Reviewing and Analyzing Efficient GCD/LCM Algorithms for Cryptographic Design

Article · September 2017

DOI: 10.17781/P002301

Ibrahim Marouf
King Faisal University
12 PUBLICATIONS **18** CITATIONS
SEE PROFILE

Mohammed Musab Mohamad Asad
King Faisal University
10 PUBLICATIONS **16** CITATIONS
SEE PROFILE

Qasem Abu Al-Haija
King Faisal University
68 PUBLICATIONS **194** CITATIONS
SEE PROFILE

Reviewing and Analyzing Efficient GCD/LCM Algorithms for Cryptographic Design

Ibrahim Marouf
King Faisal University
i.marouf@outlook.com

Mohamad Musab Asad
King Faisal University
asadmosab@gmail.com

Qasem Abu Al-Haija
King Faisal University
qalhaija@kfu.edu.sa

ABSTRACT

In this paper, we provide a practical review with numerical example and complexity analysis for greatest common divisor (GCD) and Least Common Multiple (LCM) algorithms that are commonly used in the computing coprocessors design such as Cryptoprocessor design. The paper discusses four common GCD algorithms: Dijkstra’s algorithm, Euclidian algorithm, Binary GCD algorithm, Lehmer's algorithm, and two LCM algorithms: LCM based Prime Factorizations algorithm and LCM based GCD reduction. It was found that Lehmer's algorithm can be used efficiently to compute GCD and LCM with time complexity of $\boldsymbol{O}(\frac{\boldsymbol{n}}{\boldsymbol{log(n)}})$ which enhances the linear time ($\boldsymbol{O}\ (\boldsymbol{n})$) complexity of well-known Euclidian algorithm.

KEYWORDS

Greatest Common Divisor (GCD), Dijkstra’s GCD, Euclidian GCD, Binary GCD, Lehmer's GCD, Least Common Multiple (LCM).

1 INTRODUCTION

Recently, the vast promotion in the field of information and communication technology (ICT) such as grid and fog computing has increased the inclination of having secret data sharing over the existing non-secure communication networks. This encouraged the researches to propose different solutions to ensure the safe access and store of private and sensitive data by employing different cryptographic algorithms especially the public key algorithms [1] which proved robust security resistance against most of the attacks and security halls. Public key cryptography is significantly based on the use of number theory and digital arithmetic algorithms.

Number theory [2] concerned with elementary arithmetic algorithms over the set of positive integers (i.e. natural numbers) such as divisibility, primes, congruence, GCD and others. It has rapidly grown in recent years in practical importance through its use in areas such as coding theory, cryptography and statistical mechanics.

GCD and LCM operations [2] are both essential elementary number theory algorithms that are commonly used in the design of many public key crytoprocessors such as RSA cryptosystem [3] and Schmidt-Samoa cryptosystem [4]. The good utilization of enhanced implementations of GCD/LCM will contribute in the overall cost enhancement for the coprocessor.

Recently, several solutions were proposed for speeding up GCD/LCM algorithms while few of them were efficient. For instance, Q. A. Al-Haija, M. Al-Ja'fari and M. Smadi [6] targeted Altera Cyclone IV FPGA family to design an efficient GCD (Greatest Common Divisor) coprocessor based on Euclid's method with variable datapath sizes. They tested their design using seven FPGA chip technologies in terms of maximum frequency and critical path delay of the coprocessor. As a result, they recorded the best values of maximum frequencies of 243.934 MHz down to 39.94 MHz for 32 bits and 1024-bit datapath, respectively.

Theoretically, J. Sorenson [10] provided an analysis of Lehmer’s Euclidean GCD algorithm and a modified version that is faster than Euclid’s algorithm roughly by a factor of k, where k is the number of bits in the multi-precision base of the algorithm. Also, another Euclid’s based GCD algorithm is the parallel Lehmer-Euclid GCD algorithm was discussed in [11]. For long integers, Jebelean, T [12] proposed a Double-Digit-Lehmer-

