High Dynamic Range RNS Bases for Modular Multiplication

Shirin Rezaie[1], Mohammad Esmaeildoust[2], Marzieh Gerami[1], Keivan Navi[2] and Omid Hashemipour[2]

[1] Department of Computer, Science and Research Branch, Islamic Azad University, Tehran, Iran

[2] Faculty of Electrical and Computer Engineering, Shahid Beheshti University, GC, Tehran, Iran

Abstract

Modular multiplication is the most important part of public key cryptography algorithm like RSA and elliptic curve cryptography. Residue Number System is an efficient way to speed up these applications because of its carry free nature. Efficiency of modular multiplication in RNS is depending on effective selection of RNS bases. In this work efficient design of RNS bases are reported where comparing to the state-of-the-art, the proposed RNS bases has enjoyed more efficient arithmetic operation and residue/binary to binary/residue conversion. Therefore modular multiplication in RNS is implemented with more speed. Comparison with the best work in literature shows that noticeable improvement in speed has achieved by the proposed RNS bases.

Keywords: *Montgomery Modular Multiplication, Residue Number System (RNS), modular arithmetic, Elliptic curve cryptography (ECC).*

1. Introduction

Residue Number System (RNS) has achieved more attention by researcher in recent years for its ability to perform fast arithmetic operation like addition, subtraction and multiplication [1]. Computations in RNS are done without carry propagation between residues and can run concurrently and independently, so it results in speed up and reducing the complexity of different arithmetic components. RNS is an instrumental tool in many applications like image processing, public key cryptography [2-10] and digital signal processing (DSP) [11] which require high speed computations. Modular multiplication is the main part of these applications especially in cryptography algorithms like RSA [2], [3] and elliptic curve cryptography (ECC) [9], [10] which can be implemented in RNS systems.

Moduli selection has important role in efficiency of arithmetic operation, forward and reverse conversion which are the three main parts of RNS system [12-14]. Different moduli sets with efficient reverse converter are proposed by researcher. One of the most well-formed moduli sets is $\{2^n\text{-}1, 2^n, 2^n\text{+}1\}$ which forward converter for these moduli can be done with simple process and the best reverse converter for this moduli set is reported in [15].

For applications like cryptography algorithms, more dynamic ranges are needed. Therefore five moduli sets are presented such as $\{2^n, 2^{2n+1}\text{-}1, 2^{n/2}\text{-}1, 2^{n/2}\text{+}1, 2^n\text{+}1\}$ [14] and $\{2^n, 2^n\text{-}1, 2^n\text{+}1, 2^n\text{-}2^{(n+1)/2}\text{+}1, 2^n\text{+}2^{(n+1)/2}\text{+}1\}$ when n is odd [16]. Modular multiplication is the main part of cryptography algorithms such as RSA [3] and ECC [10]. One of the most known algorithms for the modular multiplication is Montgomery modular multiplication which does not require any division [17]. This algorithm can be implemented in RNS using an auxiliary basis [18]. Proper selection of RNS bases results in increasing the efficiency of modular multiplication. In [19] RNS bases for the first and second basis are proposed in the form of $2^{k_i}-1$ and $2^{k_j}+1$ respectively where $i, j = 1,\ldots, m$. The main disadvantages of this work are unbalanced moduli sets and inefficient multiplicative inverses which yield to increasing the delay of reverse converter and inefficiency of arithmetic operation. In [2] RNS bases are presented in the form of $2^k\text{-}c_i$ where $0 \le c_i < 2^{k/2}$. Hamming weight of moduli in this work is equal to three in worse case. As discussed in [2], efficient reduction can be achieved by this form of moduli. Simple multiplicative inverses are another advantage of this work. This report is the fastest RNS implementation until now which considered the efficiency of arithmetic operation, residue/binary to binary/residue conversion. We can enjoy the efficiency of reverse conversion and arithmetic operation of moduli set $\{2^n, 2^n\text{-}1, 2^n\text{+}1, 2^n\text{-}2^{(n+1)/2}\text{+}1, 2^n\text{+}2^{(n+1)/2}\text{+}1\}$ [16] in the second basis of the work reported in [2], in order to realize RNS Montgomery multiplication with higher speed. This paper presents efficient RNS bases for public key cryptography and ECC especially. In first basis, search for moduli set in the form of $2^k\text{-}c_i$ where $0 \le c_i < 2^{k/2}$ with hamming weight equal to three are performed, and in the second basis, moduli set $\{2^n, 2^n\text{-}1, 2^n\text{+}1, 2^n\text{-}2^{(n+1)/2}\text{+}1, 2^n\text{+}2^{(n+1)/2}\text{+}1\}$ [16] is used. With these RNS bases, we can utilize the advantages of arithmetic unit and efficient forward and reverse conversion. Moreover, for the first basis, RNS moduli are proposed with variant bit lengths in order to

achieve different dynamic ranges especially for ECC, for example 192, 256 a nd 320 bits. The results show that noticeable improvement of modular multiplication is achieved compared to the method proposed in [2].
This paper is organized as follows. Section 2 introduces RNS and modular multiplication background. The proposed RNS bases are detailed in section 3. Comparison with other RNS bases is presented in Section 4 and finally section 5 concludes the paper.

