ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

An ISO 3297: 2007 Certified Organization *Vol. 5, Special Issue 4, March 2016*

National Conference on Signal Processing, Instrumentation and Communication Engineering (SPICE' 16)

Organized by

Dept. of ECE, Mar Baselios Institute of Technology & Science (MBITS), Kothamangalam, Kerala-686693, India

Montgomery Algorithm for Modular Multiplication in Cryptosystems

Nitha Thampi[1], Meenu Elizabath Jose[2]

PG Scholar, Dept. of ECE, College of Engineering, Munnar, Kerala, India[1]

Assistant Professor, Dept. of ECE, College of Engineering, Munnar, Kerala, India[2]

ABSTRACT: The majority of currently established public-key cryptosystems.RSA, ECC, requires modular multiplication in finite fields as their core operation. As a result, the throughput rate of such cryptosystem depends upon the speed of modular multiplication and upon the number of performed modular multiplications. Montgomery algorithm is one method that allows efficient implementation of multiplication modulo large number, as required by the RSA cryptosystem. Many hardware and software implementations for faster modular multiplication have been proposed, Montgomery Multiplication Algorithm is recognized as the most efficient among these. This paper presents a 32-bit implementation of a Faster Montgomery algorithm for performing modular multiplication. The algorithm is based on Montgomery method for modular multiplication and is complementary to the available techniques. Simulation shows that our design performs faster in terms of clock frequency while it requires lower area.

KEYWORDS: Cryptosystems, Encryption, Montgomery Multiplier.

I. INTRODUCTION

In this age of universal electronic connectivity, of electronic eavesdropping and fraud, it is of utmost importance to store information securely. This led to a heightened awareness to protect the data from disclosure, to guarantee the authenticity of data and messages, and to protect systems from network-based attacks. Cryptography plays a major role in mobile phone communications, e-commerce, pay-tv, sending private emails, transmitting financial information, security of ATM cards, computer passwords, electronic commerce digital signature and so on. Modular exponentiation ab mod m, and implicitly modular multiplication a · c mod m are the operations intensively used, underlying many cryptographic schemes. Rivest, Shamir, and Adleman (RSA) is one of the most widely adopted public key algorithms at present. RSA requires repeated modular multiplications to accomplish the computation of modular exponentiation [1]. Processing these cryptosystems requires a huge amount of computation and there is, therefore, a great demand for developing dedicated hardware to speed up the computations. Additionally, security requirements are increasingly important for private data transmission through mobile devices with Internet access, such as smart phones and notebook computers, which require an energy-efficient cryptosystem due to their limitation of battery power. For this kind of application, it is necessary to develop efficient hardware architectures to carry out fast modular multiplications with low energy consumption.

As the division operation in modular reduction is time-consuming, Montgomery proposed a new algorithm where division is avoided. Montgomery Multiplication(MM) algorithm is an effective way to perform modular multiplication, while avoiding division by the large modulus. The majority of existing RSA algorithms use Montgomery method and high radix number systems. It is an efficient method for modular multiplication with an arbitrary modulus. The algorithm uses simple divisions by a power of two instead of divisions by M, which are used in a conventional modular operation. An integer Z is represented as Z* R mod M, where M is the modulo and R = 2r is a radix that is co prime to M. This representation is called Montgomery residue. Multiplication is performed in this residue, and division by M is replaced with division by R. The Montgomery multiplication is the basic operation used in modular exponentiation, which is required in the Diffie-Hellman and RSA public-key cryptosystems In order to get improved performance, high-radix algorithms and designs can be used. However, these designs are usually complex and it is not so evident whether they provide the desired speed gain.Low-radix designs are usually more attractive for hardware implementation.
This paper compares the available techniques for modular multiplication and explains faster modular multiplication architecture. Section II contains a brief discussion on the various cryptographic techniques. A description of the general

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

An ISO 3297: 2007 Certified Organization *Vol. 5, Special Issue 4, March 2016*

National Conference on Signal Processing, Instrumentation and Communication Engineering (SPICE' 16)

Organized by

Dept. of ECE, Mar Baselios Institute of Technology & Science (MBITS), Kothamangalam, Kerala-686693, India

theoretical aspects of the modular multiplication is given in Section III. Montgomery techniques and algorithm is presented in Section IV. In section V, a radix 2 algorithm for faster modular multiplication for a large modulus suitable for VLSI implementation is proposed. It is a faster Montgomery modular multiplication algorithm for high-speed hardware design. The analysis results obtained using a design synthesis is presented in Section VI.

