Journal of mathematics and computer science 12 (2014), 51-64

Contents list available at JMCS

Journal of Mathematics and Computer Science

Journal Homepage: www.tjmcs.com

Efficient Implementation of RNS Montgomery Multiplication Using Balanced RNS Bases

Sakineh Sharifi[1], Mohammad Esmaeildoust[2], Mohammad Reza Taheri[3], Keivan Navi[4]

[1]*Department of Information Technology Engineering, Qom University, Qom, Iran*

[2]*Faculty of Marine Engineering, Khorramshahr University of Marine Science and Technology, Iran*

[3] *Nanotechnology and Quantum Computing Laboratory, Shahid Beheshti University, GC, Tehran, Iran*

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

m_doust@kmsu.ac.ir

Article history:
Received June 2014
Accepted July 2014
Available online August 2014

Abstract

Point multiplication is the most important part of elliptic curve cryptography which consumes remarkable time of implementation. Therefore efficiency enhancement of entire system is depending on efficiency of this part. Increasing the efficiency of the modular multiplication improve overall performance of the cryptographic system as it frequency used in some application such as Elliptic Curve Cryptography. By applying Residue Number System (RNS) to Montgomery multiplication as a method for modular multiplication, delay of modular multiplication will be reduced. Appropriate RNS moduli sets replace time consuming operation of multiplication by smaller operations. In this paper two balanced moduli set with proper dynamic range is presented and the efficiency of conversion from RNS to RNS which is the most time consuming part of the Montgomery modular multiplication will be increased.

Keywords: *residue number system (RNS), RNS Montgomery, reverse converter, elliptic curve cryptography*

1. Introduction

One of the best public key cryptography systems are RSA [1] and Elliptic curves cryptography [2] [3]. ECC can provide security equivalent RSA with smaller key and a fewer calculations. ECC systems are

face with several problems. As well as we know the important operation in ECC is point multiplication that include two basic operations are on field P: point doubling (P.D) and point addition (P.A). In order to improving performance of operation on P, Montgomery multiplication [4] can be employed. Montgomery multiplication performs modulo reduction without division. So as to increase efficiency of Montgomery, RNS Montgomery presented [5] [6]. RNS is a non-weighted number system and operations on large numbers are done over small moduli. As described in [5], two RNS bases are required to perform RNS Montgomery multiplication and conversion from one basis to another is needed in this process. In [7], RNS bases in the form of 2^n-c_i where $0 \leq c_i < 2^{k/2}$ are considered. In [8],[9], in one bases efficient RNS moduli set such as $\{2^n, 2^n\text{-}1, 2^n+1, 2^{n-2(n+1)/2}+1, 2^{n+2(n+1)/2}+1\}$ [10], $\{2^n\text{-}1, 2^n, 2^n+1, 2^{2n}+1\text{ -}1\}$ [11] and $\{2^n\text{-}1, 2^n, 2n+1, 2^{2n}+1\}$ [12] are considered and for first bases the moduli set in the form of 2^n-c_i [7] are considered. Although in [7] it is proved that in the special case of 2^n-c_i where $0 \leq c_i < 2^{k/2}$, modulo reduction can be efficiently implemented, but using well formed moduli such as 2^n-1, 2^n+1, 2^n, 2^n-3, 2^n+3, where efficient modulo adder [13-15], multiplier and residue to binary converter and binary to residue converter are presented for these moduli by researcher [16-21], can leads to more efficient RNS Montgomery implementation.
In this paper, two efficient RNS bases $\{2^n, 2^n+1, 2^{n-1}\text{-}1\}$ and $\{2^n+3, 2^n\text{-}1, 2^n\text{-}3\}$ are selected for RNS Montgomery multiplication and required conversions are efficiently implemented. The selected RNS bases are balanced and this lead to modulo channel with approximately same delay. Fast and efficient implementation of RNS Montgomery multiplication leads to meliorate the performance of P.A and P.D.
This paper organized as follow: Section 2 provides the related background of RNS and RNS Montgomery. In section 3, the RNS bases and efficient design of required conversions in RNS Montgomery regarding to proposed bases are presented. Performance of proposed work are evaluated and compared in section 4 and finally section 5 concludes the paper.

2. Related Background

This section in three subsections mathematical background of RNS Montgomery multiplication and RNS will be discussed, respectively.

2.1 Montgomery in RNS

RNS Montgomery multiplication is presented by [5]. In RNS Montgomery multiplication two RNS bases (moduli set) are required. Considering X and Y as two large integer number with RNS representation $(x_1, \ldots, x_m)$ and $(y_1, \ldots, y_m)$ in first basis $(p_1, \ldots, p_m)$ and in the second basis we consider X and Y as $(x'_1, \ldots, x'_m)$ and $(y'_1, \ldots, y'_m)$ in second basis $(p'_1, \ldots, p'_m)$.

Algorithm 1 shows the RNS Montgomery multiplication [7]. $M = p_1 \times p_2 \ldots \times p_i$ and $M' = p'_1 \times p'_2 \ldots \times p'_i$ are the dynamic range for first and second bases, respectively. Consider T which is $T < M < M'$, so that gcd (T, M) = gcd (T, M') = gcd (M, M') = 1. Montgomery multiplication performs modulo reduction without division. The most important part of Montgomery algorithm is moduli selection that leads to design pretty faster converter and efficient arithmetic unit. Choosing moduli set is necessary to provide these features, so in this approach the RNS basis in order to achieve the high performance of multiplication is proposed.

