International Journal of Engineering Research in Electronic and Communication Engineering (IJERECE) Vol 3, Issue 5, May 2016

Efficient Modular Exponentiation Architectures for RSA Algorithm

[1]S. Prasanth kumar, [2] K.J.Jegadish kumar, [3] B.Partibane
[1]Student M.E VLSI Design, Department of ECE, SSN College of Engineering
[2] Associate Professor, Department of ECE, SSN College of Engineering
[3]Assistant Professor, Department of ECE, SSN College of Engineering
[1]prasanthkumars1993@gmail.com, [2]jegadishkj@ssn.edu.in, [3]partibaneb@ssn.edu.in

***Abstract*--- Cryptosystems are used to send confidential messages in secure manner. The most important and efficient type of cryptosystem is RSA. RSA is used in various sectors such as Bank security and Internet protection. One of the various steps involved in RSA is Modular exponentiation, which is used in both Encryption and Decryption. It takes high memory and computation time of algorithm. In order to reduce the computation time of algorithm, the Modular exponentiation complexity is reduced. In this paper, we explain efficient novel Modular exponentiation. This is done by FPGA implementation using ZedBoard and comparing them with the previous methods.**

Keywords*--- *RSA cryptosystems; Modular Exponentiation; ASIC and FPGA implementation.

I. INTRODUCTION

RSA Algorithm is one of the most important public cryptosystem used for high secure transmission of data. This algorithm has several steps that are used to efficiently transmit message in secure manner. The main function of this algorithm is modular exponentiation. Several works [1], [2], [3], [5] are done to increase the bit value and there is a need for novel modular exponentiation methods which uses reduced computation time and memory. This paper focuses on hardware implantation of clock cycle method and comparing it with its conventional counterparts.

II. RSA ALGORITHM AND METHODS

A. RSA Algorithm

RSA algorithm is a widely used practical public key cryptosystem [1], [2], [3], [4], [5], [6]. In this method, the first two prime numbers are chosen as ***P*** and ***Q***. They are multiplied and the answer arrived is ***N***. There are three steps in key generation. The initial stage multiplying ***(P-1)*** and ***(Q-1)*** and the values is ***ϕ(N)***.

The secondary stage is key generation using Extended Euclidean Algorithm. Which is done by ***D×E mod ϕ(N)≡1***. Here ***E*** and ***N*** are public key ***D*** and ***N*** are private key. In third step message ***M*** is generated. This message is encrypted using public key and transmitted. This message is retrieved by decrypting it using the private key. The above explained algorithm can be further detailed as bellow Algorithm I.

Algorithm I RSA Algorithm

Input: *P, Q, E, D, M*

Output: $N = P.Q$

$$\phi(N) = (P-1).(Q-1)$$

$$C \equiv M^E \ mod \ N$$

$$M \equiv \ C^D \ mod \ N$$

Key Generation: *1<e< ϕ (N).*

$$GCD\ (E, \phi(N)) = 1$$

$$D.E \ mod \ \phi(N) \equiv 1$$

Encryption: $C \equiv M^E \ mod \ N$

Decryption: $M \equiv \ C^D \ mod \ N$

B. Modular Multiplication

Modular exponentiation is a straight forward method. But, this consumes more memory for a low bit value so modular exponentiation is done with modular multiplication [1], [3], [4]. The process can be done by equation

$$M^E \ mod \ N \ \equiv C \qquad (1)$$

The above the equation (1) is used by repeated modular multiplication [5], [6]. The process can be done by equations

$$M.C \ mod \ N \equiv C \qquad (2)$$

The above equation is further analyses in Algorithm II.

Algorithm II Modular Multiplication Method

Input: M, E, N

Output: $C \equiv M^E mod\ N$

$\boldsymbol{for}\ K = 1\ \boldsymbol{to}\ E\ \boldsymbol{do}$

$K \equiv M.K\ mod\ N$

$\boldsymbol{end}$

$C = K$

C. Fast Square Exponentiation

Algorithm II is used for memory reduction but it has some restrictions when high bit value are used, the process time become more. In order to overcome this, Fast Square Exponentiation is implemented [1], [5], [6], [7]. In this method, exponentiation value is converted into binary value. The process is further preceded based on the values. We get either '**0**' or '**1**'. If '**0**' do $\boldsymbol{M^2\ mod\ N}$ for '**1**' do $\boldsymbol{M^2\ mod\ N}$ and $\boldsymbol{D.M\ mod\ N}$ [1], [4]. The detailed algorithm is given in Algorithm III.

