RSA Algorithm Modular Multiplication

O. Prasanthi[1] M. Subba Reddy[2]

[1]*Department of Electronics and Communication, Vaagdevi institute of Technology & Science, Proddutur, A.P*
[2]*Department of Computer Science EngineeringVaagdevi institute of Technology & Science, Proddutur, A.P.*

[1]friendlyprasanthi@gmail.com
[2]subbareddy463@gmail.com

***Abstract*— This paper presents the architecture and modeling of modular multiplication for RSA public key algorithm. It supports multiple lengths like 128 bits, 256 bits, 512 bits of data. In this paper simple shift and add algorithm is used to implement the modular multiplication. It makes the processing time faster and used comparatively smaller amount of space in the FPGA due to its reusability. Each block is coded with Very High Speed Integrated Circuit Hardware Description Language. The VHDL code is synthesized and simulated using Xilinx-ISE 10.1.**

Keywords- RSA, VHDL, FPGA, modular multiplication

I. INTRODUCTION

The art of keeping messages secure is cryptography. Cryptography plays an important role in the security of data. It enables us to store sensitive information or transmit it across insecure networks so that unauthorized persons cannot read it. The urgency for secure exchange of digital data resulted in large quantities of different encryption algorithms which can be classified into two groups: symmetric key algorithms (with private key algorithms) and asymmetric key algorithms (with public key algorithms) [1]. The asymmetric key algorithm requires two different keys, one for encryption and other for decryption as shown in figure 1.

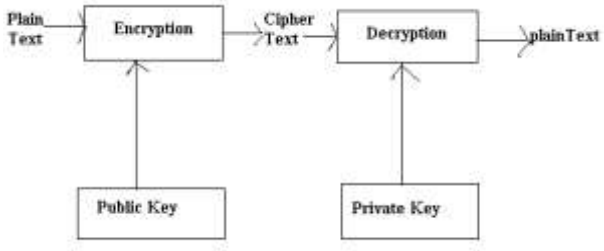

Figure 1. Public key cryptography

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 .However, the RSA algorithm is very computationally intensive, operating on very large (typically thousands of bits long) integers.A vast numbers and wide varieties of works have been done on this particular field of hardware implementation of RSA encryption algorithm.

A hardware implementation of RSA encryption scheme has been proposed by Deng Yuliang & Mao Zhigang in [2], where they use Montgomery algorithm for modular multiplication. A similar approach has been taken by C. N. Zhang & Y. Xu. in [3]. This design scheme focuses on the implementation of a RSA cryptographic processor using Bit-Serial Systolic Algorithm.

II. OVERVIEW OF RSA ALGORITHM

Figure 2 summarizes the different steps involved in RSA algorithm. An interesting feature of RSA algorithm is that, it allows most of the components used in encryption process are re-used in the decryption process [5]. So this can minimize the resulting hardware area.

RSA Key Generation

1. Choose two large primes p and q.
2. Compute n = p×q
3. Calculate φ(n) = (p-1)×(q-1)
4. Select the public exponent e ∈{ 1,2, . . . , φ(n)?1} Such that GCD (e, φ (n)) = 1.
5. Compute the private key d such that d×e ? modφ(n)

Output: public key: kpub = (n,e) and private key: kpr = (d)

Encryption/ Decryption

Plaintext block M is encrypted to a cipher text block C by:

$$C = M^e \bmod n \quad (1)$$

The plaintext block is recovered by:

$$M = C^d \bmod n \quad (2)$$

Figure 2. RSA algorithm.

RSA encryption and decryption are mutual inverses and commutative as shown in equation (1) and (2), due to symmetry in modular arithmetic. Hence the encryption engine covers both the operation of

Encryption and Decryption.The mathematics involved in modular arithmetic is as follows:

The integers *A* and *B* are congruent modulo *m* if and only if *A* – *B* is divisible by m. This congruence is written as:$A \equiv B \bmod m$
where *m* is a positive integer is called modulus.
When *A* and *B* are divided by m, the same remainder is obtained.The sign "≡" indicates congruence.
For Examples: 14 ≡ 2 mod 12.

