration can be performed in parallel using the same time as the above one. On the other hand, the number of iteration steps is reduced to the half. Nevertheless, in order to achieve a substantially better result we have to augment the exponent from e to e^2 per iteration.

Lemma 1. *Let p be a prime number, and let x be the inverse of a number a modulo p^e, where e is even. Furthermore, let*

$$x' = \left(\sum_{v=0}^{e-2} \binom{e}{v} ax^{e-v-1}x^{e-v}(-1)^{v+1} + ex\right) \bmod p^{e^2}.$$

Then $ax' \equiv 1 \bmod p^{e^2}$.

Proof. We have to show that p^{e^2} divides $ax' - 1$. Computing $(ax' - 1) \bmod p^{e^2}$ we get:

$$(ax' - 1) \bmod p^{e^2}$$

$$= \left(\sum_{v=0}^{e-2} \binom{e}{v} a^{e-v}x^{e-v}(-1)^{v+1} + eax - 1\right) \bmod p^{e^2}$$

$$= \left(- \sum_{v=0}^{e-2} \binom{e}{v} (ax)^{e-v}(-1)^{v} + eax - 1\right) \bmod p^{e^2}$$

$$= -\left(\sum_{v=0}^{e-2} \binom{e}{v} (ax)^{e-v}(-1)^{v} - eax + 1\right) \bmod p^{e^2}$$

(Remember that e has been assumed to be even)

$$= -\left(\sum_{v=0}^{e} \binom{e}{v} (ax)^{e-v}(-1)^{v}\right) \bmod p^{e^2}$$

$$= -(ax - 1)^e \bmod p^{e^2} = 0$$

since $p^e/(ax - 1)$ directly implies that $p^{e^2}/(ax - 1)^e$. □

Exploting the above lemma one directly obtains a powerful iteration technique computing the inverse.

Theorem 2. *Let a be an n-bit number, and let $p, \alpha \leqq n$.*
Then there exists a uniform family of Boolean circuits having simultaneously depth $\mathcal{O}(\mathrm{div}(n)\ \mathrm{loglog}\ n)$ and size $\mathcal{O}(n^{\mathcal{O}(1)})$ which computes the inverse of a modulo p^α.

Proof. We use the following algorithm:

(i) Compute $x_0 \in \{1, \ldots, p-1\}$ such that $ax_0 \equiv 1 \bmod p$

(ii) Compute $x_1 \in \{1, \ldots, p^2 - 1\}$ by $x_1 = (2x_0 - ax_0^2) \bmod p^2$

(iii) Compute

$$x_i = \left(2^{2^{i-2}} x_{i-1} + \sum_{v=0}^{2^{2^{i-2}}-2} \binom{2^{2^{i-2}}}{v} a^{2^{2^{i-2}}-1-v} x_{i-1}^{2^{2^{i-2}}-v} (-1)^{v+1}\right) \bmod p^{2^{2^{i-1}}}$$

for $i = 2, \ldots, \lceil \mathrm{loglog}\ \alpha \rceil$

(iv) Let $l = \lceil \mathrm{loglog}\ \alpha \rceil$. Return $x_l \bmod p^\alpha$.

The correctness of this algorithm is an immediate consequence of Lemma 1. In each iteration step there will eventually occur $\mathcal{O}(n)$ numbers the sum of which has to be computed. These numbers can be computed in parallel in depth being at most $\mathcal{O}(\mathrm{div}(n))$. Thereby only $\mathcal{O}(n^{\mathcal{O}(1)})$ processors are needed (cf. [1]). Using standard techniques it can be easily seen that each of these numbers has at most $\mathcal{O}(n)$ bits. Consequently, the sum can be computed in depth $\mathcal{O}(\mathrm{div}(n))$ using $\mathcal{O}(n^{\mathcal{O}(1)})$ processors. Since we have to perform only $\lceil \mathrm{loglog}\ \alpha \rceil + 1$ iteration steps the theorem follows. $\square$

It is an interesting open question whether or not the above result can be improved. In particular, we are interested in learning to know whether the computation of the inverse mudulo a number m having only small prime factors is more complex than division.