Algorithm III Fast Square Exponentiation

Input: M, E, N

Output: $C \equiv M^E\ mod\ N$

E converts to Binary

$E_n = \sum_{i=0}^{n-1} e(0,i)\ 2^i, \ldots,$

K = 1, P = M

$\boldsymbol{for}\ i = 0\ to\ E\ \boldsymbol{do}$

$P \equiv P^2\ mod\ N$

$\boldsymbol{if}\ e_i = 1\ \boldsymbol{then}$

$P \equiv K.P\ mod\ N$

$P \equiv P^2\ mod\ N$

return P

end

end

D. Clock Cycle Method

In Algorithm III, through time is reduced, memory consumption is high. In order to overcome this, Clock Cycle Method is implemented. In this method, memory consumption is reduced at the expense of increase in processing time to small extent when compared to method proposed in Algorithm III. In Modular multiplication methods modular function repeats after a particular cycle of values. This method stops when a value gets repeated. The values and its corresponding count (The modular multiplication process of each step is called as one count) are stored. Modulo operation is done for exponential value and count. If count value is less than exponential value, $E\ mod\ COUNT$

is done. Where E is exponential value. The output count obtained as a result of modular operation is checked with the stored value. The values corresponding to the output count obtained from modular operation is noted. The above explained method is further detailed in Algorithm IV.

Algorithm IV Clock Cycle Method

Input: *M, E, N*

Output: $C \equiv M^E\ mod\ N$

D[1000], K=1, COUNT=0, I=1, Q=1.

$D[1] \equiv M.K\ mod\ N$

$D[2] \equiv M.M\ mod\ N$

$\boldsymbol{for}\ I = 3\ \boldsymbol{to}\ D[I]! = M\ \boldsymbol{do}$

$D[I] \equiv M.D[I-1]\ mod\ N$

$COUNT = COUNT + 1$

$\boldsymbol{end}$

$\boldsymbol{if}\ (E > I)\ \boldsymbol{do}$

$Q \equiv E\ mod\ COUNT$

$C = Q$

$\boldsymbol{else}$

$C = D[I-2]$

$\boldsymbol{end}$

III. HARDWARE DESIGN AND IMPLEMENTATION

A. ZedBoard

ZedBoard is a single board computer based on Xilinx zynq device family. It is the combination of both ASIC and FPGA. The memory is saved by using both ASIC and FPGA instead of using FPGA is alone. FPGA and ASIC does Arithmetic and processor operation and quickly, whereas key generation and values applied are on ASIC, modular operation is FPGA. The time and memory is reduced by combining both ASIC and FPGA and implementing in ZedBoard. We have done the proposed methods in ZedBoard.

B. RSA Algorithm of Modular Multiplication

The basic operation of exponentiation method has been used in RSA algorithm [2], [6] and it is implemented in ZedBoard. The key and initial data is applied to ASIC and Arithmetic operations are applied to FPGA.

As shown in figure 1, the inputs are given into key generation, key distribution and message blocks. While, the Encryption and Decryption keys are obtained from modular multiplication. Modular multiplication is done by dividing the modular operation and multiplication operation in two separate blocks and its result is obtained through FPGA implementation.

