International Journal of Advanced Technology in Engineering and Science
Vol. No.4, Issue No. 01, January 2016
www.ijates.com
ijates
ISSN 2348 - 7550

IMPLEMENTATION AND PERFORMANCE EVALUATION OF MODULO AND MONTGOMERY MULTIPLIERS IN REAL TIME APPLICATION

Mr.M. Rajkumar[1], Ms.C.S.Anita[2], Ms.Pacha Shobarani[3], Ms. D.Rajalalshmi[4]

[1, 2, 3]Associate Professor, [4]Asst.Professor, [4]Asst.Professor,
R.M.D. Engineering College, Chennai, (India)

ABSTRACT

In digital life there is a growing demand for real time implementation of cryptographic algorithms which are being used in secure communication systems, networks and security systems. In this research paper a novel reconfigurable processor architecture has been presented for cryptographic applications that bridges the gap of traditional computing techniques and also sustains implementations that can show equal or even better performance results than custom hardware. This work presents an emerging reconfigurable hardware that potentially delivers flexible high performance for cryptographic algorithms. A cryptographic processor with public and private key pair generator and modulo multiplier based both encryptor and decryptor for text message was designed. The hardware design of RSA processor by employing Montgomery multiplier technique to reduce the execution delay time and the thermal power dissipation was compared and proved.

Keywords: Modulo Multiplication, Encryption, Decryption, RSA Cryptography.

I. INTRODUCTION

Digital signal processors and cryptographic cores are strategically implemented in Residue Number System (RNS) . For example, the RNS based implementation of a simplistic 16-tap transpose filter exhibited 25% reduction in delay compared to a Two's Complement System (TCS) based implementation . It was reported that the FPGA implementation of RNS adaptive filter was 65% faster than that of TCS filter. The RNS implementation of a 192-bit elliptic curve point multiplier operating at a maximum clock frequency of 53 MHz was found to exceed the performance of the contemporary TCS implementation by 33%.

Techniques such as multi-modulus and multi-function architectures to minimize the hardware redundancy as well as multi-threshold voltage and multi-supply voltage designs to lower the power dissipation have been suggested. Such control techniques are intended for algorithm level design space exploration and are applicable to generic modulo arithmetic architectures. For architecture level simplification of specific modulo arithmetic operations like modulo multiplication, techniques that explore unique number theoretic properties of special moduli of the forms 2^n and $2^n \pm 1$ have received wide spread attention amongst others. While modulo 2^n

multiplier can be simplified to a two's complement multiplier with the output truncated to n-bits, modulo $2^n - 1$ and modulo $2^n + 1$ multipliers need additional circuitries to modulo-reduce the product.

II. CRYPTOGRAPHY

Data security and cryptography are critical aspects of conventional computing. Here we provide basic terminology used in cryptography. Cryptography is the most important aspect of communications security. In data and telecommunication, cryptography is necessary when communicating over any untrusted medium, which just about any network, particularly the Internet. The goal is to transmit a message between a sender and receiver such that an eavesdropper is unable to understand it. Plaintext refers to a sequence of characters drawn from a finite alphabet, such as that of a natural language.

Encryption is the process of scrambling the plaintext using a known algorithm and a secret key. The output is a sequence of characters known as the cipher text. Decryption is the reverse process, which transforms the encrypted message back to the original form using a key.

The goal of encryption is to prevent decryption by an adversary who does not know the secret key. An unbreakable cryptosystem is one for which successful cryptanalysis is not possible. Such a system is the one-time-pad cipher. It gets its name from the fact that the sender and receiver each possess identical notepads filled with random data. Each piece of data is used once to encrypt a message by the sender and to decrypt it by the receiver, after which it is destroyed.

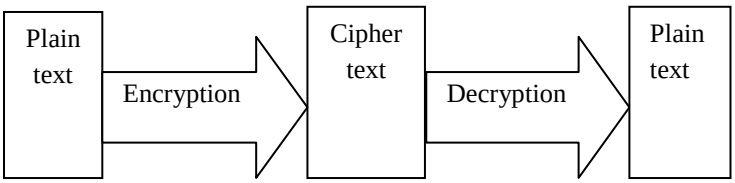

Fig 1: General diagram of cryptography

2.1 Private Key Cryptography

In private-key cryptography, the sender and recipient agree beforehand on a secret private key. The plaintext is somehow combined with the key to create the cipher text. The method of combination is such that, it is hoped, an adversary could not determine the meaning of the message without decrypting the message, for which he needs the key.