Euclid's algorithm to speed up the computation of finding the GCD for long integers.
In a related work, a generalization of the binary algorithm for GCD computing were proposed in [13] where the authors used the concept of modular conjugate to find the GCD of multi-precision integers that is faster than Lehmer-Euclid method by a factor of two. Accordingly, this new method was suitable for systolic parallelization and in "least-significant digit first" pipelined manners.
Moreover, Wang, P.S [14] discussed an enhanced EZ-GCD algorithm with it implementation aspects. His algorithm was fast and particularly suited for computing GCD of multivariate polynomials. Similarly, other three algorithms for multivariate polynomial GCD were discussed in [15].
Furthermore, an execution time comparison between three different GCD algorithms (namely: Euclidean, binary, plus-minus) was analyzed by T. Jebelean [16]. These algorithms are the known ones. In addition, the author [16] proposed new improvement of Lehmer's GCD and generalization of binary algorithm which resulted in enhancement computing speed of GCD operation.
Finally, S.M. Sedjelmaci presented new parallelization of the extended Euclidean GCD algorithm that match the best existing integer GCD algorithms since it can be achieved in parallel O/sub eps/ (n/log n) using only n/sup 1+eps/ processors on a Priority CRCW PRAM, for any constant eps>0.
In this paper, we will review and summarize four efficient algorithms to compute GCD/LCM of two positive integers. The remaining of this paper is organized as follows: Section 2 discusses the different GCD algorithms with numerical examples and flowcharts. Section 3 provides the review of LCM algorithms with numerical examples and flowcharts. Finally, Section 4 concludes the paper.

2 GREATEST COMMON DIVISOR (GCD)

GCD is well known elementary number theory algorithm that is defined as the largest positive integer number that can divide two non-negative integer numbers without reminder. The very basic method to calculate GCD is using prime factorization method (PF-GCD).
PF-GCD method depends on fact that each number can uniquely written as a product of prime numbers raised to different powers. Then, to find the GCD, take only the smallest power of common primes factors of the two numbers. An example of the algorithm is shown below:
576 = 2632, 135 = 335➔ GCD (576, 135) = 32 = 9
However, more efficient algorithms have been proposed to compute GCD such as Dijkstra's algorithm, Euclidian's algorithm, Binary algorithm and Lehmer's algorithm.

2.1 Dijkstra's Algorithm

Dijkstra's algorithm [5] depends on the idea that if: $m|d$ and $n|d$ then $(m-n)\ |d$ with no reminder, as a result we can see: If m > n, $GCD\ (m,n)$ is the same as $GCD\ (m-n,n)$. The flowchart of this algorithm is shown in Figure 1.

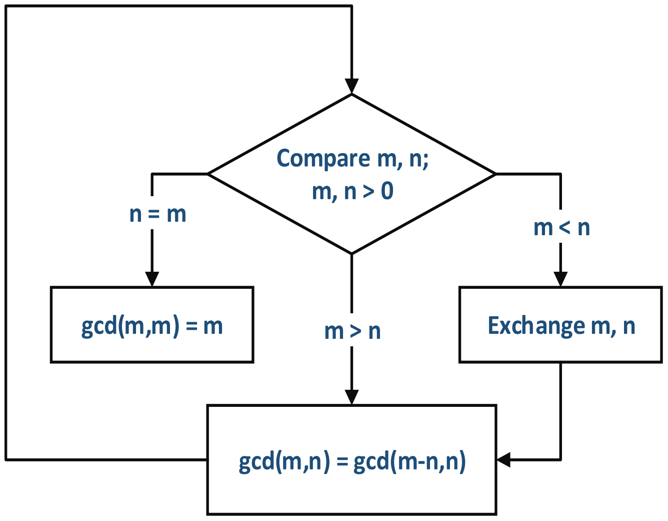

Figure 1. Dijkstra's Algorithm

An example of the algorithm is shown below:

GCD (36, 44) = GCD (36, 8) = GCD (28, 8) = GCD (20, 8) = GCD (12, 8) = GCD (4, 8) = GCD (4, 4) = 4
➔ GCD (36, 44) = 4;

2.2 Euclidian Algorithm

Euclidian algorithm [6] is simple well-known iterative method to calculate GCD. Suppose a and b are both integer numbers and $a > b$, if not exchange between a and b. The algorithm divide a by b and find the reminder. If the reminder is equal to Zero then $GCD\ (a, b)\ =$ a. If not, then divide b by the reminder and find the new reminder. Continue this procedure until the reminder reach zero, the $GCD\ (a, b)$ is the final reminder. The flowchart of this algorithm is shown in Figure 2. From the figure, we can see that the complexity of Euclidian algorithm depends on $n\ =\ (a + b)$. The number of steps can be linear, for e.g. GCD $(x, 1)$, so the time complexity is $O(n)$.

