Volume 13 Issue 48
Originally issued as F15

Global Journal of Researches in Engineering
Electrical and Electronics Engineering
Volume 13 Issue 15 Version 1.0 Year 2013
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 2249-4596 & Print ISSN: 0975-5861

# Securing Distributed FPGA System using Commutative RSA Core

By R. Ambika, S. Ramachandran & K. R. Kashwan
*Vinayaka Missions University, India*

*Abstract-* Protecting important data is of utmost concern to the organizations or multiple transceiver based communication systems and, cryptography is one of the primary ways to do the job. RSA algorithm is extensively used in the popular implementations of Public Key Infrastructures. Many cryptographic protocols and attacks on these protocols make use of the fact that the order in which encryption is performed does not affect the result of the encryption, i.e., encryption is commutative. On the other hand, the need of a security feature encompassing data authentication among multiple MIMO or transceivers has become very critical. This paper presents the implementation of a cryptography core based on Commutative RSA public key cryptography algorithm for accomplishing data security and authentication in environment comprising multiple FPGA cores without any key exchange overheads. In spite of considering conventional two terminal communications, we have implemented a scalable architecture for multi distributed FPGA based systems and realizes commutative RSA algorithm for verifying data security among multiple transceiver terminals. In this approach, a sophisticated RSA cryptographic technique based on commutative Encryption methodology has been implemented for distributed FPGA terminals. The proposed system architecture has used the Montgomery multiplication algorithm with exponential modular multiplication and Radix-2 multiplication based multiparty cryptography.

*Keywords:* *authentication, cryptography, data security, FPGA, montgomery multiplication, RSA cryptosystem, Radix-2 multiplier.*

*GJRE-F Classification :* *FOR Code: 090699*

*Strictly as per the compliance and regulations of :*

© 2013. R. Ambika, S. Ramachandran & K. R. Kashwan. This is a research/review paper, distributed under the terms of the Creative Commons Attribution-Noncommercial 3.0 Unported License http://creativecommons.org/licenses/by-nc/3.0/), permitting all non commercial use, distribution, and reproduction in any medium, provided the original work is properly cited.

# Securing Distributed FPGA System using Commutative RSA Core

R. Ambika [α], S. Ramachandran [σ] & K. R. Kashwan [ρ]

*Abstract-* Protecting important data is of utmost concern to the organizations or multiple transceiver based communication systems and, cryptography is one of the primary ways to do the job. RSA algorithm is extensively used in the popular implementations of Public Key Infrastructures. Many cryptographic protocols and attacks on these protocols make use of the fact that the order in which encryption is performed does not affect the result of the encryption, i.e., encryption is commutative. On the other hand, the need of a security feature encompassing data authentication among multiple MIMO or transceivers has become very critical. This paper presents the implementation of a cryptography core based on Commutative RSA public key cryptography algorithm for accomplishing data security and authentication in environment comprising multiple FPGA cores without any key exchange overheads. In spite of considering conventional two terminal communications, we have implemented a scalable architecture for multi distributed FPGA based systems and realizes commutative RSA algorithm for verifying data security among multiple transceiver terminals. In this approach, a sophisticated RSA cryptographic technique based on commutative Encryption methodology has been implemented for distributed FPGA terminals. The proposed system architecture has used the Montgomery multiplication algorithm with exponential modular multiplication and Radix-2 multiplication based multiparty cryptography. The proposed multiplier is able to work with any precision of the input operands, limited only by memory or control constraints. The result obtained for this approach has illustrated a very high computational efficiency with minimum memory or space occupancy and higher operational frequency. The proposed PM based CRSA cryptography core has exhibited 12.1% higher throughput as compared to Serial Montgomery based CRSA. Similarly, the frequency or speed of the proposed system is also higher. The proposed system exhibits trade-off of 0.03% in power consumption.

*Keywords: authentication, cryptography, data security, FPGA, montgomery multiplication, RSA cryptosystem, Radix-2 multiplier.*

## I. Introduction

As the telecommunication network has grown explosively and the internet has become increasingly popular, security over the network is the main concern for services like electronic commerce [1]. The fundamental security requirements include confidentiality, authentication, data integrity, and non repudiation. 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) [2]. Many systems utilize public-key cryptography to provide such security services, and the algorithms developed by Rivest, Shamir, and Adleman (RSA) [3] is one of the most widely adopted public key algorithms at present. Since, RSA is considered as an efficient and optimized solution for public-key cryptography, we have implemented the Commutative RSA (CRSA) approach for authenticating data communication between Multiple Input Multiple Output (MIMO) or transceiver systems. In most of the existing data authentication or security systems, the authentication is accomplished by key exchange approach and thus it increases the key exchange overheads. On the other hand at every terminal, encryption and decryption process is required and thus if general RSA approach is applied in that case the data authentication and security could be violated. Therefore, in order to accomplish the goal of data security with individual encryption/decryption without affecting the data security and its integrity, a modified RSA has been developed and this mechanism is termed as Commutative RSA (CRSA).

RSA is the most widely used public-key cryptosystem. An RSA operation is an exponentiation, which requires repeated multiplications. The Montgomery multiplication algorithm [4] is the most efficient multiplication algorithm available. It replaces trial division by the modulus with a series of additions and divisions by a power of two. Thus, it is well suited to hardware implementation and forms the basis of many of the currently reported RSA hardware architectures [5–7]. To date, several techniques have been proposed in order to avoid carry propagation during the addition stages of the computation, as this is a key factor in determining performance. One approach proposed by Elbirt and Paar [6] is to break these additions into x-bit stages, where x is an optimal bit length chosen to take advantage of the fast carry chains available on modern FPGAs. However, a drawback of this approach is that the circuits developed can be very heavily technology

*Author α: Vinayaka Missions University, Salem, India e-mail: ambika2810@gmail.com*
*Author σ: S J B Institute of Technology, Bangalore, India. e-mail: ramachandr@gmail.com*
*Author ρ: Sona College of Engineering, Salem, India. e-mail: drkrkashwan@gmail.com*



© 2013 Global Journals Inc. (US)

and implementation dependent. For example, it is unlikely that a design created in this manner for a specific FPGA family will show the same speed advantages if migrated to a modern ASIC technology or, indeed, an alternative type of FPGA or Programmable Logic Device (PLD). An alternative approach presented by Blum and Paar [7] is based on the use of FPGA systolic array multiplier architectures with varying processing element sizes, namely, 4, 8 and 16 bits. However, these systems are again tailored specifically for the XilinxFPGA series.

As the operands such as the plain text of a message or the cipher or possibly a partially ciphered text are usually large and, in order to improve time requirements of the encryption/decryption operations, it is essential to attempt to minimize the number of multiplications performed and to reduce the time requirement of a single multiplication. There are various algorithms that implement multiplication. But considering the versatility and robustness of Montgomery multiplication approach, we have used Montgomery Multiplication algorithm. The most attractive feature of Montgomery algorithm is that it computes multiplications without trial divisions.

