Indian Journal of Science and Technology, Vol 9(30), DOI: 10.17485/ijst/2016/v9i29/90830, August 2016
ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645

An Improved VLSI Algorithm for Modular Operation in Cryptography

G. G. Bremiga*, M. Malleswari and Sharmini Enoch

Department of Electronics and Communication Engineering, Noorul Islam University, Kumaracoil, Kanyakumari District - 629180, Tamil Nadu, India; bremigagg@gmail.com, malleswarim@gmail.com, sharminienoch@gmail.com

Abstract

Objectives: This paper presents a new proposed algorithm which performs an efficient modular multiplication method which is advantage because of its reduction in hardware and software. This proposed method implies a systematic approach which increases the parallelism level when compared to the previous versions. **Methods/Analysis:** Two conventional methods are effectively used to find the modular multiplication output. The previous work effectively combines the first conventional and next two algorithms which are invented to overcome the disadvantages of the first two algorithms. The proposed method effectively eliminates one conventional method. **Findings:** This process reduces the number of iterations hence, reducing the time consumption required to synthesis the entire algorithm. Thus, the above mentioned method efficiently condenses the hardware utilization for implementing the conventional and previous algorithm so far practiced before. **Application/Improvements:** This paper replaces the classical algorithm by other method which effectively reduces the number of iterations. This reduction in computation makes a drastic reduction in hardware and time delay to execute the algorithms. This paper shows a modification in the existing parallelism method which further shows a great improvement in reduction of hardware and time delay.

Keywords: Public-Key Cryptography, Modular multiplication, Classical Algorithm, Montgomery Algorithm, Bipartite method, Tripartite Method.

1. Introduction

The vital nucleus in the domain of cryptography is the modular operation. This modular operation is widely used in many public-key cryptographic applications. The world wide practicing cryptosystems such as RSA scheme[1], ElGamal[2] and Diffie-Hellman key exchange[3] extensively use larger number of modulus value which further makes the modular operation more complex and time consuming to get the final value. The modular operations in cryptography involve either repeated modular multiplication or repeated modular division or both which involves larger modulus value. For higher level cryptosystems, the cryptographic algorithms are implemented in higher performance hardware and the utilization of parallelism is exploited to accomplish maximum throughput.

Modular multiplication is the process of multiplying two operands and then attaining its modular value. There are many algorithms which are accomplished to calculate this modular value. Of these, the Classical and Montgomery algorithm are widely used to find the modular value in the twentieth century times. Later, both the Classical and Montgomery algorithms are combined together into a single method known as Bipartite Method and are then used to find the modular value of the multiplied operands. Here both the Classical and Montgomery computation are done in parallel and they are finally added together. At present, the Tripartite method is currently put into practice. In this method, the level of parallelism is further increased so that the computation of both the Classical and the Montgomery method dealt out are performed in a faster way. The classical method is the

*Author for correspondence

most time consuming and hardware utilizing algorithm. This paper shows an approach of completely replacing the Classical method by the Interleaved Multiplication Algorithm. This proposed method shows a significant advantage when compared to the previous algorithms in provisions to hardware utilization and time delay for synthesizing and implementing the algorithm.

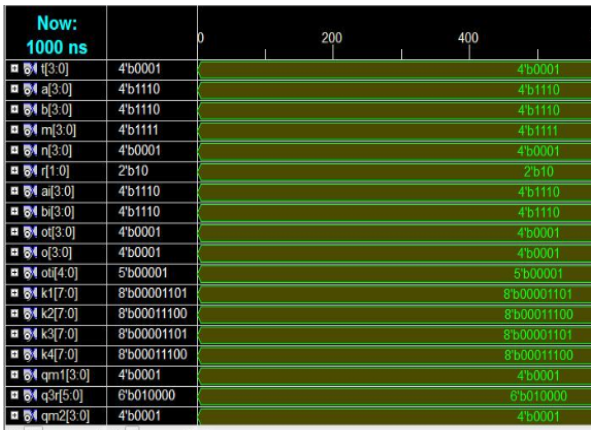

Figure 1. Simulation results of Classical Method.

