International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:8, No:1, 2014

FPGA Implementation of RSA Encryption Algorithm for E-Passport Application

Khaled Shehata, Hanady Hussien, Sara Yehia

***Abstract*—**Securing the data stored on E-passport is a very important issue. RSA encryption algorithm is suitable for such application with low data size. In this paper the design and implementation of 1024 bit-key RSA encryption and decryption module on an FPGA is presented. The module is verified through comparing the result with that obtained from MATLAB tools. The design runs at a frequency of 36.3 MHz on Virtex-5 Xilinx FPGA. The key size is designed to be 1024-bit to achieve high security for the passport information. The whole design is achieved through VHDL design entry which makes it a portable design and can be directed to any hardware platform.

***Keywords*—**RSA, VHDL, FPGA, modular multiplication, modular exponential.

I. Introduction

E-PASSPORT is a passport that includes a smart card embedded in the back. This card contains the traveler's personal data. Many countries adopted e-passports to facilitate people traveling and Visa issuing. About 53 different countries including the United State (US) and Canada have used e-passport [1]. However, the security and integrity of the e-passport are very critical. The International Civil Aviation Organization (ICAO) created sets of e-passport standard [2], [3]. 1024-bit RSA is one of the recommended algorithms used for Active Authentication (AA) protocol. This protocol is used to prevent e-passport cloning [4].

The RSA is a public key encryption algorithm invented by Rivest, Shamir, and Adleman in 1977. RSA operation is based on modular exponentiation which requires repeated modular multiplications. Moreover for security reasons RSA operand sizes is recommended to be 1024 bits or more [5]. As a result the modular operations for 1024 bits or higher make RSA is difficult to achieve a high throughput. To address this problem many algorithms are invented such as add and shift, Montgomery multiplication and carry save adder (CSA) [6], [7].

This paper presents the implementation of RSA encryption/decryption algorithm with 1024-bit key length on FPGA. RSA algorithm adopts square and multiply algorithm for modular exponential. The modular multiplier is implemented using add and shift algorithm presented.

The paper is organized as follows: Section II explains RSA algorithm, Section III explains the mathematical algorithms used to execute RSA algorithm. Then in Section IV is discussed the RSA implementation and shows the simulation results. Finally, Section V draws the conclusion.

Khaled Shehata and Hanady Hussien are Lecturers in Arab Academy for Science and Technology, Cairo, Egypt (e-mail: khaledshehata58@gmail.com Hanady.issa@gmail.com).

Sara Yehia is Teaching Assistant, Lecturer in the Higher Institute of Engineering, New Cairo, Egypt (e-mail: Hawa3dy@yahoo.com).

II. RSA Algorithm

RSA is a public encryption algorithm which has a public key for encryption (e) and private key for decryption (d). RSA algorithm is summarized to three main steps [8].

A. Key Generation

In this step the private and public keys are generated as shown in Fig. 1 by:

1. Choose two large prime numbers p and q.
2. Compute modulus number n = p x q.
3. Calculate the Euler function φ(n) = (p-1) x (q-1).
4. Select an integer number e randomly as a public key. It should satisfy Greater Common Divisor GCD(e, φ(n)) = 1, 1< e < φ(n).
5. Compute the private key d such that d x e =1(mod φ(n)).

B. Encryption

In RSA both plain text (M) and cipher text (C) are blocks with length less than [$\log_2 n$]. In encryption, the cipher text is generated by

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

C. Decryption

The decipher text is recovered using the private key (d) by

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

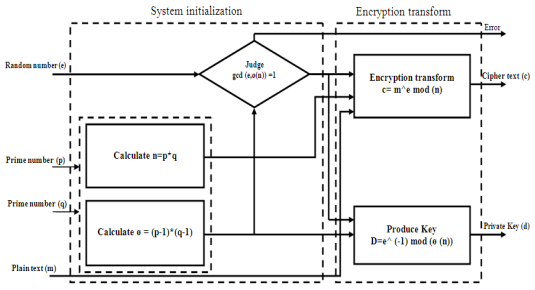

Fig. 1 Block diagram of RSA encryption and decryption algorithms

III. RSA Mathematical Operations

The RSA encryption/decryption algorithm is based on computation of modular exponentiation operation. The strength of RSA depends on the difficulty of factoring the modulus n to get the prime numbers p and q. Hence, the larger

prime numbers the harder the factorization of modulus n. Therefore the modular exponentiation operation becomes harder to accomplish on a hardware platform. This section details the main modular mathematical operations used for hardware implementation.