According to algorithm 1, in the process of Montgomery multiplication conversion from one basis to another is required. Figure 1 shows the required conversions.

Algorithm 1: RNS Montgomery multiplication

```
1: $D = X \times Y \; (d_i = |x_i \times y_i|_{m_i}$ in base $B_n$
   $d'_i = |x'_i \times y'_i|_{m'_i}$ in base $B'_n)$
2: $q_i = \left| d_i \times |-T|^{-1}_{m_i} \right|_{m_i}$ in $B_n$
3: $q_i$ in $B_n \longrightarrow q'_i$ in $B'_n$
4: $r' = \left| (d'_i + q'_i \times N'_i) |M^{-1}|_{m'_i} \right|_{m'_i}$ in $B'_n$
5: $r$ in $B_n \longleftarrow r'$ in $B'_n$
```

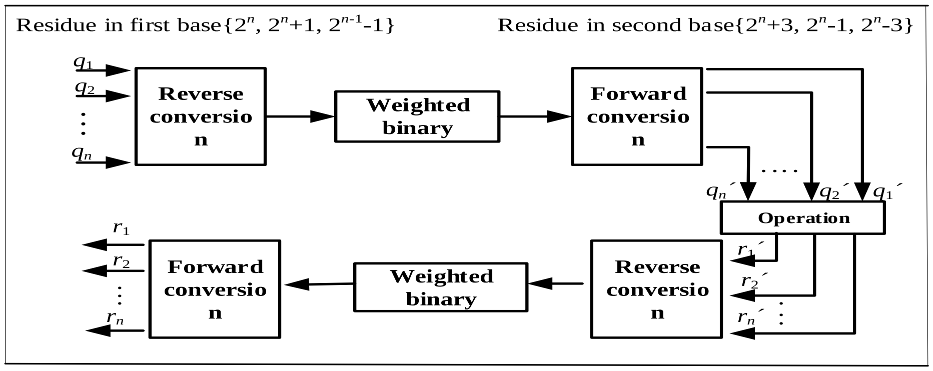

Figure 1. Overview of base extension in Montgomery algorithm

Conversion from one basis to another needs the mathematical background of the RNS. Therefore in the following the related background of the RNS is detailed.

2.2 Residue Number System

The RNS is an unconventional number system which is defined in terms of relatively-prime moduli set $\{m_1, m_2, \ldots, m_n\}$ that is gcd $(m_i, m_j) = 1$ for $i \neq j$. A weighted number X can be represented as $X = (x_1, x_2, \ldots, x_n)$, where

$$x_i = X \bmod m_i = |X|_{m_i}, 0 \leq x_i < m_i \tag{1}$$

Such a representation is unique for any integer X in the range [0, M-1], where M is the dynamic range of the moduli set $\{m_1, m_2, \ldots, m_n\}$ which is equal to the product of mi terms ($M = \prod_{i=1}^{n} m_i$)[22].

RNS includes three main parts: forward converter, reverse converter and arithmetic operation [23]. Several algorithms for reverse conversion can be employed such as Chinese reminder Theorem (CRT), Mixed Radix Conversion (MRC) or the modified version of these algorithms [23]. Since MRC is used in the required conversion in this paper, the mathematical details of MRC is discussed in the following.

In MRC [24], [25], the number X can be calculated from residues by:

$$X = v_n \prod_{i=1}^{n-1} P_i + \ldots + v_3 P_2 P_1 + v_2 P_1 + v_1 \quad (2)$$

The coefficients $\{v_1, v_2, \ldots, v_n\}$ can be obtained from residues as follows:

$$v_1 = x_1 \quad (3)$$

$$v_2 = \left| (x_2 - v_1) \left| P_1^{-1} \right|_{P_2} \right|_{P_2} \quad (4)$$

$$v_3 = \left| ((x_3 - v_1) \left| P_1^{-1} \right|_{P_3} - v_2) \left| P_2^{-1} \right|_{P_3} \right|_{P_3} \quad (5)$$

In the general case

$$v_n = \left| (((x_n - v_1) \left| P_1^{-1} \right|_{P_n} - v_2) \left| P_2^{-1} \right|_{P_n} - \cdots - v_{n-1}) \left| P_{n-1}^{-1} \right|_{P_n} \right|_{P_n} \quad (6)$$

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

3. Selected RNS Bases

In order to increase the efficiency of Montgomery in RNS, efficient RNS bases are required. To achieve this, for first and second bases $\{2^n, 2^n+1, 2^{n-1}-1\}$ and $\{2^n+3, 2^n-1, 2^n-3\}$ when n is even, are considered as moduli sets, respectively. As mentioned before, conversion from one base to another is needed in the process of RNS Montgomery multiplication. In the following, the required conversion will be detailed.