2. Related background

2.1 Overview of RNS

We consider a s et of integers (p_1, p_2, …, p_m), which is called RNS basis with $M=\prod_{i=1}^{m} p_i$. The p_i's are pair wise relatively prime, where gcd (p_i, p_j) = 1, for $1 \le i, j \le m$, $i \ne j$. The RNS representation of an integer $X \in [0, M]$, is (x_1, x_2,…, x_m), i.e. $x_i = X \bmod p_i$. There are several algorithms for the reverse converter which translate residues into its equivalent weighted number. The Mixed Radix Conversion (MRC) is one of them calculated by:

$$X = v_m \prod_{i=1}^{m-1} p_i + \ldots + v_3 p_2 p_1 + v_2 p_1 + v_1 \quad (1)$$

$$X = v_1 + p_1(v_2 + p_2(v_3 + \ldots + p_{m-1} v_m)\ldots) \quad (2)$$

Where

$$v_1 = x_1$$

$$v_2 = \left|(x_2 - v_1)\left|p_1^{-1}\right|_{p_2}\right|_{p_2}$$

$$v_3 = \left|((x_3 - v_1)\left|p_1^{-1}\right|_{p_3} - v_2)\left|p_2^{-1}\right|_{p_3}\right|_{p_3}$$

And in the general case:

$$v_m = \left|(((x_m - v_1)\left|p_1^{-1}\right|_{p_m} - v_2)\left|p_2^{-1}\right|_{p_m} - \ldots - v_{m-1})\left|p_{m-1}^{-1}\right|_{p_m}\right|_{p_m}$$

$\left|p_i^{-1}\right|_{p_j}$ is the multiplicative inverse of p_i modulus p_j.

The other algorithm is Chinese Remainder Theorem (CRT) that convert residue number into weighted number X as follows:

$$X = \left|\sum_{i=1}^{m} |x_i N_i|_{p_i} M_i\right|_M \quad (3)$$

Where $M = p_1 p_2 \ldots p_m$, $M_i = \frac{M}{P_i}$ and $N_i = \left|M_i^{-1}\right|_{p_i}$ is multiplicative inverse of M_i modulus p_i. The CRT is implemented in parallel channel followed by a modulus M adder which is very large, but MRC is a sequential algorithm. For the moduli set with more than four moduli set, combination of these two algorithms could be applied to achieve higher speed of inverse converter [13-14].

2.2 Overview of RNS Montgomery multiplication

In this section we discuss the calculation of Montgomery modular multiplication in RNS introduced in [2]. Consider X and Y as two large numbers, B_m= {p_1,…,p_m} and B'_m={p'_1,…,p'_m} as two bases. Where $M=\prod_{i=1}^{m} p_i$ and $M'=\prod_{i=1}^{m} p'_i$ are the products of the elements of the RNS bases. The RNS representation of X and Y in these bases is equal to (x_1,…,x_m) and (y_1,…,y_m) in the first basis, and (x'_1,…,x'_m) and (y'_1,…,y'_m) in the second basis. We consider T where $T < M < M'$ and gcd(T, M) = gcd(T, M') = gcd(M, M') = 1. The term, $A \times B \times M^{-1} \bmod T$, can be calculated by modular multiplication as following:

RNS Montgomery Multiplication

1. Consider D as product of A and B in two bases B_m and B'_m. This means $d_i = |x_i \times y_i|_{p_i}$ in the first basis and $d'_i = |x'_i \times y'_i|_{p'_i}$ for i=1,…, m in the auxiliary basis and in the general case $D = A \times B$.
2. Consider $Q = \left|D \times |T|_P^{-1}\right|_P$ which is evaluated just in the first basis thus $q_i = \left|d_i \times |T|_{p_i}^{-1}\right|_{p_i}$.
3. Representation of Q is extended to auxiliary basis B'_m.
4. Consider $R = (D - Q \times T) \times |M|_{P'}^{-1}$ which is computed just in the auxiliary basis B'_m. Thus $r'_i = \left|(d'_i - q'_i \times \mathrm{N}'_i) \times |M|_{p'_i}^{-1}\right|_{p'_i}$.
5. Representation of R is extended to the first basis B_m.

3. Proposing RNS bases

Basic operation for RNS Montgomery multiplication consists of two conversions included several products and one addition [2]. Hence proper selection of RNS bases leads to speed up these operations in each moduli, and reverse and forward conversion can be done with more speed.

3.1 Selecting RNS bases for modular multiplication

Form of the moduli is very important for the efficiency of the modular multiplication, so selecting efficient RNS bases is the main purpose of this work in order to achieve efficient modular multiplication. In [2], RNS bases are presented in the form of 2^k-c_i where $0 \le c_i < 2^{k/2}$. Reduction in moduli 2^k-c_i is easy and efficient arithmetic operation will be achieved comparing to general moduli [2]. Simple multiplicative inverses is another advantages of the work reported in [2] which result in replacing multiplication required in calculation of v_i's in Eq. 2 by some simple shift and addition . Cost of reduction in [2] for moduli in the form of $2^k - c_i$ is reported as $2w(c_i)+2$ additions of k bit words where $w(c_i)$ is the hamming weight of c_i.