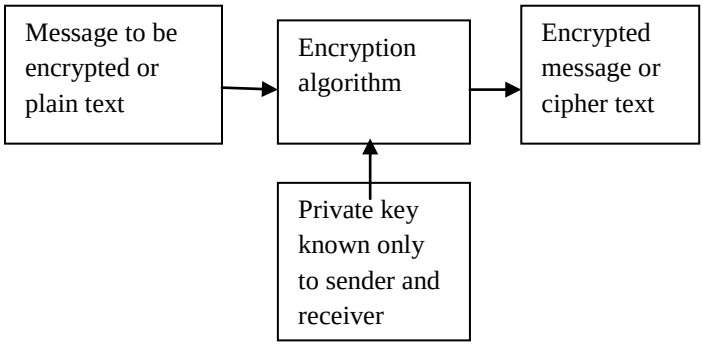

Fig 2 : Encryption process in Secret Key Cryptography

To break a message encrypted with private-key cryptography, an adversary must either exploit a weakness in the encryption algorithm itself, or else try an *exhaustive BLAKErch* of all possible keys (brute force method). If the key is large enough (*e.g.*, 128 bits), such a BLAKErch would take a very long time (few years), even with very powerful computers.

Private-key methods are efficient and difficult to break. However, one major drawback is that the key must be exchanged between the sender and recipient beforehand, raising the issue of how to protect the secrecy of the key. When the President of the United States exchanges launch codes with a nuclear weapons site under his command, the key is accompanied by a team of armed couriers. Banks likewise use high security in transferring their keys between branches. These types of key exchanges are not practical, however, for e-commerce between, say, amazon.com and a casual web surfer.

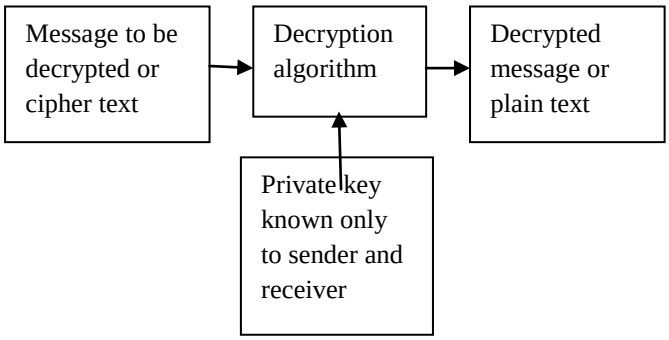

Fig 3: Decryption Process in Secret Key Cryptography

The main problem with secret-key cryptosystems is getting the sender and receiver to agree on the secret key without anyone else finding out. This requires a method by which the two parties can communicate without fear of eavesdropping. However, the advantage of secret-key cryptography is that it is generally faster than public-key cryptography.

2.2 Public Key Cryptography

Public Key cryptography uses two keys Private key (known only by the recipient) and a Public key (known to everybody). The public key is used to encrypt the message and then it is sent to the recipient who can decrypt the message using the private key.

The message encrypted with the public key cannot be decrypted with any other key except for its corresponding private key. The following Diagram illustrates the encryption process in the public key cryptography.