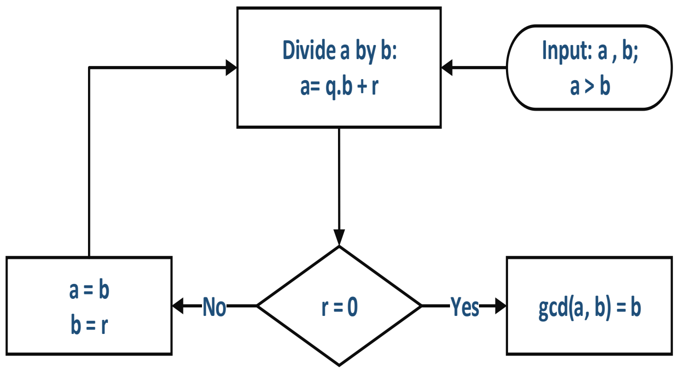

Figure 2. Euclidian Algorithm

An example of the algorithm is shown below:

GCD (24, 16): $24 = 1 \times 16 + 8\,; 8 \neq 0 \rightarrow$
$16 = 2 \times 8 + 0$
Since r = 0 ➔ GCD (24, 16) = 8

2.3 Binary GCD

Binary GCD algorithm [7] replaces the division operations by arithmetic shifts, comparisons, and subtraction depending on the fact that dividing binary numbers by its base 2 is equivalent to the right shift operation. This algorithm is also known as Stein's algorithm which is illustrated in figure 3. The algorithm depends on the following facts:

1. If both a & b are 0, then GCD is zero: $GCD\ (0, 0)\ =\ 0$.
2. $GCD\ (a, 0)\ =\ a$ & $GCD\ (0, b)\ =\ b$ because everything divides 0.
3. If a & b are both even, $GCD\ (a, b)\ =\ 2 * GCD\ (a/2, b/2)$ because 2 is a common divisor. Multiplication with 2 can be done with bitwise shift operator.
4. If a is even & b is odd, $GCD\ (a, b)\ =\ GCD\ (a/2, b)$. Similarly, if a is odd & b is even, then GCD $(a, b)\ =\ GCD\ (a, b/2)$. It is because 2 is not a common divisor.
5. If both a & b are odd, then $GCD\ (a, b)\ =\ GCD\ (|a - b|/2, b)$ where b is min (a, b). Note that difference of two odd numbers is even
6. Then to complete the algorithm repeat steps 3–5 until a = b, or until a = 0. In either case, $GCD(a, b)\ =\ 2^k.b$, where k is the number of common factors of 2 found in step 3.