In order to increase efficiency of arithmetic unit and speed of reverse converters, RNS moduli set, $\{2^n, 2^n-1, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1\}$ [16], is used for the second basis. For the first basis like the method proposed in [2] the exhaustive search for the moduli in the form of 2^k-c_i where $0 \le c_i < 2^{k/2}$ is done. Although the first basis has simple multiplicative inverses [2], the second basis enjoys more efficient arithmetic operation, forward and reverse converter.

Three RNS bases with different dynamic ranges are proposed for each basis shown in table 1. Selecting RNS moduli sets with various bit lengths leads to having different dynamic ranges which is suitable for ECC, for example 192, 256 a nd 320 bits. Reverse converter for moduli set $\{2^n, 2^n-1, 2^n+1, 2^n-2^{(n+1)/2}+1, 2^n+2^{(n+1)/2}+1\}$ [16] is designed for odd n, therefore k is considered even in first basis and n is considered as k+1 which is shown in table 1.

Table 1: proposed RNS bases for various dynamic ranges

RNS bases	***First basis B_m***	***Auxiliary basis B'_m***
The first 5-moduli RNS bases	2^{64}-2^{10}-1 , 2^{64}-2^{31}-1 , 2^{64}-2^{16}-1 , 2^{64}-2^{19}-1 , 2^{64}-2^{20}-1 .	2^{65} , 2^{65}-1 , 2^{65}+1 , 2^{65}-2^{33}+1 , 2^{65}+2^{33}+1 .
The second 5-moduli RNS bases	2^{52}-2^{10}-1 , 2^{52}-2^{31}-1 , 2^{52}-2^{15}-1 , 2^{52}-2^{19}-1 , 2^{52}-2^{20}-1 .	2^{53} , 2^{53}-1 , 2^{53}+1 , 2^{53}-2^{27}+1 , 2^{53}+2^{27}+1 .
The third 5-moduli RNS bases	2^{40}-2^{8}-1 , 2^{40}-2^{10}-1 , 2^{40}-2^{16}-1 , 2^{40}-2^{19}-1 , 2^{40}-2^{20}-1 .	2^{41} , 2^{41}-1 , 2^{41}+1 , 2^{41}-2^{21}+1 , 2^{41}+2^{21}+1 .

The process of conversion from one basis to another needed in line 3 and 5 of modular multiplication algorithm prescribed in section 2.2 is shown in figure 1. Delay of conversion from first basis to second basis and vice versa must be considered in order to achieve overall delay.

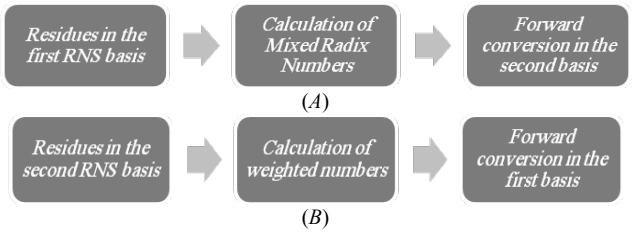

Fig.1 (*A*) conversion from first to second basis, (*B*) conversion from second to first basis

3.2 RNS to RNS conversion from first to second basis

As shown in figure 1, RNS to RNS conversion from first to second basis includes two steps: RNS to mixed radix system (MRS) in first basis and MRS to RNS from first to second basis. Eq. 4 shows the delay of these conversions:

$$Delay_{RNS-RNS} = Delay_{RNS-MRS} + Delay_{MRS-RNS} \quad (4)$$

Delay of RNS to MRS based on [2] in first basis is:

$$Delay_{RNS-MRS} = \left(\sum_{i=1}^{m-1} \max_{j=2,m;i<j} \left(w\left(\left| p_i^{-1} \right|_{p_j} \right) + 2w(c_j) + 4 \right) \right) kD_{FA} \quad (5)$$

Where $\left| p_i^{-1} \right|_{p_j}$ is multiplicative inverse of p_i modulus p_j, $w(c_j)$ is the hamming weight of c_j, D_{FA} is delay of one bit full adder and m is the number of moduli in each basis. For five moduli set Eq. 5 is reformed to:

$$Delay_{RNS-MRS} = \left(\sum_{i=1}^{4} \max_{j=2,5;i<j} \left(w\left(\left| p_i^{-1} \right|_{p_j} \right) + 2w(c_j) + 4 \right) \right) kD_{FA} \quad (6)$$

Delay of RNS to MRS conversion from first to second basis with different bit lengths is shown in table 2. Note that the moduli in first basis are same with [2]. Therefore comparing to [2], same RNS to MRS conversion from first to second basis is achieved.

Table 2: cost of conversion from RNS to MRS with various bit lengths

Key length	***Conversion's Delay from RNS to MRS***
320	5632 D_{FA}
256	4836 D_{FA}
192	3220 D_{FA}

After calculation of MRS, conversion of MRS numbers to residues in the next basis must be done. Since conversion of MRS to residues in second basis can be perform in

parallel, hardware implementation for critical moduli in second bases which are the moduli $2^n+2^{(n+1)/2}+1$ and $2^n-2^{(n+1)/2}+1$ are done. The proposed hardware implementation represented in the following subsection which has delay

$$Delay_{MRS-RNS}=\left(\sum_{i=1}^{m-1}(MA(2^n\pm 2^{(n+1)/2}+1)+3CSA+CPA)\right)D_{FA} \quad (7)$$