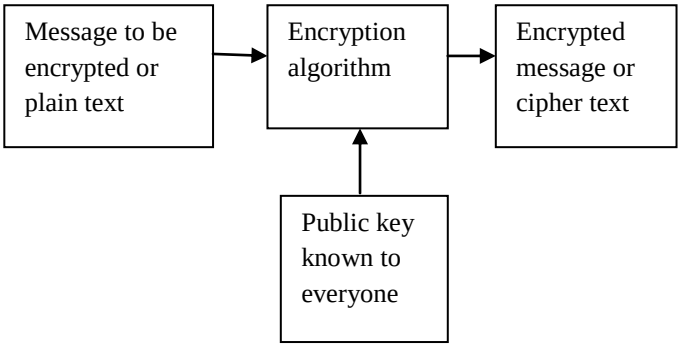

Fig 4: Encryption Process in the public key cryptography

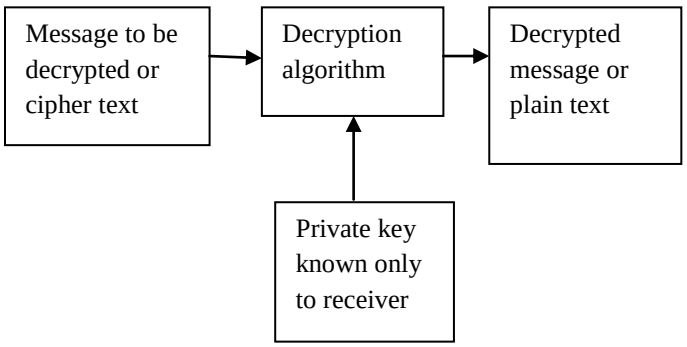

Fig 5: Decryption Process in the private key cryptography

The public-key algorithm uses a one-way function to translate plain text to cipher text. Then, without the private key, it is very difficult for anyone (including the sender) to reverse the process (i.e., translate the cipher text back to plaintext). A one-way function is a function that is easy to apply, but extremely difficult to invert. The most common one-way function used in public-key cryptography involves factoring very large numbers. The idea is that it is relatively easy to multiply numbers, even large ones, with a computer; however, it is very difficult to factor large numbers. The only known algorithms basically have to do a sort of exhaustive BLAKErch (Does 2 go in to? Does 3? 4? 5? 6? and so on). With numbers 128 bits long, such a BLAKErch requires performing as many tests as there are particles in the universe.

In a public-key cryptosystem, the private key is always linked mathematically to the public key. Therefore, it is always possible to attack a public-key system by deriving the private key from the public key. Typically, the defence against this is to make the problem of deriving the private key from the public key as difficult as possible. For instance, some public-key cryptosystems are designed such that deriving the private key from the public key requires the attacker to factor a large number, it this case it is computationally infeasible to perform the derivation. This is the idea behind the RSA public-key cryptosystem. Public-key cryptography is a cryptographic approach which involves the use of asymmetric key algorithms instead of or in addition to symmetric key algorithms. Unlike symmetric key algorithms, it does not require a secure initial exchange of one or more secret keys to both sender and receiver. The asymmetric key algorithms are used to create a mathematically related key pair: a secret private key and a published public key. Use of these keys allows protection of the authenticity of a message by creating a digital signature of a message using the private key, which can be verified using the public key. It also allows protection of the confidentiality and integrity of a message, by public key encryption, encrypting the message using the public key, which can only be decrypted using the private key. The most widely used public key cryptosystem is RSA. The RSA scheme is a block cipher in which the plaintext and cipher text are integers between 0 and n-1 for some n.

III. RSA ALGORITHM

It is based on a very simple number-theoretical idea, and yet it has been able to resist all cryptanalytic attacks. The idea is a clever use of the fact that, while it is easy to multiply two large primes, it is extremely difficult to factorize their product.

Thus, the product can be publicized and used as the encryption key. The primes themselves cannot be recovered from the product and are used for decryption. There is no formal proof whatsoever that factorization is

intractable or is intractable in the special case needed for RSA, and that factorization is needed for the cryptanalysis of the RSA.

RSA algorithm simply capitalizes on the fact that there is no efficient way to factor very large integers. The security of the whole algorithm relies on that fact. If someone comes up with an easy way of factoring a large number, then that's the end of the RSA algorithm. Then any message encrypted with the RSA algorithm is no more secure.

Briefly, the algorithm involves multiplying two large prime numbers (a prime number is a number divisible only by that number and 1) and through additional operations deriving a set of two numbers that constitutes the public key and another set that is the private key. Once the keys have been developed, the original prime numbers are no longer important and can be discarded. Both the public and the private keys are needed for encryption /decryption but only the owner of a private key ever needs to know it. Using the RSA system, the private key never needs to be sent across the Internet.

The private key is used to decrypt text that has been encrypted with the public key. Thus, while a sending a message, it is easy to find out the public key (but not the private key) from a central administrator and message is encrypted using public key. While receiving it, the message is decrypted with the private key.