II. MODULAR ARITHMETIC IN RSA CRYPTOGRAPHY

Cryptography is an essential aspect for secure communication. It is the practice and study of techniques for secure communication in the presence of third parties called adversaries. The rising growth of data communication and electronic transactions over the internet has made security to become the most important issue over the network. Cryptography not only protects data from theft or alteration, but can also provide Privacy, confidentiality, user authentication, Integrity, Non-repudiation. Two types of cryptographic schemes are used mainly to accomplish these goals. It includes secret key or symmetric cryptography (DES), public-key or asymmetric cryptography (RSA). The asymmetric key algorithm requires two different keys, one for encryption and other for decryption.

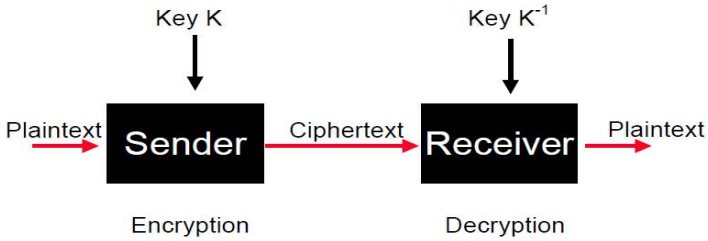

Fig. 1: Block Diagram of Cryptosystem

The RSA algorithm is a secure, high quality, public key algorithm. It can be used as a method of exchanging secret information such as keys and producing digital signatures. It uses modular exponentiation of large numbers to encrypt data, which is a slow process due to repeated modular multiplications. Encryption is the process of converting a plain text into a format which is not easily readable and is called as cipher. The conversion from plain text to cipher text involved some mathematical operation only. Hence after generation of both keys, the RSA encryption/decryption is just a modular exponentiation operation. This mathematical operation is represented as:
Plaintext block M is encrypted to a cipher text block C by:
$C = M^e \bmod m$
The plaintext block is recovered by:
$M = C^d \bmod m$
where C is cipher text, M is plain text, E is the public key exponent, and m is the modulus. This operation has involved a few modular operations like modular multiplication, modular addition, and subtraction. RSA also requires modular multiplication for private key generation. Thus the efficiency of an RSA encryption system depends on the speed of modular multiplications.

III. MODULAR MULTIPLICATION

Modular multiplication problem is defined as the computation of $P = A \times B \pmod m$, given the integers A, B and m. It is usually assumed that A and B are positive integers with $0 \leq A, B < m$ [2]. The square or multiplication operation is just a simple multiplication. There are many approaches to perform multiplication such as multiply then divide, interleaving multiplication and reduction, Brickell's method. Normally modular multiplication is done by repeated subtraction of modulus from the multiplicand until the result is smaller than modulus. This technique is time consuming when the value of modulus is too large. Modular Multiplication can be also performed by division of the modulus. Division is again a time consuming task.
Modular exponentiation ($a^n \bmod m$) and its key constituent operation, modular multiplication ($a \cdot b \bmod m$), are the fundamental operations underlying cryptographic algorithms. Since modular multiplications account for most of the time spent for encryption and decryption, their optimization is crucial. This can be achieved either by reducing the number of modular multiplications or by reducing the latency of each modular multiplication. Modular exponentiation operation can further simplified in to series of modular multiplication and squaring operation. This simplification is

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

An ISO 3297: 2007 Certified Organization Vol. 5, Special Issue 4, March 2016

National Conference on Signal Processing, Instrumentation and Communication Engineering (SPICE' 16)

Organized by

Dept. of ECE, Mar Baselios Institute of Technology & Science (MBITS), Kothamangalam, Kerala-686693, India

based on an algorithm known as square and multiply algorithm. This algorithm is based on scanning the bit of the exponent from the left (the most significant bit) to the right (the least significant bit). In every iteration, i.e., for every exponent bit, the current result is squared, If and only if the currently scanned exponent bit has the value 1, a multiplication of the current result by M is executed following the squaring.

