Efficient Montgomery Modular Multiplication by using Residue Number System

Elham Khani[1], Mohammad Esmaeildoust[2], Keivan Navi[3]
[1]Deptt.of Computer Engineering, Science and Research Branch, Islamic Azad University,Tehran, Iran
[2]Faculty of Marine Engineering, Khoramshahr Marine Science and Technology University, Iran
[3]Faculty of Electrical and Computer Engineering, Shahid Beheshti University, GC Tehran, Iran

ABSTRACT

By applying Residue Number System (RNS) to Montgomery modular multiplication the delay of modular multiplication will be decreased. Modular multiplication over large number is frequently used in some application such as Elliptic Curve Cryptography. By choosing appropriate RNS moduli sets the time consuming operation of multiplication can be replaced by smaller operations. In addition because of the property of RNS, arithmetic operations are done over smaller numbers called residues. In this paper by choosing appropriate moduli sets the efficiency of conversion from RNS to RNS that is the most time consuming part of the Montgomery modular multiplication will be increased.

General Terms Algorithm.

Keywords Residue Number System, Montgomery Multiplication, Modular Multiplication, Moduli set.

1. INTRODUCTION

Residue number system is a carry free system that performs arithmetic operation on residues instead of the weighted binary number. This property leads to fast operation over smaller numbers. RNS has been used in some applications such as public key cryptography [1], digital signal processing [2], and digital image processing [3] which requires fast arithmetic unit and less power consumption. Forward converter, reverse converter and arithmetic unit are the components of RNS. Appropriate moduli selection, dependent to the application, plays an important role in efficiency of RNS system. The most prominent moduli set that have been introduced is $\{2^k-1, 2^k, 2^k+1\}$ [4]. Efficient Reverse converter for this moduli set has been proposed but dynamic range of this moduli set for modern applications that require high dynamic range was not enough. So moduli sets with higher dynamic rage like [5] and [6] were proposed. Modular multiplication is the main part of application like ECC [7] and RSA[1]. Montgomery modular multiplication needs two moduli set that called bases. These two bases determine the architecture and delay of the RNS Montgomery modular multiplication. Several moduli set for RNS Montgomery multiplication have been introduced till now, in [8] moduli in the form of and are selected for first and second basis, respectively. Although the arithmetic operation in modulo $2^k\pm1$ is simple, but due to unbalanced moduli, the efficiency of arithmetic unit is inapplicable. In [9] RNS bases in the form of 2^k-ci were $0 \leq ci \leq 2^k/2$ were proposed and exhaustive search to find moduli with small Hamming weight was done. The advantage of this work is small Hamming weight of moduli set that leads to simple multiplicative inverses and finally modular multiplication can be replace by some shift and addition. In this paper, in order to improve the efficiency of RNS Montgomery multiplication, the second basis in [9] are replaced by moduli set $\{2^{2k+1}-1, 2^{k/2}-1, 2^{k/2}+1, 2^k+1, 2^k\}$ [10] and the required conversion between basis are designed.

This paper organized as follows: section 2 includes a brief background about RNS and Montgomery modular multiplication. In section 3, RNS bases are discussed. Conversion of RNS to RNS between bases is detailed in section 4. In section 5, the proposed reduction methods are proposed. Section 6 includes the comparison with state-of-the-art works and finally section 7 concludes the paper.

2. BACHGROUND MATERIAL

2.1 Residue Number System

A residue number system is introduced by a set of integer number $\{P_1, P_2, \ldots, P_n\}$ that are relatively prime. In residue representation each number X is shown by a set of n small integer $(x_1, x_2, \ldots, x_n)$ for $i = 1$ to n; where $x_i = (X \bmod P_i)$. Dynamic range M is multiplication of moduli P_i ($M = P_1 \times P_2 \times \ldots \times P_n$). There are some algorithms for converting residue numbers into their binary equivalent, one of them is Mix-Radix Conversion (MRC). In MRC method, a binary number X is obtained from its residue by the below formula:

$$X = v_n \prod_{i=1}^{n-1} P_i + \ldots + v_3 P_2 P_1 + v_2 P_1 + v_1 \qquad (1)$$

