International Journal of Electrical, Electronic and Communication Sciences
ISSN: 2517-9438
Vol:2, No:8, 2008

FPGA Implementation of RSA Cryptosystem

Ridha Ghayoula, ElAmjed Hajlaoui, Talel Korkobi, Mbarek Traii, Hichem Trabelsi

***Abstract*—**In this paper, the hardware implementation of the RSA public-key cryptographic algorithm is presented. The 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. An efficient architecture for modular multiplications based on the array multiplier is proposed. 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.

***Keywords*—**RSA, Cryptosystem, Montgomery, Implementation. FPGA.

I. Introduction

In a secure telecommunications network such as is increasingly required for electronic commerce and internet privacy; security requirements include confidentiality, authentication, data integrity and non-repudiation. These services are offered by public key cryptosystems, the most popular of which is the RSA encryption scheme [1]. The fundamental operation of the algorithm is modular exponentiation which is achieved by repeated modular multiplications. The Montgomery modular multiplication algorithm [2] is often used to perform these calculations. However, the high bit lengths required to provide adequate security (1024 bits is considered secure against attack in the near future), mean a high hardware throughput is difficult to achieve.

An efficient algorithm for the calculation of (Ax B) mod M was developed by P. L. Montgomery [4], and forms the basis of the designs presented here. It should be noted that Montgomery's algorithm only works if the modulus is relatively prime to the radix, although this is always the case in RSA.

The objective of this work is the modeling and implementation on FPGAs the RSA cryptosystem while basing on the Montgomery Multiplication.

This paper is organized as follows: Section 2 presents the RSA algorithm and Montgomery multiplication. Section 3 presents the implementation architecture of Montgomery Multiplication while using the double adder architecture. Sections 4, 5 present the modular exponentiation architecture. Then, Sections 6, 7 present results of simulation. , and finally some conclusions are drawn in Section 8.

Authors are with Laboratory of Electronics, Department of Physics, Faculty of Sciences of Tunis, 2092, El-Manar, Tunisia (e-mails: ridha_ghayoula_fst@yahoo.fr, hajlamjed@yahoo.fr, korkobi_talel@yahoo.fr, traii_moncef@yahoo.fr).

II. Montgomery Modular Multiplication

A. The RSA Algorithm

Suppose Alice wishes to send a secret message to Bob. In a secret key system, Alice and Bob need to know the same secret key and so it must be communicated via some secure channel. In a public key system, Bob broadcasts his public key and Alice can use it to encode a message. The design of the cryptosystem is such that only Bob can decode the encrypted message, but no exchange of secret keys is necessary.

In the RSA cryptosystem, [3] Alice must first find Bob's public key M, E which are the modulus and exponent respectively. She calculates the cipher-text (C) from the plaintext (P) by:

$$C = P^E \ (mod\ M) \tag{1}$$

To decode the message, Bob uses his private key (D) to recover the plain text by:

$$P = C^D \ (mod\ M) \tag{2}$$

To generate the key, two large prime numbers P and Q are first generated, and two equations are used to calculate D, E and M:

$$M = PQ$$
$$DE \equiv 1 \ \ (mod\ (P - 1)(Q - 1)) \tag{3}$$

where E is relatively prime to (P - 1) (Q - 1). In practice, E is often chosen to be a small number which reduces the amount of computation required to perform encryption. The strength of RSA depends on the key size k, the number of bits in M. Breaking RSA is believed to be as hard as factorizing M to P, Q which is intractable for k ≥ 1024 with current Technology [4].

B. Description of the Montgomery Algorithm

The encryption and decryption operations are, indeed, of modular exponentiation operations. The exponentiation operation is a succession of modular products, it is clear that the efficiency of this last is going to determine the performance of the system because the Montgomery algorithm achieves the modular multiplication of effective way, and it is easily adaptable to the material implementation.