The RSA algorithm and Diffie-Hellman key exchange scheme need exponentiation, which binary or m-ary methods can break into a series of multiplications. It is effectively accomplished by Montgomery multiplication algorithm. Montgomery algorithm speeds up the multiplications and squaring required for exponentiation. The efficient implementation of this long-word length multiplication is crucial for the performance of public-key cryptography like our proposed CRSA. Exponentiation with a large modulus, which is usually accomplished by repeated multiplications, has been widely used in public key cryptosystems for secured data communications. To speed up the computation, the Montgomery multiplication algorithm is used to relax the process of quotient determination and, the carry-save addition (CSA) is employed to reduce the critical path delay. Basically, the exponentiation with a large modulus is usually accomplished by performing repeated multiplications, which is considerably time-consuming. As a result, the throughput rate of RSA cryptosystem will be entirely dependent on the speed of multiplication and the number of performed multiplications. One way to achieve this is to use carry save adders (CSAs) to perform the addition stages of Montgomery's algorithm. For example, Kim et al. [8] used two levels of carry save logic (CSL) and a 32-bit carry propagate adder along with a 32 x 32-bit shift register in order to perform the 1024-bit additions required. Bunimov et al. [9] improved this by replacing one level of CSL with a look-up table.

In order to accomplish the goal of data security and authentication among multiple MIMO or transceiver terminals with proposed Commutative RSA cryptographic algorithm, we have implemented an enhanced and optimized noble data authentication architecture called Commutative RSA algorithm with multiple MIMO or transceiver systems, and simulated on FPGA devices. In this approach, three FPGA cores have been considered in simulation framework and simulation for RSA encryption and decryption has been accomplished at every considered terminal. The developed architecture encompasses the Montgomery modular multiplication approach to speed up the computation and to relax the process of quotient determination and similarly the carry-save addition has been employed to reduce the critical path delay. The proposed multiplier is able to work with any precision of the input operands, limited only by memory or control constraints. In order to make the system compatible with Very Large Scale Integration and to get optimized performance, the system architecture has been developed with Montgomery multiplication with Radix-2 multiplier based architecture. We have implemented two different CRSA implementation architectures. One is Serial Montgomery implementation and another one represents Parallel Montgomery based CRSA core. The performance for both architectures for delay, frequency, efficiency, power consumption as well as throughput have been calculated and we have found that the proposed Parallel Montgomery (PM) based CRSA performs far better than serial Montgomery (SM) based CRSA core.

The remaining paper has been divided into the following sections. Section 2 discusses in brief the literature survey conducted for the research work with emphasis on RSA algorithm and implementation of Montgomery multiplication with Radix-2 architecture. Section 3 discusses the proposed Commutative RSA algorithm and presents the mathematical derivation for CRSA approach. Section 4 represents the proposed commutative RSA core based on serial Montgomery and parallel Montgomery multipliers. The hardware implementation has been presented in Section 5 followed by Section 6 that presents the results and analysis of the research work. The conclusion has been given in the last section.

## II. Related Works

Gustavo D. Sutter et. al [10] optimized the Montgomery's multiplication and proposed architectures to perform the least significant bit first and the most significant bit first algorithms. The developed architecture has the following distinctive characteristics: 1) use of digit serial approach for Montgomery multiplication. 2) Conversion of the CSA representation of intermediate multiplication using carry–skip addition. This allows the critical path to be reduced, albeit with a small-area speed penalty; and 3) recomputed the quotient value in Montgomery's iteration in order to

© 2013 Global Journals Inc. (US)

speed up the operating frequency. In this paper, researchers presented results in Xilinx Vertex 5 and in 0.18-µm application-specified integrated circuit technologies.

Jin-Hua and Cheng-Wen [11] proposed a radix-4 modular multiplication algorithm based on Montgomery's algorithm, and a fast radix-4 modular exponentiation algorithm for RSA public-key cryptosystem. The proposed multiplier is four-times faster than a direct radix-2 implementation of Montgomery's algorithm. Extending the design for a larger modulus is straightforward. High-radix bit-level and digit-level modular multipliers have also been discussed.

C. McIvor et.al [12] presented Modified Montgomery multiplication and associated RSA modular exponentiation algorithms and circuit architectures. Practical approach presented is based on a reformulation of the solution to modular multiplication within the context of RSA exponentiation.

Alexandre F. Tenca and C¸ etin K. Koc [13] presented a scalable architecture for the computation of modular multiplication based on the Montgomery multiplication algorithm. A word-based version of is presented and used to explain the main concepts in the hardware design. The proposed multiplier is able to work with any precision of the input operands, limited only by memory or control constraints.

Marcelo E. and Naofumi Takagi [14] proposed a mixed radix-4/2 algorithm for modular multiplication/division for a large modulus suitable for VLSI implementation. The calculation of modular multiplication is based on the Montgomery multiplication algorithm and the modular division on the extended Binary GCD algorithm. The researchers exploit these similarities to modify the algorithms in order to share almost all hardware components for both operations.

Koç, C.K., et.al [15] studied the operations involved in computing the Montgomery product and describe several high-speed, space-efficient algorithms for computing MonPro (a, b), and analyzed their time and space requirements. Their focus is to collect several alternatives for Montgomery multiplication, three of which are new. However, the researchers do not compare the Montgomery techniques to other modular multiplication approaches.

Ching-Chao Yang et. al [16] proposed a new algorithm based on Montgomery's algorithm to calculate modular multiplication that is the core arithmetic operation in an RSA cryptosystem. The modified algorithm eliminates over-large residue and has very short critical path delay that yields a very high-speed processing. The researchers have implemented a 512-bit single-chip RSA processor based on the modified algorithm with Compass 0.6-$\mu$m SPDM CMOS cell library.

Along with the strong momentum of shifting from single-core to multicore systems, Zhimin Chen et. al [17] present a parallel-software implementation of the Montgomery multiplication for multicore systems. Their comprehensive analysis shows that the proposed scheme, pSHS, partitions the task in a balanced way so that each core has the same amount of job to do. In addition, we also comprehensively analyze the impact of inter-core communication overhead on the performance of pSHS. The analysis reveals that pSHS is high performance, scalable over different number of cores, and stable when the communication latency changes.

GuilhermePerin et. al [18] described a comparison of two Montgomery modular multiplication architectures: a systolic and a multiplexed. Both implementations target FPGA devices. The modular multiplication is employed in modular exponentiation processes, which are the most important operations of some public-key cryptographic algorithms, including the most popular of them, the RSA. The proposed systolic architecture presents a high-radix implementation with a one-dimensional array of Processing Elements.

The RSA algorithm proposed by P. Fournaris and O. Koufopavlou [19] has gained wide acceptability and has been well used algorithm in many security applications. Its main mathematical function is demanding in terms of speed, operation of modular exponentiation. In this article, a systolic, scalable, redundant carry-save modular multiplier and RSA encryption architecture are proposed using the Montgomery modular multiplication algorithm.

Perovic, N. S. et. al [23] presented FPGA implementation of RSA algorithm, where a key is 1024 bits long and the project synthesis results like resource occupancy, maximal operating frequency, etc. were examined for the system implementation.