Where MA $(2^n\pm2^{(n+1)/2}+1)$ is modular adder in modulo $2^n\pm2^{(n+1)/2}+1$, CSA represent carry save adder and CPA is carry propagation delay where ripple carry adder is used in this design. Based on achieved hardware and delay for reduction in moduli $2^n\pm2^{(n+1)/2}+1$, MRS to RNS conversion in second basis has delay

$$Delay_{MRS-RNS}=\left(\sum_{i=1}^{m-1}(MA(2^n\pm 2^{(n+1)/2}+1)+3CSA+CPA)\right)D_{FA} \quad (8)$$

By using modulo m adder [21] and considering $2n$ delay of FA for moduli $2^n-2^{(n+1)/2}+1$, we have:

$$Delay_{MRS-RNS}=\sum_{i=1}^{4}(2n+4+(n+3))=(12n+28)D_{FA} \quad (9)$$

Similarly considering $(2n+2)$ delay of FA for moduli $2^n+2^{(n+1)/2}+1$, we have

$$Delay_{MRS-RNS}=\sum_{i=1}^{4}(2n+6+(n+3))=(12n+30)D_{FA} \quad (10)$$

3.2.1 Reduction in modulo $2^n-2^{(n+1)/2}+1$

Efficient binary to residue conversion in moduli 2^n, 2^n-1 and 2^n+1 are proposed by researcher [22] which can be employed in MRS to RNS conversion in this work. For the calculation of MRS to RNS delay, moduli in the form of $2^n-2^{(n+1)/2}+1$ and $2^n+2^{(n+1)/2}+1$ must be considered. Reduction in modulo $2^n-2^{(n+1)/2}+1$ is based on Eq. 2. Let us rewrite it for simplicity.

$$x_i=\left|v_1+p_1(v_2+p_2(v_3+\ldots+p_{m-1}v_m)\ldots)\right|_{p_j} \quad (11)$$

Where p_1, p_2,…, p_{m-1} are moduli in the form of $2^k-2^{t_i}-1$ and p_j is $2^n-2^{(n+1)/2}+1$. Therefore Eq. 11 can be rewritten as:

$$x_i=\left|\begin{matrix}v_1+(2^k-2^{t_1}-1)(v_2+\\(2^k-2^{t_2}-1)(\underbrace{v_3+(2^k-2^{t_3}-1)v_4}_{L}+\ldots)\ldots)\end{matrix}\right|_{2^n-2^{\frac{(n+1)}{2}}+1} \quad (12)$$

Note that $n = k+1$ in Eq. 12. Considering L as basic operation in Eq. 12 result in

$$L=\left|v_i+v_{i+1}\underbrace{0\ldots0}_{k}-v_{i+1}\underbrace{0\ldots0}_{t_i}-v_{i+1}\right|_{2^n-2^{\frac{(n+1)}{2}}+1} \quad (13)$$

$(k+1)$-bit separation results in

$$L=\left|v_i^1+v_{i+1}^1+v_{i+1}^2-v_{i+1}^3-v_{i+1}^4-v_{i+1}^5\right|_{2^n-2^{\frac{(n+1)}{2}}+1} \quad (14)$$

Where

$$v_i^1=0v_i$$

$$\left.\begin{matrix}v_{i+1}^1=v_{i+1,0}\underbrace{0\ldots0}_{k}\\v_{i+1}^2=00v_{i+1,k-1}\ldots v_{i+1,1}\end{matrix}\right\}v'_{i+1}=v_{i+1,0}0v_{i+1,k-1}\ldots v_{i+1,1}$$

$$\left.\begin{matrix}v_{i+1}^3=v_{i+1,k-t}\ldots v_{i+1,0}\underbrace{0\ldots0}_{t_i}\\v_{i+1}^4=\underbrace{0\ldots0}_{k-t_i+2}v_{i+1,k-1}..v_{i+1,k-t_i+1}\end{matrix}\right\}v''_{i+1}=v_{i+1,k-t}\ldots v_{i+1,0}0v_{i+1,k-1}..v_{i+1,k-t_i+1}$$

$$v_{i+1}^5=0v_{i+1}$$

Negative number in modulo $2^n-2^{(n+1)/2}+1$ can be expressed as

$$\begin{aligned}\left|-v\right|_{2^n-2^{\frac{(n+1)}{2}}+1}&=\left|(2^n-2^{(n+1)/2}+1)-v\right|_{2^n-2^{\frac{(n+1)}{2}}+1}\\&=\left|(2^n-1-v)-2^{(n+1)/2}+2\right|_{2^n-2^{\frac{(n+1)}{2}}+1}\\&=\left|\bar{v}+(-2^{(n+1)/2}+2)\right|_{2^n-2^{\frac{(n+1)}{2}}+1}\end{aligned}$$

Since n is 41, 53 and 65 shown in table 1, the value $(-2^{(n+1)/2}+2)$ can be computed according to value of n. Therefore $(-2^{(n+1)/2}+2)$ is considered as constant r which is determined according to n. Thus

$$L=\left|v_i^1+v'_{i+1}+\bar{v}''_{i+1}+\bar{v}_{i+1}^5+2r\right|_{2^n-2^{\frac{(n+1)}{2}}+1} \quad (15)$$

Hardware implementation of Eq. 15 is shown in figure 2. In this figure, output of the binary adder S has $n+3$ bit. In order to calculate S in modulo $2^n-2^{(n+1)/2}+1$, with $(n-1)$-bit separation S_1 and S_2 are achieved where S_1 is $n-1$ bits *LSB* and S_2 is the rest. Therefore $\left|S_1\right|_{2^n-2^{(n+1)/2}+1}=S_1$ and S_2 is applied to a combinational circuit that computes $\left|S_2\right|_{2^n-2^{(n+1)/2}+1}$ [20], and produce the variable, S_3. Finally modulo adder are used to calculate the results.