Where $v_1 = x_1$ (2)

$$v_2 = \left| (x_2 - v_1) \left| P_1^{-1} \right|_{P_2} \right|_{P_2} \qquad (3)$$

$$v_3 = \left| ((x_3 - v_1) \left| P_1^{-1} \right|_{P_3} - v_2) \left| P_2^{-1} \right|_{P_3} \right|_{P_3} \qquad (4)$$

In general

$$v_n = \left| (((x_n - v_1) \left| P_1^{-1} \right|_{P_n} - v_2) \left| P_2^{-1} \right|_{P_n} - \cdots - v_{n-1}) \left| P_{n-1}^{-1} \right|_{P_n} \right|_{P_n} \qquad (5)$$

$\left| P_i^{-1} \right|_{P_j}$ Denotes the multiplicative inverse of P_i in modulus P_j.

2.2 MRS Representation

In MRC algorithm, the intermediate number vi is MRS representation and $(v_1, v_2, \ldots, v_n)$ is the MRS representation of X where v_i is obtained by Eq. (2-5). According to Honor's scheme Eq.(1) is converted to:

$X = v_1 + P_1 (v_2 + P_2 (v_3 + \ldots + P_{n-1} v_n) \ldots)$ (6)

2.3 Montgomery Modular Multiplication

One of the most efficient algorithms for modular multiplication is Montgomery algorithm that was reported in [11]. By using Montgomery multiplication, modular multiplication can be done without any division. Adopting Montgomery algorithm to RNS leads to benefit from both Montgomery method and properties of RNS. The RNS version of Montgomery modular multiplication is as follow:

Suppose X and Y are two large numbers that are multiplied to each other in modulo N. For implementing Montgomery multiplication in RNS two moduli set are needed (two basis). let $s_1=\{P_1, P_2, \ldots, P_n\}$ with dynamic range M_1 be the first base and $s_2=\{P'_1, P'_2, \ldots, P'_n\}$ with dynamic range M_2 second base where $N < M_1 < M_2$ and $\gcd(M_1, M_2) = \gcd(M_1, N) = \gcd(M_2, N) = 1$.

X and Y have the RNS representation $(x_1, x_2, \ldots, x_n)$, $(y_1, y_2, \ldots, y_n)$ and $(x'_1, x'_2, \ldots, x'_n)$, $(y'_1, y'_2, \ldots, y'_n)$ in first and second basis respectively. Modular multiplication computes $X \times Y \times M^{-1} \bmod N$ according to the following algorithm:

```
Algorithm 1: Montgomery Modular Multiplication in RNS

Step 1: Let $D = X \times Y$ in both basis(
        $d_i = |x_i \times y_i|_{P_i}$ in first base and
        $d'_i = |x'_i \times y'_i|_{P'_i}$ in second base )
Step 2: Calculation of $Q$ as $Q = \left| D \times |N|_{P_i}^{-1} \right|_{M_1}$
        in first base ($q_i = \left| d_i \times |N|_{P_i}^{-1} \right|_{P_i}$)
Step 3: Converting the representation of $Q$ from first
        to second basis(base extension)
Step 4: Calculation of
        $R = (D - Q \times N) \times |M_1|_{M_2}^{-1}$ in second basis
        where $r'_i = \left| (d'_i - q'_i \times n'_i) \times |M_1|_{P'_i}^{-1} \right|_{P'_i}$
Step 5: Converting the representation of $R$ from
        second to first basis(base extension)
```

In algorithm 1, first step consists of two modular multiplications in both basis, second step has one RNS product in first basis, in step 3 and 5 of algorithm 1, two conversions from first basis to second basis and second base to first base are done. In step 4, two RNS product and one addition in second basis is comprised. We can divide the arithmetic operation in two sections, one section is consisting of operations in three steps 1, 2 and 4 and another section contains base extension or RNS to RNS conversion. Selection of the moduli with appropriate arithmetic unit play an important role in efficiency of modular multiplication, on the other hand base extensions improvement due to the complicity in term of delay and hardware cost is important issue in Mont. Modular multiplication. By reducing the delay of third and fifth step overall delay of multiplication will be improved.