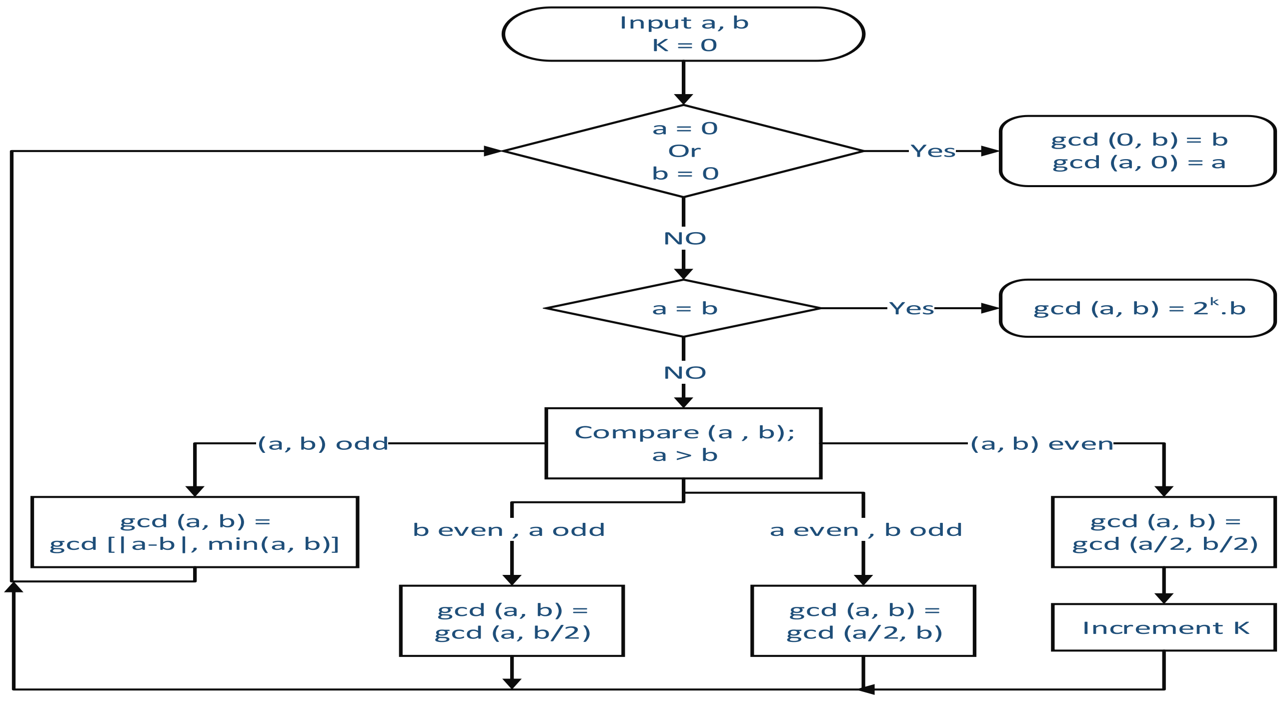

Figure 3. Binary GCD algorithm

The flowchart of this algorithm is shown in fig. 3. It's clearly seen that the complexity of binary GCD algorithm depends of the number of bits in the larger of the two numbers (n) which grow quadratically $(O(n^2))$.

An example of the algorithm is shown below:

GCD (16, 12) = GCD (8, 6) = GCD (4, 3) = GCD (2, 3) = GCD (1, 3) = GCD (2,1) = GCD (1, 1) = GCD (0,1) = 22 = 4

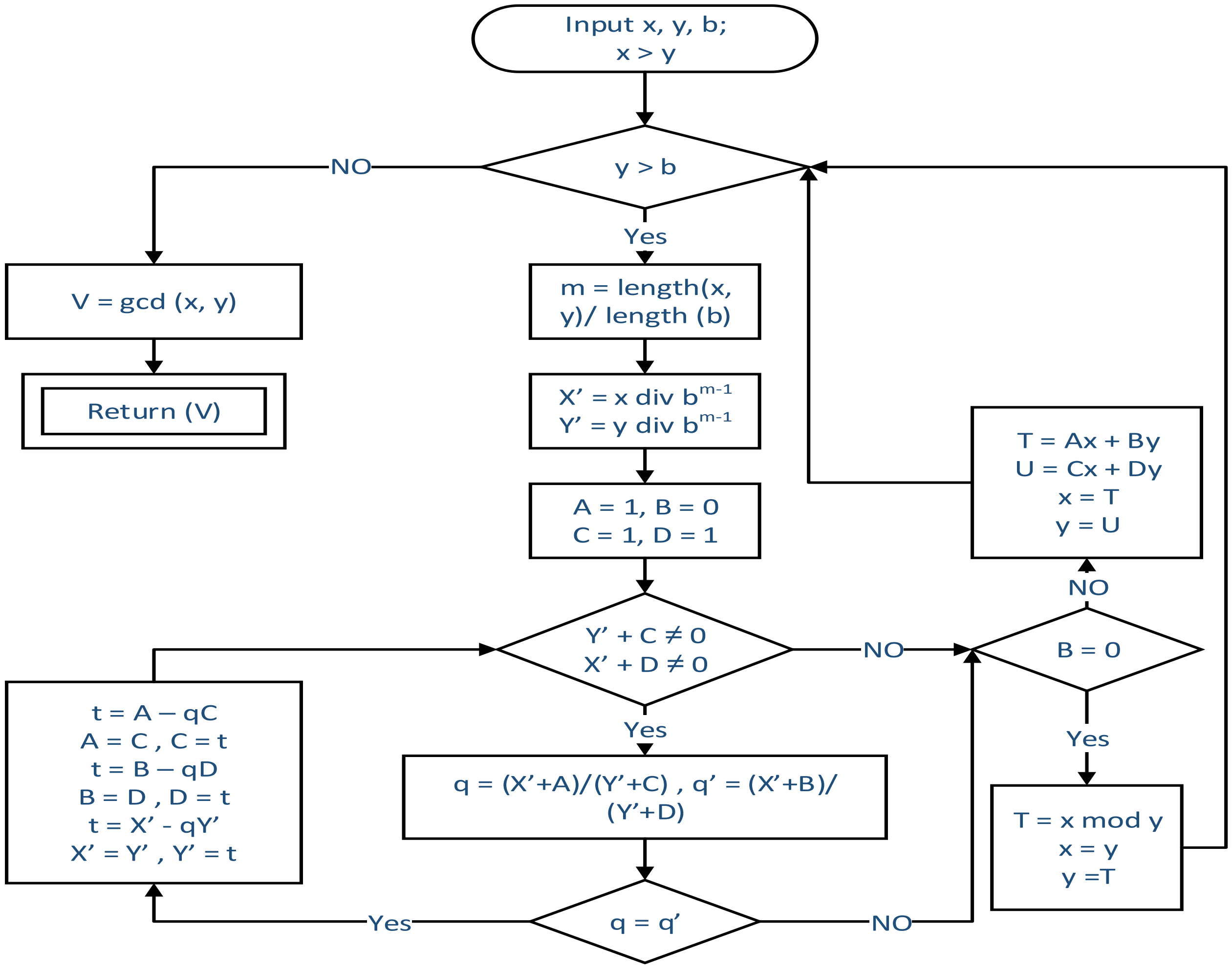

Figure 4. Lehmer's GCD algorithm

2.4 Lehmer's GCD algorithm

When the two numbers of GCD are very long, Euclidean algorithm will take longer time to compute GCD. However, Lehmer's GCD algorithm [8], is a fast GCD algorithm when two numbers are very long. The algorithm decreases the large numbers to smaller numbers of chosen base. The primary advantage of this algorithm is the reduction of the two numbers to the chosen base b. The "div" function in diagram means:

$$X' = x\ div\ b^{1-m} = \left\lfloor \frac{x}{b^{1-m}} \right\rfloor \qquad (1)$$

The function $V = GCD\ (x, y)$ is computed using and simpler GCD algorithms such as Euclidian Algorithm. The flowchart of this algorithm is shown in Figure 4. The cost complexity of lehmer's GCD algorithm is approaching faster with $O\ (n\ /\ log\ n)$.