III. MODULAR MULTIPLICATION

Encryption is the process of converting the 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 $C = Me \bmod n$ [5], where *C* is cipher text, *M* is plain text, *e* is the public key exponent, and *n* is the modulus. This operation has involved a few modular operations: modular multiplication, modular addition, and subtraction.

The modular multiplication problem is defined as the computation of $P = A \times B \pmod{n}$, given the integers *A*, *B*, and *n*. It is usually assumed that A and B are positive integers with $0 \leq A, B < n$.

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.

But in this paper Montgomery's algorithm is used. It avoids the traditional "division" operation and uses "shift and addition" operations to perform modular multiplication.

Let *A* and *B* are two *k*-bit positive integers, respectively. Let *Ai* and *Bi* are the *ith* bit of *A* and *B*, respectively. The algorithm is stated as follows:

```
Input: A, B, n
Output: M = A×B mod n
M = 0
FOR I = 0 TO K-1
M=M+(A×Bi)
if M0 = 1
M = M/2;
else
M=(M+n)/2;
return M;
```

Figure 3. Algorithm for Modular Multiplication

In this paper carry save adder is used to perform both addition operations. The shift register block performs the division by two operations. A 2×1 multiplexer is used to perform the selection operations between two numbers.

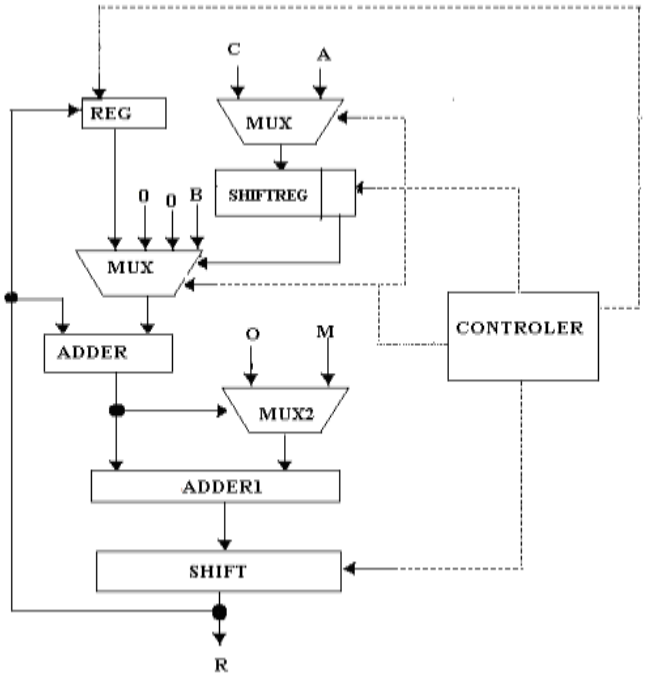

Figure 4. Montgomery's modular multiplier architecture.

IV. RESULT & DISCUSSIONS

The RTL schematic diagram for modular multiplication block is shown in figure 5. It takes three inputs of 128 bit each. The synthesis report for 128 bit modular multiplication is given below. By changing the generic parameter, a block of different size for modular multiplication for RSA encryption module may be obtained.

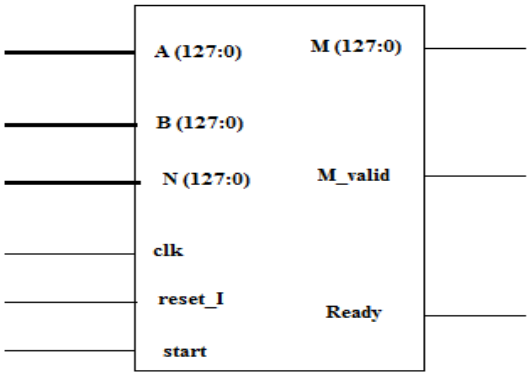

Figure 5. RTL Schematic Diagram of modular multiplication.

The simulation wave form of modular multiplication is shown in figure 6. In this figure three numbers (*A*, *B* & *N*) are 7, 4 and 15 respectively. Hence the result (*M*) should be 13, which is shown in the simulation.