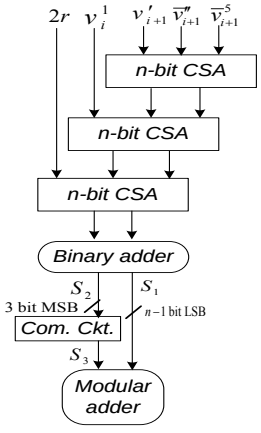

Fig.2 hardware implementation of reduction of L in modulo $2^n-2^{(n+1)/2}+1$

Since L has (k+1) bit, in the next step $L' = v_i + (2^k - 2^{t_i} - 1) \times L$ must be calculated as follows:

$$L' = \left| v_i + (2^k - 2^{t_i} - 1)L \right|_{2^n - 2^{\frac{(n+1)}{2}} + 1} \quad (16)$$

$$L' = \left| v_i + L^1 + L^2 + \bar{L}^3 + \bar{L}^4 + \bar{L} + 3r \right|_{2^n - 2^{\frac{(n+1)}{2}} + 1} \quad (17)$$

Where

$$\left.\begin{aligned} v_i^1 &= 0 v_i \\ L^1 &= L_0 \underbrace{0 \ldots 0}_{k} \\ L^2 &= 0 L_k \ldots L_1 \end{aligned}\right\} L' = L_0 L_k \ldots L_1$$

$$\left.\begin{aligned} L^3 &= L_{k-t_i} \ldots L_0 \underbrace{0 \ldots 0}_{t_i} \\ L^4 &= \underbrace{0 \ldots 0}_{k-t_i+1} L_k \ldots L_{k-t_i+1} \end{aligned}\right\} L'' = L_{k-t_i} \ldots L_0 L_k \ldots L_{k-t_i+1}$$

So Eq. 17 changes to

$$L' = \left| v_i^1 + L' + \bar{L}'' + \bar{L} + 2r \right|_{2^n - 2^{\frac{(n+1)}{2}} + 1} \quad (18)$$

Hardware implementation of L' is similar to figure 2. Based on this hardware implementation, the delay and area of conversion from MRS to RNS can be calculated as

$$Delay_{MRS-RNS} = \left(\sum_{i=1}^{m-1} (MA(2^n - 2^{(n+1)/2} + 1) + 3CSA + CPA) \right) D_{FA} \quad (19)$$

$$Area_{MRS-RNS} = \left(\sum_{i=1}^{m-1} (MA(2^n - 2^{(n+1)/2} + 1) + 3CSA + CPA) \right) A_{FA} \quad (20)$$

Where A_{FA} is area of one bit full adder.

3.2.2 Reduction in modulo $2^n+2^{(n+1)/2}+1$

Based on Eq. 12 we have

$$x_i = \left| v_1 + (2^k - 2^{t_1} - 1)(v_2 + (2^k - 2^{t_2} - 1)(\underbrace{v_3 + (2^k - 2^{t_3} - 1)v_4}_{I} + \ldots)\ldots) \right|_{2^n + 2^{\frac{(n+1)}{2}} + 1} \quad (21)$$

Note that n=k+1 in Eq. 21. Basic operation in Eq. 21 is calculation of I which can be done as:

$$I = \left| v_i + v_{i+1} \underbrace{0 \ldots 0}_{k} - v_{i+1} \underbrace{0 \ldots 0}_{t_i} - v_{i+1} \right|_{2^n + 2^{\frac{(n+1)}{2}} + 1} \quad (22)$$

(k+2)-bit separation results in

$$I = \left| v_i^1 + v_{i+1}^1 + v_{i+1}^2 - v_{i+1}^3 - v_{i+1}^4 - v_{i+1}^5 \right|_{2^n + 2^{\frac{(n+1)}{2}} + 1} \quad (23)$$

Where

$$v_i^1 = 00 v_i$$

$$\left.\begin{aligned} v_{i+1}^1 &= v_{i+1,1} v_{i+1,0} \underbrace{0 \ldots 0}_{k} \\ v_{i+1}^2 &= 0000 v_{i+1,k-1} \ldots v_{i+1,2} \end{aligned}\right\} v'_{i+1} = v_{i+1,1} v_{i+1,0} 00 v_{i+1,k-1} \ldots v_{i+1,2}$$

$$\left.\begin{aligned} v_{i+1}^3 &= v_{i+1,k-t_i+1} \ldots v_{i+1,0} \underbrace{0 \ldots 0}_{t_i} \\ v_{i+1}^4 &= \underbrace{0 \ldots 0}_{k-t_i+4} v_{i+1,k-1} \ldots v_{i+1,k-t_i+2} \end{aligned}\right\} v''_{i+1} = v_{i+1,k-t_i+1} \ldots v_{i+1,0} 00 v_{i+1,k-1} \ldots v_{i+1,k-t_i+2}$$