The MonPro2 function calculates the Montgomery Multiplication:

$$MonPro2\ (A, B, M) = ABr^{-1}\ Mod\ M$$

Algorithm 2: MonPro2(A, B, M)
MonPro2 (A,B,M)
{
S-1:= 0;
For i = 0 to n - 1 do
qi := (Si-1+biA) Mod 2 ; (LSB of Sum)
Si := (Si-1 + qiM + biA)/2 ;
end For
Return Sn-1;
}

Fig. 1 Algorithm 2: MonPro2

Where:

$$A = \sum_{i=0}^{k-1} a_i 2^i , B = \sum_{i=0}^{k-1} b_i 2^i , M = \sum_{i=0}^{k-1} m_i 2^i$$

$$a_i , b_i , m_i \in \{0,1\}$$

With M is an integer of k-bit (ie: $0 \prec M \prec 2^k$), $A, B \prec M$ and n is equal to k +2 and S are remainders (ie: $0 \leq S \prec 2^k$).

III. Montgomery Architecture

A. Block Diagram

This block present, the Montgomery multiplication MonPro2 (A, B, M), therefore it contains three variable of entrances A, B and M and a signal clock (CLK) and two signals of RST commands and LOAD for the loading variables and R exit that represent the result of calculation. Finally this algorithm is going to calculate the operation following: $AB\ (2^n)^{-1} mod\ N$.

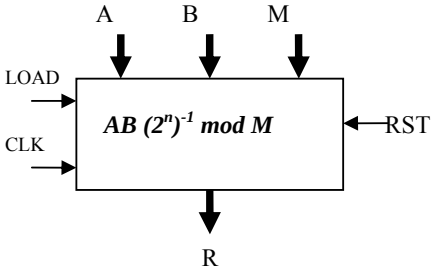

Fig. 2 Diagram block of the Montgomery multiplication

B. Double Adder Architectures

The architecture of double adder represents the Montgomery multiplication based on the algorithm (2), therefore she is constituted of two adders of 8 bits and two multipliers (A.bi and M.qi).

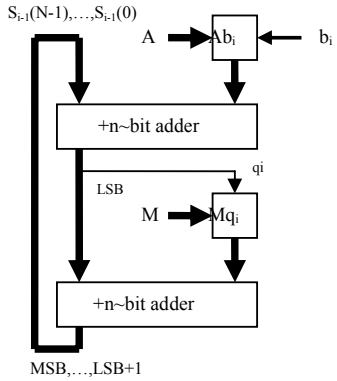

Fig. 3 MonArch1: Montgomery architecture based on the algorithm 2

To remove the dependency of qi on the addition of biA and Si-1, A can be shifted up 1 bit, thus forcing the LSB of Si-1 + biA to always be zero, as illustrated in figure 4. However, this extra factor of 2 must be removed by an extra iteration, meaning that the number of clock cycles is again equal to n + 1 [4].

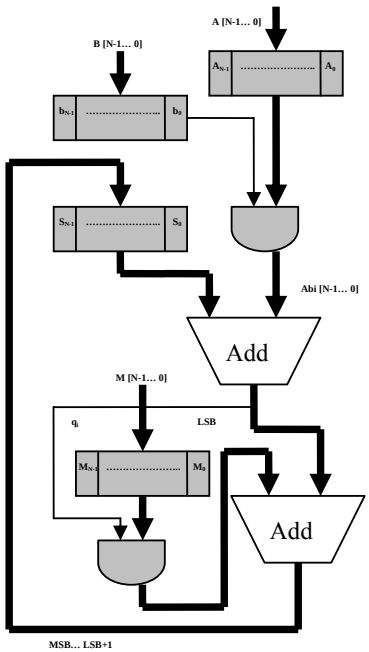

Fig. 4 Montgomery multiplier architecture based on algorithm (2)

IV. Modular Exponentiation

The modular exponentiation is executed by the repeated modular multiplications [4]. There are two common algorithms that can be to use: The binary method of L-R and the binary method of R-L. These are given in algorithms (4) & (5), where P is the E plaintext is the exhibitor, $\mathcal{M}$ is the module, C is constant and equal to $2^{2n} mod\ \mathcal{M}$, and $\mathcal{R}$ is the result.