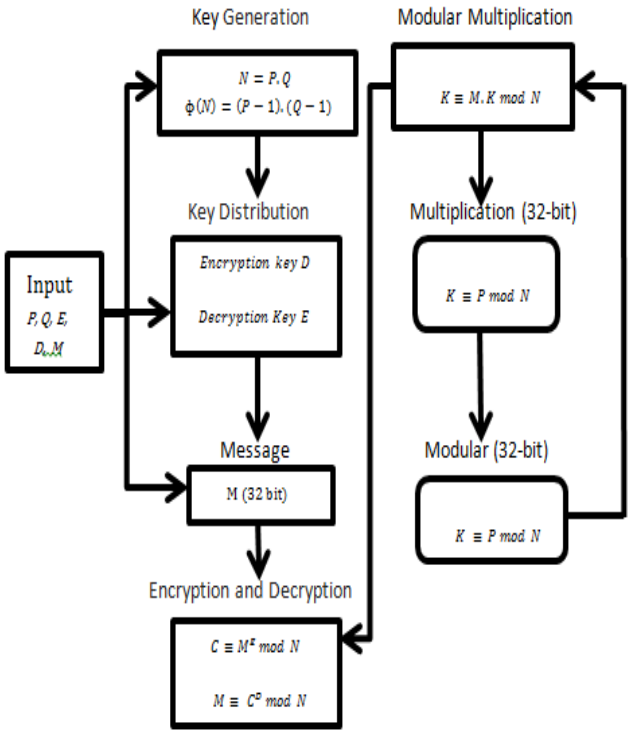

Figure.1. Block diagram of Modular Multiplication in RSA

As shown in the Output window (figure 2) 2347, 7663 are the given public key values, 67, 7663 are the private key values and sent message is 3452, encryption value is 2978, and decryption values 3452, decryption value and message value are equal. So, it ensures secure transmitted of message. Time cycle of the process is 2414.

Figure. 2. RSA using Modular multiplication

C. *Fast Square Exponentiation*

The proposed Fast Square Exponentiation method is implemented in ZedBoard ASIC and FPGA. In this method, the exponential value is converted into binary value and the operations are executed [7], [8], [10] based on Fast Square Exponentiation method

As shown in figure 3, the inputs are given to key generation, key distribution and message blocks. While, the Encryption and Decryption keys are obtained from modular multiplication. Exponentiation values are converted into binary value and based upon these binary values [9], operation is performed. Modular multiplication is acquired by dividing the modular operation and multiplication operation in two separate blocks and its result is obtained through FPGA implementation.

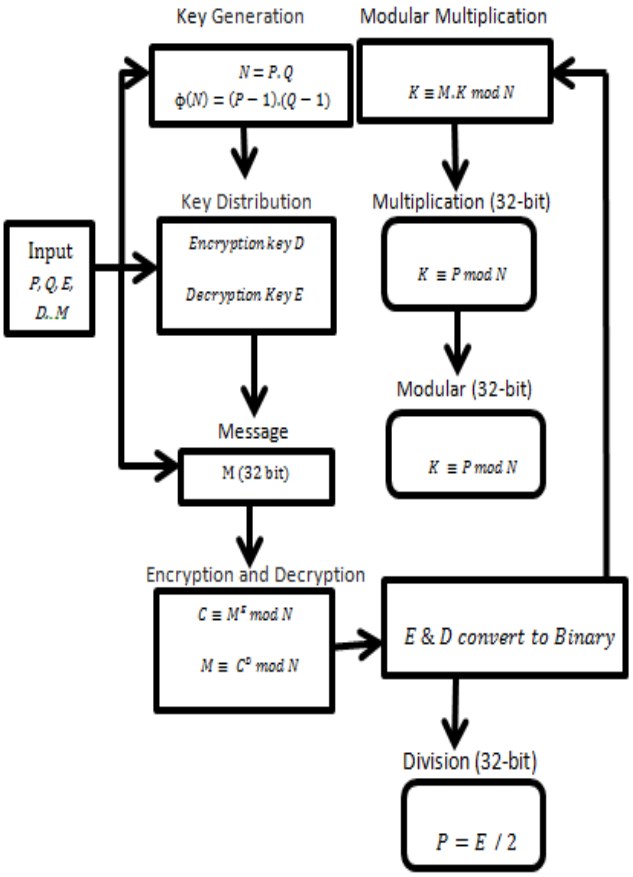

Figure.3. Block diagram of RSA using Fast Square Exponentiation

In Fast Square Method, same inputs are given as in Modular Multiplication method. Here encryption value is 2978, and decryption value is 3452, decryption value and message value are equal. The process of time cycle is 28. The execution time is very low. The obtained output in figure 4

```
enter two prime p and q
 97
 79
n=7663
pn=7488

ENCRYPTION
Enter the key ENCRYPTION Key DECRYPTION key 2347 67
 Enter message 3452
 1
 3452
 339
 7639
 576
 3635
 2213
 712
 1186
 2030
 5869
 7639
 1445
 3689
 6896
 3714
 396
 2978

ENCRYPTION is 2978
 1
 2978
 2393
 2188
 5632
 2267
 5079
 6163
 4741
 3452

DECRYPTION is 3452
```