3.1 Residue-to-binary conversion in first bases

By using MRC and considering $P_1 = 2^n$, $P_2 = 2^n+1$ and $P_3 = 2^{n-1}-1$ and residues $Z = (x_1, x_2, x_3)$, we have

$$X = v_1 + v_2 2^n + v_3 (2^n + 1)(2^n) \quad (7)$$

Where

$$v_1 = x_1 \quad (8)$$

$$v_2 = \left| (x_2 - x_1) \left| P_1^{-1} \right|_{P_2} \right|_{P_2} \quad (9)$$

$$v_3 = \left| (x_3 - x_1) \left| P_1^{-1} \right|_{P_3} - v_2 \left\| P_2^{-1} \right|_{P_3} \right\|_{P_3} \quad (10)$$

Theorem 1: Required multiplicative inverses in Eq. (10), are $\left| P_1^{-1} \right|_{P_3} = 2^{n-2}$, $\left| P_1^{-1} \right|_{P_2} = -1$ and $\left\| P_2^{-1} \right|_{P_3} \right| = \sum_{i=0}^{(n/2)-1} 2^{2i}$.

Proof.

$$\left|p_1^{-1}\right|_{p_2} = -1 \longrightarrow \left|2^n \times (-1)\right|_{2^n+1} = \left|-2^n\right|_{2^n+1} = 1$$

$$\left|P_1^{-1}\right|_{P_3} = 2^{n-2} \longrightarrow \left|2^n \times 2^{n-2}\right|_{2^{n-1}-1} = \left|2^{2n-2}\right|_{2^{n-1}-1} = 1$$

$$\left|\left|P_2^{-1}\right|_{P_3}\right| = \sum_{i=0}^{(n/2)-1} 2^{2i} \longrightarrow \left|\left(2^n+1\right)\times\left|P_2^{-1}\right|_{P_3}\right|_{2^{n-1}-1} = 1 \longrightarrow \left|3\times\left|P_2^{-1}\right|_{P_3}\right|_{2^{n-1}-1} = 1$$
$$\longrightarrow \left|P_2^{-1}\right|_{P_3} = \left|\frac{1}{3}\right|_{2^{n-1}-1} = \left|\frac{2^n-1}{3}\right|_{2^{n-1}-1} = \sum_{i=0}^{(n/2)-1} 2^{2i}$$

Using theorem 1, v_2 can be calculated as:

$$v_2 = \left|x_1 - x_2\right|_{2^n+1} \tag{11}$$

Based on equation of v_3 in MRC we have:

$$v_3 = \left|\left(x_3 - x_1\right)\left|P_1^{-1}\right|_{P_3} - v_2\left|\left|P_2^{-1}\right|_{P_3}\right|\right|_{P_3} \tag{12}$$

Using theorem 1, v_3 can be calculated as:

$$v_3 = \left|\underbrace{\left(x_3 - x_1\right)2^{n-1}}_{Y} - \underbrace{v_2\left(2^0 + 2^2 + \ldots + 2^{n-2}\right)}_{K}\right|_{2^{n-1}-1} \tag{13}$$

***Lemma* 1**: In modulo 2^n-1, multiplication of *n*-bit residue number *x* by 2^k is equal to *k*-bit circular left shift residue number *x* [26].

***Lemma* 2**: In modulo 2^n-1, the negative of residue number *x* is obtained by one's complement of *x*, where $0 \leq x < 2^n - 1$ [26].

By using Lemma 1 and 2 we have:

$$v_3 = \left|Y + \bar{K}\right|_{2^{n-1}-1} \tag{14}$$

Where

$$\bar{K} = \left|(CLS(\bar{v}_2, 0) + \ldots.. + CLS(\bar{v}_2, n-2))\right|_{2^{n-1}-1} \tag{15}$$

CLS (*k*, *p*) denotes *p*-bit left shift of *k* [27]. For *Y* calculation we have:

$$Y = \left| \left(x_{3,n-2} \ldots . x_{3,0} - \underbrace{00 \ldots .0}_{n-2 \text{ bit}} x_{1,n-1} \ldots x_{1,0} \right) 2^{n-2} \right|_{2^{n-1}-1} \tag{16}$$

Based on Lemma 1 and 2, Y can be rewritten as follows:

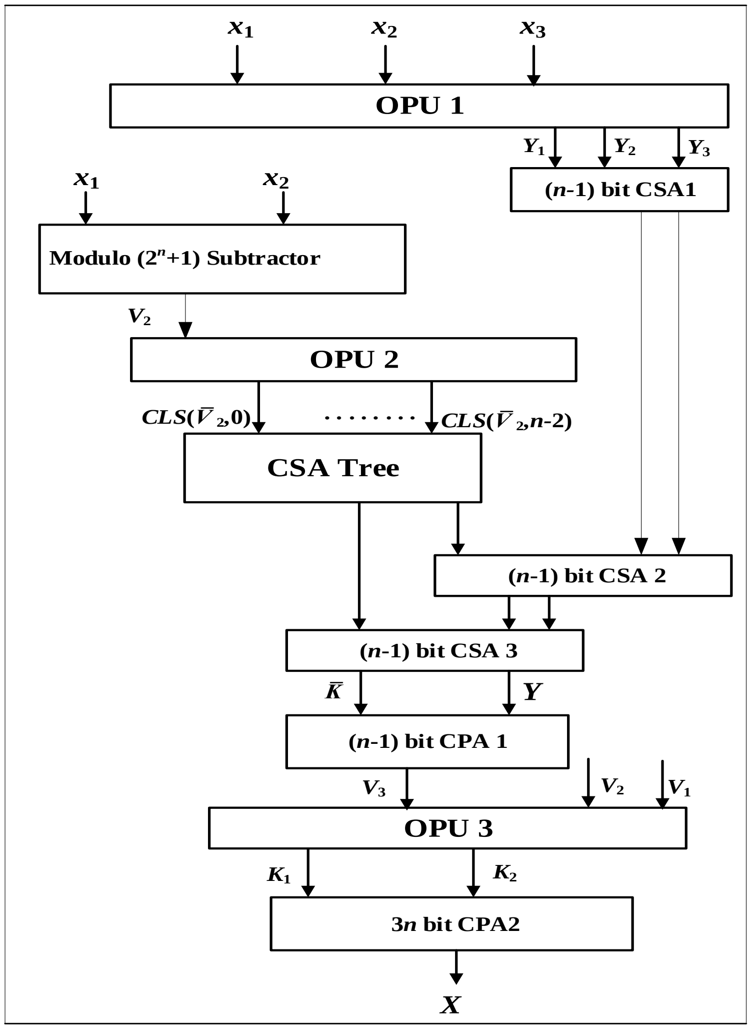

Figure 2. Hardware implementation of X conversion in first basis

$$Y = \left| \left(x_{3,0} x_{3,n-2} \ldots x_{3,1} + \underbrace{\bar{x}_{1,n-1} \ldots \bar{x}_{1,0} \overbrace{11 \ldots 1}^{(n-2)}}_{2(n-1) \text{ bit}} \right) \right|_{2^{n-1}-1} \tag{17}$$

$$Y_1 = \left| \left(x_{3,0} x_{3,n-2} \ldots x_{3,1} \right) \right|_{2^{n-1}-1} \tag{18}$$

$$Y_2 = \left|\left(\overline{x}_{1,n-1}\ldots\overline{x}_{1,1}\right)\right|_{2^{n-1}-1} \tag{19}$$

$$Y_3 = \left|\left(\overline{x}_{1,0}\,\overbrace{11\ldots1}^{(n-2)}\right)\right|_{2^{n-1}-1} \tag{20}$$

Finally after calculation of coefficients {v_1, v_2, v_3} we came back to general equation of X. Thus:

$$X = v_1 + v_2 2^n + v_3 2^{2n} + \underbrace{00\ldots0}_{n\ \text{bit}} v_3 2^n \tag{21}$$

$$X = \underbrace{v_3 v_1}_{(2n-1)\ \text{bit}} + v_2 2^n + \underbrace{0\ldots0}_{2n\ \text{bit}} v_3 2^{2n} \tag{22}$$

$$X = \underbrace{v_3\ \overbrace{v_3 v_1}^{(2n-1)-\text{bit}}}_{k_1} + \underbrace{v_2 2^n}_{k_2} \tag{23}$$

Hardware implementation of X in Eq. (23) is shown in figure 2. Operand preparation unit (OPU) 1, 2 and 3 provides the required negation and shift according to Eq. (18-23). Details report of the required hardware and considered Carry save adder (CSA) with end around carry (EAC) [27], Modulo adder [29][30], subtractor [31] and carry propagate adder (CPA) with EAC [32] which used in this paper are included in section 4 and table 2.

3.2 binary-to-Residue conversion in second bases

Calculation of number X from moduli set of {2^n+3, 2^n-1, 2^n-3} can be done with two MRC levels as shown in figure 3:

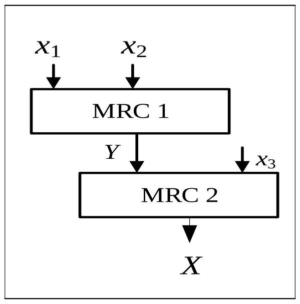

Figure 3. Two MRC levels of conversion in second basis

As mentioned before the coefficients {v_1, v_2, v_3} have to be obtained. Assume $P_1 = 2^n+3$, $P_2 = 2^n-1$, $P_3 = 2^n-3$ and $Z = (x_1, x_2, x_3)$. Thus:

$$v_1 = x_1 \tag{24}$$

$$Y = v_1 + v_2\left(2^n + 3\right) \tag{25}$$

$$v_2 = \left|\left(x_2 - x_1\right)\left|P_1^{-1}\right|_{P_2}\right|_{P_2} \tag{26}$$

***Theorem* 2**: Required multiplicative inverse in Eq. (26), is $\left|P_1^{-1}\right|_{2^n-1} = 2^{n-2}$

Proof.

$$\left|P_1^{-1}\left(2^n+3\right)\right|_{2^n-1} = 1 \longrightarrow \left|P_1^{-1}\right|_{2^n-1} = 2^{n-2}$$

