Design and Implementation of RSA Algorithm using FPGA

Article *in* INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY · September 2015

DOI: 10.24297/ijct.v14i12.1737

Ari Shawkat Tahir
University of Zakho
6 PUBLICATIONS **21** CITATIONS

ISSN 2277-3061

Design and Implementation of RSA Algorithm using FPGA

Ari Shawkat Tahir
University of Zakho, Department of Computer Science
ari.kochar@gmail.com

ABSTRACT

RSA cryptographic algorithm used to encrypt and decrypt the messages to send it over the secure transmission channel like internet. The RSA algorithm is a secure, high quality, public key algorithm. In this paper, a new architecture and modeling has been proposed for RSA public key algorithm, the suggested system uses 1024-bit RSA encryption/decryption for restricted system. The system uses the multiply and square algorithm to perform modular operation. The design has been described by VHDL and simulated by using Xilinx ISE 12.2 tool. The architectures have been implemented on reconfigurable platforms FPGAs. Accomplishment when implemented on Xilinx_Spartan3 (device XC3S50, package PG208, speed -4) which confirms that the proposed architectures have minimum hardware resource, where only 29% of the chip resources are used for RSA algorithm design with realizable operating clock frequency of 68.573 MHz.

Indexing terms/Keywords

Cryptography; RSA; VHDL; FPGA.

Academic Discipline And Sub-Disciplines

Computer Science

SUBJECT CLASSIFICATION

Data Security , Cryptography , VHDL

TYPE (METHOD/APPROACH)

Design an architecture for RSA algoroithms using FPGA

Council for Innovative Research
Peer Review Research Publishing System
Journal: INTERNATIONAL JOURNAL OF COMPUTERS & TECHNOLOGY
Vol. 14, No. 12
www.ijctonline.com, editorijctonline@gmail.com

Now these days electronic data communications and Computer networks have made, it very much important to develop new ways to guarantee their security. As the time passes demand of security in the communication channel increases, and the development of a new and efficient hardware security module has started to get the primary preference [1].

The rising growth of data communication technique and electronic transactions over the web has made system security to become the most important issue over the network. To provide modern security features, public-private key cryptosystems are used. One of such cryptosystem is RSA algorithm. Though computation in RSA takes more time by if the message to be encrypted is generated randomly then RSA will prove to be good cryptography algorithm for system security.

For the better working of RSA based cryptosystem the system has the public key for decryption and the user will have the device containing the private key assigned to the user. And instead of entering the password the user will just have to insert the device to the system and the system will do the cross checking of the password for that particular user and allow access accordingly. As the user will not know the password as well as the password length so he can't give the password to any other person and the person will be solely responsible for any wrong doing in the system.

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 [2]. 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) [3,4].

New architecture has been presented in this paper to design and implement RSA algorithm using multiplication and square algorithm to perform modular operation during encryption and decryption proses. The whole system has been implemented using Xilinx ISE Design Suite 12.2. The Simulation results show the validity of the RSA encryption design while the synthesis results show that 29% of the chip resource are used when implementing the design on Xilinx_Spartan3 (device xc3s500, package pq208). This means minimum resources are used for the design with clock frequency of 68.573 MHz.

II. The Related work

In 2013, Gurpreet Kaur, Vishal Arora,[5] uses RSA cryptographic algorithm to encrypt and decrypt the messages for the citation of message in user's Communication over the secure transmission channel like internet and throw some light on the flaws of some other public key cryptographic algorithms in comparison to RSA algorithm. The suggested algorithm used to generate key pairs (private key and public key), which are generated using Big integer 19 digit prime value, which will be used to encrypt and decrypt the messages. Algorithm is implemented using GMP library try to makes Algorithm work efficiently and time complexity is also reduced.

In 2011, Sushanta Kumar Sahu and Manoranjan Pradhan, [6] presents the architecture and modeling of RSA public key encryption/decryption systems supporting multiple keysizes like 128 bits, 256 bits, 512 bits. Using simple shift and add algorithm is used to implement the blocks, 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. It is verified that this architecture support multiple key of 128bits, 256bits, and 512 bits.