TABLE I: HDL SYNTHESIS REPORT

Macro Statistics	
Component	**Nos.**
# Adders/Subtractors	2
32-bit adder	1
32-bit subtractor	1
# Registers	292
1-bit register	272
128-bit register	10
32-bit register	6
96-bit register	4
# Comparators	1
32-bit comparator equal	1
# Xors	192
1-bit xor2	192

TABLE II. TIMING SUMMARY

Speed Grade	-5
Minimum period	8.735 ns
Maximum Frequency	114.475MHz
Minimum input arrival time before clock	5.854ns
Maximum output required time after clock	4.063ns

TABLE III. DEVICE UTILIZATION SUMMARY

Selected Device	3s250etq144-5
Number of Slices	1719 out of 2448 70%
Number of Slice Flip Flops	2009 out of 4896 41%
Number of 4 input LUTs	3190 out of 4896 65%
Number of IOs	517
Number of bonded IOBs	517 out of 108 478% (*)

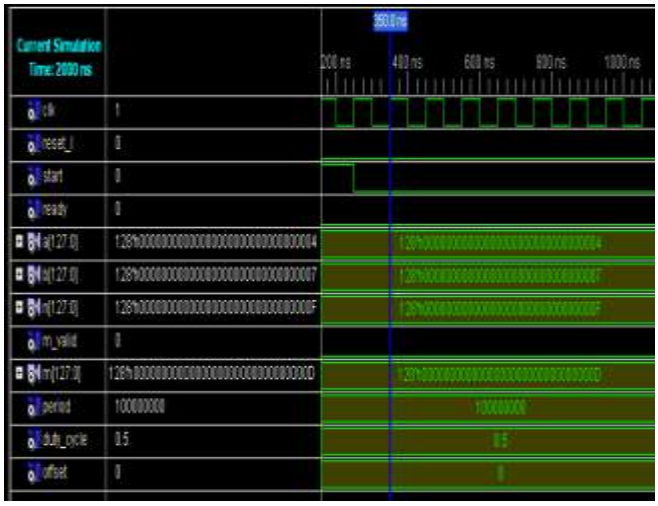

Figure 6. Simulation waveform of modular multiplier

V. CONCLUSION

As the modular multiplication is one of the most important operations in public-key cryptography, therefore, the efficient implementation of modular multiplication has become the key factors affecting the performance of public key cryptosystems. The VHDL code of modular multiplication is developed for RSA algorithm. Optimized and Synthesizable VHDL code for each block synthesized using Xilinx ISE 10.1 and verified that functionally correct. From the synthesis estimate, the minimum clock period was found to be 8.735 ns, from which the maximum clock frequency is 114.475MHz.

REFERENCES

[1] SCHNEIER, B. : Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley & Sons, 1996.

[2] Deng Yuliang, Mao Zhigang, Ye Yizheng, Wang Tao "Implementation of RSA Crypto processor Based on Montgomery Algorithm, 1998 IEEE

[3] C. N. Zhang, Y. Xu and C. C. Wu 'A Bit-Serial Systolic Algorithm and VLSI Implementation for RSA, 1997 IEEE

[4] M. Jason Hinek, "Cryptanalysis of RSA and Its Variants" CRC press, 2010.

[5] Rivest, R., Shamir, A., and Adleman, L, "A Method for Obtaining Digital Signatures and Public Key Cryptosystems", Communications of the ACM, 1978, vol. 21, no. 2, pp. 120-126.

[6] William Stallings, "Cryptography and Network Security: Principles and Practices." 3rd edition, 2003.

[7] Steve Burnett and Stephen Paine, "RSA Security's Official Guide to Cryptography" , McGraw-Hill, 2001

[8] Peter J Ashenden & Jim Lewis, "The Designer's Guide to VHDL",Morgan Kaufmann Publishers.

[9] Enoch O. Hwang,"Digital Logic and Microprocessor Design With VHDL"

[10] Parth Mehta, Dhanashri Gawali, "Conventional versus Vedic Mathematical Method for Hardware Implementation of a Multiplier",2009 International Conference on Advances in Computing, Control, and Telecommunication Technologies, Trivandrum, Kerala, India, December 2009, pp. 640-642, 2009.

[11] Christof Paar • Jan Pelzl, "Understanding Cryptography" Springer,2010