A Straight Line Program Computing the Integer Greatest Common Divisor

S. M. Sedjelmaci
LIPN, CNRS UPRES-A 7030
Université Paris-Nord,
Av. J.B.-Clément, 93430 Villetaneuse, France.
sms@lipn.univ-paris13.fr

Extended Abstract

While NC algorithms have been discovered for the basic arithmetic operations, the parallel complexity of some fundamental problems as integer gcd is still open, since first being raised in a paper of Cook [2]. Many authors attempt to design fast parallel integer GCD algorithms. Chor and Goldreich [1] proposed $O(n/\log n)_\epsilon$ parallel time with $O(n^{1+\epsilon})$ number of processors, for any $\epsilon > 0$. Sorenson [4] and the author [3] also suggest other parallel algorithms with the same parallel performance. Since then, no major improvements have been made. In this paper, we propose a straight line program computing the integer GCD. It has polynomial size, but the outputs are polynomials with exponential degree. This work is a first attempt to improve the parallel complexity of integer GCD, thanks to Valiant et al. [5] contraction method, and, as far as we know, it is the first straight line program for computing the integer GCD. Throuhough this paper, we represent the input integers as formal strings of bits.

The Integer GCD Algorithm

Input: $x,\ y > 0$ odds ;
Output: $\gcd(x, y)$;

$$\begin{pmatrix} u \\ v \end{pmatrix} \leftarrow \begin{pmatrix} x \\ y \end{pmatrix};$$

While ($u \neq v$)

$$\begin{pmatrix} u \\ v \end{pmatrix} \leftarrow \begin{pmatrix} v \\ (u+v)/2^t \end{pmatrix}; \text{ s.t.: } (u+v)/2^t \text{ is odd.}$$

EndWhile

Return u.

Example: Let $(x, y) = (35, 19)$ we obtain in turn:

$$\begin{pmatrix} 35 \\ 19 \end{pmatrix} \to \begin{pmatrix} 19 \\ 27 \end{pmatrix} \to \begin{pmatrix} 27 \\ 23 \end{pmatrix} \to \begin{pmatrix} 23 \\ 25 \end{pmatrix} \to \begin{pmatrix} 25 \\ 3 \end{pmatrix} \to \begin{pmatrix} 3 \\ 7 \end{pmatrix} \to \begin{pmatrix} 7 \\ 5 \end{pmatrix} \to \begin{pmatrix} 5 \\ 3 \end{pmatrix} \to \begin{pmatrix} 3 \\ 1 \end{pmatrix} \to \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

Theorem 0.1 *: Let $u, v \geq 1$ be two odd integers of n bits, $n \geq 1$, such that $|u - v| = r2^t > 1$, with $r \geq 1$ odd, and $t \geq 1$. Let (u_k, v_k) be the sequence of consecutive values of u and v, obtained in the GCD algorithm. Then*

i) $\max\{u_{t+1}, v_{t+1}\} < (3/4)\ \max\{u, v\}$.

ii) The algorithm terminates after at most $n^2/\log_2(4/3)$ iterations and returns gcd(u, v).

iii) The (**While** *$u \neq v$) condition can be replaced by (* **For** *$i = 1$* **to** *$3n^2$) in the GCD algorithm.*

Proof: The case $u = v$ is trivial. We assume that $t \geq 3$, $u \neq v$ and $(u, v) = (v_0 + r2^t, v_0)$, the case $(u, v) = (u_0, u_0 + r2^t)$ is similar. The t first iterations give in turn

$$\begin{pmatrix} u_0 = v_0 + r2^t \\ v_0 \end{pmatrix} \rightarrow \begin{pmatrix} v_0 \\ v_0 + r2^{t-1} \end{pmatrix} \rightarrow \begin{pmatrix} v_0 + r2^{t-1} \\ v_0 + r2^{t-2} \end{pmatrix} \cdots \rightarrow \begin{pmatrix} v_0 + r2^{t-2} + \ldots + 2r \\ 1/2^m\{v_0 + r2^{t-2} + \ldots + r\} \end{pmatrix}$$

After t iterations, the integer $2^m v_t = v_0 + r2^{t-2} + \ldots r$ is even. So $v_t < (1/2)u_0$ and $u_t < u_0$. Then, after $t+1$ iterations, we have $u_{t+1} = v_t < (1/2)u_0$ and $v_{t+1} \leq (1/2)(u_t + v_t) < (3/4)u_0$. Similarly, if $u_{t+1} = v_{t+1}$, it stops and returns the result: $u_{t+1} = \gcd(u, v)$. Otherwise $|u_{t+1} - v_{t+1}| = r_2 2^{t_2} > 1$, then we repeat the same process to the pair (u_{t+1}, v_{t+1}). Since $t_1 = t < n$, $t_2 < n, \ldots, t_p < n$, then after pn iterations we have $1 \leq \max\{u_{pn}, v_{pn}\} < (3/4)^p \max\{u, v\} < (3/4)^p 2^n$. Moreover, the **For** and the **While** versions of the algorithm give the same pair (u_i, v_i) until we reach a pair (u_k, v_k) such that $u_k = v_k$, with $k \leq pn < \lfloor n^2/\log_2(4/3)\rfloor$. At this point, the **While** version of the algorithm terminates and returns u_k, and the **For** algorithm loops with the same consecutive pair (u_k, v_k), with $v_k = u_k$, until the $(3n^2)$th iteration. The cases $t = 1$ or $t = 2$ are trivial. Hence the result.

While the addition of two n bits is trivial, the instruction $A \rightarrow A/2^t$, $A > 0$, can be done as follows (we set $A = (a_n, a_{n-1}, \cdots, a_1)$, and $a_{n+1} = 0$) :

For $k = 1$ **to** $n - 1$ **Do**
  $c = (1 - a_1)$;
  For $i = 1$ **to** n **Do** $a_i = c \cdot a_{i+1} + (1 - c) \cdot a_i$
EndFor
Return $A' = (a_n, a_{n-1}, \cdots, a_1)$.

The **For** version of the GCD algorithm is clearly a straight-line program using only the ring operations $+, -$, and $\times$ on bits with $O(n^4)$ steps, however the degree of the polynomials generated by the program is exponential.

References

[1] B. Chor and O. Goldreich, An improved parallel algorithm for integer GCD, *Algorithmica.* **5** (1990).

[2] S. Cook, A Taxonomy of Problems with Fast Parallel Algorithms, *Information and Control.* **64** (1985) 2–22.

[3] M.S. Sedjelmaci, On a Parallel Lehmer-Euclid GCD Algorithm, *Proceedings of the International Symposium on Symbolic and Algebraic Computation ISSAC'2001* (2001) 303–308.

[4] J. Sorenson, Two Fast GCD Algorithms, *J. of Algorithms* **16** (1994) 110–144.

[5] L.G. Valiant, S. Skyum, S. Berkowitz and C. Rackoff, Fast parallel computation of polynomials using few processors, *SIAM J.Computing* **12** No.4 (1983) 641–644.