3. OPTIMAL RNS BASIS FOR MODULAR MULTIPLICATION

Efficiency of modular multiplication by RNS Montgomery algorithm is dramatically dependent to moduli selection. As mentioned in section 2, for RNS Montgomery modular multiplication two bases are needed. The efficiency of arithmetic operation and revere converters of these two bases determine the overall performance of the modular multiplication. In first base, moduli are selected in the form of 2^k-c_i where 0< c_i <$2^{k/2}$ [9]. By considering c_i=2^t-1, moduli in the form of 2^k-c_i benefit from small hamming weight that results in simple multiplicative inverses. Based on [9] multiplicative inverses for moduli in the form of P_i=2^k-2^t-1 are simple and this property leads to replacing multiplication by some shifts and additions. To achieve proper moduli set an exhaustive search is done to find the moduli that are pair wise relatively primes to other moduli in first and second basis, in addition the selected moduli must have a less Hamming weight for calculating multiplicative inverses. For cryptographic system with key length of 256 and 320 two moduli set is obtained that are shown in Table 1.

Another base is needed in second part of RNS Montgomery modular multiplication. For these second base five moduli set s_2={2^{2k+1}-1, $2^{k/2}$-1, $2^{k/2}$+1, 2^k+1, 2^k}[10] is employed. The second base has the advantages of efficient arithmetic operation, forward and reverse converter. In [10] the efficient reverse converter for these moduli set is introduced that is used in this work. Dynamic range is equal to 5*k* so for key length 256 and 320, *k* is considered 52 and 64 respectively. As mentioned before the dynamic range of first base must be less than second base, because in second base there is a moduli 2^{2k+1}-1we can consider *k* equal for both base and the condition of M_1<M_2 will be satisfied.

Table 1. Proposed RNS bases

Proposed RNS bases	256 Key length		320 key length	
	First Base	Second base	First Base	Second base
5Moduli RNS bases	2^{52}-2^7-1 2^{52}-2^8-1 2^{52}-2^{16}-1 2^{52}-2^{17}-1 2^{52}-2^{19}-1	2^{105}-1 2^{26}-1 2^{26}+1 2^{52}+1 2^{52}	2^{64}-2^{16}-1 2^{64}-2^{19}-1 2^{64}-2^{28}-1 2^{64}-2^{20}-1 2^{64}-2^{31}-1	2^{129}-1 2^{32}-1 2^{32}+1 2^{64}+1 2^{64}

4. RNS to RNS CONVERSION

In step 3 of the algorithm RNS to RNS conversion from first to second basis and in step 5 conversion from second to first basis are done.

4.1 Conversion from First to Second Basis

As mentioned before in first basis moduli are in the form of 2^k-c_i where 0< c_i <$2^{k/2}$. To achieve small hamming weight c_i is considered as $2^{t_i}-1$ where $0<t_i<k/2$. Conversion from first basis to second basis is done in two steps. In first step numbers in modulo $\left(2^k-2^{t_i}-1\right)$ convert to mixed radix representation, and in next step MRS numbers are reduced to second basis. So the total delay of the conversion from first to second basis is:

$$Delay_{RNS\to RNS}=Delay_{RNS\to MRS}+Delay_{MRS\to RNS}$$

4.1.1 RNS to MRS delay

As proved in [9] the total delay of conversion from RNS to MRS is terms of *k*-bit delay FA is equivalent to:

$$Delay_{RNS\to MRS}=\sum_{i=1}^{n-1}\max_{j=2,n;i<j}(w(P^{-1}{}_{i,j})+2w(c_j)+4) \qquad (7)$$

w(*m*) is the hamming weight of *m* and *n* is the number of moduli. In this paper with five moduli set *n*=5. Table 2 shows the delay of RNS to MRS conversion.