In 2008, Ridha and ets,[3] presented the hardware implementation of the RSA public-key cryptographic algorithm. The suggested RSA cryptographic algorithm is depends on the computation of repeated modular exponentials. The Montgomery algorithm is used and modified to reduce hardware resources and to achieve reasonable operating speed for FPGA. Proposing An efficient architecture for modular multiplications based on the array multiplier. We have implemented a RSA cryptosystem based on Montgomery algorithm. As a result, it is shown that proposed architecture contributes to small area and reasonable speed.

In 2014, M. A. Smadi and ets, ,[7] propose and discuss An efficient FPGA design and implementation of RSA Crypto processor using scalable modules.The design has been executed for 32-bit encryption and decryption keys using a new parallelized architecture involving several hardware modules and modular arithmetic units. The final coding of the coprocessor were synthesized using both ALTERA Cyclone IV EP4CE115F29C7 and VERTIX VII VC707 FPGA kits and resulted in a maximum frequencies of 15.725 MHz, 17.629 MHz respectively as well as analyzed and compared with several RSA designs.

III. RSA ALGORITHM

The RSA cryptosystem was invented by Rivest, Shamir, and Adleman in 1977. This is the most commonly used public-key cryptographic algorithm, and it is considered secure when sufficiently long keys are used. The security of RSA depends on the difficulty of factoring large integers. Difficulty of factoring *n to* find the original primes *p,q* defines the strength of RSA[1]. 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 both (private and public) keys are generated to be used in Encryption and decryption process. Generating both keys private and public as shown in below steps:

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

It is the transformation of data into a form that becomes as difficult as possible to read without the appropriate knowledge (a key). In the RSA algorithm the ciphertexte is generated by below equation. The fig. 1 shows the RSA encryption process.

$$C=M^{e} \bmod n \qquad (1)$$

C. **Decryption**

It is the reverse of encryption. It is the transformation of encrypted data back into an intelligible form. In RSA algorithm uses below equation to generate the plaintext. The fig. 1 shows the encryption process

$$M=C^{d} \bmod n \qquad (2)$$

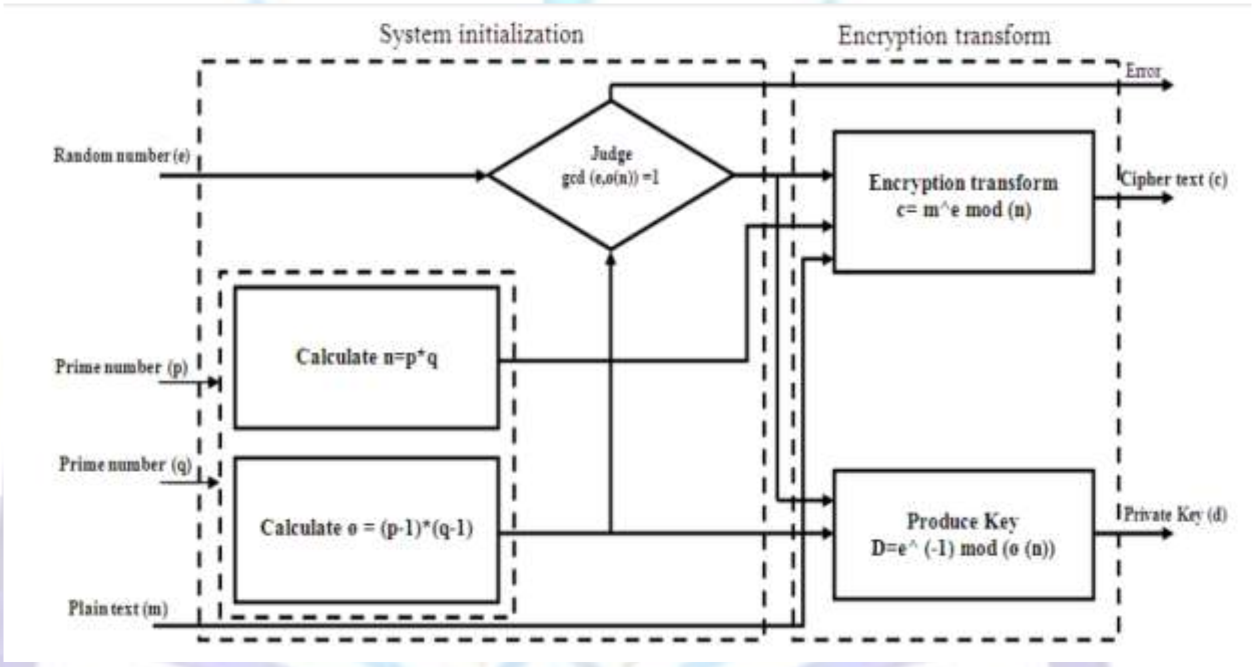
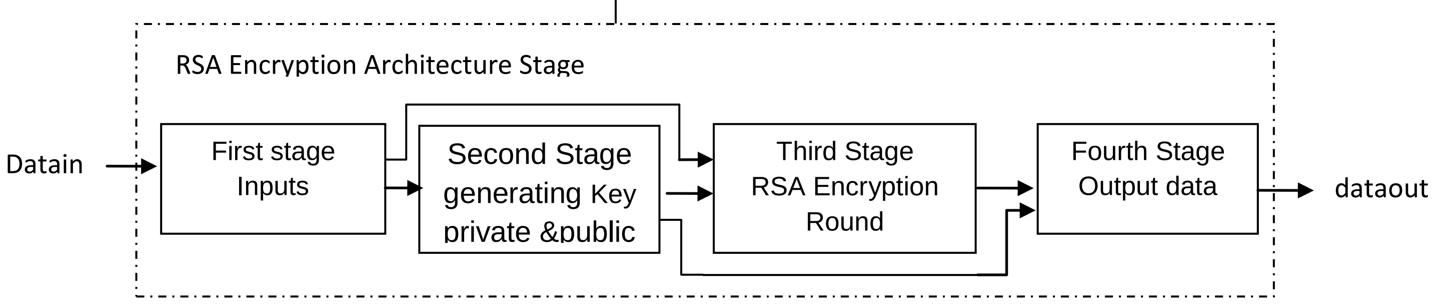