Then Eq. (26) can be rewritten as follow:

$$v_2 = \left|\left(x_2 - x_1\right)2^{n-2}\right|_{2^n-1} \tag{27}$$

By using Lemma 1 and 2 we have:

$$v_2 = \left|(\underbrace{x_{2,n-1}.....x_{2,0}}_{n\ \text{bit}} - \underbrace{00..0}_{n-1\ \text{bit}}\underbrace{x_{n,1}.....x_{0,1}}_{n+1\ \text{bit}}) \times 2^{n-2}\right|_{2^n-1} \tag{28}$$

$$v_2 = \left|\underbrace{x_{2,1}x_{2,0}x_{2,n-1}.....x_{2,2}}_{n\ \text{bit}} - 0\underbrace{x_{n,1}.....x_{0,1}}_{n+1\ \text{bit}}\underbrace{00..0}_{n-1\ \text{bit}}\right|_{2^n-1} \tag{29}$$

$$v_2 = \left|\underbrace{x_{3,1}x_{3,0}x_{3,n-1}.....x_{3,2}}_{n\ \text{bit}} + \underbrace{1\overline{x}_{n,1}....\overline{x}_{2,1}}_{n\ \text{bit}}\underbrace{\overline{x}_{1,1}\overline{x}_{0,1}111....1}_{n\ \text{bit}}\right|_{2^n-1} \tag{30}$$

$$v_2 = \left|\underbrace{x_{3,1}x_{3,0}x_{3,n-1}.....x_{3,2}}_{n\ \text{bit}} + \underbrace{1\overline{x}_{n,1}....\overline{x}_{2,1}}_{n\ \text{bit}} + \underbrace{\overline{x}_{1,1}\overline{x}_{0,1}111....1}_{n\ \text{bit}}\right|_{2^n-1} \tag{31}$$

$$v_2 = \left|K_1 + K_2 + K_3\right|_{2^n-1} \tag{32}$$

Where

$$K_1 = x_{3,1}x_{3,0}x_{3,n-1}.....x_{3,2} \tag{33}$$

$$K_2 = 1\overline{x}_{n,1}....\overline{x}_{2,1} \tag{34}$$

$$K_3 = \overline{x}_{1,1}\overline{x}_{0,1}111....1 \tag{35}$$

At the end of MRC1, *Y* can be computed by:

$$Y = v_1 + \left(2^n + 3\right)v_2 \tag{36}$$

$$Y = v_1 + 2^n v_2 + 3v_2 = v_1v_2 + 3v_2 = \underbrace{v_1v_2}_{v_{21}} + \underbrace{v_20}_{v_{22}} + v_2 \tag{37}$$

Hardware implementation of *Y* in Eq. (37) in first level of MRC conversion is shown in figure 4.

For implementation of second level of MRC conversion as shown in figure 3 we used Eq. (40) as follows:

$$v_1 = Y \tag{38}$$

$$X = v_1 + v_2 P_3 \tag{39}$$

$$v_2 = \left| (x_3 - Y) \left| P_{1,2}{}^{-1} \right|_{P_3} \right|_{P_3} \tag{40}$$

***Theorem* 3**: Required multiplicative in Eq. (40) is $\left| P_{1,2}{}^{-1} \right|_{P_3} = \left| \left((2^n+3)(2^n-1) \right)^{-1} \right|_{(2^n-3)}$

Proof.

$$\left| (2^n+3)(2^n-1) \left| P_{1,2}{}^{-1} \right|_{P_3} \right|_{(2^n-3)} = 1 \longrightarrow \left| 12 \left| P_{1,2}{}^{-1} \right|_{P_3} \right|_{(2^n-3)} = 1$$

$$\longrightarrow \left| 12 \left| P_{1,2}{}^{-1} \right|_{P_3} \right|_{(2^n-3)} = \left| \frac{1}{12} \right|_{(2^n-3)} = \left| \frac{-(2^n-4)}{12} \right|_{(2^n-3)} = -\left(1 + 2^2 + \ldots + 2^{n-4}\right)$$