a) Modular Exponentiation Operation

Modular exponentiation for large numbers is considerably difficult to compute. Therefore, this operation can be simplified into series of modular multiplication and squaring operations [9], [10]. This algorithm is known as square and multiply algorithm. In this algorithm the exponent number e is scanned either from Left to Right (LR) or Right to Left (RL). In LR method, which is common used, if the scanned bit is logic zero a squared operation is performed. However if the scanned bit is logic one a multiplication operation is computed. This operation is performed k-time where k is the modulus length. The square and multiply algorithm is described by the following code [8]-[10].

```
Input:  m, e and  n.
Output: c = m^e mod n,  e > 1
        Initialization  c = m if e_{k-1} = 1 else c=1
        for j = k -1  downto 0 do
            c = c * c  mod n
            if ( e[j] == 1) then
            c = c * m mod e
        end for
        return c
```

b) Modular Multiplication Operation.

The modular multiplication operation is essential to compute the exponentiation modular as shown in previous algorithm. Shift and add algorithm is one of the algorithms used to perform modular multiplication. This algorithm computes y x z (mod n). The numbers y and z are k-bit integers and yi and zi are the ith bit of y and z respectively. The detailed algorithm is described as follows [10], [11].

```
Input: y, z, n
Output: Mul = y × z mod n
        Initialization Mul = 0;
        For i = 0 to k
            Mul=Mul +(y×zi)
            if Mul_0 = 1 then
                Mul = Mul / 2;
            else
                Mul=(Mul + n)  /2;
return M;
```

IV. Simulation Results of RSA Encryption/ Decryption Hardware Implementation

The RSA Encryption / Decryption modules with key length 1024 are designed and implemented based on VHDL code. The design adopts the square and multiply algorithm for modular expatiation. The modular multiplier is performed based on add and shift algorithm. The public and private keys are generated using C# program. The results are stored in ROM. There are two different ROMs, one is used to store (n, e) keys and the other is to store (n, d) keys. The design is simulated using Xilinx ISE 12.3 targeting Virtex-5 XC5VTX240T-2FF175 FPGA from Xilinx.

A. Add and Shift Algorithm Simulation Results

As discussed before add and shift algorithm is used to perform modular multiplier. It computes Mul = y x z mod n. As shown in Fig. 2, the algorithm inputs are 'mpand', 'mplier', and 'modulus' each with length 1024 bits. These inputs represent y, z and n respectively. The algorithm output is 'product' signal which represents 'Mul' output. As shown in Fig. 1 mpand = e_{hex}, mplier = 3_{hex}, modulus = 21_{hex} and product = 9_{hex} (9 = e x 3 mod 21). At clock: 209.048ns