$$v_{i+1}^5 = 00 v_{i+1}$$

Following the same approach that is used in calculation of $\left| -v \right|_{2^n - 2^{\frac{(n+1)}{2}} + 1}$, then $\left| -v \right|_{2^n + 2^{\frac{(n+1)}{2}} + 1}$ can be expressed as

$$\begin{aligned} \left| -v \right|_{2^n + 2^{\frac{(n+1)}{2}} + 1} &= \left| (2^n + 2^{(n+1)/2} + 1) - v \right|_{2^n + 2^{\frac{(n+1)}{2}} + 1} \\ &= \left| (2^n - 1 - v) + 2^{(n+1)/2} + 2 \right|_{2^n + 2^{\frac{(n+1)}{2}} + 1} \\ &= \left| \bar{v} + 2^{(n+1)/2} + 2 \right|_{2^n + 2^{\frac{(n+1)}{2}} + 1} \end{aligned}$$

As mentioned before, n is 41, 53 and 65, so the value of $(2^{(n+1)/2}+2)$ can be computed according to value of n. Therefore $(2^{(n+1)/2}+2)$ is considered as constant r' which is determined according to n. Thus

$$I = \left| v_i^1 + v'_{i+1} + \bar{v}''_{i+1} + \bar{v}^5_{i+1} + 2r' \right|_{2^n + 2^{\frac{(n+1)}{2}} + 1} \quad (24)$$

By replacing I in Eq. 21 with (k+2)-bit binary form, $v_i + (2^k - 2^{t_i} - 1) \times I$ must be calculated. Therefore we have:

$$I' = \left| v_i + (2^k - 2^{t_i} - 1)I \right|_{2^n + 2^{\frac{(n+1)}{2}} + 1} \quad (25)$$

$$I' = \left| v_i + I^1 + I^2 + \bar{I}^3 + \bar{I}^4 + \bar{I} + 3r' \right|_{2^n + 2^{\frac{(n+1)}{2}} + 1} \quad (26)$$

Where

$$\left.\begin{aligned} v_i^1 &= 00 v_i \\ I^1 &= I_1 I_0 \underbrace{0 \ldots 0}_{k} \\ I^2 &= 00 I_{k+1} \ldots I_2 \end{aligned}\right\} I' = I_1 I_0 I_{k+1} \ldots I_2$$

$$\left.\begin{aligned} I^3 &= I_{k-t_i+1} \ldots I_0 \underbrace{0 \ldots 0}_{t_i} \\ I^4 &= \underbrace{0 \ldots 0}_{k-t_i+2} I_{k+1} \ldots I_{k-t_i+2} \end{aligned}\right\} I'' = I_{k-t_i+1} \ldots I_0 I_{k+1} \ldots I_{k-t_i+2}$$

So Eq. 26 changes to

$$I' = \left| v_i + I' + \bar{I}'' + \bar{I} + 2r' \right|_{2^n + 2^{\frac{(n+1)}{2}} + 1} \quad (27)$$

Hardware implementation of Eq. 27 is also similar to figure 2. Therefore delay of conversion from MRS to RNS for moduli, $2^n-2^{(n+1)/2}+1$ and $2^n+2^{(n+1)/2}+1$, is equal to:

$$Delay_{MRS-RNS} = \left(\sum_{i=1}^{m-1} (MA(2^n \pm 2^{(n+1)/2} + 1) + 3CSA + CPA) \right) D_{FA} \quad (28)$$

3.3 RNS to RNS conversion from second to first basis

As shown in figure 1, RNS Montgomery modular multiplication required conversion from second to first basis. The second basis is 5-moduli set, $\{2^n, 2^n\text{-}1, 2^n\text{+}1, 2^n\text{-}2^{(n+1)/2}\text{+}1, 2^n\text{+}2^{(n+1)/2}\text{+}1\}$ [16]. Delay of conversion from second to first basis can be calculated as:

$$Delay_{RNS-RNS} = Delay_{RNS-Weighted} + Delay_{Weighted-RNS} \quad (29)$$

Delay and area of conversion from RNS to weighted number for these five moduli set based on [16] is shown in the table 3.

Table 3: Delay and area of reverse conversion in second RNS basis

RNS basis	***Area of Conversion from RNS to Weighted***	***Delay of Conversion from RNS to Weighted***
$\{2^n, 2^n\text{-}1, 2^n\text{+}1, 2^n\text{-}2^{(n+1)/2}\text{+}1, 2^n\text{+}2^{(n+1)/2}\text{+}1\}$ [16]	$(19n)A_{FA}$+ $(7n)A_{XOR}$+ $(7n)A_{AND}$+ $(2n)A_{XNOR}$+ $(2n)A_{OR}$+ $(4n)A_{NOT}$	$(8n\text{+}4)D_{FA}$

The forward converter for modulo in the form of $2^k - 2^{t_i} - 1$ where $0 < t_i < k/2$ is presented in [2]. This modulo achieves small hamming weight and simple multiplicative inverses. In [2] to calculate the residue numbers from MRS in five moduli set based on Eq. 2 the following operation must be considered:

$$x_j = \left| v_1 + p_1(v_2 + p_2(v_3 + p_3(\underbrace{v_4 + p_4 v_5}_{H}))) \right|_{2^k - 2^{t_j} - 1} \quad (30)$$