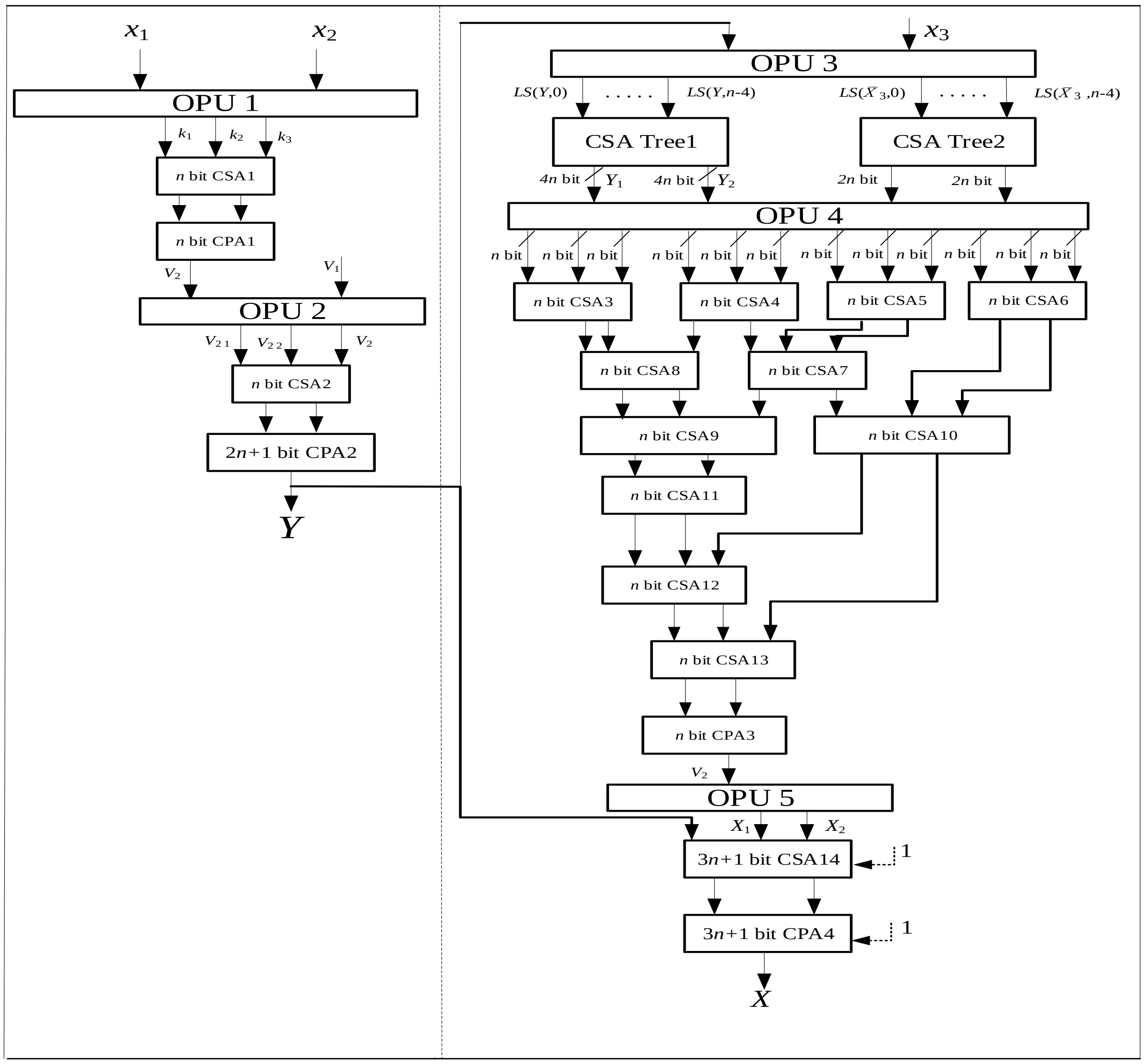

Figure 4. Hardware implementation of *X* conversion in second basis

Based on Theorem 3 and Lemma 1 and 2 Eq. (40) can be written as the following equation:

$$v_2 = \left| \left(Y - x_{3,n-1} \dots x_{3,0} \right) \left(1 + 2^2 + \dots + 2^{n-4} \right) \right|_{(2^n-3)} \tag{41}$$

$$v_2 = \left| \begin{matrix} CLS(Y,0) + \dots\dots + CLS(Y,n-4) + \\ CLS(\overline{x}_3,0) + \dots\dots + CLS(\overline{x}_3,n-4) \end{matrix} \right|_{(2^n-3)} \tag{42}$$

Finally *X* is calculated as the following:

$$\begin{aligned} X &= Y + v_2(2^n-3)(2^n-1) = Y + v_2\left(2^{2n} + 2^{n+1} - 3\right) \\ &= Y + 2^{2n}v_2 + 2^{n+1}v_2 - v_2 0 - v_2 \end{aligned} \tag{43}$$

$$X = Y + 2^{2n}\left(\underbrace{0\dots0}_{2n \text{ bit}} v_2 \right) + 2^{n+1}\left(\underbrace{0\dots0}_{(n+1) \text{ bit}} v_2 \right) + \overline{v}_2 1 + \overline{v}_2 + 2 \tag{44}$$

$$X = Y + \underbrace{v_2 \underbrace{00\dots0}_{n \text{ bit}} \overline{v}_2}_{x_1} + \underbrace{v_2 \overline{v}_2 1}_{x_2} + 2 \tag{45}$$

By calculating X in Eq. (45) for second level of MRC conversion, the residue to binary converter for the moduli set $\{2^n+3, 2^n-1, 2^n-3\}$ is designed completely and shown in figure 4. Details report of the required hardware is included in section 4 and table 3.

3.3 Forward conversion

In order to achieve RNS to RNS conversion according to figure 1, Forward conversion in the considered moduli set is required. Forward converter for the moduli 2^n, 2^n+1, 2^n+3, 2^n-1 and 2^n-3 are designed by researchers [19] [20].