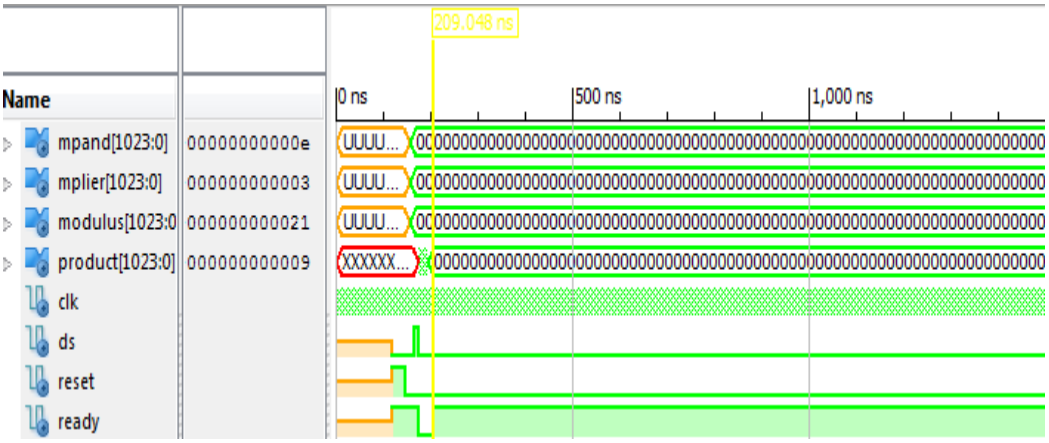

Fig. 2 Simulation results of add and shift algorithm

B. Square and Multiply Algorithm Simulation Results

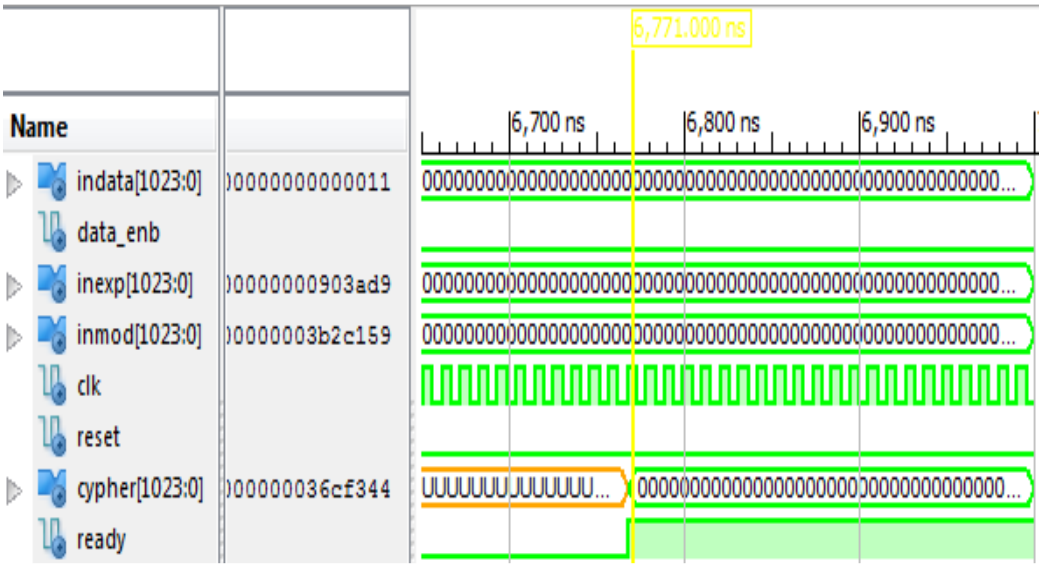

Fig. 3 Simulation results of square and multiply algorithm

Similarly the square and multiply algorithm is designed and tested. This algorithm computes $c = m^e \bmod n$. As shown in Fig. 3 the applied inputs are 'indata', 'inexp', 'inmod' and the output delivered is 'cipher'. These signals represent m, e, n and c respectively. From the simulation results shown in Fig. 2, indata = 11_{hex}, inexp=$903ad9_{hex}$, inmod = $3b2c159_{hex}$ the output cipher = $36cf344_{hex}$ ($36cf344 = 11^{903ad9} \bmod 3b2c159$). At [6,771.000ns] clock.

RSA Encryption/Decryption Algorithm

The whole system is tested by applying 1024-bit plain text. The used public keys are loaded form ROM module. The simulation result is shown in Fig. 4. By applying the generated cipher text on the RSA decryption algorithm the deciphering output is identical with the original plain text as shown in Fig. 5.