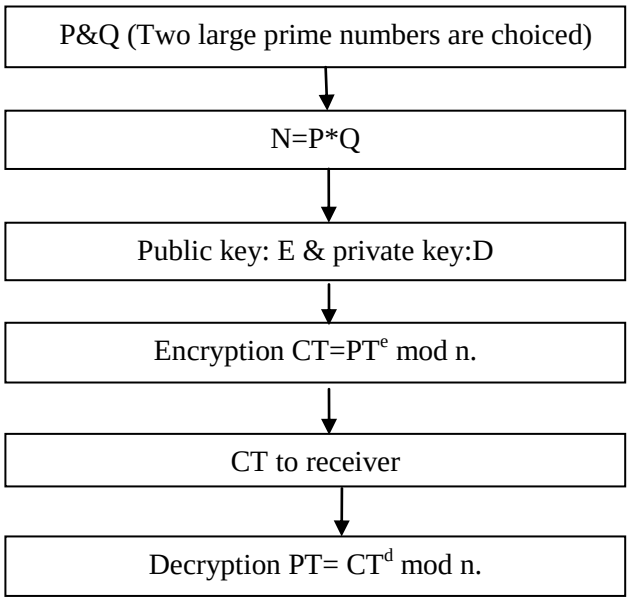

Fig 6: RSA algorithm

3.1 Features of Rsa

It is the easiest to understand as well as the most popular to implement RSA obtains its security from the difficulty of factoring large numbers. The algorithm is patented in North America (although algorithms cannot be patented elsewhere in the world) this is a source of legal difficulties in using the scheme.

3.2 Rsa Security

The security of RSA algorithm depends on the ability of the hacker to factorise numbers. Newer faster and better methods for factoring numbers are constantly being devised. The current best for long numbers of the number field sieve. Prime number of a length that was unimaginable a mere decade ago are now factored easily.

Obviously the larger the number is, the harder it is to fact and so the better the security of RSA.As theory and computers improve large and larger keys will have to be used.

IV. KEY GENERATION ALGORITHM

1. Generate two large random primes, *p* and *q*, of approximately equal size such that their product n = pq is of the required bit length, e.g. 1024 bits.
2. Compute n = pq and (φ) phi = (p-1)(q-1).
3. Choose an integer *e*, 1 < e < phi, such that gcd (e, phi) = 1.
4. Compute the secret exponent *d*, 1 < d < phi, such that ed ≡ 1 (mod phi).
5. The public key is (n, e) and the private key is (n, d). Keep all the values d, p, q and phi secret.

- n is known as the *modulus.*
- e is known as the *public exponent* or *encryption exponent* or just the *exponent.*
- d is known as the *secret exponent* or *decryption exponent.*

A. ENCRYPTION

Sender A does the following:-

1. Obtains the recipient B's public key (n, e).
2. Represents the plaintext message as a positive integer *m*
3. Computes the cipher text $c = m^e \bmod n$.
4. Sends the cipher text *c* to B.

B. DECRYPTION

Recipient B does the following:-

1. Uses his private key (n, d) to compute $m = c^d \bmod n$.
2. Extracts the plaintext from the message representative *m.*

C. DIGITAL SIGNING

Sender A does the following:-

1. Represents this digest as an integer *m* between 0 and *n*-1..
2. Uses her private *key* (n, d) to compute the signature $s = m^d \bmod n$.
3. Creates a *message digest* of the information to be sent.
4. Sends this signature *s* to the recipient, B.

D. SIGNATURE VERIFICATION

Recipient B does the following:-

1. Uses sender A's public key (n, e) to compute integer $v = s^e \bmod n$.
2. Extracts the message digest from this integer.
3. Independently computes the message digest of the information that has been signed.
4. If both message digests are identical, the signature is valid.

E. KEY LENGTH

On considering the key length of an RSA key, with reference to the length of the modulus, n in bits. The minimum recommended key length for a secure RSA transmission is currently 1024 bits. A key length of 512

bits is now no longer considered secure, although cracking it is still not a trivial task for the likes of the user. The longer the information is needed to be kept secure, the longer the key one should use.