3. Improved powering

The powering problem has been intensively studied by *v. z. Gathen* ([5]). Thereby for P-uniform families of Boolean circuits he obtained the best possible result, i.e., he described a P-uniform family of Boolean circuits having simultaneously depth $\mathcal{O}(\log n)$ and size $\mathcal{O}(n^{\mathcal{O}(1)})$ which computes POW for a modulus that has only small prime factors. On the other hand, considering log-space uniform families of Boolean circuits much more difficulties arise. Using the well-known decomposition $\mathbb{Z}/p^\alpha = G \times H$ (cf. [6]) *v. z. Gathen* ([5]) has divided the problem POW into appropriate computations in G and H. Thereby $H = \{r \in \mathbb{Z}/p^\alpha \mid r \equiv 1 \bmod p\}$ and G is defined to be the image set of the following injective group homomorphism $h\colon \mathbb{Z}/p \rightarrowtail \mathbb{Z}/p^\alpha$, defined as:

$$r \rightarrowtail r^{p\alpha-1} \bmod p^\alpha .$$

The hardest part here is the computation of the homomorphism h. In [5] again an iteration technique has been presented which computes h as follows:

(i) $s_0 = r$

(ii) $s_i = s_{i-1} + (s_{i-1}^{p-1} - 1)(1 + p + \ldots + p^{2^i-1})\, s_{i-1}^{(p-2)^2} \bmod p^{2^i}$

for $i = 1, \ldots, \lceil \log \alpha \rceil$.

What we like to present here is an improvement of the above technique which considerably reduces the size of the circuit.

Algorithm HOMOMORPHISM

Input: Let p, α, r be numbers satisfying $r \in \mathbb{Z}/p^\alpha$, $p, \alpha \leq n$.

Output: The number $s \in \mathbb{Z}/p^\alpha$ fulfilling $s \equiv r^{p^{\alpha-1}} \bmod p^\alpha$

```
Method:
  (i) $s_0 = r$
  (ii) For $i = 1$ to $l = \lceil \log \alpha \rceil$ do
       $s_i = s_{i-1} + (s_{i-1}^{p-1} - 1)(1 + p + \dots + p^{2^i-1})\, s_{i-1} \bmod p^{2^i}$.
  Return $s_l \bmod p^\alpha$.
```

Theorem 3. *The Algorithm* HOMOMORPHISM *correctly computes the group homomorphism* $h\colon \mathbb{Z}/p \rightarrowtail \mathbb{Z}/p^\alpha$, *with* $r \rightarrowtail r^{p^{\alpha-1}} \bmod p^\alpha$.

Proof. First of all we note that $\operatorname{card}(\mathbb{Z}/p^\alpha) = (p-1)\, p^{\alpha-1}$. Hence, if $s \equiv r^{p^{\alpha-1}}$ $\bmod p^\alpha$ then

$$s \equiv r \bmod p, \text{ and} \tag{1}$$

$$s^{p-1} \equiv 1 \bmod p^\alpha \tag{2}$$

Conversely, if s satisfies (1) and (2) then $s \equiv r^{p^{\alpha-1}} \bmod p^\alpha$. Consequently it suffices to show that each s_i satisfies $s_i \equiv r \bmod p$ and $s_i^{p-1} \equiv 1 \bmod p^{2^i}$.