Algorithm for Modular Exponentiation
Input: M, e, n
Output: $C = M^e \bmod n$
Let e contain k bits)
If $e_{k-1}=1$ then C=M
else C=1
For i=k-2 down to 0
C=C×C
If $e_i=1$ then C=C×M

IV. MONTGOMERY MULTIPLICATION

In 1985 a method for modular multiplication using Residue Number System (RNS) representation of integers is proposed by Peter L. Montgomery. In this method the costly division operation usually needed to perform modular reduction is replaced by simple shift operations by transforming the operands into the RNS domain before the operation and re-transforming the result after operation. A radix R is selected to be two to the power of a multiple of the word size and greater than the modulus, i.e. $R = 2^w > M$. For the algorithm to work R and M need to be relatively prime i.e. must not have any common non-trivial divisors. With R a power of two, this requirement is easily satisfied by selecting an odd modulus. This also fits in nicely with the cryptographic algorithms that we are targeting, where the modulus is either a prime always odd with the exception of 2 or the product of two primes and therefore odd as well.
The Montgomery algorithm computes $c = (a*b*(2n)^{-1}) \bmod M$ [3]. The idea of Montgomery is to keep the lengths of the intermediate results minimum.RNS representations of integers are called M residues and are usually denominated as the integer variable name with a bar above it. An integer a is transformed into its corresponding M-residue $\bar{a}$ by multiplying it by R and reducing modulo M. The back-transformation is done in an equally straight forward manner by dividing the residue by R modulo M. Thus here are the following equations as transformation rules between the integer and the RNS domain:

$$\bar{a} = a(mod\ M) \quad (1)$$
$$=> a = \bar{a}R^{-1} \quad (2)$$

Montgomery Multiplication can be defined simply as the product of two M residues divided by the radix modulo M ie; $\bar{c} = \bar{a}\bar{b}R^{-1}(mod\ M)$. Division by the Radix is required to make the result again an M-residue. The key concepts of the Montgomery algorithm are the following :
i) Adding a multiple of M to the intermediate result does not change the value of the final result; because the result is computed modulo M. M is an odd number.
ii) After each addition in the inner loop the least significant bit of the intermediate result is inspected. If it is 1, i.e., the intermediate result is odd, we add M to make it even. This even number can be divided by 2 without remainder. This division by 2 reduces the intermediate result to n+1 bits
again.
iii) After n steps these divisions add up to one division by 2n.
Radix-2 Montgomery Multiplication Algorithm
Let X and Y be two n-bit operands and M be any odd integer which is greater than zero for satisfying radix-2 operation. Montgomery multiplication involves first transformation of operands into Montgomery domain and then after result is re-transformed into Montgomery domain. This conversion process replaces division by several shift operations then Montgomery multiplication process for inputs X, Y, M and output Z is described as follows:
Output to be obtained : Z= (X,Y) mod M
Where $X' = X.2^n \bmod M$
$Y' = Y.2^n \bmod M$
$Z' = MP(X', Y', M) = X \cdot Y \cdot 2^n \bmod M$
Hence $Z' = Z \cdot 2^n \bmod M$

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

An ISO 3297: 2007 Certified Organization *Vol. 5, Special Issue 4, March 2016*

National Conference on Signal Processing, Instrumentation and Communication Engineering (SPICE' 16)