## III. Proposed System

Highly robust and optimized system architecture for implementation of $\boldsymbol{Commutative\ RSA}$ algorithm for data authentication among multiple MIMO terminals (here simulated on FPGA devices) has been proposed in this paper. In order to facilitate the secure data communication among multiple MIMO or transceiver systems, a noble commutative RSA approach that states that, the order in which encryption is performed does not affect the result of the encryption, has been implemented and simulated on multiple FPGA devices. In order to optimize the performance of the system with minimum space and higher speed, the robust Montgomery modular multiplication mechanism has been adopted with $\boldsymbol{Radix-2}$ multiplication architecture. We have proposed the implementation of Serial Montgomery as well as Parallel Montgomery based CRSA cryptography core, with a goal to enhance the system performance for its less memory occupancy, fast rate, higher throughput and less power consumption.



© 2013 Global Journals Inc. (US)

*a) Commutative RSA*

A secure plane is realizable provided the data communicated over the plane is protected and cannot be colluded. The use of cryptographic techniques is generally preferred, hence the *Secure Multi FPGA Communication Protocol* (*SMFCP*) proposed in this paper adopts the commutative RSA algorithm. The *SMFCP* considers two prime numbers $Param_P_p^{CRSA}$ and $Param_Q_q^{CRSA}$ initialized amongst all the group members. $G_A$ Let and $G_B$ represent the group members required to communicate over the secure plane. To compute the encryption keys and decryption key pairs of the commutative RSA algorithm, the Property $Prop_N^{CRSA}$ and $Prop_\phi^{CRSA}$ are computed using the following equations:

$$Prop_N^{CRSA} = \left[\left(Prop_P_p^{CRSA}\right) \times \left(Prop_Q_q^{CRSA}\right)\right] \quad (1)$$

$$Porp_\phi^{CRSA} = \left[\left(Prop_P_p^{CRSA} - 1\right) \times \left(Prop_Q_q^{CRSA} - 1\right)\right] \quad (2)$$

From the above equations, it is clear that

$$Param_N_A{}^{CRSA} = Param_N_B{}^{CRSA} \quad (3)$$

and $Porp_\phi_A{}^{CRSA} = Prop_\phi_B{}^{CRSA}$ for $A$ and $B$ (4)

The encryption key pair of $A$ and $B$ represented as

$$(Prop_N_A{}^{CRSA}, Porp_E_A{}^{CRSA}) \text{ and } (Prop_N_B{}^{CRSA}, Prop_E_B{}^{CRSA})$$

are to be obtained. The $Param_E^{CRSA}$ is obtained by randomly selecting numbers such that it is a co prime of $Prop_\phi^{CRSA}$ or in other terms:

$$\mathcal{FN}_{GCD}(Prop_E^{CRSA}, Prop_\phi^{CRSA}) = 1 \quad (5)$$

where $\mathcal{FN}_{GCD}(x, y)$ represents the greatest common divisor function between two variables $x$ and $y$.

The decryption key pair of $A$ and $B$ is represented by $(Prop_N_A{}^{CRSA}, Prop_D_A{}^{CRSA})$ and $(Prop_N_B{}^{CRSA}, Prop_D_B{}^{CRSA})$ and the Property $Prop_D^{CRSA}$ is computed based on the following equation:

$$Prop_D^{CRSA} = (Prop_E^{CRSA})^{-1} Mod(Prop_N^{CRSA}) \quad (6)$$

Let $Enc_X$ represent the encrypted data $X$. The encryption operation is defined as follows:

$$Enc_X = Y^{Prop_E^{CRSA}} Mod(Prop_N^{CRSA}) \quad (7)$$

The commutative RSA decryption operation on the encrypted data $\mathbb{B}$ is defined as

$$Dec_Y = Y^{Prop_D^{CRSA}} Mod(Prop_N^{CRSA}) \quad (8)$$

*b) Commutative property of RSA Algorithm*

The commutative property of the RSA algorithm adopted in SMFCP can be proved if data X encrypted by A and then encrypted by B provides the same resultant if the encryption is performed by B followed by the encryption performed by A, i.e.,

$$\mathrm{Enc^B(Enc_x{}^A)} \equiv \mathrm{Enc^A(Enc_x{}^B)} \quad (9)$$

$$\mathrm{Enc^B}\left(\mathrm{X^{Prop_E_A{}^{CRSA}}\ Mod(Prop_N_A{}^{CRSA})}\right) \equiv \mathrm{Enc^A}\left(\mathrm{X^{Prop_E_B{}^{CRSA}}\ Mod(Prop_N_B{}^{CRSA})}\right) \quad (10)$$

$$\mathrm{X^{\left(Prop_E_A{}^{CRSA} \times Prop_E_B{}^{CRSA}\right)}\ Mod\left(Prop_{N_A}{}^{CRSA}\right) = X^{\left(Prop_E_B{}^{CRSA} \times Prop_E_A{}^{CRSA}\right)}\ Mod(Prop_N_B{}^{CRSA})} \quad (11)$$

As $\mathrm{Prop_{N_A}{}^{CRSA} = Prop_N_B{}^{CRSA}}$ it can be concluded that

$$\mathrm{X^{\left(Prop_E_A{}^{CRSA} \times Prop_E_B{}^{CRSA}\right)}\ Mod\left(Prop_{N_A}{}^{CRSA}\right) = X^{\left(Prop_E_B{}^{CRSA} \times Prop_E_A{}^{CRSA}\right)}\ Mod(Prop_N_A{}^{CRSA})} \quad (12)$$

And hence $\mathrm{Enc^B(Enc_X{}^A) \equiv Enc^A(Enc_X{}^B)}$ (13)



© 2013 Global Journals Inc. (US)

## IV. Proposed Commutative RSA Core Based on Serial Montgomery and Parallel Montgomery

The dominant goal of this research work is to implement and illustrate the efficiency and robustness of commutative RSA cryptography approach for multiple MIMO or transceiver systems and for this purpose, we have implemented Commutative RSA cryptography core among multiple FPGA devices. In order to optimize the performance as well as memory occupancy, highly effective system architectures like Montgomery modular multiplication based on Radix-2 has been developed. Such implementation causes the reduction in memory occupancy as well as the speed is also enhanced many folds. These implemented approaches have been discussed in the following sections.

*a) Montgomery Algorithm*

Montgomery multiplication [20] is an efficient method for modular multiplication with an arbitrary modulus, particularly suitable for implementation on general-purpose computers and embedded micro-processors. The method is based on a representation of the residue class modulo $M$. The algorithm uses simple divisions by a power of two instead of divisions by $M$, which are used in a conventional modular operation. The Montgomery multiplication (MM) is the basic operation used in modular exponentiation, which is required in the Diffie-Hellman and RSA public-key cryptosystems.

Montgomery's modular multiplication algorithm employs only simple additions, subtractions, and shift operations to avoid trial division, a critical and time-consuming operation in conventional modular multiplication. The price paid is the need to convert operands into and out of Montgomery's domain, which is almost negligible in some particular applications such as cryptosystems.