2. Materials and Methods

2.1 Classical Modular Multiplication Algorithm

The first algorithm which was followed so far is the Classical algorithm proposed by G. R. Blackey in 1983[4]. The brief explanation of the same algorithm is explained in the work done by K.R. Sloan[5]. This method is the conventional way of computing the modular value for the multiplication of two input operands. It is mandatory that these input operands should be in residue class ring of integers. Assume the two k bit operands whose modular value has to be obtained are R and S which are in residue class ring of integers, where k is an integer. The k bit modulus value is taken as T, which should be considerably chosen between w^{k-1} to w^k, where w is th[illegible] of each digit in modulus T and k is the number o[illegible] The value of k is usually chosen to be 2^r, where r is [illegible] ord size. The second operant S can be written as $S = \sum_{j=0}^{k-1} s_j * r^j$ for computation purposes.

For two integer inputs R, S and k digit odd modulus value T, the Classical Modular Multiplication method is defined as,

$$O = R * S \bmod T$$

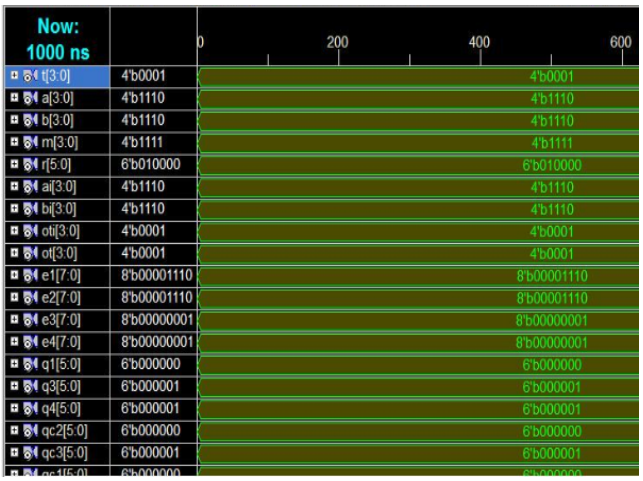

Figure 2. Simulation results of Montgomery Method.

2.1.1 Algorithm

Input: T: $w^{k-1} \le T < w^k$, where k is the number of bits.
R, S: $0 \le R, S < T$
Output: $O = R * S \bmod T$
Algorithm:

```
O = 0;
for j = k − 1 downto 0 do
    O = w * O + sj * R;
    Uc = [ O / T ];
    O = O − Uc * T;
End for
```

2.2 Montgomery Modular Multiplication Algorithm

The second algorithm Montgomery Modular Multiplication Algorithm was introduced by P. L. Montgomery in 1985[6]. The hardware and time consumption of the previously practiced algorithm is larger since the processing is based on the k bit modulus value T. The procedure of calculating Uc = [O/T] is an typical time consuming and expensive operation. This operation can be overcome by using the Montgomery method as its processing is based on the radix w. The two operands R and S should be in residue class ring of integers. The two input elements H and N are converted into residue class ring of integers modulo M and are taken as R and S. These converted integers are entitled as image or Montgomery residue. This image R and S are then given as input to the Montgomery algorithm.

For a given k bit modulus and image inputs R and S, the Montgomery Modular Multiplication is described as,

$O = R * S * W^{-1} \bmod T$

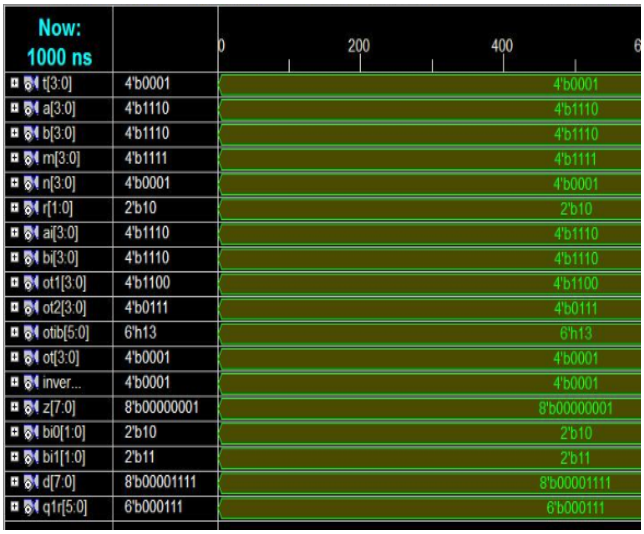

Figure 3. Simulation results of Bipartite Method.