Table 2:Delay of RNS to MRS conversion for *k*=64 and 52

Key length	Delay of RNS to MRS conversion
256	93 k D_{FA}=4836 D_{FA}
320	86 k D_{FA}=5504 D_{FA}

4.1.2 MRS to RNS delay

After calculation of mixed radix number conversion from MRS to RNS will be completed. This conversion is done according to Eq.(8)

$$x_i = \left| v_1 + P_1(v_2 + P_2(v_3 + \ldots + P_{n-1}v_n)\ldots) \right|_{P'_i} \qquad (1)$$

In Eq. (8) P_i is moduli in first basis that is in the form of 2^k-2^t-1 and P'_i is moduli in second basis {2^{2k+1}-1, $2^{k/2}$-1, $2^{k/2}$+1, 2^k+1, 2^k}. Eq.(8) is done over all five moduli in parallel so In this step the worst case delay determines the delay of MRS to RNS conversion. Moduli (2^{2k+1}-1) is a critical moduli and reduction in modulo 2^{2k+1}-1 has more delay in compare with other moduli so this moduli defines the delay of MRS to RNS conversion. The reduction circuit for third moduli was reported in [12]. We have:

$$Delay_{MRS \to RNS} = \left(\sum_{i=1}^{n-1} \left(MA(2^{2k+1} - 1) + 2 \right) \right)$$

In terms of delay FA where n- the number of moduli- is 5. By considering the delay of MA[1](2^{2k+1}-1) equal to delay of (2k+1) full adder, the total delay of conversion from MRS to RNS is (8k+12)D_{FA}. So the total delay of conversion first to second basis is equal to (93k+(8k+12)) D_{FA} and (86k+(8k+12)) D_{FA} for key length 256 and 320 respectively. Table 3 shows the delay for two key lengths 256 and 320. The reduction in modulo 2^k+1 is reported in [12] which can be used in this work and reduction in modulo $2^{k/2}$±1 is presented in section 5.

Table 3:total delay of conversion from first to second basis

Key length	Delay of RNS to RNS conversion from first to second basis
256	5268 D_{FA}
320	6028 D_{FA}

[1] Modular Adder

4.2 Conversion from Second to First Basis

4.2.1 Conversion delay of RNS to weighted

Delay of conversion from second to first basis consists of two parts, first conversion from second basis to weighted number.

$$Delay_{RNS \to RNS} = Delay_{RNS \to weighted} + Delay_{wieghted \to RNS}$$

For first part efficient reverse converter was designed in[10] thus

$$Delay_{RNS \to weighted} = (10k + 9)\ D_{FA}$$

Table 4 shows the area and delay of first part.

Table 4: Delay and area of conversion from RNS to weighted

Proposed moduli	Area	Delay
{2^{2k+1}-1, $2^{k/2}$-1, $2^{k/2}$+1, 2^k+1, 2^k}	(20k +k/2+2)A_{FA}+(k-1)A_{XNOR} +(k-1)A_{OR} +(2k+3)A_{XOR} +(2k+3)A_{AND}+(9k/2)A_{NOT}	(10k+9)t_{FA}

4.2.2 Conversion delay of weighted to RNS

Second part in conversion from second to first basis is reduction of a weighted (binary) number to modulo 2^k-c_i , where $c_i = 2^{t_i} - 1$. According to [9] cost of reduction in the moduli in the form of 2^k-c_i is equal to 2w(c_i)+2 in terms of addition of k-bit word where c_i is the Hamming weight of c_i. For converting binary number to second basis a reduction in modulo $(2^k - 2^{t_i} - 1)$ must be done. According to [9] delay of reduction $|a_i+P_i \times y|_{P'j}$ is evaluated with w(c_i)+2w'(c_i)+2 addition of k-bit words. For converting MRS to RNS, Eq. (8) must be done. In [9] the total delay for Eq. (8) is considered as:

$$Delay_{MRS \to RNS} = \sum_{i=1}^{n-1} \max_{j=2,n; i<j} (w(c_i) + 2w(c'_j) + 2)kD_{FA} \qquad (9)$$

A 5k-bit binary number X can be rewritten as X=x_0+2^k(2^k(2^k($2^k x_4$+x_3)+x_2)+x_1) . The reduction of X in modulo $(2^k - 2^{t_i} - 1)$ (second basis) is:

$$x_i = \left| x_0 + 2^k \left(2^k \left(2^k \left(2^k x_4 + x_3 \right) + x_2 \right) + x_1 \right) \right|_{P_i} \qquad (10)$$