Algorithme 4 :L-R Algorithme :MonExp1($\mathcal{P}$,ε,$\mathcal{M}$)

MonExp1 ($\mathcal{P}$, ε, $\mathcal{M}$)

{

$C := 2^{2n}$ Mod $\mathcal{M}$;

P:= Monpro (C ,$\mathcal{P}$,$\mathcal{M}$); (Mapping)

R := Monpro (C ,1 ,$\mathcal{M}$);

for i: =k-1 **downto** 0 **do**

R := Monpro (R,R,M); (Square)

if (ε_i=1) **then**

R := Monpro (R,P,M); (Multiply)

end if

end for

$\mathcal{R}$:= Monpro (1,R, $\mathcal{M}$); (Re-Mapping)

Return $\mathcal{R}$;

}

Fig. 5 Algorithm L-R

Algorithme 5 :R-L Algorithme :MonExp2 ($\mathcal{P}$,ε,$\mathcal{M}$)

MonExp2 ($\mathcal{P}$, ε, $\mathcal{M}$)

{

C: = 2^{2n} Mod $\mathcal{M}$;

P:= Monpro ($\mathcal{P}$, ε, $\mathcal{M}$); (Mapping)

R: = Monpro (c, 1, $\mathcal{M}$);

For i: =0 **to** k-1 **do**

if (ε_i = 1) **then**

R: = Monpro (R, P, M); (Multiply)

end if

P: = Monpro (P, P, M); (Square)

end For

$\mathcal{R}$: = Monpro (1, R, $\mathcal{M}$); (Re-Mapping)

Return $\mathcal{R}$;

}

Fig. 6 R-L algorithm

In the algorithm (4), the square and Multiply are operations that must be executed sequentially, and therefore multiplications must be executed in series, it means that all two are operations that can be executed in the same multiplier, simple to achieve point of view material, so one optimizes the surface of implantation.

In the algorithm (5), the square and multiplied it are some independent operations, and can be executed in parallel. Thus, less 50% cycles of clocks are required to accomplish the modular exponentiation. However, the two physical multipliers are required to achieve the acceleration of the algorithm. Therefore, products of surface x speed of the two algorithms are very similar. For our work, the algorithm of R-L (the algorithm (5)) will be used since the primary goal is to increase the binary debit of the exponentiation in real time for encrypt and to decrypt data.

V. Exponentiator Architecture

The fundamental operation some modular exponentiation is the repeated modular multiplication, in the RSA cryptosystem this exponentiation is based on the modular multiplication of Montgomery, two operations are necessary for the modular exponentiation, in the first, operation(Mapping) it is necessary to convert data of P entrance (message of origin) in $\mathcal{P}r$ mod M. After this result, the modular exponentiation becomes $\mathcal{P}^e rMod\ M$. In the second operation, the process re-mapping is executed. It calculates the function *Monpro (1, R, M)* finally to remove the supplementary factor r, one can reach the wanted result that is: $\mathcal{P}^e Mod\ M$. We apply the method binary R-L (right - left) that consists to optimize the speed while adopting the multiplication of Montgomery to the modular exponentiation.

A. Modular Exponentiation

Modular exponentiation block is considered as the operation of cryptography to all entrances that are the message of P origin, the key deprives (e, M) and R exit that represents encrypt message.