Montgomery modular multiplication is one of the fundamental operations used in cryptographic algorithms, such as RSA and Elliptic Curve Cryptosystems. The Multiple-Word Radix-2 Montgomery Multiplication algorithm represents a now-classic architecture for implementing Montgomery multiplication in hardware. With properties optimized for minimum latency, this architecture performs a single Montgomery multiplication in approximately 2n clock cycles, where "n" is the size of operands in bits.

In many cryptosystems, such as RSA, computing $M$ is a crucial operation. The reduction of $M$ is a more time-consuming step than the multiplication A . B without reduction. Montgomery introduced a method for calculating products $(\mathrm{mod\ M})$ without the costly reduction $(\mathrm{mod\ M})$ , since then known as Montgomery multiplication. $M$ is assumed to be an odd integer. Montgomery multiplication of $A$ and B $(\mathrm{mod\ M})$, denoted by $\mathrm{MP(A, B, M)}$ is defined as $\mathrm{A\,.\,B.}\,2^n\,(\mathrm{mod\ M})$ for some fixed integer n. Since Montgomery multiplication is not an ordinary multiplication, there is a conversion process between the ordinary domain (with ordinary multi-plication) and the Montgomery domain. The conversion between the ordinary domain and the Montgomery domain is given by the relation $\mathrm{A} \leftrightarrow \mathrm{A}'$ where $A' = \mathrm{A.}\,2^n(\mathrm{Mod\ M})$.

Mathematically, it can be written as:

$$\mathrm{MP}(A', \mathrm{B}', \mathrm{M}) = \mathrm{A}'.\mathrm{B}'.2^{-\mathrm{n}} = (\mathrm{A}.2^n).(\mathrm{B}.2^n).2^{-\mathrm{n}} = \mathrm{A.B.}\,2^n = (\mathrm{A.B})'(\mathrm{mod\ M}). \quad (14)$$

The conversion between each domain can be done using the same Montgomery operation, in particular $\mathrm{A}' = \mathrm{MP}(\mathrm{A}, 2^{2\mathrm{n}}\,(\mathrm{mod\ M}), \mathrm{M})$ and $\mathrm{X} = \mathrm{MP}(\mathrm{A}', 1, \mathrm{M})$ , where $2^{2n}\,(\mathrm{mod\ M})$ can be precomputed. Despite the initial conversion cost, we achieve an advantage over ordinary multiplication if we do many Montgomery multiplications followed by an inverse conversion at the end, which is the case, for example, in our proposed RSA.

*b) Radix-2 Modular Multiplier*

The optimized algorithm for Radix-2 Modular multiplier for Montgomery multiplication is given as follows:

$$\boldsymbol{Input}: Odd\ M, n = \lfloor log_2 M \rfloor + 1, \quad (15)$$

$$A = \textstyle\sum_{i=0}^{n-1} a_i\,.\,2^i, with\ 0 \le A, B < M \quad (16)$$

$$\boldsymbol{Output}: \boldsymbol{C} = MP(A, B, M) \equiv A.B.2^{-n}(mod\ M), 0 \le C < M \quad (17)$$

$$1.1\ \mathrm{X}[0] = 0; \quad (18)$$

$$1.2\ For\ \ i = 0\ to\ n-1\ \ \boldsymbol{do};$$

$$1.3 \begin{cases} q_{\ i} = (a_i.b_0) \oplus \mathrm{X}[i]o; \\ \mathrm{X}[i+1] = \frac{X[i] + a_i.b + q_{\ i}.m}{2}; \end{cases} \quad (19)$$



© 2013 Global Journals Inc. (US)

$$1.4\ \mathrm{X}[n] > M\ \ \textbf{then}$$

$$1.5[\mathrm{X}[n] = X[n] - M;$$

$$1.6\ \textbf{return}\ \ \boldsymbol{C} = \mathrm{X}[n] \tag{20}$$

The above mentioned algorithm represents the Pseudocode for the Radix-2 Montgomery multiplication, where we choose $n = \lfloor log_2 M \rfloor + 1$. $n$ is the size of M in bits.

The verification of the above algorithm may be presented as follows:

Consider X[i] given as

$$\mathrm{X}[i] \equiv \frac{1}{2^i}\left(\sum_{j=0}^{i-1} a_j . 2^j\right).B(Mod\ \ M), \tag{21}$$

With X[0]=0. Then $X[n] \equiv A.B.2^{-n}(mod\ M) = MP(A,B,M).X[n]$ (22)

can be computed iteratively using the following dependence:

$$\equiv X[i+1] \equiv \frac{1}{2^{i+1}}\left(\sum_{j=0}^{i} a_j . 2^j\right).B \tag{23}$$

$$\equiv \frac{1}{2^{i+1}}\left(\sum_{j=0}^{i} a_j . 2^j + a_i . 2^i\right).B \tag{24}$$

$$\frac{1}{2}\left(\frac{1}{2^i}\left(\sum_{j=0}^{i-1} a_j . 2^j\right).B + a_i . B\right) \tag{25}$$

$$\frac{1}{2}(X[i] + a_i . B)(mod\ \ M). \tag{26}$$

Therefore, depending on the parity of $\mathrm{X}[i] + a_i.Y$, we do compute $\mathrm{X}[i+1]$ as or $X[i+1] = \frac{a[i]+a.B+M}{2}$ so as to make the numerator divisible by 2.

Since $B < M$ and $\mathrm{X}[0] = 0$, one has $0 \le X[i] < 2M$ for all $0 \le i < n$. In References [21] and [22], the result of a Montgomery multiplication is presented as $A.B.2^{-n}(mod\ M) < 2M$ when $A, B < 2M$ and $2^n > 4M$. As a result, by redefining "n" to be the smallest integer such that $2^n > 4M$, the subtraction at the end of algorithm can be avoided and the output of the multipication can be directly used as an input for the next Montgomery multiplication.

*c) Modular Multiplication Algorithms*

In RSA, the public encryption key is a pair of positive integers (E, N) and the private decryption key is another pair of positive integers (D, N). To encrypt a message using the key (E, N) the following structural approach have been implemented. Fig. 1 represents the Serial Montgomery multiplication, whereas the parallel Montgomery is presented in Fig. 2. It encompasses two Montgomery multipliers connected in parallel. In our research work, we have implemented $Radix-2$ Modular multiplier based multiplication architecture. A brief description of the employed algorithm is as follows:

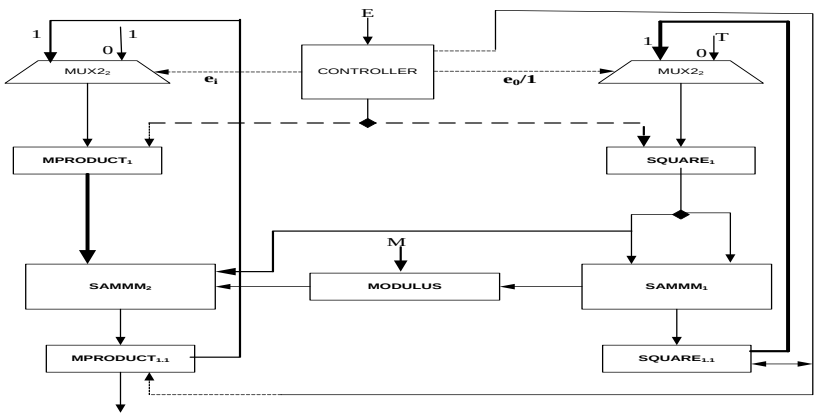


*Figure 1 :* Serial Montgomery Multiplication Architecture



© 2013 Global Journals Inc. (US)