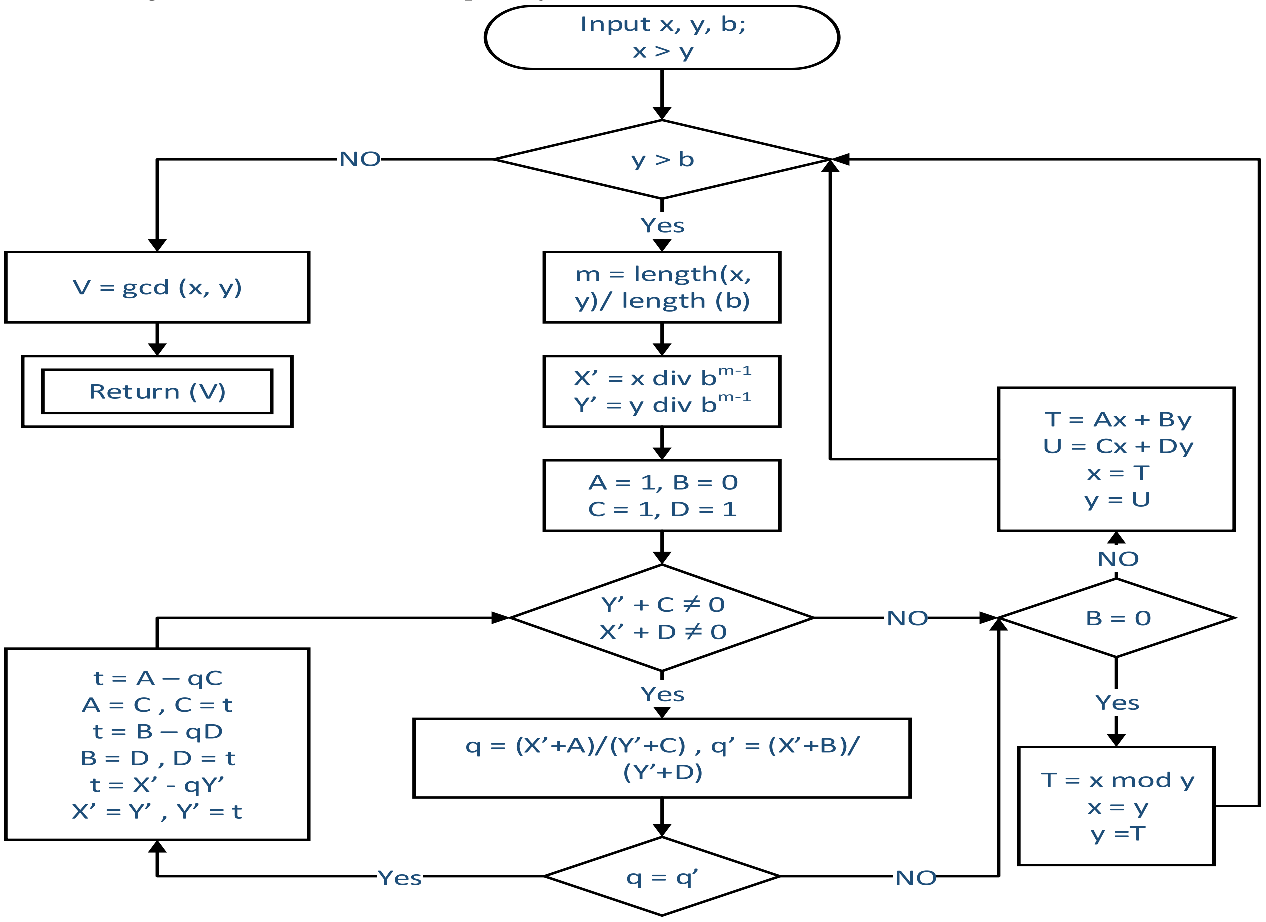

Figure 4. Lehmer's GCD algorithm

An example of the algorithm is shown below: Let: x= 768,454,923 and y= 542,167,814, find $GCD\ (x, y)$. Firstly, we use Lehmer's algorithm to reduce x and y to the chosen base b = 103. Now, the high-order digit of x and y are x' = 768 and y' = 542 then run the algorithm as in table 1 and the table 2.

Table 2. Lehmer's GCD algorithm Example

x	y	q	q'	Reference
768,454,923	542,167,814	1	1	i
89,593,596	47,099,917	1	1	ii
42,493,679	4,606,238	10	8	
4,606,238	1,037,537	5	2	
1,037,537	456,090	5	2	
456,090	125,357	3	3	iii
34,681	10,657	3	3	iv
10,657	2,710	5	3	
2,710	2,527	1	0	
2,527	183			

Table 2. Reference Table

	x'	y'	A	B	C	D	q	q'
i	768	542	1	0	0	1	1	1
	542	226	0	1	1	-1	2	2
	226	90	1	-1	-2	3	2	2
	90	46	-2	3	5	-7	1	2
ii	89	47	1	0	0	1	1	1
	47	42	0	1	1	-1	1	1
	42	8	1	-1	-1	2	10	5

iii	456	425	1	0	0	1	3	3
	125	81	0	1	1	-3	1	1
	81	44	1	-3	-1	4	1	1
	44	37	-1	4	2	-7	1	1
	37	7	2	-7	-3	11	9	1
iv	34	10	1	0	0	1	3	3
	10	4	0	1	1	-3	2	11

Finally, we get x = 2,527 and y = 183, therefore, $v = GCD\,(x, y) = GCD\,(2{,}527, 183)$ which can be calculated easily using Euclidian's algorithm.

3 LEAST COMMON MULTIPLE (LCM)

LCM is well known elementary number theory algorithm that is defined as the smallest multiple integer of two numbers. The very basic method to calculate LCM is using prime factorization method (PF-LCM).

PF-LCM method depends on fact that each integer number can uniquely written as a product of prime numbers raised to different powers. Then, we can calculate LCM by taking all common factors with greatest power.

An example of the algorithm is shown below:

40 = 235, 26 = 2.13➔ LCM (40, 26) = 23.5.13 = 520

Alternatively, LCM can be efficiently calculated by using the GCD reduction method [9]. LCM is usually calculated using GCD from the formula:

$$LCM\,(a, b) = \frac{ab}{\mathrm{GCD}\ (a, b)} = \left(\frac{a}{\mathrm{GCD}(a, b)}\right) b$$

Where GCD can be calculated using different algorithms without need to factor the numbers.