Fig 1: Block diagram of RSA encryption algorithms

IV. Design of the RSA Encryption Algorithm

The proposed design uses to translate the data from the original data (plaintext) to ciphertext. The proposed design of the RSA Encryption Algorithm consists of four stages. Each stage has its work. Fig 3 shows the block diagram of the whole RSA algorithm system. The Fig 2 shows the four stages that suggested for the propose architecture of the RSA algorithm.


RSA Encryption Architecture Stage
Datain
First stage Inputs
Second Stage generating Key private &public
Third Stage RSA Encryption Round
Fourth Stage Output data
dataout

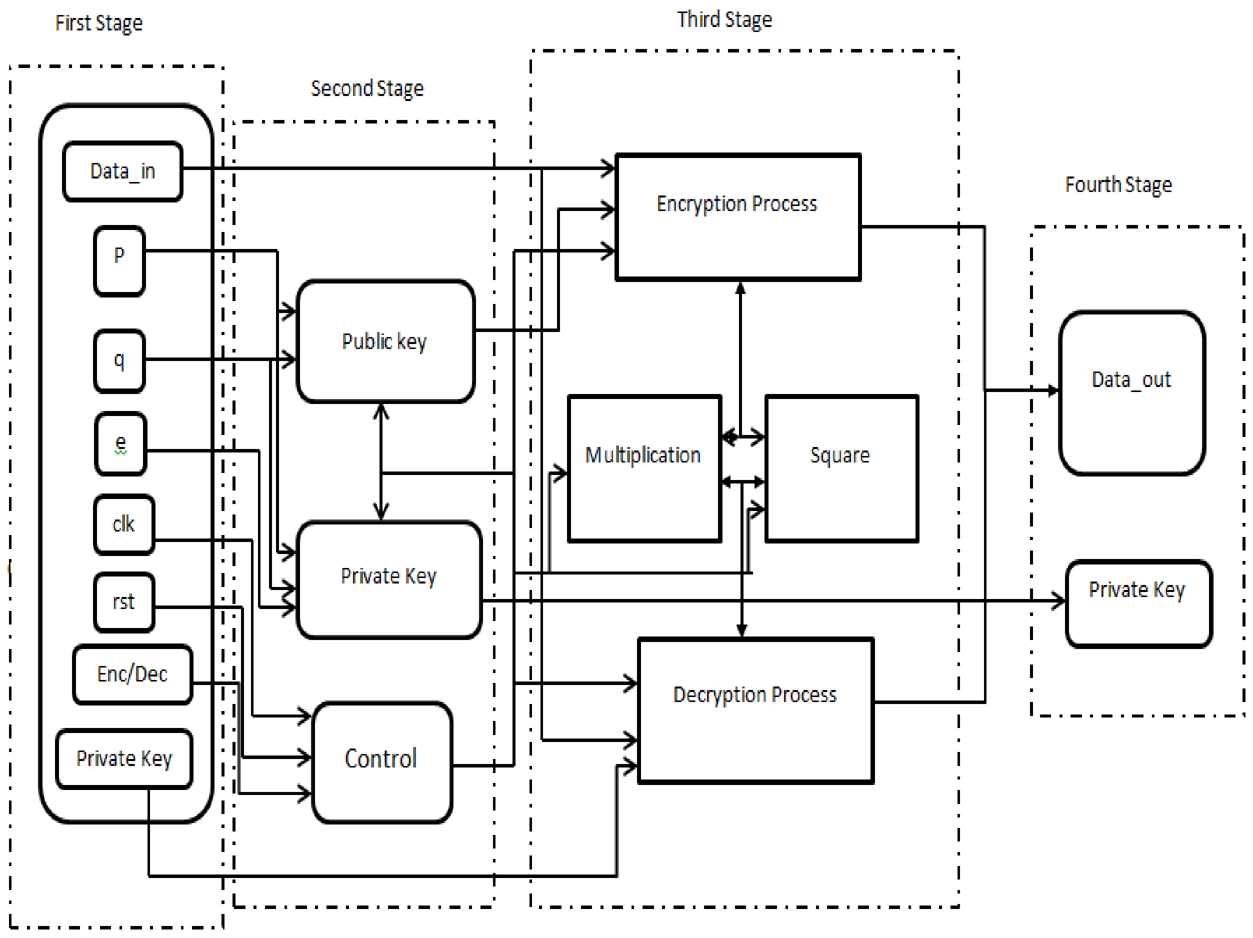


Fig 3: RSA Algorithm Process

- The first stage

  In this stage, reading 1024-bit Data and entered to a variable called input, reading two prime number p&q to generate public key , reading e which uses to generate private key and reading another signal which are using for controlling such as clk, rst and En/Dec. The input array contains 1024 rows, each row contains 1-bit.

- Second Stage

  In this stage, it receiving both prime numbers (p&q) from the previous stage then it is operated to produce both keys private and public. The public key used to encrypt data from original to unreadable data. And the private key uses to decrypt data to back to the original data from unreadable data.

- Third stage

  It contains many components such as Encryption, Decryption, Multiplication and Square componen. In this stage, it receives public key from previous stage and sending it to Encryption component during Encryption process. In encryption component, to complete the process it send message and key to both multiplication and Square component because modular exponentiation and multiplication the large number is difficult to compute. The output form this stage is sending to next stage which is the last one.

- Fourth stage