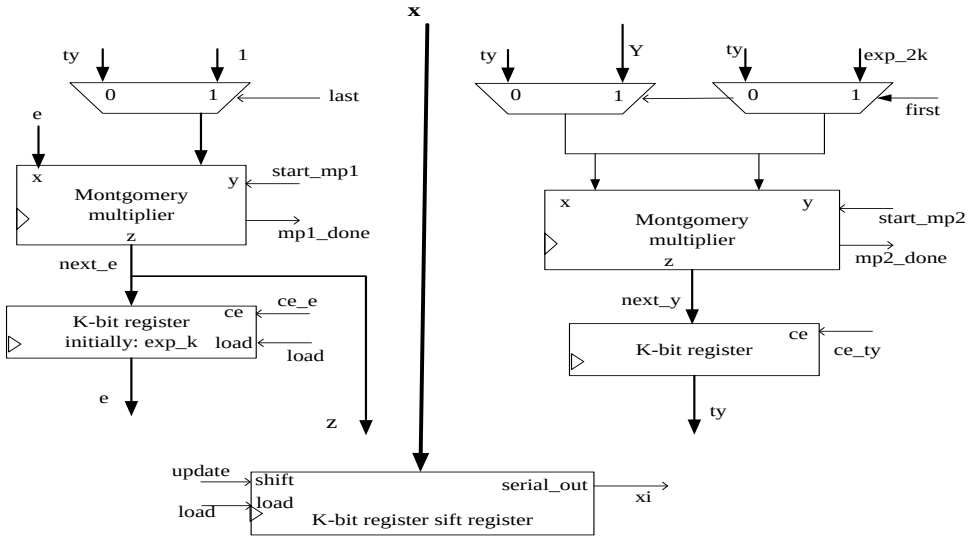


*Figure 2 :* Montgomery exponentiation (MSB first) with two Montgomery's multipliers in parallel

## V. Hardware Design

Fig. 2 presented earlier shows the architecture of a 32-bit RSA processor based on the proposed Commutative RSA algorithm. We use four 32-bit linear shift registers to store operands needed in computing 32-bit RSA operation. The operations of the RSA processor are described in the following. In the initial stage, commutative RSA operands are loaded into shift registers serially through an input buffer. While loading message M into the text register, we shift the exponent register until the first nonzero is the most significant bit and count the number of bits of exponent $\log_2 E$. After the initial stages, we start the multiplier. Once the first output bit of the multiplier is ready, we start the Montgomery module immediately. So the execution time of CPA, multiplier, and Montgomery module is almost overlapped. Therefore, the function units of our design are fully utilized during computation.

*Carry-Propagation Adder and Serial Parallel Multiplier:* The carry-propagation adder converts the carry-save form of the output from the Montgomery module to non-redundant binary form. It generates one bit output per cycle to the serial-parallel multiplier for the next iteration. The serial-parallel multiplier is used to realize the multiplication and square of two n +1 bit numbers. It first generates the $n+2$ lower bits of a product serially to the Montgomery module, and then it stops and holds the n higher bit of the product. The n higher bits of the product will be added with the output of the Montgomery module to get the modular multiplication result.

The multiplier itself is a linear array type with a special input circuit. When the multiplier is generating a product of two numbers, the parallel input $M0$ is ready in the text register and another operand can arrive in serial. However, if we want to square one number, a serial input of the operand will make the multiplier fail. We solved this problem by scheduling the serial input operands and insert some zeros to avert the failure of the squaring operation.

*Montgomery Module:* The Montgomery module is shown in Fig. 2 and the overall operation for Montgomery modular multiplication and its functional approach has already been presented in previous sections. The variable $X[0]$ refers the n+2 lower bit of the product from the multiplier. $X[0]$ enters the Montgomery module one bit per cycle from the lower bit to the higher bit in series. The reduction step is a shift-and-add operation that is very similar to the basic step of a multiplication. The quotient determination is a parity decision on the summation of the intermediate result and the carry. This can be done simply by an exclusive-OR gate with inputs of $X$[i] and the $LSB$ of the intermediate result in the previous iteration. After $n+2$ iterations, the Montgomery module will add X $[n+2]$ and the $n$ higher bits of the product from the multiplier together. The result is then sent to the carry-propagation adder for the next modular multiplier iteration.

In this work, we have developed two CRSA cryptography cores. First model represents the Serial Montgomery multiplier based design, while the second describes the optimized Parallel Montgomery based CRSA cryptography core implementation. In parallel Montgomery approach, two Montgomery multipliers have been used in parallel.

The results obtained after implementation have been summarized in the following sections.

## VI. Results

The robust commutative RSA core, whose details were presented in earlier sections, has been implemented on multiple FPGA devices for simulation and illustration of data authenticity among multiple user terminals in a communication environment. The



© 2013 Global Journals Inc. (US)

proposed work for the implementation of commutative RSA cryptography core has been simulated with three individual FPGA devices. The implementation of FPGA cores do signify the MIMO or multiple transceiver terminals in multiuser communication environment. The design has been coded in VHDL and has been simulated using Xilinx Design Suite 14.3 targeted on Virtex-5, xc5vlx330t-2-ff1738FPGA. In this work, two systems have been developed as mentioned earlier. One is the Serial Montgomery based Cryptography core and the second is our proposed Parallel Montgomery based cryptography core. The results obtained for both architectures have been compared. Considering the performance parameters like Memory occupancy, speed, power consumption, delay and throughput, it has been found that the Parallel Montgomery performs better than Serial Montgomery (SM) based Commutative RSA implementation. The delay in Parallel Montgomery based CRSA is 13.78% lower as compared to Serial Montgomery based CRSA cryptography core. Similarly, the throughput of Parallel Montgomery based CRSA is 12.11% higher than the serial Montgomery based CRSA architecture. Even in the proposed system, the trade-off between power consumption is also very small and it is only 0.03% higher in Parallel Montgomery based CRSA.

The simulation results for encryption and decryption obtained by the Serial Montgomery based $CRSA$ core is presented in Fig. 3 and Fig. 4 respectively. The functional verification of the Parallel Montgomery based $CRSA$ cryptographic core is shown in Fig. 5 and Fig. 6.