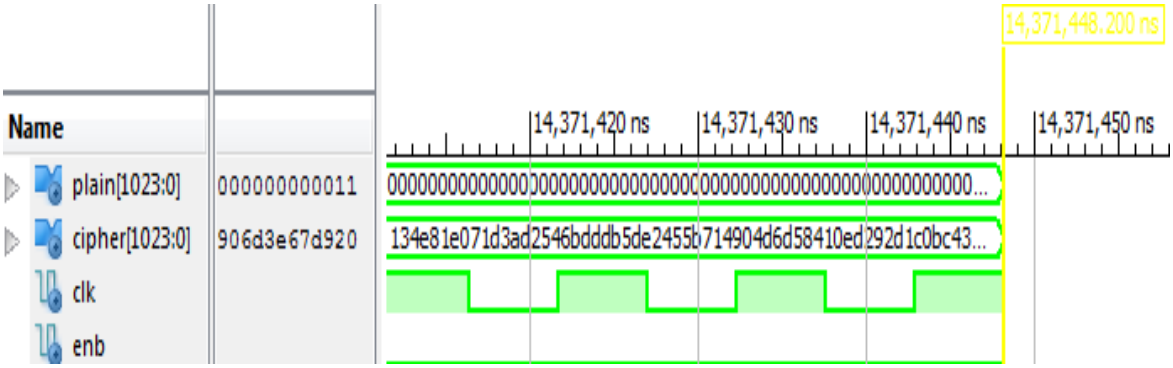

Fig. 4 Simulation Results of RSA encryption algorithm

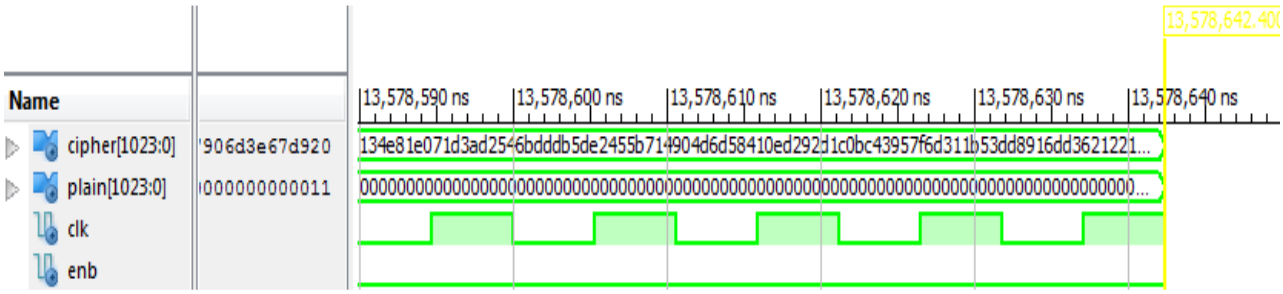

Fig. 5 Simulation Results of RSA decryption

TABLE I
THE RESULTS OF ENCRYPTION AND DISCUSSION

Reference	Clock Frequency (MHz)	Chip(plate form)	Key generation
R. Srinivasan [13]	54.7	Alter a Apex 0KE EP20k200EBC356	64 bits
Mostafizur Rahman [14]	100Mhz	Xilinx's vertex II pro FPGA	8bits
Sushanta Kumar Sahu [12]	101.06MHZ	XilinxISE10.1 3s100evq100-4	128 bits
Proposed	36.290MHz	Xilinx'svirtex5 xc5vtx240t-2ff175	1024 bits

The algorithms in Table I are used firstly R. Srinivasan [13]: Square and multiply algorithm is used for modular exponent and the Montgomery algorithm is used for modular multiply. R. Srinivasan uses 54.7 MHz clock frequency.

Secondly, Mostafizur Rahman [14]: Square and multiply algorithm is used for modular exponent and the add and shift algorithm is used for modular multiply. Mostafizur Rahman uses 100 MHz clock frequency.

Thirdly, Sushanta Kumar Sahu [12]: Square and multiply algorithm is used for modular exponent and the Montgomery algorithm is used for modular multiplier. Sushanta Kumar Sahu uses 101.06 MHz clock frequency.

It is clear that when the number of bits is increased; the frequency is decreased, compared with other works. The previous work is much more complicated than the present work; because of the length of RSA public key and private key under1024- bit are insecurity. There are certain procedures in the selection of the p,q and e in addition to the generation of public key apart from the need for a high speed computer.