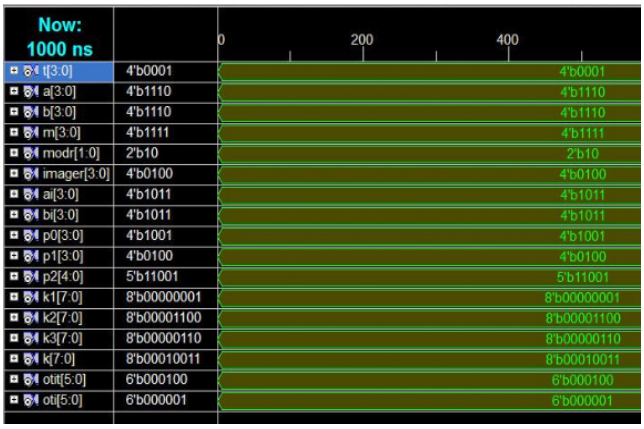

Figure 4. Simulation results of Tripartite Method.

The transformation of integer set to Montgomery residue is done using the Classical method by calculating R = H * W mod T and S = N * W mod T. The term W is obtained from $W = w^k$, where w is the radix of each digit in the operands and k is an integer. The same transformation can be done by applying the Montgomery method to H and W^2 mod T, where W^2 mod T is pre-computed. Similarly the next image is obtained by applying the Montgomery method to R and W^2 mod T. The result obtained is always in Montgomery residue or in image form. The reverse transformation from Montgomery residue or image to original integer set is achieved by applying the Montgomery method to this result obtained and integer 1

2.2.1 Algorithm

Input: T: $w^{k-1} \le T < w^k$, where k is the number of bits.
R, S: $0 \le R, S < T$

Output: $O = R * S * W^{-1} \bmod T$

Algorithm:

```
O = 0;
for j = 0 to n − 1 do
        O = O + sj * R;
        UM = ( ( O mod w ) T′ ) mod w;
        O = ( O + UM * T ) / w;
endfor
  if O ≥ T then O = O − T;
```

This algorithm uses a pre-computed value T′ whose value is to be produced from $T' = -T^{-1} \bmod w$. For simple level of computation, the radix w value is always assumed to be $W = w^k$, where k the radix of each digit. The value of k is given by $k = 2^r$ where r is the corresponding word size (usually r = 1). The condition D > N is always used in the method to assure G is bounded within 2N. The Montgomery method implies the condition of W > T to make sure that the output T is bounded between 2W. This condition completely reduces any extra arithmetic operation to stabilize the integer output.

Table 1. Comparison of modular multiplication algorithms

Hardware Parameters					
Classical		Montgomery	Bipartite	Tripartite	Proposed
No. of Slices	1141	960	603	465	388
LUTs	2022	1744	1045	808	677
Adders/Subtractors	213	173	115	93	53
Comparators	194	97	90	70	32
Time Delay Parameters					
Path Delay	358.443ns	259.023ns	156.693ns	140.753ns	99.794ns
CPU	84s	58.30s	28.94s	21.06s	17.13s

2.3 Bipartite Modular Multiplication Method

The third algorithm is the Bipartite Modular Multiplication Method proposed by Marcelo E. Kaihara and Naofumi Takagi in 2008[7]. In this method one of the input elements S is split into two halves as SH and SL. The condition D < N should be satisfied with $W = w^n$ where n is chosen as $0 < n < k$. For ease of computation n is always taken as $n = k/2$. It is represented as $S = SH * w^k + SL$ where $SH < w^{k-n}$ and $SL < w^n$. The upper half calculation $R * SH \mod T$ is done by using the Classical method and the lower half calculation $R * SL * W^{-1} \mod T$ is done using the Montgomery method. These two values are finally added into a single value. All the calculations are done in image or Montgomery residue format.