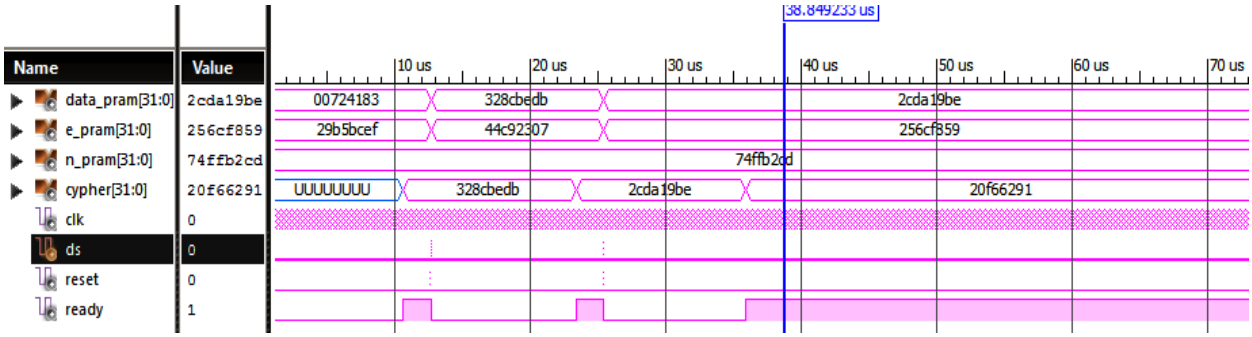


*Figure 3 :* Simulation Waveforms Using Serial Montgomery based : Encryption

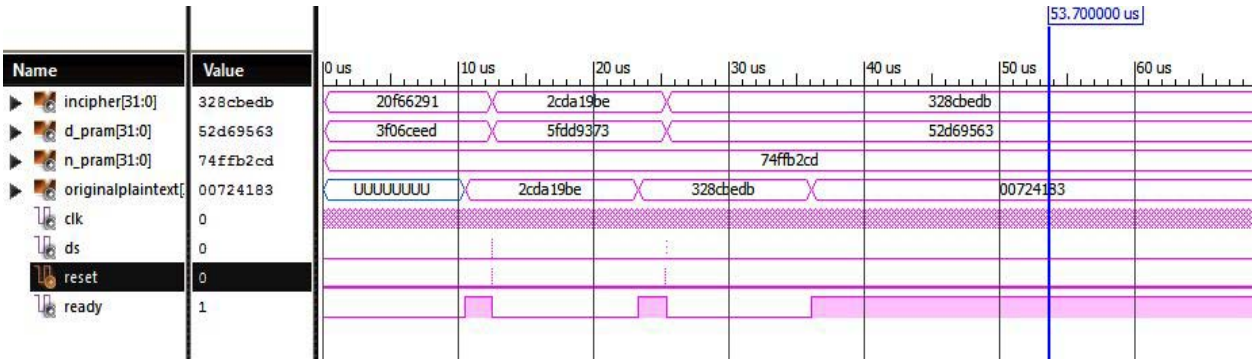


*Figure 4 :* Simulation Waveforms Using Serial Montgomery based : Decryption

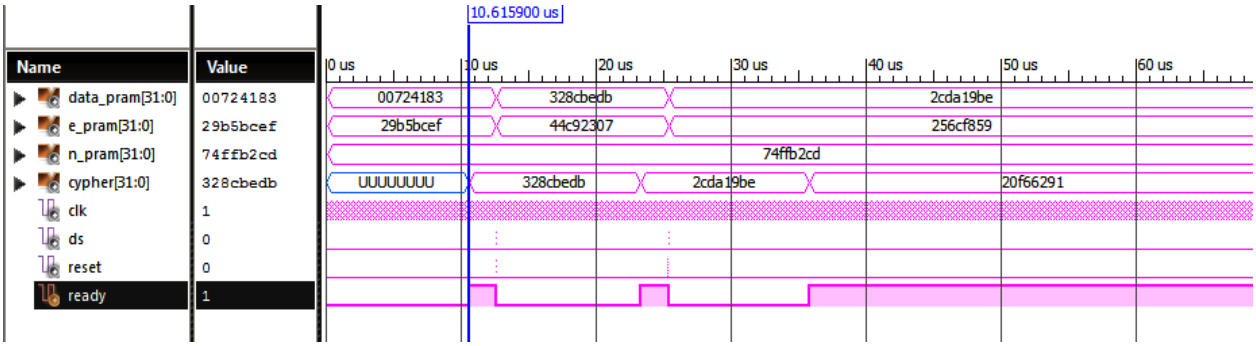


*Figure 5 :* Simulation Waveforms Using Parallel Montgomery based : Encryption



© 2013 Global Journals Inc. (US)

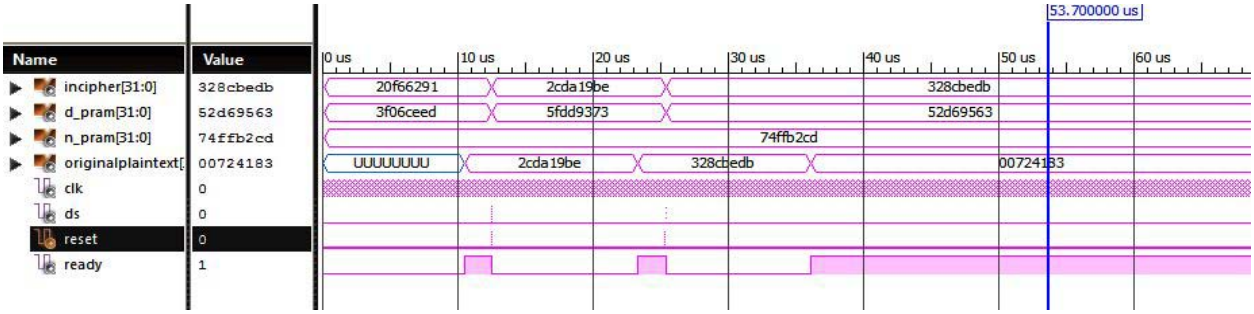


*Figure 6 :* Simulation Waveforms Using Parallel Montgomery based : Decryption

The results obtained for comparative simulation are presented in the following.

*Table 1 :* Comparison for Chip Resource Utilization in Serial and Parallel Montgomery based CRSA cryptography Core

| CRYPTOGRAPHY CORE | CRSA | CRSA |
|---|---|---|
| CIRCUIT | SERIAL MONTGOMERY | PARALLEL MONTGOMERY |
| DEVICE | xc5vlx330t-2-ff1738 | xc5vlx330t-2-ff1738 |
| SLICE LUT | 913 | 844 |
| LUT USED AS LOGIC | 913 | 813 |
| OCCUPIED SLICES | 290 | 311 |

*Table 2 :* Comparison for Power consumption in Serial and proposed Parallel Montgomery based Commutative RSA cryptography core

| CRYPTOGRAPHY CORE | CRSA | CRSA |
|---|---|---|
| CIRCUIT | SERIAL MONTGOMERY | PARALLEL MONTGOMERY |
| DEVICE | xc5vlx330t-2-ff1738 | xc5vlx330t-2-ff1738 |
| STATIC POWER (mW) | 3516.7 | 3516.75 |
| DYNAMIC POWER (mW) | 4.76 | 5.72 |
| TOTAL POWER (mW) | 3521.46 | 3522.47 |

*Table 3 :* Performance (Delay, Frequency and Throughput) Analysis for Serial and Parallel Montgomery Based CRSA Cryptography Core

| CRYPTOGRAPHY CORE | CRSA | CRSA |
|---|---|---|
| CIRCUIT | SERIAL MONTGOMERY | PARALLEL MONTGOMERY |
| DEVICE | xc5vlx330t-2-ff1738 | xc5vlx330t-2-ff1738 |
| FREQUENCY (MHz) | 199.57 | 227.08 |
| DELAY (ns) | 5.01 | 4.40 |
| THROUGHPUT(kbps) | 779.58 | 887.02 |

The graphical comparison for the performance of Serial and Parallel Montgomery based Commutative RSA architectures has been presented in Fig. 7 to Fig. 10.