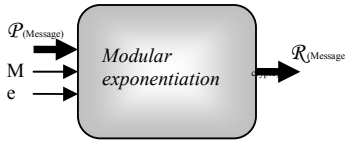

Fig. 7 Diagram block of the modular exponentiation

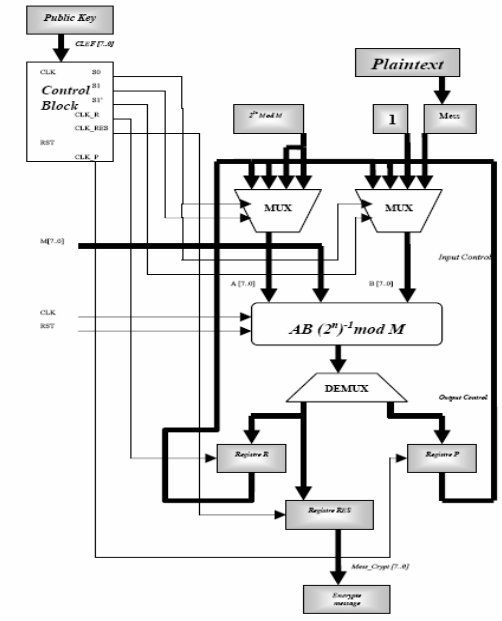

Fig. 8 Architecture - MonExp2

This figure present the architecture of implantation of the modular exponentiation Monexp2, indeed, it presents the different constituent blocks of our application. This figure represents all stages of calculation MonExp2 (P, ε, $\mathcal{M}$), the first stage dedicated to the initialization indeed, (Mapping):

$C: = 2^{2n}$ *Mod* $\mathcal{M}$;
$P: =$ *Monpro* $(C, \mathcal{P}, \mathcal{M})$;
$R: =$ *Monpro* $(C, 1, \mathcal{M})$;

The second is reserved to the modular exponentiation to calculate the intermediate parameters (R, P) while using the key (e) [4]. Finally the third stage (Re-Mapping), is reserved to the calculation of encrypt signal.

$$\mathcal{R}: = Monpro\ (1, R, \mathcal{M}).$$

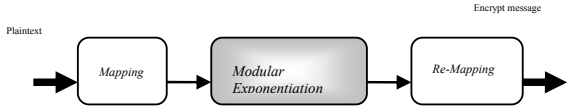

Fig. 9 RSA cryptosystem based on the Montgomery algorithm

VI. Application

In this part, we have developed our application. Indeed, to lead the correct results it has been necessary to pass by various stages. First, we started with developing programs in VHDL for all blocks of the application. Then, we used MAX+PLUS II [5] software to compile and simulate our system. The elements of our application are regrouped to implant the architecture of the algorithm monpro2. Fig. 10 shows entrances and exits of this block; Figs. 11 and 12 present algorithm simulation's results.

Fig. 10 Block of the Montgomery multiplication

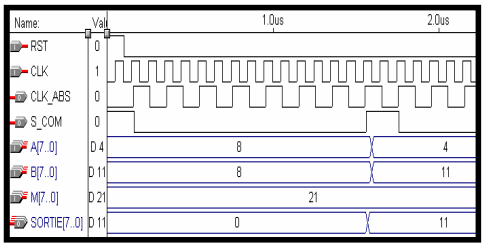

Fig. 11 Monpro2 (8, 8, 21)

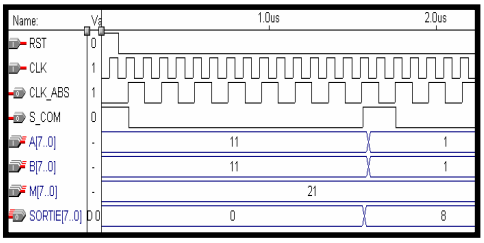

Fig. 12 Monpro2 (11, 11, 21)

Figs. 11 and 12 represent the results of simulations of the Montgomery multiplication Monpro2, indeed in this part, we tried to verify this multiplication while using examples and to visualize commands signals while basing on the logic of algorithm 2.

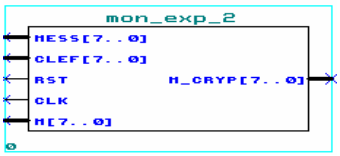

Fig. 13 Monexp2

This figure presents the block diagram of the algorithm of the modular exponentiation Monexp2.