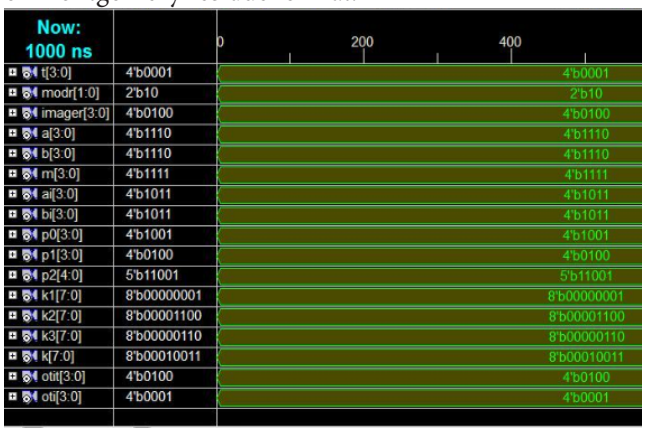

Figure 5. Simulation results of Proposed Method.

For a given k bit modulus and image inputs R and S, the Bipartite Modular Multiplication Method is described as,

$$O = R * S * W^{-1} \mod T$$
$$= (R * (SH * W + SL) * W^{-1} \mod T$$
$$= (R * SH \mod T + R * SL * W^{-1} \mod T) \mod T$$

The image form R and S from integer set H and N can be obtained by giving $R = H * W \mod T$ and $S = N * W \mod T$ to the classical algorithm. The same transformation can also be obtained from Bipartite Modular Multiplication method by computing $R = H * W^2$ and $S = N * W^2$, where $W^2 = w^{2k} \mod T$ is pre-computed. These Montgomery residues are given as input to the Bipartite method and result obtained hence is in image or Montgomery residue form. The inverse image conversion from Montgomery residue to original integer set can be calculated either by giving the Bipartite algorithm to this result obtained along with integer 1 or by calculating $H = Output * 1 * w^k \mod T$ using the Montgomery algorithm. This value gives the final result which is the modular value of two input operands that are multiplied together. Since one of the operand is partitioned into half, half the number of iterations is enough to compute the entire process. This shows a greater advantage when the cryptosystems encompass with higher level of bits.

2.4 Tripartite Modular Multiplication Method

The latest method so far practicing is the Tripartite algorithm proposed by Kazuo Sakiyama, Miroslav Knezevic, Junfeng Fan, Bart Prenee, and Ingrid Verbauwhede in 2011[8]. This method shows more parallelism that makes the computation of Modular Multiplication more efficiently. Here, both the operands are split into two halves and they are divided into three components of computation. The processing of these three components are done separately and finally all the three valued attained are added together to get the final modular value. As the computations are done in three ways this parallel processing will consume least amount of time delay and hardware.

All the input elements should be in the residue class ring of integers. The condition of D < N must be satisfied with $W = w^k$ where k is chosen as $0 < n < k$. Both the operands are split into upper half and lower half as $R = RH * w^k + RL$ and $S = SH * w^k + SL$ where RH, SH < w^{k-n} and RL, SL < w^n.

For the k digit odd modulus and two input operands split into upper half and lower half, the tripartite method is given as,

$$O = R * S * W^{-1} \mod T$$
$$= (RH * W + RL)(SH * W + SL) * W^{-1} \mod T$$
$$= (RH * SH W + (RH * SL + RL * SH) + RL * SL W^{-1}) \mod T$$
$$= \{ z1 * W \mod T + (z2 + z0 + z1) \mod M + z0 * W^{-1} \mod T \} \mod T$$

where,

$z0 = RL * SL$, $z1 = RH * SH$ and $z2 = (RH + RL)(SH + SL)$.