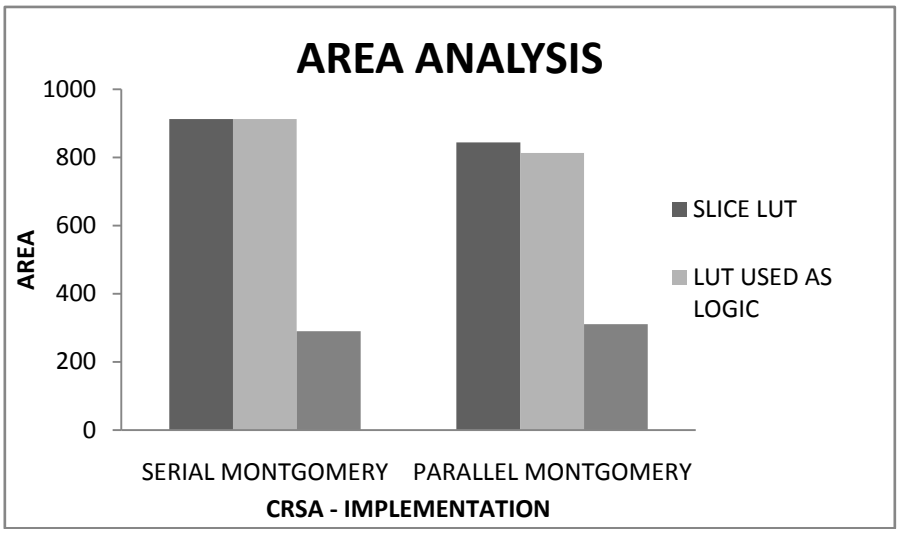


*Figure 7 :* Chip Resource Utilization Analysis for serial and parallel Montgomery based CRSA



© 2013 Global Journals Inc. (US)

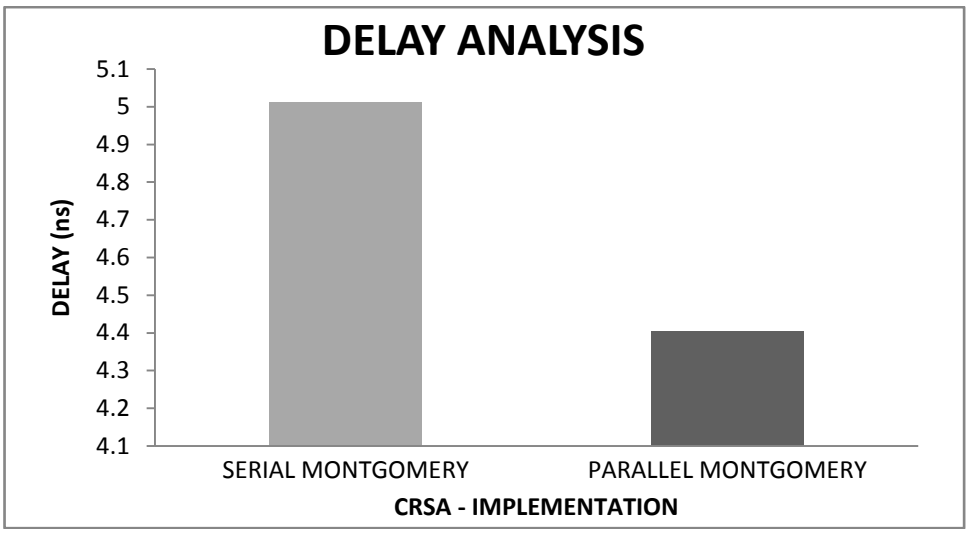


*Figure 8 :* Delay analysis for serial and parallel Montgomery based CRSA

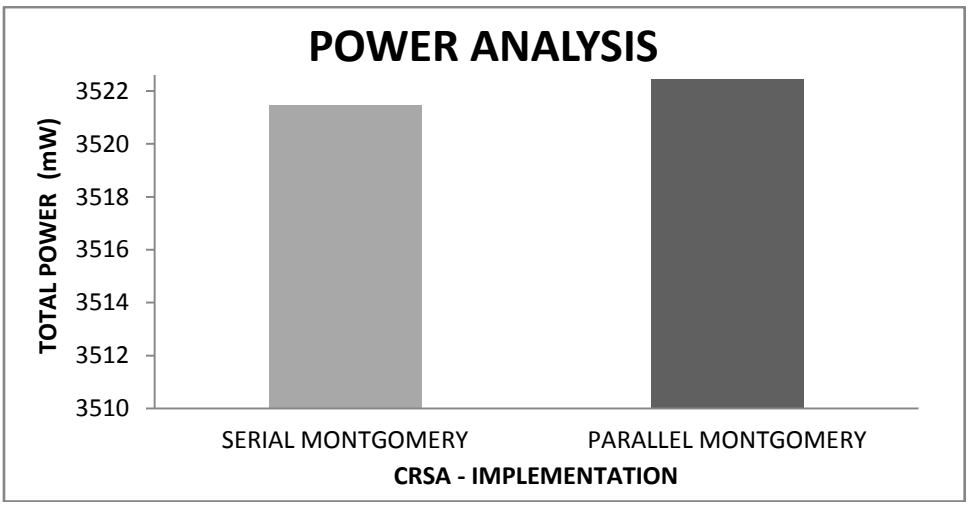


*Figure 9 :* Power analysis for serial and parallel Montgomery based CRSA

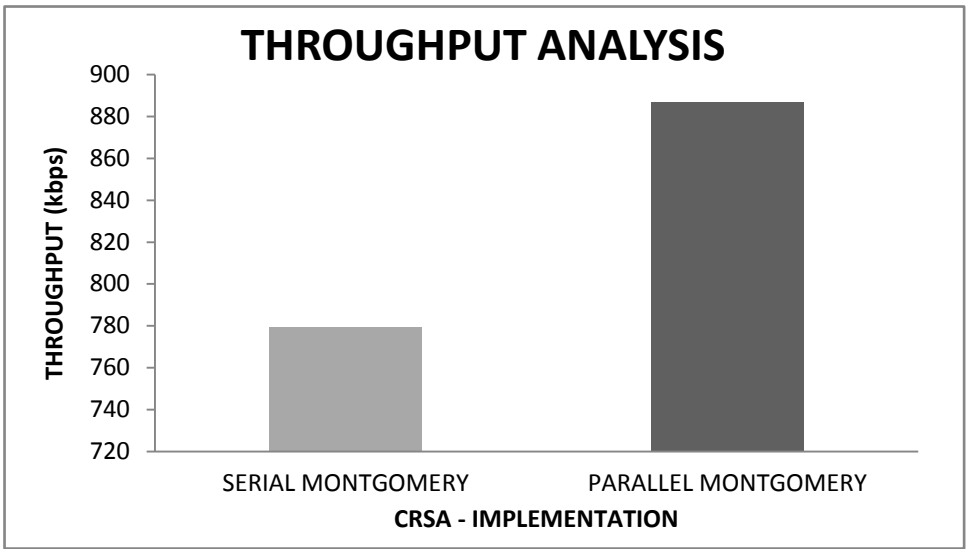


*Figure 10 :* Throughput analysis for serial and parallel Montgomery based CRSA



© 2013 Global Journals Inc. (US)