F. STEPS IN RSA

- n = pq, where p and q are distinct primes.
- phi, φ = (p-1)(q-1)
- e < n such that gcd(e, phi)=1
- $d = e^{-1}$ mod phi.
- $c = m^e$ mod n, 1<m<n.
- $m = c^d$ mod n.

V. MONTGOMERY MULTIPLIER

For modular multiplication Montgomery's technique is chosen. Montgomery multiplication is defined as follows:

$$MONT(X,Y) = XYR^{-1} \mod N$$

For a word base b = 2°, R should be chosen such that R = 2r = (2°)' > N. There is a one-to-one correspondence between each element x G ZN and its Montgomery representation xR mod N. This Montgomery representation allows very efficient modular arithmetic especially for multiplication. Montgomery's method for multiplying two integers x and y (called N-residues) modulo N, avoids division by N which is the most expensive operation in hardware. The method requires conversion of x and y to an N-residue domain and conversion of the calculation result back to ZN. The procedure is as follows. To compute Z = xy mod N, one first has to compute the Montgomery multiplication of x and R2 mod N to get Z' = xR mod N. Mont(Z', y) gives the desired result. When computing the Montgomery product T = Mont(x, y) = xyR-1 mod N, the procedure shown is performed.

In the original notation of Montgomery after each multiplication a reduction was needed. The input had the restriction X, Y < N and the output T was bounded by T < 2N. As a consequence, if T > N , N must be subtracted so that the output can be used as input of the next multiplication. To avoid this subtraction a bound for R is known such that for inputs X,Y < 2N the output is also bounded by T < 2N.

In the need of avoiding reduction after each multiplication is addressed. In practice this means that the output of the multiplication can be directly used as an input of the next Montgomery multiplication. We want to find a bound on R such that with X,Y < 2N the output of the Montgomery multiplication T < 2N. Write R > kN, then:

$$T = \frac{XY + mN}{R} = \frac{XY}{R} + \frac{m}{R}N < \frac{4}{k}N + N$$

where, **m = (XY** mod **R)N'** mod **R** [1].

Hence, **T < 2N** for **k** > 4, implying: 4N < R. We will use 4N < **R** = 2'+2, by taking **a** = 1 for simplicity and making the iteration starting from Step 2 execute **I** + **2** times. As the result of the decision for a, $-\mathbf{N}^{-i}$ mod 2a can be written as $(2 - n_0)$ **- 1** mod 2. Because **N** is odd for RSA and an odd prime for ECC, $n_0 = 1$ which results to **N'** = 1. We will use the Algorithm 2 for MMM which includes these improvements.

Example:

Montgomery Arithmetic:

Several modular multiplications are involved in RSA. The encryption formula is $y=b^{\wedge a} \bmod m$

The multiplication come modular operation is done at each single level and the immediate results are stored in temporary registers and the same steps are repeated till the exponent or index value exceeds. For example consider a number 2 with exponent value 5 and the final result is stored in the output register y.

E.g. 2^5 mod 3

A= (2x2) mod 3

A= (Ax2) mod 3

A= (Ax2) mod 3

A= (Ax2) mod 3

Output Y= A mod n the simulation output of implementing Montgomery multiplication for the example is obtained as below. By viewing the simulation output, timing analyzer, power analyzer and flow summary tool in Quartus II software. The area, power and time have been improved.