Figure. 4. RSA using Fast Square Exponential

A. *Clock Cycle Method*

As shown in figure 5, the inputs are given into key generation, key distribution and message blocks. In Modular multiplication methods modular function repeats after a particular cycle of values.

This method stops when a value gets repeated. The values and its corresponding count are stored. Modulo operation is done for exponential value and count. If count value is less than exponential value, $E \ mod \ COUNT$ is done. Where E is exponential value. The output count obtained as a result of modular operation is checked with the stored value.

The values corresponding to the output count obtained from modular operation is noted. Finally, the Encryption and Decryption keys are obtained from modular multiplication. This result is obtained through FPGA implementation.

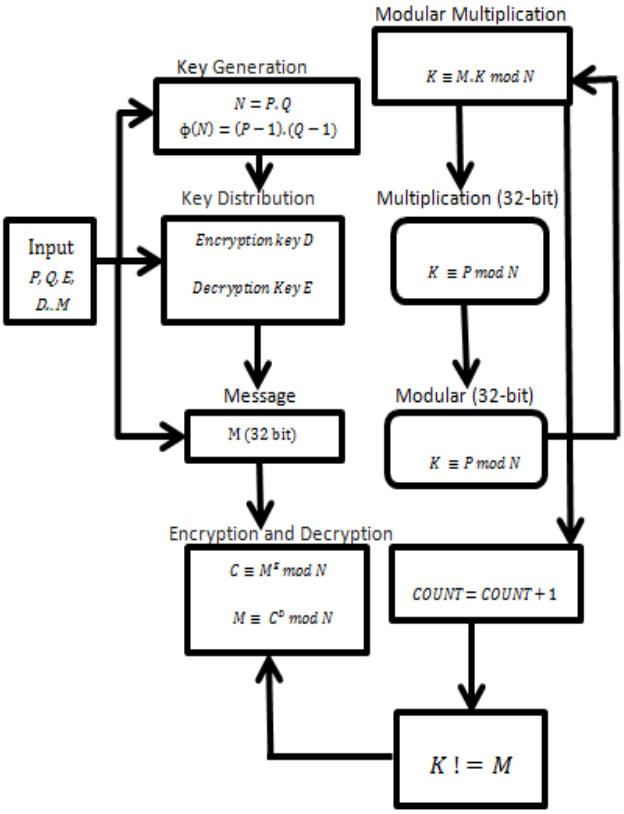

Figure.5. Block diagram of RSA using Clock Cycle Method

In this method, we give same input value in Modular multiplication method. Here encryption value is 2978, and decryption value is 3452, decryption value and message value are equal. Time cycle in this process is 148. The power and memory have been significantly reduced in clock cycle method than the fast square method. The time consumption has been reduced than the modular multiplication method but not as much as Algorithm II. The obtained output is given in figure 6

```
Enter two prime number p and q 97 79
n=7663
 pn=7488
Enter public and private keys 2347 67
Enter your message 3452
3452  339  5452  7639  1445  7190  7086  576
5079  7427  5269  4089  712  5664  3815  4346  5901  1998
4447  2055  5585  6975  554  4321  3894  1186  2030  3578
3635  3689  6185  1502  4716  3420  4820  2267  1761  2213
396  2978  3973  5689  5822  5158  4267  1398  5869  6479
6163  2188  4921  6084  5348  1129  4504  7244  1919  3556
6928  6896  3714  529  2314  3082  2820  2630  5768  2662
4874  4763  4741  5427  5632  633  1161  23  2766  134
6849  2393  7585  6612  4210  3872  1872  2235  6242  6691
2978
Encryption message 2978
2978  2393  7427  2188  2314  2055  4716  5632  5452  5822
4447  1502  5427  339  5689  6612  4089  6084  2820  6975
4210  712  5348  2630  554  2267  23  7190  1398  2235
4820  1161  1445  4267  1872  3815  4504  2662  3894  2213
1186  6928  2788  3635  4874  1050  396  6849  5079  6163
6691  1998  3556  7165  3578  3714  2583  6185  4741  3452
Decryption message 3452
```