Eq.(10) is similar to Eq.(8) but in Eq.(10) Hamming weight of c_i=0, because the $P_i = 2^k - 2^{t_i} - 1$ in Eq.(8) is changed to 2^k in

Eq.(10) $(c_i = 2^{t_i} - 1)$ thus Eq.(9) can be rewritten as below:

$$Delay_{weighted \to RNS} = \sum_{i=1}^{n-1} (2w(c'_j) + 2)kD_{FA} \tag{2}$$

Table 5 shows the total delay of conversion from second to first basis

Table 5: Total Delay of conversion from second to first basis

Key length	Delay of conversion from second to first basis
256	1777 D_{FA}
320	2185 D_{FA}

5. REDUCTION of MRS NUMBERS IN MODULO $2^{k/2}$+1, $2^{k/2}$-1, 2^k

By reduction in modulo 2^k, $2^{k/2}$+1, $2^{k/2}$-1 MRS representation of numbers will convert to RNS representation in second basis. By using Eq.(8) we have

$$x_i = \left| v_1 + P_1\left(v_2 + P_2\left(v_3 + P_3 + \cdots + P_{n-1} v_n\right)\cdots\right) \right|_{P'_i}$$

Where $P_i = 2^k - 2^{t_i} - 1$ and P'_i is moduli $2^{k/2}$-1, $2^{k/2}$+1, 2^k. By replacing these moduli in Eq. (8) it can be rewritten as

$$x_i = \left| v_1 + \left(2^k - 2^{t_1} - 1\right)\left(v_2 + \left(2^k - 2^{t_2} - 1\right)\left(\underbrace{v_3 + \left(2^k - 2^{t_3} - 1\right)v_4}_{I} + \cdots\right)\cdots\right)\right|_{p'_i} \tag{12}$$

Calculation of I is the main process in determining x_i. Realization of delay and architecture of I leads to the obtainment of total delay and architecture of x_i. By considering P'_i=2^k, the reduced number in this moduli is equivalent to k-bit LSB of the number.

5.1 Reduction in Modulo $2^{k/2}$+1

Eq.(12) for P'_i=$2^{k/2}$+1and five moduli is rewritten as

$$x_i = \left| v_1 + \left(2^k - 2^{t_1} - 1\right)\left(v_2 + \left(2^k - 2^{t_2} - 1\right)\left(v_3 + \left(2^k - 2^{t_3} - 1\right)\underbrace{\left(v_4 + \left(2^k - 2^{t_4} - 1\right)v_5\right)}_{I}\right)\right)\right|_{2^{k/2}+1} \tag{3}$$

I is the main operation in Eq.(13) so by considering that $(2^k - 2^{t_i} - 1)$ in modulo $2^{k/2}$+1 is equal to -2^{t_i} , we have

$$I = \left| v_i - 2^{t_i} v_{i+1} \right|_{2^{k/2}+1} = \left| v_i + v_{i+1}\underbrace{00\ldots0}_{t} \right|_{2^{k/2}+1} = \left| v_1 + v_2 - v_3 - v_4 \right|_{2^{k/2}+1} \tag{4}$$

Where

$$v_1 = v_{k/2} \ldots v_{i,0}$$
$$v_2 = 000 v_{i,k-1} \ldots v_{i,k/2+1}$$
$$v_3 = v_{i+1,k/2-t} v_{i+1,0} \underbrace{00\ldots0}_{t}$$
$$v_4 = \underbrace{00\ldots0}_{k/2-t+2} v_{i+1,k/2} \ldots v_{i+1,k/2-t+1}$$

Lemma1: in modulo 2^k+1 the negative of a number is equal to one's complement of the number add to 2

$$\left| -v_i \right|_{2^k+1} = \left| 2^k + 1 - v_i \right|_{2^k+1} = \left| \bar{v}_i + 2 \right|_{2^k+1}$$

