# Fast parallel GCD algorithm of many integers

Sidi M. SEDJELMACI
LIPN, CNRS UMR 7030
University of Paris-Nord
Av. J.-B. Clément, 93430 Villetaneuse, France
sms@lipn.univ-paris13.fr

**Abstract:** We present a new parallel algorithm which computes the GCD of $n$ integers of $O(n)$ bits in $O(n/\log n)$ time with $O(n^{2+\epsilon})$ processors, for any $\epsilon > 0$ on CRCW PRAM model.

The computation of the GCD of two integers is not known to be in the NC parallel class, nor it is known to be P-complete [1]. The best parallel performance was first obtained by Chor and Goldreich [2], then by Sorenson [7] and Sedjelmaci [5] since they propose, with different approaches, parallel integer GCD algorithms which can be achieved in $O(n/\log n)$ time with $O(n^{1+\epsilon})$ number of processors, for any $\epsilon > 0$, in PRAM CRCW model. A naive approach, using a binary tree computation to compute the GCD of $n$ integers of $O(n)$ bits would require $O(n)$ parallel time with $O(n^{2+\epsilon})$ processors. One may also use the existing parallel GCD algorithms of two integers and try to adapt them to design a GCD for many integers. However, it is not obvious how to find a parallel GCD for $n$ integers which conserve the same $O(n/\log n)$ time, with $O(n^{2+\epsilon})$ processors, which is roughly the bit-size of all the $n$ input integers. In this paper, we prove that we can compute the GCD of $n$ integers of $O(n)$ bits, in only $O(n/\log n)$ parallel time with $O(n^{2+\epsilon})$ processors, for any $\epsilon > 0$ on CRCW PRAM model, in the worst case. Another probabilistic approach is given in [3]. To our knowledge, it is the first deterministic algorithm which computes the GCD of many integers with this parallel performance and polynomial work. Our algorithm, called $\Delta$-GCD is the following:

**Input**: A set $A = \{ a_0, a_1, \cdots, a_{n-1} \}$ of $n$ distinct positive integers, $a_i < 2^n$, with $n \geq 4$.
**Output**: $\gcd(a_0, a_1, \cdots, a_{n-1})$.

$\alpha := a_0\,;\ I := 0\,;\ p := n\,;$
**While** $(\alpha > 1)$ **Do**
  **For** $(i = 0)$ **to** $(n-1)$ **ParDo**
    **If** $(0 < a_i \leq 2^n/p)$ **Then** $\{\ \alpha := a_i\,;\ I := i\,;\ \}$
  **Endfor**
  **If** $(\alpha > 2^p/n)$ **Then** /* Compute in parallel $I, J$ and $\alpha$ */
    $\alpha := \min\{\,|\,a_i - a_j\,| > 0\,\} = a_I - a_J\ ;\quad a_I := \alpha\ ;$
  **Endif**
  **For** $(i = 0)$ **to** $(n-1)$ **ParDo** /* Reduce all the $a_i$'s */
    **If** $(i \neq I)$ **Then** $a_i := a_i \bmod \alpha\ ;$
  **Endfor** /* $\forall i,\ 0 \leq a_i \leq \alpha$ */
  **If** $(\forall i \neq I,\ a_i = 0)$ **Then Return** $\alpha$ ; /* Here $\alpha = \gcd(a_0, \cdots, a_{n-1})$ */
  $p := np\,;$
**Endwhile**
**Return** $\alpha$.

We use a weak version of the function min based the pigeonhole principle, where only the $O(\log n)$ leading bits of the integers are considered. The integer $\alpha$ is, at each while iteration, $O(\log n)$ bits less. More details for the computations of $I, J$ and $\alpha$ are given in [6], as well as a first C program checking the correctness of the $\Delta$-GCD algorithm.

**Theorem :** The $\Delta - GCD$ algorithm computes in parallel the GCD of $n$ integers of $O(n)$ bits in length, in $O(n/\log n)$ time using $O(n^{2+\epsilon})$ processors on CRCW PRAM model, with $\epsilon > 0$.

**Proof:** (Sketch, see [6]). The algorithm terminates after $O(n/\log n)$ loop iterations. Let $t_i$ be the time cost at iteration $i$, $1 \leq i \leq N$, with $N = O(n/\log n)$. Let $k_i$ be the maximum bit length of all the quotients $q_j = \lfloor a_j/\alpha \rfloor$, with $\sum_{i=1}^{N} k_i \leq n$. We prove that $t_i = O(\min\{ \frac{k_i}{\log n}, \log n \})$. The total number of processors is $n \times O(n^{1+\epsilon}) = O(n^{2+\epsilon})$ and the parallel time is then $t(n) = \sum_{i=1}^{N} t_i = \sum_{i=1}^{N} \min(\{ \frac{k_i}{\log n}, \log n\}) = \sum_{k_i < \log n} 1 + \sum_{\log n < k_i < \log^2 n} \frac{k_i}{\log n} + \sum_{k_i > \log^2 n} \log n = O(n/\log n)$. $\square$

A Blankinship-like algorithm can be easily designed to compute Extended GCD, and an upper bound of the multipliers [4] could be considered as well. A slightly modified Rosser's algorithm (pivoting with $\alpha$) can be used to solve linear Diophantine equations. Moreover, a $O(n^2/\log n)$ sequential version of $\Delta$-GCD should be considered with precomputed lookup tables for arithmetic operations on $O(\log n)$ bit integers.

# References

[1] A. Borodin, J. von zur Gathen and J. Hopcroft, Fast parallel matrix and GCD computations, *Information and Control,* 52, 3, 1982, 241–256

[2] B. Chor and O. Goldreich, An improved parallel algorithm for integer GCD, Algorithmica, 5, 1990, 1-10

[3] G. Cooperman, S. Feisel, J. von zur Gathen and G. Havas, GCD of many integers, *Lect. Notes in Comp. Sci., Springer-Verlag, Berlin,* 1627 (1999), 310–317

[4] G. Havas, S. Majewski, Extended gcd calculation, *Congressus Numerantium, 111,* 1627 (1998), 104-114

[5] S.M. Sedjelmaci, On A Parallel Lehmer-Euclid GCD Algorithm, in Proc. of the International Symposium on Symbolic and Algebraic Computation (ISSAC'2001), 2001, 303-308

[6] S.M. Sedjelmaci, Fast Parallel GCD algorithm of many integers, lipn.univ-paris13.fr/∼sedjelmaci, Rapport interne, LIPN, April, 2013

[7] J. Sorenson, Two Fast GCD Algorithms, J. of Algorithms, 16, 1994, 110-144