In this proposed technique, the Xilinx'svirtex5 xc5vtx240t-2ff175is used, number of slice (LUTS) used is 28,350, while the available (LUTS) is 149,760 so the proposed technique utilizes 18%. The key generation is 1024 bit, it is the most security code compared to previous work in the latest papers.

V. CONCLUSION

In this paper, a detailed implementation technique for 1024-bit RSA encryption/decryption algorithm is presented. The modular exponential for encryption and decryption process is performed by using square and multiply algorithm. The add and shift algorithm is used to perform the modular multiplier. All these algorithms are implemented using VHDL code targeting Virtex-5 XC5VTX240T-2FF175 FPGA from Xilinx. The whole design is tested using Xilinx ISE 12.3 tool. The system speed achieved is 36.3 MHz which comply with the speed of smart card used in e-passport.

REFERENCES

[1] Sungbae Ji, Zeen Kim, Kwangjo Kim," Design of an RFID-embedded e-ID System for Privacy Protection", Symposium on Cryptography and Information Security, Miyazaki, Japan, Jan. pp. 22-25, 2008.

[2] Albert B. Jeng, Lo-Yi Chen, " How to enhance the Security of e-passport", Proceedings of the Eighth International Conference on Machine Learning and Cybernetics, Baoding, pp. 2922-2929, 12-15 July 2009.

[3] Saeed, M.Q., Masood, A.; Kausar, Firdous," Securing e-Passport System: A Proposed Anti-Cloning and Anti-Skimming Protocol", 17th International Conference on telecommunications & Computer Networks, 2009.

[4] Zdenek Riha, Vashek Matyas, "Privacy issues of electronic passports" Journal of medical informatics and technologies, Vol. 17, ISSN 1642-6037, 2011.

[5] Na Qi Jing Pan Qun Ding "The implementation of FPGA –based RSA public key algorithm and its application in mobile –phone SMS encryption system" International Conference on Instrumentation, Mesurment, Computer, Communication and Control volume 21-No.5, pp. 700-703, 2011.

[6] Ridha Ghayoula, El Amjed Hajlaoui, Talel Korkobi, Mbarek Traii, Hichem Trabelsi, "*FPGA Implementation of RSA Cryptosystem*", International Journal of Engineering and Applied Sciences pp 2-3, 2006.

[7] Chiranth E, Chakravarthy H.V.A, Nagamohanareddy P, Umesh T.H, Chethan Kumar M." Implementation of RSA Cryptosystem Using Verilog "in International Journal of scientific & Engineering Research Volume 2, Issue 5, May-2011, 1ISSN 2229-5518

[8] Vibhor Garg, V. Aruna chalams."Architectural analysis of RSA crypto system on FPGA "International *Journal of Computer Applications* ", Volume 26-No8, July 2011.

[9] Chiranth E, Chakravarthy H.V.A, Nagamohanareddy P, Umesh T.H, Chethan Kumar M., "Implementation of RSA Cryptosystem Using Verilog", International Journal of Scientific & Engineering Research Volume 2, Issue 5, pp.1-7, May-2011

[10] Muhammad I. Ibrahimy, Mamun B.I. Reaz, handaker Asaduzzaman and Sazzad Hussain, "FPGA Implementation of RSA Encryption Engine with Flexible Key Size", International journal of communication, Issue 3, Volume 1, pp. 107-113, 2007

[11] M., Rokon, I.R., Rahman, M., "Efficient hardware implementation of RSA cryptography", 3rd International Conference on Anti-counterfeiting, Security, and Identification in Communication, 20-22 Aug. , pp. 316-319, Hong Kong, 2009.

[12] Sushanta Kumar Sahu, Manoranjan Pradhan," FPGA Implementation of RSA Encryption System" International Journal of Computer Applications", Volume 19– No.9, pp. 10 –12, April 2011.

[13] R. Srinivasan, Dr. V. Vaidehi, J. Balaji, S. Heema "A single chip efficient FPGA implementation of RSA and DES for digital envelop heme" Madras Institute of Technology Campus, Anna University, Chennai – 600 044 .India.2011.

[14] Mostafizur rahman, Iqbalurrahmanrok on and Miftahurrahman "Efficient hard ware implementation of RSA cryptography"2009.