The delay of H is addition of k bit words where $w(c_i)$ and $w(c'_j)$ are the hamming weight of $2^k - 2^{t_i} - 1$ and $2^k - 2^{t_j} - 1$, respectively. In [2] total delay for m moduli is reported as

$$Delay_{MRS-RNS} = \left(\sum_{i=1}^{m-1} \max_{j=1,m} (w(c_i) + 2w(c'_j) + 2) \right) kD_{FA} \quad (31)$$

For designing forward converter in modulo $2^k - 2^{t_j} - 1$ for $5k$-bit dynamic ranges we have

$$X = \sum_{i=0}^{4} 2^{ik} x_i = 2^{4k} x_4 + 2^{3k} x_3 + 2^{2k} x_2 + 2^k x_1 + x_0 \quad (32)$$

Eq. 32 can be rewritten as

$$X = 2^k (2^k (2^k \underbrace{(2^k x_4 + x_3)}_{z} + x_2) + x_1) + x_0 \quad (33)$$

Unlike the calculation needed in Eq. 31, the hamming weight of c_i in calculation of z as the basic operation in Eq. 33 is equal to zero. Therefore the delay of reduction in modulo $2^k - 2^{t_j} - 1$ reported in Eq. 31 for the proposed RNS bases is changed to

$$Delay_{Weighted-RNS} = \left(\sum_{i=1}^{m-1} (2w(c'_j) + 2) \right) kD_{FA} \quad (34)$$

Therefore for five moduli set, Eq. 32 changes to

$$Delay_{Weighted-RNS} = \left(\sum_{i=1}^{4} (2w(c'_j) + 2) \right) kD_{FA} \quad (35)$$

Total delays from second to first basis for proposed moduli sets are shown in table 4.

Table 4: total cost of conversion from second to first basis

key length	***5moduli proposed***
320	$2060\ D_{FA}$
256	$1676\ D_{FA}$
192	$1292\ D_{FA}$

4. Complexity of modular multiplication and comparison

The main aim of this work is increasing the efficiency of arithmetic operation. Since the second basis of our approach is the moduli set $\{2^n, 2^n\text{-}1, 2^n\text{+}1, 2^n\text{-}2^{(n+1)/2}\text{+}1, 2^n\text{+}2^{(n+1)/2}\text{+}1\}$, reduction in this moduli set can be implemented with more simple process (four levels of CSA and MA ($2^n \pm 2^{(n+1)/2}+1$) in worse case). With using the moduli set, $\{2^n, 2^n\text{-}1, 2^n\text{+}1, 2^n\text{-}2^{(n+1)/2}\text{+}1, 2^n\text{+}2^{(n+1)/2}\text{+}1\}$ in the second basis, the delay of MRS to RNS conversion is implemented with faster hardware. As shown in table 5, the proposed RNS bases have achieved noticeable improvement in delay of RNS to RNS conversion required in the process of RNS Montgomery multiplication.

Table 5: Comparison delay of different RNS bases for P_{256}

RNS bases	***Total delay***	***Improvement (%)***
5moduli bases [2]	$(11840)D_{FA}$	-
The first 5moduli proposed	$(8502)D_{FA}$	28%

5. Conclusion

This paper presents five moduli RNS bases in order to increase the efficiency of RNS Montgomery multiplication. RNS moduli sets with various bit lengths are proposed which cover different dynamic ranges for ECC 192, 256 and 320 bits. Higher speed in RNS to RNS conversion is achieved by the proposed RNS bases. Comparison with the best RNS bases achieved 28% improvement in delay of RNS to RNS conversion in five moduli RNS bases. Therefore more efficient modular multiplication is achieved by the proposed RNS bases.

References

[1] K. Navi, A. S. Molahosseini, M. Esmaeildoust, "How to Teach Residue Number System to Computer Scientists and Engineers," IEEE Transactions on Education, vol. 53, no. 3, 2010.

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

[3] J. C. Bajard, L. Imbert, "A Full RNS Implementation of RSA," IEEE Transactions on Computers, vol. 53, no. 6, pp. 769-774, 2004.

[4] J. Bajard, L. Didier, and P. Kornerup, "An RNS Montgomery's Modular Multiplication Algorithm," IEEE Trans. Computers, vol. 47, no. 2, pp. 167-178, Feb. 1998.

[5] J. Bajard, L. Didier, and P. Kornerup, "Modular Multiplication and Base Extensions in Residue Number Systems," Proc. 15th IEEE Symp. Computer Arithmetic (ARITH '01), pp. 59-65, 2001.

[6] A.F. Tenca and C .K. Koc, "A Scalable Architecture for Modular Multiplication Based on Montgomery's Algorithm," IEEE Trans. Computers, vol. 52, no. 9, pp. 1215-1221, Sept. 2003.

[7] C. McIvor, M. McLoone, and J.V. McCanny, "Modified Montgomery Modular Multiplication and RSA Exponentiation," IEE Proc. Computers and Digital Techniques, vol. 151, pp. 402-408, 2004.

[8] C. McIvor, M. McLoone, J.V. McCanny, A. Daly, and W. Marnane, "Fast Montgomery Modular Multiplication and RSA Cryptographic Processor Architectures," Proc. 37th Ann. Asilomar Conf. Signals, Systems, and Computers, 2003.