We proceed inductively. For $i = 0$ the assertion is obviously fulfilled. So let us now assume that $s_i \equiv r \bmod p$ and $s_i^{p-1} \equiv 1 \bmod p^{2^i}$. We get:

$$\begin{aligned} s_i^{p-1} &= \left(s_i + (s_i^{p-1} - 1)(1 + p + \dots + p^{2^{i+1}})\, s_i\right)^{p-1} \bmod p^{2^i} \\ &\equiv \sum_{v=0}^{p-1} \binom{p-1}{v} s_i^v \left((s_i^{p-1} - 1)(1 + p + \dots + p^{2^{i+1}-1})\, s_i\right)^{p-1-v} \\ &\equiv \left((p-1)\, s_i^{p-2}(s_i^{p-1} - 1)(1 + p + \dots + p^{2^{i+1}-1})\, s_i + s_i^{p-1}\right) \bmod p^{2^{i+1}}, \end{aligned}$$

since by the assumption p^{2^i} divides $(s_i^{p-1} - 1)$, and consequently $p^{2^{i+1}}(s_i^{p-1} - 1)^v$ for every $v \geq 2$.

Considering the latter term we obtain:

$$\begin{aligned} &-(1-p)(s_i^{p-1} - 1)(1 + p + \dots + p^{2^{i+1}-1})\, s_i s_i^{p-2} + s_i^{p-1} \\ &\equiv (p^{2^{i+1}} - 1)(s_i^{p-1} - 1)\, s_i s_i^{p-2} + s_i^{p-1} \\ &\equiv (p^{2^{i+1}} - 1)(s_i^{p-1} - 1)\, s_i^{p-1} + s_i^{p-1} \\ &\equiv (p^{2^{i+1}} - 1)\left((s_i^{p-1})^2 - s_i^{p-1}\right) + s_i^{p-1} \\ &\equiv p^{2^{i+1}}\left((s_i^{p-1})^2 - s_i^{p-1}\right) - (s_i^{p-1})^2 + s_i^{p-1} + s_i^{p-1} \\ &\equiv (-1)\left((s_i^{p-1})^2 - 2s_i^{p-1} + 1\right) + 1 \equiv (-1)(s_i^{p-1} - 1)^2 + 1 \\ &\equiv 1 \bmod p^{2^{i+1}}, \end{aligned}$$

since by assumption p^{2^i} divides $(s_i^{p-1} - 1)$. Thus $p^{2^{i+1}}/(s_i^{p-1} - 1)^2$. $\square$

Since our method is less complicated than *v. z. Gathen*'s one the algorithm HOMOMORPHISM either reduces the parallel time by a constant factor or it considerably reduces the number of gates.

On the other hand it remained open whether or not the number of iterations can be decreased up to loglog n.

References

[1] *Beame, P. W., S. A. Cook, H. J. Hoover*: Log depth circuits for division and related problems. SIAM J. Comput. 15 (1986), 994–1003.

[2] *Borodin, A.*: On relating time and space to size and depth. SIAM J. Comput. 6 (1977), 733–744.

[3] *Cook, S. A.*: The classification of problems which have fast parallel solutions. In: Proc. Foundations of Computation Theory; (Lecture Notes in Computer Science, 158); Springer, 1983; pp. 78–93.

[4] *v. z. Gathen, J.*: Parallel arithmetic computations: a survey. In: Proc. Mathematical Foundations of Computer Science; (Lecture Notes in Computer Science, 233); Springer, 1986; pp. 93–112.

[5] *v.z. Gathen, J.*: Computing powers in parallel. SIAM J. Comput. 16 (1987), 930–945.

[6] *Hasse, H.*: Number Theory. (Grundlehren der mathematischen Wissenschaften). Springer, 1980.

[7] *Zeugmann, T.*: Parallel algorithms. (Series: Encyclopedia of Computer Science and Technology). Marcel Dekker, New York (to appear)

Kurfassung

Es werden verbesserte parallele Algorithmen zur Berechnung der Inversen oder einer großen Potenz einer Zahl für Module mit ausschließlich kleinen Primfaktoren vorgestellt.

Резюме

Представлены улучшенные параллельные алгоритмы вычисления обратных элементов и больших степеней чисел для модулей с исключительно малыми простыми сомножителями.

Author's address:

Th. Zeugmann
Humboldt-Universität zu Berlin
Sektion Mathematik
Postfach 1297
1086 Berlin
German Democratic Republic