Organized by

Dept. of ECE, Mar Baselios Institute of Technology & Science (MBITS), Kothamangalam, Kerala-686693, India

Algorithm 1
Algorithm for Radix 2 Montgomery Multiplier

```
P=0;
For(i=;i<n;i++)
{
P=P+Xi)*Y;
P=P+P[0]*M
P=P div 2
}
If(P>M)
P=P-M;
```

Hardware reduction of this algorithm is possible by precomputation. The values to be added to the intermediate result within the loop can be precomputed. Delay due to carry propagation must be avoided

V. FASTER MONTGOMERY MULTIPLIER

The motivation behind this optimized algorithm is that of reducing the chip area for practical hardware implementation. This is possible if we can precompute four possible values to be added to the intermediate result . These are the four possible scenarios:

i) if the sum of the old values of *S* and *C* is an even number, and if the actual bit x_i of X is 0, then we add 0 before we perform the reduction of *S* and *C* by division by 2.

ii) if the sum of the old values of *S* and *C* is an odd number, and if the actual bit x_i of X is 0, then we must add *M* to make the intermediate result even.

Afterwards, we divide *S* and *C* by 2.

iii) if the sum of the old values of *S* and *C* is an even number, and if the actual bit x_i of X is 1, but the increment x_i **Y* is even, too, then we do not need to add *M* to make the intermediate result even. Thus, in the loop we add *Y* before we perform the reduction of *S* and *C* by division by 2. The same action is necessary if the sum of *S* and *C* is odd, and if the actual bit x_i of X is 1 and *Y* is odd as well. In this case, *S*+*C*+*Y* is an even number, too.

iv) if the sum of the old values of *S* and *C* is odd, the actual bit x_i of X is 1, but the increment x_i **Y* is even, then we must add *Y* and *M* to make the intermediate result even. Thus, in the loop we add *Y*+*M* before we perform the reduction of *S* and *C* by division by 2. The same action is necessary if the sum of *S* and *C* is even, and the actual bit x_i of X is 1, and *Y* is odd. In this case, *S*+*C*+*Y*+*M* is an even number, too.

The computation of *Y*+*M* can be done prior to the loop. This saves one of the two additions which are replaced by the choice of the right operand to be added to the old values of *S* and *C*. Algorithm 2 is a modification of Montgomery's method which takes advantage of this idea.

Algorithm 2
Algorithm for Faster Montgomery Multiplier
Inputs: X, Y, M with $0 \leq X, Y < M$
Output: $P = (X*Y\ (2^n)^{-1}) \bmod M$
n: number of bits in X;
x_i : i^{th} bit of X;
s_0 : LSB of S, c_0: LSB of C, y_0 : LSB of Y;
R: precomputed value of Y+ M;
S = 0 ; C = 0 ;
for (i=0 ; i <n; i ++)
{
if (($s_0 = c_0$) and not x_i) then I =0;
if (($s_0 \neq c_0$) and not x_i) then I= M
if (not(s_0^ c_0^ y_0) and x_i) then I = Y;
if ((s_0^c_0^ y_0) and x_i) then I=R;
S,C= S+ C+ I;
S : S div ; C= C div 2 ;}
P= S+ C;

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

An ISO 3297: 2007 Certified Organization *Vol. 5, Special Issue 4, March 2016*

National Conference on Signal Processing, Instrumentation and Communication Engineering (SPICE' 16)

Organized by

Dept. of ECE, Mar Baselios Institute of Technology & Science (MBITS), Kothamangalam, Kerala-686693, India

if (P≥ M) then P =P-M;

The advantage of Algorithm 2 in comparison to Algorithm 1 can be seen in the implementation of the loop of Algorithm 2 . The possible values of I are stored in a lookup-table, which is addressed by the actual values of x_i, y_0, s_0 and c_0. The operations in the loop are now reduced to one table lookup and one carry save addition. Both these activities can be performed concurrently. Note that the shift right operations that implement the division by 2 can

Faster Montgomery consumes less area and also less power.