VI. SIMULATION RESULTS

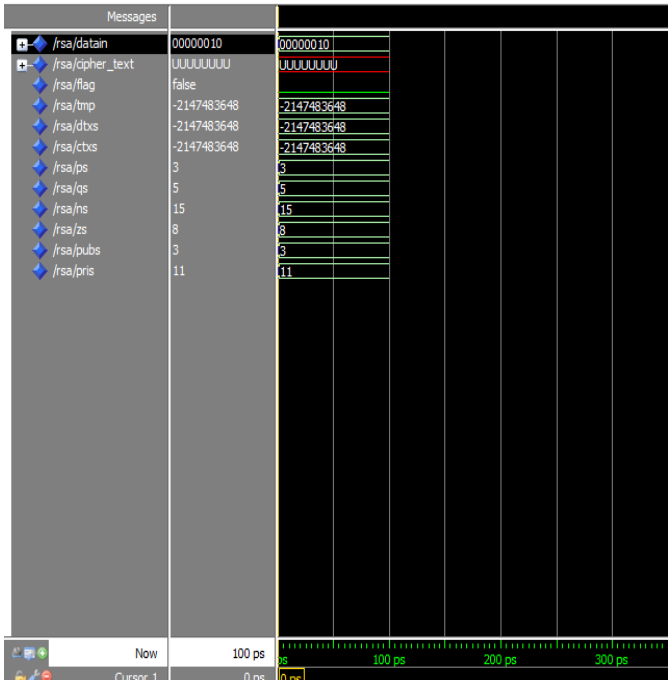

Fig 7 Simulated Result of Key Generation

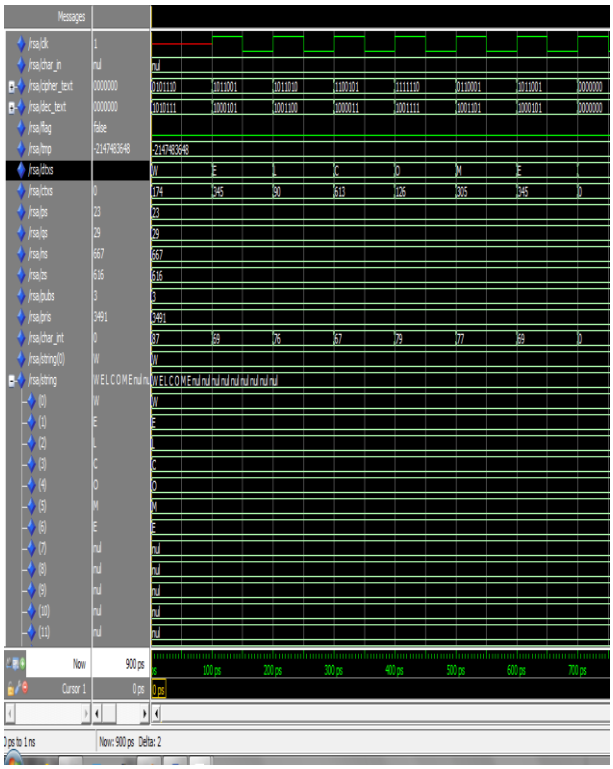

Fig 8 Simulated Result of Encryption and Decryption

VII. COMPARISON CHART OF MODULO AND MONTGOMERY MULTIPLIERS

PARAMETERS	MODULO MULTIPLIER	MONTGOMERY MULTIPLIER
TOTAL NO. OF REGISTERS USED	64	20
TOTAL POWER DISSIPATION	37.70mw	35.11mw
DELAY TIME	150.692ns	7.186ns

VIII. CONCLUSION

Area-power efficient modulo and Montgomery multipliers employing RSA algorithm were proposed. RSA uses a variable size encryption block and a variable size key. The key-pair is derived from a very large number, *n*, that is the product of two prime numbers chosen according to special rules; these primes may be 100 or more

digits in length each, yielding an *n* with roughly twice as many digits as the prime factors. A cryptographic processor with public and private key pair generator and modulo multiplier based both encryptor and decryptor for text message was designed. The hardware design of RSA processor by employing Montgomery multiplier technique to reduce the execution delay time and the thermal power dissipation was also designed and the comparison results were shown.

REFERENCES

[1]. Cristophe bhode 'Introduction to reconfigurable computing' architectures algorithms and applications springer.

[2]. Diffie W and HellmanM. E. (1976) 'New directions in cryptography,' IEEETrans. Inf. Theory, vol. IT-22, no. 6, pp. 644–654.

[3]. Eslami Y, Sheikholeslami A, GulakP. G.,Masui S, and Mukaida K(2006).'An area-efficient universal cryptography processor for smart cards,'IEEE Trans. Very Large Scale Integr. (VLSI) Syst., vol. 14, no. 1, pp.43–56.

[4]. Goodman J and A. P. Chandrakasan, (2001) 'An energy-efficient reconfigurablepublic-key cryptography processor,' in the proceedings of IEEE J. Solid-State Circuits,vol. 36, no. 11, pp. 1808–1820.