  The fourth stage receives the input array from the previous stage and entered in the new array named dataout. The output array consists of 1024 rows, each row consisting of 1-bit. The output array represents the Ciphertext in Encryption process and represents plaintext in Decryption process. At the end, it gets the 1024-bit ciphertext from 1024-bit plaintext and Vice versa.

V. RSA MATHEMATICAL OPERATIONS

The RSA encryption/decryption algorithm is based on computation of modular exponentiation operation. The difficulty of factoring the modulus n to get the prime numbers p and q made RSA stronger. 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. In order to compute large numbers use two modular mathematical operations for hardware implementation. The both modular are:

- Modular Exponentiation Operation
- Modular Multiplication Operation.

Modular exponentiation for large numbers is considerably difficult to compute. In order to simplify modular exponentiation for large number the process can be divided 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, 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 n-time where n is the modulus length. The square and multiply algorithm is described by the following pseudo code [8,10].

The pseudo code for modular exponentiation is as follows.

```
Input: M, e and n.
Output: $C = M^e$ mod n, e > 1
Initialization: **Set** C **to** M **if** ek-1 is equal 1 else **Set** C **to** 1
**for** j **equal** k -1 **downto** 0  **Loop**
        **Set** C **to** C * C **mod** n
        **if** e[j] is **equal to** 1 **then**
        **Set** C **to** (C **mul** M) **mod** e
**end Loop**
**return** C
```

In modular multiplication, this operation is a fundamental process to compute the exponentiation modular as shown in previous algorithm. The algorithm uses to permeate modular multiplication is Shift and add algorithm. This algorithm computes y * z (mod n). The numbers y and z are k-bit integers and y_i and z_i is the ith bit of y and z respectively. The detailed algorithm is described as follows [10,11].