4 CONCLUSIONS

GCD/LCM operations are essential number theory algorithms that commonly used as underlying operations of many computing processors. The efficient utilization of such algorithms contributes in the overall leverage of system execution. For instance, Lehmer's algorithm reduces the large integers to common base (b) with fast convergence rate at complexity of O (n/log (n)) which make it faster than other algorithms. Eventually, Lehmer's algorithm can be efficiently implemented (in hardware or software) to compute both GCD and LCM operations for any two large integer numbers.

REFERENCES

[1]. A.J. Menezes, P.C. Van Oorschot and S.A. Vanstone. (1996). Handbook of Applied Cryptography", CRC Press, Boca Raton, Florida

[2]. W. Trappe and L. C. Washington, (2002) 'Introduction to Cryptography with Coding Theory', Prentice Hall, vol. 1: p.p. 1-176.

[3]. Q. Abu Al-Haija, M. Smadi, M. Jaffri and A. Shua'ibi, "Efficient FPGA Implementation of RSA Coprocessor Using Scalable Modules", 9th International Conference on Future Networks and Communications (FNC-2014), by Elsevier, Ontario, Canada, 17-20, Aug-2014.

[4]. M. R. K. Ariffin, M. A. Asbullah, N. A. Abu and Z. Mahad, " A New Efficient Asymmetric Cryptosystem Based on the Integer Factorization Problem of $\boldsymbol{N} = \boldsymbol{p^2q}$", Malaysian Journal of Mathematical Sciences 7(S): 19-37 (2013)

[5]. K. J. Goldman, "Recursive Algorithms", Computer Science I, lecture notes, Washington University in St. Louis, 1997.

[6]. Q. A. Al-Haija, M. Al-Ja'fari and M. Smadi, (2016) 'A comparative study up to 1024-bit Euclid's GCD algorithm FPGA implementation & synthesizing', 2016 5th International Conference on Electronic Devices, Systems and Applications (ICEDSA), Ras Al Khaimah, United Arab Emirates, pp. 1-4.

[7]. D. Stehlé and P. Zimmermann (2004), "A binary recursive gcd algorithm", Algorithmic number theory (PDF), Lecture Notes in Comput. Sci., 3076, Springer, Berlin, pp. 411–425, MR 2138011, doi:10.1007/978-3-540-24847-7_31.

[8]. S. M. Sedjelmac, " On a Parallel Lehmer-Euclid GCD Algorithm", International Symposium on Symbolic and Algebraic Computation (ISSAC 2001), By ACM, UWO, Canada, 2001.

[9]. W. Stein, (2011) 'Elementary Number Theory: Primes, Congruence, and Secrets', Springer, vol. 1.

[10]. J. Sorenson. (1995). "An analysis of Lehmer's Euclidean GCD algorithm", Proceedings of the 1995 International Symposium on Symbolic and Algebraic Computation, pp. 254-258. Montreal, Canada, ACM Press.

[11]. Sidi Mohammed Sedjelmaci, “On a parallel Lehmer-Euclid GCD algorithm”, Proceedings of the 2001 international symposium on Symbolic and algebraic computation, p.303-308, July 2001, London, Ontario, Canada. doi:10.1145/384101.384142
[12]. T. Jebelean. “A double-digit Lehmer–Euclid algorithm for finding the GCD of long integers”, Journal of Symbolic Computation, 19 (1995), pp. 145-157
[13]. Jebelean, T., 1993. “A generalization of the binary GCD algorithm”. In Proceedings of the 1993 international symposium on Symbolic and algebraic computation (pp. 111-116). ACM.
[14]. Wang, P.S., 1980. “The eez-gcd algorithm”. ACM SIGSAM Bulletin, 14(2), pp.50-60.
[15]. Sasaki, T. and Suzuki, M., 1992. “Three new algorithms for multivariate polynomial GCD”. Journal of symbolic computation, 13(4), pp.395-411.
[16]. T. Jebelean, "Comparing Several GCD Algorithms", ARITH-11: IEEE Symposium on Computer Arithmetic, 1993-June.
[17]. S. M. Sedjelmaci, "On a parallel extended Euclidean algorithm," Proceedings ACS/IEEE International Conference on Computer Systems and Applications, Beirut, 2001, pp. 235-241.