Figure. 6. RSA using clock cycle method

Table 1: Comparison of Hardware Resource Utilization

Algorithm	Slice LUTs (53200)	LUTs Memory (17400)	Power consumption (w)	Cycles	Time (ns) Clock frequency 164.26 MHz
I. Modular multiplication	1803	245	1.723	2414	14696
II. Fast square method	5458	421	1.814	28	653
III. Clock cycle method	3282	276	1.760	148	824

IV. ZEDBOARD IMPLEMENTATION AND ANALYSIS

In this section, we present the results of the proposed methods using Xilinx SDSoC and VIVADO. The design is developed in C language and Verilog HDL. The function accuracy is verified by TeraTerm software.

The above three Algorithm II, III and IV are implemented in RSA. Here we use 32-bit key and message. Results are tabulate in table 1. The utilization reports and power consumptions are also tabulated.

The result in Table 1 prove that memory, LUT, time and power consumption are better than the Fast square method. The number of clock cycle is little more than fast square method.

V. CONCLUSION

The paper explains the detailed functioning process of the clock cycle method. This modular exponentiation method is hardware implemented and power consumption, utilization report, clock, time and memory are calculated. We prove that the clock cycle method is better when compared to the previous methods. This method is implemented in FPGA using Verilog HDL language.

REFERENCES

[1] Bagus Hanindhito, Nur Ahmadi, Hafez Hogantara, Annisa I. Arrahmah, Trio Adiono (2015), 'FPGA Implementation of Modified Serial Montgomery Modular Multiplication for 2048-bit RSA Cryptosystems' ,IEEE 978-1-4799-7711-6/15

[2] Ashbir A.Fadila, Naufal Shidqi, Trio Adiono (2015), 'Hardware Implementation of Montgomery Modular Multiplication Algorithm Using Iterative Architecture', IEEE Vol.35,No. 4.

[3] Miaoqing Huang, Kris Gaj, and Tarek El-Ghazawi (2011), 'New Hardware Architectures for Montgomery Modular Multiplication Algorithm' IEEE Transactions on computers, Vol.60,No.7.

[4] Dimitrios Schinianakis, Thanos Stouraitis (2014), 'Multifunction Residue Architectures for Cryptography' IEEE Transactions on circuit and system, Regular papers, Vol. 61, No.4.

[5] Nahed Mulla, Dr.Bncoet (2014), 'FPGA Implementation of An Efficient Montgomery Multiplier for Adaptive Filtering Application' IEEE Vol. 25, No.4.

[6] Donald Donglong Chan (2000), 'Parameter Space for the Architecture of FFT-based Montgomery Modular Multiplication', IEEE Transaction on VLSI, Vol 32, No. 11.

[7] A.Gonzale (2006), 'Double-Residue Modular Range Reduction for Floating-point Hardware Implementation', IEEE Transactions on computers, Vol.55,No. 3.

[8] Pingjian Wang, Zongbin Liu, Lei Wang, Neng Gao (2013), 'High Radix Montgomery Modular Multiplier on Modern FPGA', 12th IEEE International Conference on Trust, Security and Privacy in Computing and Communications.

[9] Ho Won Kim, and Sunggu Lee(2004), 'Design and Implementation of a Private and Public Key Crypto Processor and its Application to a Security System', IEEE Transactions on Consumer Electronics, Vol.35,No. 4.

[10] K.P.Sridhar, M.Raguram, Prakash, S.Koushighan, (2014), 'Secured Elliptic Curve Cryptosystem for Scan Based VLSI Architecture', IEEE ISBN 1-7758-774-91