The computation of $z1 * W \mod T$ and $z0 * W^{-1} \mod T$ is done using the Classical method and the Montgomery method. The computation of $(z2 + z0 + z1) \mod M$ involves merely a modular program. The Tripartite method directly implies the value of $W = w^n$, where n can be determined from $n = k/2$. This makes the estimation of hardware much lesser when compared to conventional means.

2.5 Proposed Modular Multiplication Method

The proposed method startlingly decreases maximum amount of hardware utilization and trim down the time consumption required for the processor to synthesis and simulate the hardware description language inscribed to process the modular multiplication algorithm. This change is due to complete elimination of the Classical algorithm which involves one division algorithmic step which is expensive while constructing the cryptosystem hardware. The Classical method is replaced using Interleaved Modular Multiplication which completely replaces the heavy division step by repeated subtraction method.

2.5.1 Issue

The general tripartite modular multiplication method involves three parts of computation. The first part z1 * W mod T employs the Classical method and the second part z0 * W^{-1} mod T uses the Montgomery method while the third part apply the basic modular operation. Of these, the Classical method is the most complex algorithm in which the difficulty be positioned in the following step, Uc = [O/T]. This step needs a division program which has a complexity in the order of n^2. If the input operands are taken in radix 2 and for initial k value as 4, the intermediate quotient calculation needs a division program which runs for $O(n^2)$. This higher level of iteration for executing $O(n^2)$, produces greater number of iterations to get the quotient value Uc. This in turn increases the number of hardware and time delay needed for the calculation of Uc. This is one considerable disadvantage on using the Classical method.

2.5.2 Solution

The conventional algorithm computation is replaced using Interleaved Modular Multiplication Method. This method directly subtracts the divisor by dividend repeatedly until getting a value lesser than the dividend instead of implementing a repeated division program which runs until the iteration ceases to $O(n^2)$. This method completely replaces set of instructions needed for performing the division with recurring subtraction. The important aspect lies in the point that these repetitive subtraction need not to be in the order of $O(n^2)$. The computation of subtraction can be done twice or thrice to get the final value and this working out is more than essential to get the modular value using the classical method

2.5.3 Algorithm

The Classical algorithm has been replaced by a simple Interleaved Modular Multiplication program which is similar to normal way multiplication which will be a initially

- a bit by bit multiplication (LSB bit of second multiplied by the first operand),
- a shift operation to accommodate the next level partial product,
- addition for the parallel level partial products and
- finally subtracting the final value from the modulus to obtain the final modular value of the multiplication of to operands.

2.5.4 Advantage

The following are the advantages on implementing the above level modification.

- The procedure for calculating the division problem is eliminated
- i.e.., Uc = [O/T].
- As this procedure is a time consuming operation, on doing this repeated multiplication by modulus T by increasing integers to precede multiplication is completely eliminated.
- This in turn reduces consequent small level operations too (subtraction and addition used to perform division), thereby lowering the number of adders and subtractions.
- Apart from this as total operand size multiplication is completely eliminated, this in turn drastically reduces the number of slices and Look-up-tables used to perform the entire multiplication.

In short, this new method replaces Classical algorithm by new and simple way of modular multiplication which reduces the number of multiplications, additions and subtractions used for division program. This improvement shows the following significant advantage in reduction of number of slices, adders/subtractions and comparators. On implementing this both Bipartite Algorithm and Tripartite Algorithm itself will become the Proposed Algorithm.

3. Results and Discussion

The following figures titled Figure 1, Figure 2, Figure 3, Figure 4 and Figure 5 implies the simulation results of the Classical, Montgomery, Bipartite, Tripartite and for the Proposed algorithm. The result analysis had been done for all possible values of four bit wide. The same algorithms were also extended for higher level of bits. The output is given in the terminal t. All the above algorithms are coded in VLSI Hardware Description Language. The HDL codes are synthesized and simulated in Xilinx 9.1i version of ISE simulator. The simulation results are compared with the manually calculated value and the logic of HDL code is verified.