The pseudo code for modular multiplication is as follows.

```
Input: y, z, n
Output: **Set** M **to** (y **mul** z) mod n
        Initialization :  **Set** M **to** 0;
        **For** i = 0 **to** k **Loop**
                Set M to (M plus (y mul zi)
                **if** $M_0$ is **equal**  1 **then**
                        **Set** M **to** (M **div** 2);
                **else**
                        **Set** M **to** ((M **plus** n)  **div** 2;
                **End if**
        **End Loop**
**Return** M;
```

VI. EXPERIMENTAL RESULTS:

The RSA Encryption / Decryption algorithm 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 proposed architecture has been synthesizes and implemented successfully on the device family Virtex4 (device XC4VLX80, package FF1148). The device utilization summary is shown in Table 1. The timing summary and memory summary are shown in Table 2.

The system firstly tested separately, first time the RSA Encryption tested on 1024-bit of plaintext and get 1024-bit ciphertext using public key 1024-bit. Second time tried to recover the original data using RSA Decryption to get back 1024-bit plaintext from 1024-ciphertext using 1024-bit private key. At the end, the whole system has been tested.

Table (5.3): Device Utilization Summery of RSA Encryption Design

Logic utilization	Used	Available	Utilization
Number of Slice	557	1,536	29%
Number Of Slice Flip Flops	459	1,536	29%
Number of 4 input LUTs	932	1,536	60%
Number of bounded IOBs	132	124	106%
Number of BUFGMUXs	1	8	12%

Table (5.4): Timing and Memory Summary of RSA Algorithm Design.

Speed Grade	-4
Minimum period	14.583 ns
Maximum Frequency	68.573 MHz
Minimum input arrival time before clock:	7.188 ns
Maximum output required time after clock:	9.008 ns
Total memory usage	609292 kilobytes

VII. CONCLUSION

In this paper, new an efficient architecture has been proposed to implement an optimized 1024-bit RSA encryption/decryption algorithm for restricted system using multiply and square algorithm to process the Modular exponential for encryption and decryption. The whole system is implemented using VHDL code targeting Spartan3 (device XC3S50, package PG208, speed -4) from Xilinx. The whole design is tested using Xilinx ISE Design Suite 12.2 tool. The system speed achieved is 68.573 MHz.

REFERENCES

[1] Ankit A. and Pushkar P., (2012)," Implementation of RSA Algorithm on FPGA", International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181, Vol. 1 Issue 5.

[2] Na Qi, Jing P. and Qun D., (2011), "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.

[3] Ridha G., El Amjed H., Talel K., Mbarek T. and Hichem T., (2006), "FPGA Implementation of RSA Cryptosystem", International Journal of Engineering and Applied Sciences, pp 2-3.

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

[5] **Gurpreet K. and Vishal A., (2013), "**An Efficient Implementation of RSA Algorithm using FPGA and Big Prime Digit", International Journal of Computer & Communication Engineering Research (IJCCER), Volume 1 - Issue 4.

[6] Sushanta K. S. and Manoranjan P., (2011)," FPGA Implementation of RSA Encryption System", International Journal of Computer Applications (0975 – 8887), Volume 19– No.9,

[7] M. A. Smadi,Qasem A. and Abdullah A.,(2014), "Efficient FPGA Implementation of RSA Coprocessor Using Scalable Modules" , International Symposium on Emerging Inter-networks, Communication and Mobility, Procedia Computer Science 34, pp 647 – 654.

[8] Vibhor G. , Aruna c. (2011)."Architectural analysis of RSA crypto system on FPGA ", International Journal of Computer Applications , Volume 26-No8.

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

[10] Muhammad I. I., Mamun B.I. R., handaker A. and Sazzad H., (2007), "FPGA Implementation of RSA Encryption Engine with Flexible Key Size" , International journal of communication, Issue 3, Volume 1, pp. 107-113.

[11] Rokon I.R., Rahman M.,(2009), "Efficient hardware implementation of RSA cryptography" , 3rd International Conference on Anticounterfeiting, Security, and Ident