VI. DESIGN EVALUATION

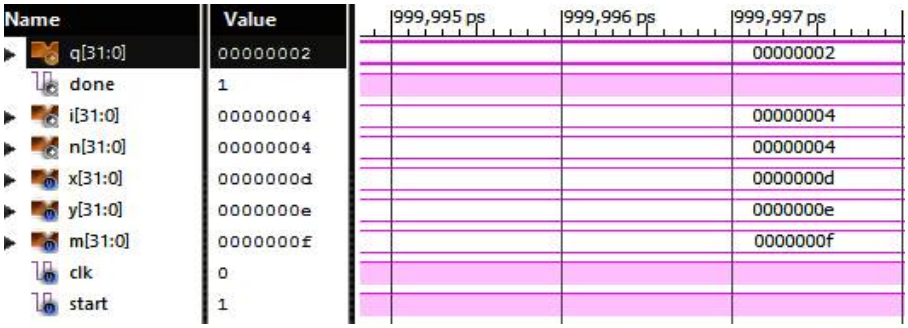

Fig. 2: Simulation Result

Delay analysis of Montgomery multiplier and faster Montgomery multiplier gives the following result :

Table. 1: Delay Analysis

Technique	Delay
Montgomery multiplier	41.702ns
Faster Montgomery multiplier	32.55ns

VII. CONCLUSION

The estimated total circuit area and critical path delay of the modular multiplier based algorithm show that it can be implemented in much smaller hardware than that necessary to implement multiplier and divider separately. We conclude that, among the various algorithms proposed in literature for calculating modular multiplication, the Montgomery modular multiplication algorithm seem to be the suitable ones to be combined. This paper presented an efficient algorithm to reduce the energy consumption and enhance the throughput of Montgomery modular multipliers simultaneously. The work can be further extended to modular exponentiation and squaring. A reversible architecture can be implemented for lower power consumption.

REFERENCES

[1] Tenca, Alexandre F., and Cetin K. Koc. "A scalable architecture for modular multiplication based on Montgomery's algorithm." *Computers, IEEE Transactions on* 52.9 (2003): 1215-1221.

[2] Kaihara, Marcelo E, and Naofumi Takagi. "A hardware algorithm for modular multiplication/division." *Computers, IEEE Transactions on* 54.1 (2005): 12-21.

[3] Kuang, Shiann-Rong, et al. "Energy-efficient high-throughput Montgomery modular multipliers for RSA cryptosystems." *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on* 21.11 (2013): 1999-2009.

[4] Knezevic, Miroslav, Frederik Vercauteren, and Ingrid Verbauwhede. "Faster interleaved modular multiplication based on Barrett and Montgomery reduction methods. "*Computers, IEEE Transactions on* 59.12 (2010): 1715-1721.

[5] Kong, Yinan, and Braden Phillips. "Comparison of Montgomery and Barrett modular multipliers on FPGAs." (2006).

ISSN (Print) : 2320 – 3765
ISSN (Online): 2278 – 8875

International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering

An ISO 3297: 2007 Certified Organization *Vol. 5, Special Issue 4, March 2016*

National Conference on Signal Processing, Instrumentation and Communication Engineering (SPICE' 16)

Organized by

Dept. of ECE, Mar Baselios Institute of Technology & Science (MBITS), Kothamangalam, Kerala-686693, India

[6] Chen, Jun Hong, et al. "High-speed modular multiplication design for public-key cryptosystems." *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*. IEEE, 2008.
[7] Cho, Koon-Shik, Je-Hyuk Ryu, and Jun-Dong Cho. "High-speed modular multiplication algorithm for RSA cryptosystem." *Industrial Electronics Society, 2001. IECON'01. The 27th Annual Conference of the IEEE*. Vol. 1. IEEE, 2001.
[8] Shieh, Ming-Der, et al. "A new algorithm for high-speed modular multiplication design." *Circuits and Systems I: Regular Papers, IEEE Transactions on* 56.9 (2009).