Critical moduli in first and second bases are 2^n+1 and 2^n+3, respectively. Forward conversion for (3*n*)-bit word in moduli 2^n+1 and (3*n*+1)-bit word in critical moduli 2^n+3 in second bases have been calculated and show in table 1 as follow:

Table 1. Delay of critical channel of forward conversion in 2^n+1 and 2^n+3

	Critical moduli	Hardware cost	Delay for conversion
First bases	$\{2^n+1\}$	$(4n+10)t_{FA}$	$(2n+5)t_{FA}$ [19]
Second bases	$\{2^n+3\}$	$(5n+19)t_{FA}$	$(3n+10)t_{FA}$ [20]

4. Performance evaluation

In this section, in order to evaluate the performance of the proposed conversion for comparison with [7], we are going to calculate the hardware cost and conversion delay of two proposed sets $\{2^n, 2^n+1, 2^{n-1}-1\}$ and $\{2^n+3, 2^n-1, 2^n-3\}$ when t_{NOT}, t_{FA}, $t_{XNOR/OR}$ and $t_{XOR/OR}$ are delay represented of NOT gate, a full adder (FA), a pairs of XNOR/OR and a pairs of XOR/OR, respectively. As shown in figure 2 for first set, for calculating Eq. (18-20), (2*n*-2) NOT gates in OPU1 is used, so delay OPU1 equals to t_{NOT}. In Eq. (11) for calculating v_2 is used a modulo (2*n*+1) subtractor which has (2*n*+2)-bit delay [30]. Hence, for obtaining Eq. (14) from Eq. (15) and Eq. (17-20) is used from (*n*+1) NOT gates in OPU2 and a 4- operand modulo $(2^{n-1}-1)$ adder is required which include three (*n*-1)-bit CSA with EAC following by (*n*-1)-bit CPA with EAC. Each CSA with EAC has the delay of FA, and the delay of a

CPA with EAC is twice the delay of a regular CPA. Since Eq. (19) and Eq. (20) have constant value of 1 then FAs in CSA1 reduced to pairs of XNOR/OR gates. Area for CSA Tree is (n-3)(n-1)-bit CSA with EAC tree which arranged in l levels. Delay of circular shifting or bits rearrangement in OPU3 is ignored, since it is a rearrangement of wires. Realization of Eq. (23) required ($3n$)-bit CPA. Finally total delay and hardware cost for first set are calculated in table 2.

Table 2. Hardware and delay specification of reverse converter for the moduli set $\{2^n, 2^n+1, 2^{n-1}-1\}$

Parts	FA	NOT	XNOR/OR pairs	Delay
OPU1		$2n$-2		t_{NOT}
CSA1	1		n-2	t_{FA}
Modulo subtractor [28]	$2n$+2			$(n+1)t_{\mathrm{FA}}$
OPU2		n+1		t_{NOT}
CSA Tree	n^2-$4n$-3			$*l.t_{\mathrm{FA}}$
CSA2	n-1			t_{FA}
CSA3	n-1			t_{FA}
CPA1	n-1			$(2n-2)t_{\mathrm{FA}}$
CPA2	$3n$			$(3\mathrm{n})t_{\mathrm{FA}}$
Total delay	n^2+$4n$-3	$3n$-1	n-2	$(6n+2+l)\ t_{\mathrm{FA}}+2t_{\mathrm{NOT}}$

*Here l is the number of levels of CSA tree with (n-1) input

In second set, required hardware for convert from Residue to binary in Eq. (29) is ($2n$) NOT gates in OPU1 following by n-bit CSA with EAC and n-bit CPA with EAC are used. Since Eq. (34) and Eq. (35) have constant value of 1 then FAs in CSA1 reduced to pairs of XNOR/OR. For calculating the Eq. (37) need to n-bit CSA that following by ($2n$+1)-bit CPA with EAC has been used. In Eq. (41) OPU3 has n NOT gates and two CSA Trees which one of them consists of (n-5)($4n$)-bit CSA with EAC tree area which arranged in k levels for calculation of Y into two ($4n$)-bit parts which has constant value of 0 then FAs in CSA tree reduced to pairs of XOR/AND gates and following by OPU4, eleven n-bit CSA with EAC and n-bit CPA with EAC. As mention before delay of circular shifting or bits rearrangement in OPU4 was ignored, since it is a rearrangement of wires. Another CSA Tree for x_3 calculation has (n-5)($2n$)-bit CSA with EAC tree area which arranged in I levels which has constant value of 1 then FAs in CSA tree reduced to pairs of XNOR/OR gates. Finally for operations of Eq. (45) OPU5 with (n+1) NOT gates, one ($3n$+1)-bit CSA and ($3n$+1)-bit CPA has been applied. Since Eq. (45) has constant values of 0 then full adders in CSA9 reduced to pairs of XOR/AND gates. Total delay for second set is calculated in table 3.

Table 3. Hardware and Delay Specification of Reverse Converter for the moduli set $\{2^n+3, 2^n-1, 2^n-3\}$

Parts	FA	NOT	XOR/AND pairs	XNOR/OR Pairs	Delay
OPU1		$2n$			t_{NOT}
CSA1	1			n-1	t_{FA}
CPA1	n				$(2n)\ t_{FA}$
CSA 2	$2n+1$				t_{FA}
CPA2	$2n+1$				$(2n+1)\ t_{FA}$
OPU3		n			t_{NOT}
CSA Tree1	$4n^2-20n$		$2n^2-7n+3$		$*k.t_{FA}$
CSA Tree2	$2n^2-10n$			n^2-3n	$*I\ t_{FA}$
CSA 3-13	$n\times 11$				$(1\times 11)\ t_{FA}$
CPA3	n				$(2n)\ t_{FA}$
OPU 5		$n+1$			t_{NOT}
CSA9	$2n+1$		n		t_{FA}
CPA3	$3n+1$				$(3n+1)t_{FA}$
Total delay	$6n^2-8n+5$	$4n+1$	$2n^2-6n+3$	n^2-2n-1	$(9n+16+k+I)t_{FA}+3t_{NOT}$

*Here I and k are the number of levels of two CSA trees with (n-3) input

Total delay and hardware cost for RNS to RNS conversion from first to second basis and inverse are computed in table 4.