[5]. HoWon Kim, Member, IEEE, and Sunggu Lee,(2004).'Design and Implementation of a Private and Public Key Crypto Processor and Its Application to a Security System' in the proceedings of IEEE Transactions on Consumer Electronics, Vol. 50, No. 1.

[6]. Jun-Hong Chen, Ming-Der Shieh, Member, IEEE, and Wen-Ching Lin(2010).'A High-Performance Unified-Field Reconfigurable Cryptographic Processor' in the proceedingsof IEEE transactions on very large scale integration (vlsi) systems, vol. 18, no. 8.

[7] Jan Zutter, Max Thalmaier, Karsten-Olaf Laux Wipotec (2009).'Acceleration of RSA Cryptographic Operations using FPGA Technology', in the proceedings of GmbH 20th International Workshop on Database and Expert Systems Application IEEE.

[8]. C. E. Leiserson and J. B. Saxe,(1991). 'Retiming synchronous circuitry,' in the proceedings of Algorirhmica, vol. 6, pp. 5-35.

[9]. Neil Smyth, Máire McLoone and John V McCanny (2005).'Reconfigurable Processor for Public-Key Cryptography' in the proceedings of IEEE.

[10]. Nibouche', M. Nibouche', and A.Bouridane(2004) 'High speed FPGA implementation of RSA encryption algorithm', in the proceedings of IEEE.

[11] .Omar Nihouche, Mokhtar Nibouche, Ahmed Bouridane, and Ammar Belatreche, (2004). 'Fast Architectures For FPGA-Based Implementation of RSA EncryptionAlgorithm', in the proceedings of IEEE.

[12]. Qiang Liu' Fangzhen Dong Tong' Xu Cheng.(2004).'A Regular Parallel RSA Processor' in the proceedings of the 47rh IEEE International MidwestSymposium on Circuits and Systems in the proceedings of IEEE.

[13]. K. Sakiyama, L. Batina, B. Preneel, and I. Verbauwhede,(2007) 'Multicorecurve-based cryptoprocessor with reconfigurable modular arithmeticlogic units' in the proceedings ofIEEE Trans Comput., vol. 56, no. 9, pp.1269–1282.

[14]. Z.Shi, R.B. Lee (2000) 'Bit Permutation Instructions forAcceZerading Software Cryptography', in the Proceedings. of the IEEE IntlConference on Application-specific Systems, Architecturesand Processors, Boston, US, pp. 138-144.

[15]. M. C. Sun, C. P. Su, C. T. Huang, and C. W.Wu,(2003) 'Design of a scalableRSA and ECC crypto-processor,' in Proc. IEEE Asia South Pacific DesignAutom. Conf., pp. 495–498.

[16] .N. Smyth, M. McLoone, and J. V. McCanny,(2006) 'An adaptable and scalableasymmetric cryptographic processor,' in Proc. IEEE Int. Conf.Appl.-Specific Syst. Arch. Processors, pp. 341–346.

[17]. Y. Wang, J. Leiwo, and T. Srikanthan,(2005) 'A unified architecture forcrypto-processing in embedded systems' in Proc. IEEE Conf. EmbeddedSoftw. Syst., pp. 1–7.

[18]. Yang Qian, Wu Xingiun, Zhou Runde, Lu Ruibing (2008)'An Embedded RSA processor for encryption and Decryptio' in the proceedings of IEEE., 1-9.

[19]. X. Zeng, C. Chen, and Q. Zhang,(2004) 'A reconfigurable public-key cryptography coprocessor' in Proc. IEEE Asia Pacific Conf. Adv. Syst. Integr.Circuits, pp. 172–175.

[`20]. Zhu Kejia , Xu Ke, Wang Yang, and Min Hao (2009) 'A Novel ASIC Implementation of RSA Algorithm' ,in the proceedings of IEEE.

[`21]. Ramya Muralidharan and Chip-Hong Chang (2012) 'Area-Power Efficient Modulo 2^n - 1 and Modulo 2^n + 1 Multipliers for $\{2^n - 1, 2^n, 2^n + 1\}$ Based RNS' ,in the proceedings of IEEE.