[9] D. M. Schinianakis, A. P. Fournaris, H. E. Michail, A. P. Kakarountas, and T. Stouraitis, "An RNS Implementation of an Fp Elliptic Curve Point Multiplier", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS-I, VOL. 56, NO. 6, JUNE 2009.

[10] D. M. Schinianakis, A. P. Kakarountas, and T. Stouraitis, "A new approach to elliptic curve cryptography: An RNS architecture," in Proc. IEEE Mediterranean Electrotech. Conf., pp. 1241-1245, May 2006.

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

[12] K. Navi, M. Esmaeildoust, A. S. Molahosseini, "A General Reverse Converter Architecture with Low Complexity and High Performance", IEICE TRANSACTIONS on Information and Systems Vol.E94-D No.2 pp.264-273, 2011.

[13] A. S. Molahosseini, K. Navi, C. Dadkhah, O. Kavehei and S. Timarchi, "Efficient Reverse Converter Designs for the new 4-Moduli Set $\{2^n\text{-}1, 2^n, 2^n\text{+}1, 2^{2n+1}\text{-}1\}$ and $\{2^n\text{-}1, 2^n\text{+}1, 2^{2n}, 2^{2n}\text{+}1\}$ Based on New CRTs," IEEE Transactions on Circuits and Systems-I, vol. 57, no. 4, (2010), pp. 823-835.

[14] Mohammad Esmaeildoust, Keivan Navi and MohammadReza Taheri, "High speed reverse converter for new five-moduli set $\{2^n, 2^{2n+1}\text{-}1, 2^{n/2}\text{-}1, 2^{n/2}\text{+}1, 2^n\text{+}1\}$," IEICE Electronics Express, Vol. 7, No. 3 pp.118-125, 2010.

[15] Y. Wang, X. Song, M. Aboulhamid and H. Shen, "Adder based residue to binary numbers converters for $\{2^n\text{-}1, 2^n, 2^n\text{+}1\}$," IEEE Transactions on Signal Processing, vol. 50, no. 7, pp. 1772-1779, 2002.

[16] A. A. Hiasat, "VLSI implementation of New Arithmetic Residue to Binary decoders," IEEE Transactions on VLSI systems, vol. 13, no. 1, pp. 153-158, 2005.

[17] P.Montgomery, "Modular Multiplication without Trial Division," Mathematics of Computation, vol. 44, no. 170, pp. 519-521, Apr. 1985.

[18] K.C Posch and R. Posch, Modulo reduction in residue number systems. IEEE Transaction on Parallel and Distributed Systems, 6:5 (1995) p.: 449-454

[19] Kooroush Manochehri Kalantari, Saadat Pour Mozafari and Babak Sadeghiyan, "Improved RNS for RSA Hardware Implementation", Vol. 2, No. 2&4 (b), pp. 31-39, 2004.

[20] A.A. Hiasat, "Arithmetic binary to residue encoders for moduli $(2^n \pm 2^k\text{+}1)$", IEE Proc.-Comput. Digit. Tech., Vol. 150, No. 6, November 2003.

[21] M.A. Bayoumi, G.A. Jullien, W.C. Miller, "A VLSI implementation of residue adder," IEEE Transactions on Circuits and Systems, vol. 34, no. 3, pp. 284-288, 1987.

[22] B. Guan and E.V. Jones, "Fast conversion between binary and residue numbers, " Electronics Letters, vol. 24, no. 19, pp. 1195-1197, 1988.

Shirin Rezaie was born in Tehran, Iran, in 1986. She received the B.Sc. degree from Islamic Azad University (IAU), South Tehran Branch, Tehran, Iran in 2009. She is M.Sc. student in Computer Architecture at IAU, Science and Research Branch. She is working on computer arithmetic especially on residue number system.

Mohammad Esmaeildoust received the B.Sc. degree in hardware engineering from Shahed University, Tehran, Iran, in 2006, and he M.Sc. degree in computer architecture from Shahid Beheshti University of Technology, Tehran, Iran, in 2008. He is currently a Ph.D. candidate in computer architecture at Shahid Beheshti University of Technology. He is working on reconfigurable computing and computer arithmetic, especially in the area of the residue number system.

Marzieh Gerami is M.Sc. student in Computer Architecture at IAU, Science and Research Branch (Tehran, Iran). She received the B.Sc. degree from Shahid Bahonar University Of Kerman, Iran in 2007. She is working on computer arithmetic especially on residue number system.

Keivan Navi received the M.Sc. degree in Electrical Engineering (Computer Hardware) from Sharif University of Technology, Tehran, Iran, in 1990 and the Ph.D. degree in computer architecture from Paris XI University, Paris, France, in 1995. He is currently an Associate Professor in the Faculty of Electrical and Computer Engineering, Shahid Beheshti University. His research interests include the Residue Number System, carbon nanotubes, single electron transistors, reversible logic design, interconnection network, and quantum computing.

Omid Hashemipour (BS'85, MS'87, Phd'91) in electrical engineering all received from university of Arkansas at Fayetteville USA. From 1991, he is with the Electrical and Computer engineering Faculty at Shahid Beheshti University, G.C., Tehran, Iran as an associate professor. His research interest includes Low Power, Low Voltage Analog and Digital integrated Circuits.