So Eq.(14) can be rewritten as:

$$I = \left| v_1 + v_2 + \bar{v}_3 + \bar{v}_4 + 4 \right|_{2^{k/2}+1} \tag{5}$$

As mentioned in section 5.1, I is the fundamental part in calculation of x_i , and for achieving the final result, I must be repeated n times, which n is the number of moduli in first basis. So the total delay of calculation of x_i is

$$Delay_{MRS \to RNS} = \left(\sum_{l=1}^{n-1} I\right) D_{FA} \tag{16}$$

Figure 1 illustrates hardware implementation of *I:*

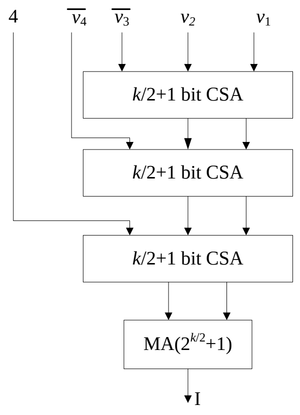


Fig 1:Hardware implementation of *I*

By considering the delay of Carry Save Adder equal to delay of one bit full adder cell, delay of I is equal to MA($2^{k/2}$+1)+3, and the total delay is calculated as:

$$Delay_{MRS \to RNS} = \left(\sum_{l=1}^{n-1} MA(2^{k/2}+1) + 3\right) D_{FA} \tag{6}$$

5.2 Reduction in Modulo $2^{k/2}$-1

Like section 5.1 we have

$$x_i = \left| v_1 + (2^k - 2^t - 1)\left(v_2 + (2^k - 2^t - 1)\left(v_3 + (2^k - 2^t - 1)\underbrace{\left(v_4 + (2^k - 2^t - 1) v_5 \right)}_{J} \right)\right) \right|_{2^{k/2}-1} \tag{7}$$

Same as previous section J is a core for calculation of x_i , by considering that $(2^k - 2^{t_i} - 1)$ in modulo $2^{k/2}$-1 is equal to -2^{t_i} , J can be rewritten as $J = \left| v_i - 2^{t_i} v_{i+1} \right|_{2^{k/2}-1}$ then we have

$$I = \left| v_i - v_{i+1} \underbrace{00\cdots0}_{t_i} \right|_{2^{k/2}-1} = \left| v_1 + v_2 - v_3 - v_4 \right|_{2^{k/2}-1} \tag{8}$$

where

$$v_1 = v_{k/2-1} \ldots v_{i,0}$$

$$v_2 = v_{i,k-1} \ldots v_{i,k/2}$$

$$v_3 = v_{i+1,k/2-t-1} \cdots v_{i+1,0} \underbrace{00\ldots0}_{t}$$

$$v_4 = \underbrace{00\ldots0}_{k/2-t} v_{i+1,k/2} \ldots v_{i+1,k/2-t}$$

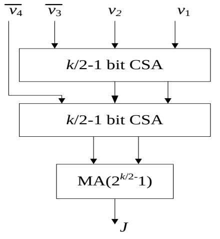
Lemma2: the negative of residue number *a* in modulo 2^k-1 is equal to one's complement of the number *a*.

Using Lemma 2 we have

$$I = \left| v_1 + v_2 + \bar{v}_3 + \bar{v}_4 \right|_{2^{k/2}-1} \tag{20}$$

Figure 2 shows hardware implementation of *j*. Based on Eq.(20) the delay of conversion from MRS to RNS for moduli $2^{k/2}$-1 is equal to

$$Delay_{MRS \to RNS} = \left(\sum_{l=1}^{n-1} MA(2^{k/2} - 1) + 2 \right) D_{FA} \tag{21}$$



2 Fig 3. Hardware implementation of *J*

6. COMPARISON

This work is an effort to speed up conversion from RNS to RNS that is the critical part of a Montgomery modular multiplication. By selecting the appropriate moduli set the delay of the base extension can be reduced but in other side it must be noticed that the efficiency of arithmetic operation must be preserved. So for the second basis the moduli set {2^{2k+1}-1,$2^{k/2}$-1,$2^{k/2}$+1,2^k+1,2^k} was selected, because this moduli set benefit from simple process of reduction (for MRS to RNS conversion) and simple hardware implementation of arithmetic unit. Table 5 shows the improvement for base extension. In this work we have a remarkable improvement in delay of base extension.