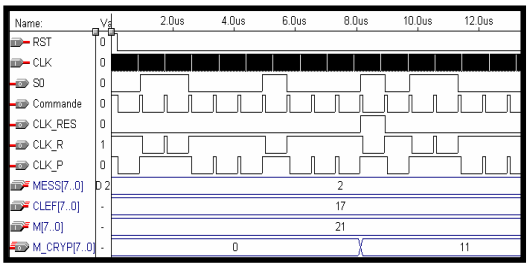

Fig. 14 Simulation Monexp2 (2, 17,21)

Fig. 14 presents the result of main program simulation (monexp2) that permits to encrypt a 8 bits message. It visualizes the different signals permitting manage of the modular exponentiation algorithm considering the public-key value KEY [7 ...0]. Then, we have used a key of 17, module M of 21 and a message of 2. Program execution presents an encrypt message equal to 11[6].

VII. Results

We used in our implantation the programmable circuit MAX 9000 - EPM9320LC84-15. Table I shows the number of Configurable Logic Block (CLB) occupied for the implementation of the architecture Monexp2, Input/Outputs blocks (I/OB) and the clock frequency.

TABLE I
Results of the Implantation

Architecture	Length of key	Frequency	I/OB	No.CLBs
Monpro2	8 bits	20MHZ	34	114 (35%)
Monexp2	8 bits	20 MHZ	34	224 (70%)

We have chosen an optimal architecture that permits to use 224 CLBs (70%) and 34 Input/Output Block of this circuit with a clock frequency of 20 MHz [8].

VIII. Conclusion

We proposed an efficient design to implement an optimized RSA cryptosystem for restricted system [9]. Reconfiguration can be used to load the values of M, C, and possibly E or D in the design before encryption or decryption beginning. Generally this is not always performing with many exponentiation calculations being executed with the same values of M,C, E, and D. Also, by having both E and D stored in registers, digital signatures can be appended to messages by simply encrypting part (or all) of the plaintext with the sender's private key[9], and then reloading this ciphertext into the encryptor to further encrypt it with the recipient's public key. To decrypt this message the receiver must first decrypt it with his own private key and then with the senders public key to confirm its origin.

References

[1] R. L. Rivest, A. Shamir, and L. Adleman. "A Method for obtaining digital signatures and public-Key cryptosystems". Comm. ACM, 21:120 126,1978.

[2] Schneier Bruce, Cryptographie appliquée - Algorithmes, protocoles et codes source en C - 2ème édition, International Thomson Publishing France, 1997-Applied Cryptography - Protocols, Algorithms, and Source Code in C - 2nd Edition.

[3] Bouallegue Ridha, Hamdi Omessaad « Sécurité des Crypto Systèmes ». *ENIT; SUPCOM, Tunis, Tunisie.2003.*

[4] Alan Daly and William Marnane "Efficient Architectures for implementing Montgomery Modular Multiplication and RSA Modular Exponentiation on Reconfigurable Logic". -University College Cork Ireland 2001.

[5] Young Sae Kim, Woo Seok Kang, Jun Rim Choi "Implementation of 1024-bit modular processor for RSA cryptosystem" School of Electronic and Electrical Engineering, Kyungpook National University, Korea.2001.

[6] John Fry - Martin Langhammer. "RSA & Public Key Cryptography in FPGA"2000.

[7] A.Mazzero, L.Romano "FPGA-based Implementation of a serial RSA" processor, G.P.Saggese-Universita'degli Studi Napoli "Federico II" 2002.

[8] Tom Kean "Cryptography Rights Management of FPGA Intellectual Property Cores" Edinburgh EH8 8YB United Kingdom.1999.

[9] S.H. Tang, K.S. Tsui and P.H.W. Leong "Modular Exponentiation using Parallel Multipliers" The Chinese University of Hong Kong Shatin, NT, Hong Kong 2001.

Ridha Ghayoula received the degree in automatic electric engineering in 2002 and the M.S. degree in electronics device from the Faculty of Sciences of Tunis, Tunisia, in 2005, He is currently working toward the Ph.D. degree in electrical engineering at the Faculty of Sciences of Tunis. His current research interests include intelligent antennas and microwave integrated circuits.