The comparison table for all the above algorithms are based on the hardware utilized and time delays consumed for synthesis and to implement all the above stated algorithms. The hardware computation can be summarized by considering the amount of slices, LUTs used, adders/subtractions and total number of comparators needed. The time make use of can be stated from the CPU time usage and time delay. The comparison table is stated in Table 1. Thus the above results show that the proposed algorithm produces significant reduction in time delay and in hardware computation.

4. Conclusion and Future Work

These results clearly depicts that the proposed algorithm produces a significant advantage in hardware and time consumption in executing and implementing the algorithm. This drastic reduction in hardware and time delay is due to complete elimination of the classical algorithm by normal interleaved multiplication method. On practicing this scenario, the bulk program which is needed to execute the heavy division algorithm is completely avoided. The replacement of the division process by repeated subtractions is the reason behind the trim down of the hardware. This makes a greater advantage while implementing the cryptosystems while considering heavy input operands of larger size.

The current work is extended to make any modifications further to produce a significant reduction in hardware utilization and time delay. This may be achieved by reducing the steps in algorithm which skip a bulk operation in this case a division operation, say. The future work focuses on the modular multiplication to produce a speed in the calculation of final value. Any improvement in algorithm is done to enhance the efficiency and speed of the entire processing.

5. Acknowledgement

The first author thanks Head of the department, research guide and joint supervisor, Department of Electronics and Communication Engineering, Noorul Islam University, Noorul Islam Center for Higher Education, Kumaracoil for the support and the facilities provided during this project. Also thank all the project committee members for their suggestions and valuable ideas during the project review.

6. References

1. Rivest RLShamir, A, Adleman L. A Method for Obtaining Digital Signatures and Public-Key Cryptosystems, Comm ACM. 1978 Feb; 21(2):120–6.
2. Diffie W, Hellman ME. New Directions in Cryptography. IEEE Trans Information Theory. 1976 Nov; 22(11):644–54.
3. ElGamal T. A Public Key Cryptosystem and a Signature Scheme based on Discrete Logarithms. IEEE Trans Information Theory. 1985 Jul; 31(4):469–72.
4. Blakley GR. A Computer Algorithm for Calculating the Product AB Modulo M. IEEE Trans Computers. 1983 May; 32(5):497–500.
5. Sloan KR. Comments on a Computer Algorithm for Calculating the Product AB Modulo M. IEEE Trans. Computers. 1985 Mar; 34(3):290–2.
6. Montgomery PL. Modular Multiplication without Trial Division. Mathematics of Computation. 1985 Apr; 44(170):519–21.
7. Kaihara ME, Takagi N. Bipartite Modular Multiplication Method. IEEE Trans Computers. 2008 Feb; 57(2):157–64.
8. Sakiyama K, Knezevic M, Fan J, Prenee B, Verbauwhede I. Tripartite Modular Multiplication, Intergration. The VLSI Journal. 2011 Sep; 44(4):259–69.
9. George Amalarethinam DI, Sai Geetha J, Mani K. Analysis and Enhancement of Speed in Public Key Cryptography using Message Encoding Algorithm. Indian Journal of Science and Technology. 2015 Jul; 8(16). Doi: 10.17485/ijst/2015/v8i16/69809.
10. Vijayakumar P, Indupriya S, Rajashree R. A Hybrid Multilevel Security Scheme using DNA Computing based Color Code and Elliptic Curve Cryptography. Indian Journal of Science and Technology. 2016 Mar; 9(10). Doi:10.17485/ijst/2016/v9i10/88987.
11. Dawahdeh ZE, Yaakob SN, Sagheer AM. Modified ElGamal Elliptic Curve Cryptosystem using Hexadecimal

Representation. Indian Journal of Science and Technology. 2015 Jul; 8(15). Doi: 10.17485/ijst/2015/v8i15/64749.
12. Kalaivani D, Karthikeyen S. VLSI Implementation of Area-Efficient and Low Power OFDM Transmitter and Receiver. Indian Journal of Science and Technology. 2015 Aug; 8(18). Doi:10.17485/ijst/2015/v8i18/63062.