## VII. Conclusion

A noble security or authentication public key cryptography technique called Commutative RSA has been implemented for multiple MIMO or transceiver terminals for accomplishing the goal of data security in multiuser communication environment. The commutative RSA approach has been implemented with multiple FPGA cores that functions as individual transceiver terminal and performs its encryption and decryption individually without affecting the original data. The two approaches based on Montgomery multiplication with Radix-2 multiplier have been designed and individual modules for Serial Montgomery (SM) and Parallel Montgomery have been simulated. The results obtained have been compared and it has been found that the proposed Parallel Montgomery (PM) architecture performs better as compared to Serial Montgomery. The proposed PM based CRSA cryptography core has exhibited 12.1% higher throughput as compared to Serial Montgomery based CRSA. Similarly, the frequency or speed of the proposed system is also higher. The proposed system exhibits trade-off of 0.03% in power consumption. Thus considering various aspects of this research work, it can be stated that the proposed Parallel Montgomery based Commutative RSA performs better than the serial based Montgomery multiplication application.

## References Références Referencias


1. B. Premkumar, "An RNS to binary converter in 2n+1, 2n, 2n-1 moduli set," IEEE Trans. Circuits Syst. II, Vol. 39, pp. 480–482, July 1992.
2. SCHNEIER, B., Applied Cryptography: Protocols, Algorithms, and Source Code in C, John Wiley & Sons, 1996.
3. R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signature and public-key cryptosystems," Commun. ACM, Vol. 21, No. 2, pp. 120–126, Feb. 1978.
4. Montgomery, P. L.: 'Multiplication without Trial Division', Math. Computation, 1985, 44, pp. 519–521.
5. Eldridge, S. E., and Walter, C. D., 'Hardware Implementation of Montgomery's Multiplication Algorithm', IEEE Trans. Comput., 1993, 42, pp. 693–699.
6. Elbirt, A. J., and Paar, C., 'Towards an FPGA Architecture Optimized for Public-Key Algorithms'; the SPIE Symposium on Voice, Video and Communications, Sept. 1999.
7. Blum, T., and Paar, C., 'Montgomery Exponentiation on Re-configurable Hardware'. Proc. 14th Symposium on Computer Arithmetic, 1999, pp. 70–77.
8. Kim, Y. S., Kang, W. S., and Choi, J. R., 'Implementation of 1024-bit processor for RSA cryptosystem'. http://www.ap-asic.org/2000/proceedings/10-4.pdf.
9. Bunimov, V., Schimmler, M., and Tolg, B., 'A Complexity-Effective Version of Montgomery's Algorithm'. Presented at the Workshop on Complexity Effective Designs (WECD02), May 2002.
10. Gustavo D. Sutter, Jean-Pierre and José Luis "Modular Multiplication and Exponentiation Architectures for Fast RSA Cryptosystem Based on Digit Serial Computation", IEEE TRANSACTIONS ON INDUSTRIAL LECTRONICS, Vol. 58, No. 7, JULY 2011.
11. Jin-Hua Hong and Cheng-Wen Wu, "Cellular-Array Modular Multiplier for Fast RSA Public-Key Cryptosystem Based on Modified Booth's Algorithm", IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION SYSTEMS, Vol. 11, No. 3, JUNE 2003.
12. McIvor, M. McLoone and J. V. McCanny, "Modified Montgomery modular multiplication and RSA exponentiation techniques", IEE Proceedings online No. 20040791 DOI: 10.1049/ip-cdt: 20040791.
13. Alexandre F. Tenca and C¸ etin K. Koc, "A Scalable Architecture for Modular Multiplication Based on Montgomery's Algorithm": IEEE TRANSACTIONS ON COMPUTERS, Vol. 52, No. 9, SEPTEMBER 2003.
14. Marcelo E. Kaihara and NaofumiTakagi, "A Hardware Algorithm for Modular Multiplication/ Division", IEEE TRANSACTIONS ON COMPUTERS, Vol. 54, No. 1, JANUARY 2005.
15. Koç, C. K., Acar, Tolga, and Kaliski, B. S., "Analyzing and comparing Montgomery multiplication algorithms", Micro, IEEE Publication: Jun 1996 Page(s): 26 - 33 ISSN: 0272-1732 INSPEC Accession Number: 5298231 Vol. 16, Issue: 3.
16. Ching-Chao Yang, Tian-Sheuan Chang, and Chein-Wei Jen, "A New RSA Cryptosystem Hardware Design Based on Montgomery's Algorithm", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS—II: ANALOG AND DIGITAL SIGNAL PROCESSING, Vol. 45, and No. 7, JULY 1998.
17. Zhimin Chen and Patrick Schaumont, "A Parallel Implementation of Montgomery Multiplication on Multicore Systems: Algorithm, Analysis, and Prototype", IEEE TRANSACTIONS ON COMPUTERS, Vol. 60, No. 12, DECEMBER 2011.
18. GuilhermePerin, Daniel Gomes Mesquita, and JoãoBaptista Martins, "Montgomery Modular Multiplication on Reconfigurable Hardware: Systolic versus Multiplexed Implementation", International Journal of Reconfigurable Computing Volume 2011 (2011), Article ID 127147, 10 pages DOI: 10.1155/2011/127147.
19. P. Fournaris and O. Koufopavlou, "A new RSA encryption architecture and hardware implemen-




© 2013 Global Journals Inc. (US)

tation based on optimized Montgomery multiplication," in Proc. IEEE ISCAS, May 23–26, 2005, pp. 4645–4648.
20. P. L. Montgomery, "Modular Multiplication without Trial Division", Math. of Computation, Vol. 44, No. 170, pp. 519-521, Apr., 1985.
21. L. Batina and G. Muurling, "Montgomery in Practice: How to Do It More Efficiently in Hardware," Proc. Cryptographer's Track at the RSA Conf., Topics in Cryptology (CT-RSA '02), pp. 40-52, Feb. 2002.
22. D. Walter, "Precise Bounds for Montgomery ModularMultiplication and Some Potentially Insecure RSA Moduli,"Proc. Cryptographer's Track at the RSA Conf. Topics in Cryptology (CT-RSA '02), pp. 30-39, Feb. 2002.
23. Perovic, N. S., Popovic-Bozovic, M., "FPGA implementation of RSA cryptoalgorithm using shift and carry algorithm"; 20th Telecommunications Forum (TELFOR), 2012 on 20-22 Nov. 2012, Page(s): 1040 – 1043.



© 2013 Global Journals Inc. (US)