Table 5 . Comparison of the delay of base extension for key lenght 256

RNS base	Total delay(in terms of Full-adder delay)	improvement
[9]	11840	40 percent
[13]	8502	17 percent
Proposed	7561	-

7. CONCLUSION

In cryptographic systems like ECC, modular multiplications over large numbers are frequently used. Montgomery modular multiplication in RNS was introduced as an approach to increase the efficiency of modular multiplication which needs two moduli set called bases. In this work, efficient bases which provides suitable arithmetic unit. Moreover this moduli set selection benefit from high dynamic range and speeding up RNS to RNS conversion that is proper for ECC. Comparison with literature shows 40% and 17% improvement in delay of RNS to RNS conversion. Therefore by speeding up the RNS to RNS conversion the total speed of the modular multiplication is increased by proposed RNS bases.

8. REFERENCES

[1] M. Esmaeildoust, D. Schinianakis, H.Javashi, T. Stouraitis, K. Navi, "Efficient RNS Implemenation of Elliptic Curve Point Multiplication Over GF(p)," IEEE Transaction on Very Large Scale Integration(VLSI) Systems, 2012, 1.

[2] G. C. Cardarilli, A. Nannarelli, M. Re, "Residue Number System for Low-Power DSP Applications", Procceding of 41nd IEEE Asilomar Conference on Signals, Systems, and Computers, Nov. 2007, 1412-1416 .

[3] W.Wei, . M.N.S Swamy, M.O Ahmad., "RNS application for digital image processing", *Proceedings* of the 4th IEEE international workshop on system-on-chip for real time applications, Jul. 2004, 77-80.

[4] Y. Wang, X. Song, M. Aboulhamid, and H. Shen," Adder based residue to binary numbers converters for {2^n-1, 2^n, 2^n+1}", IEEE Transactions on Signal Processing, vol. 5, Issue 7, 2002, 1772-1779.

[5] B. Cao, C. H Chang., T. Srikanthan," a residue-to-Binary Converter for a New Five-Moduli Set", IEEE Trans Circuits Syst. I, *Reg. Papers*, Vol. 54, Issue 5, 2007,1041–1048.

[6] A. Hariri, K. Navi, R. Rastegar, "A new high dynamic range moduli set with efficient reverse converter", Journal of Computers & Mathematic Applications, Vol. 55, Feb. 2008, 660-668.

[7] D.M.Schinianakis, A. P. Fournaris, H. E.Michail, A. P. Kakarountas, T. Stouraitis, "An RNS Implementation of an Fp Elliptic Curve Point Multiplier", IEEE Transaction on Circuits and Systems I, , *Reg. Papers*, Vol. 56, Issue 9, 2009, 1202-1213.

[8] K. M. Kalantari, S. P. Mozafari, B. Sadeghiiyan, "Improved RNS for RSA Hardware Implementation," The CSI Journal on Computer Science and engineering, Reg. Papers, Vol. 2, No. 2&4(b), 2004, 31-39.

[9] J. C. Bajard, M. Kaihara, T. Plantard, "Selected RNS Bases for Modular Multiplication", 19th IEEE International Symposium on Computer Arithmetic, 2009, 25-32.

[10] M. Taheri, E. Khani, M. Esmaeildoust, K. Navi, "Efficient Revese Converter Design for Five Moduli set {2^n , 2^{2n+1}-1 , $2^{n/2}$-1 , $2^{n/2}$+1 , 2^n+1 }", Journal of Computational &Modeling, Vol, 2, No. 1, 2012, 93-108.

[11] Montgomery, P. L. "Modular Multiplication Without Trial Division", Mathematics of Computation, Vol. 44, No. 170, Apr. 1985, 519-521.

[12] M. Gerami, M. Esmaeildoust, Sh. Rezaie, K. Navi, O. Hashemipour, "Four Moduli RNS Bases for Efficient Design of Modular Multiplication", Journal of Computations & Modeling, Vol. 1, No. 2, 2011, 73-96.

[13] Sh. Rezaie, M. Esmaeildoust, M. Gerami, K. Navi, O. Hashemipour," High Dynamic Range RNS Bases for Modular Multiplication", International Journal of Computer Science, Vol. 8, No. 1, Issue 4, Jul. 2011, 69-76.