Table 4. Total delay and hardware cost for two proposed sets

	Hardware cost	Delay
RNS to RNS conversion from first to second basis	$(n^2+8n+7)t_{FA}+(3n-1)t_{NOT}+(n-2)t_{XNOR/OR}$	$(8n+7+l)t_{FA}+2t_{NOT}$
RNS to RNS conversion from second to first basis	$(6n^2-3n+24)t_{FA}+(4n+1)t_{NOT}+(2n^2-6n+3)t_{NOR/AND}+(n^2-2n-1)\ t_{XNOR/OR}$	$(12n+26+k+I)t_{FA}+3t_{NOT}$

For p = 192-b implementation according to NIST report [33], $p = 2^{192}-2^{64}-1$ is considered. By the proposed RNS bases, moduli sets $\{2^{66}, 2^{66}+1, 2^{65}-1\}$ and $\{2^{66}+3, 2^{66}-1, 2^{66}-3\}$ when n = 66 are achieved. In table 5 delay and hardware cost of proposed sets with [7] has been compared. Before beginning, in order to have a better comparison to obtain total area and delay estimation, the unit gate model is considered [14]. According to this model [14], time and area requirements for each of the following components are explained as follows: each two-input monotonic gate (e.g., AND, NAND) counts as one gate in area and delay, an XOR/XNOR gate has two gates in area and delay and a FA counts as seven gates in area and has four gates in delay.

Table 5. Total delay of required RNS to RNS conversion for 192 bit key length

	Delay	Unit gate delay
proposed	(1298) t_{FA}	5192
[7]	(5376) t_{FA}	21504

Table 5 shows the delay comparison with three moduli RNS bases proposed in [7]. As the result shows, noticeable improvement in RNS to RNS conversion in RNS Montgomery multiplication is achieved. Besides, using well formed RNS moduli results arithmetic operation in RNS Montgomery multiplication with higher efficiency [23].

5. CONCLUSION

Montgomery modular multiplication in RNS is an efficient way to achieve higher speed of modular multiplication. Conversion from RNS to RNS is the critical part of this approach. In this paper, efficient RNS bases are selected to achieve high speed operation and the required RNS to RNS conversions are designed in efficient way. The proposed design enjoys efficient bases with suitable arithmetic unit as well as high speed RNS to RNS conversion that is proper for ECC. The results shows noticeable improvements in terms of delay of conversions are achieved compared to state-of-the-art-work in literature.

References

[1] R. L. Rivest, A. Shamir, and L. M. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems,"*Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.

[2] V. S. Miller, "Use of elliptic curves in cryptography," *in Lecture notes in computer sciences; 218 on Advances in cryptology—CRYPTO 85. New York, NY, USA: Springer-Verlag New York, Inc*., 1986, pp. 417–426.

[3] N. Koblitz, "Elliptic Curve Cryptosystem," *J. Cryptology Math. Comp*., vol. 48, pp. 203–209, 1987.

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

[5] 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.

[6] 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.

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

[8] M. Gerami, M. Esmaeildoust, Sh. Rezaie, K. Navi and O. Hashemipour, "Four Moduli RNS Bases for Efficient Design of Modular Multiplication," *Journal of Computations & Modelling*, vol. 1, no. 2, pp 73-96, 2011.

[9] Sh. Rezaie, M. Esmaeildoust, M. Gerami, K. Navi and O. Hashemipour, "High Dynamic Range RNS Bases for Modular Multiplication*," IJCSI International Journal of Computer Science Issues*, Vol. 8, Issue 4, No 1, July 2011.

[10] 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.

[11] A. S. Molahosseini, K. Navi, C. Dadkhah, O. Kavehei and S. Timarchi, "Efficient Reverse Converter Designs for the new 4-Moduli Set $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ and $\{2^n-1, 2^n+1, 2^{2n}, 2^{2n}+1\}$ Based on New CRTs, " *IEEE Transactions on Circuits and Systems-I*, 57(4), (2010), 823-835.

[12] B. Cao, C. Chang and T. Srikanthan, "An Efficient Reverse Converter for the 4-Moduli Set $\{2^n-1, 2^n, 2^n+1, 2^{2n}+1\}$ Based on the New Chinese Remainder Theorem," *IEEE Transactions on Circuits and Systems-I: Fundamental Theory and Applications*, 50(10), (October, 2003), 1296-1303.

[13] P. M. Matutino, R. Chaves, and L. Sousa, "Arithmetic units for RNS moduli $\{2^n - 3\}$ and $\{2^n + 3\}$ operations," in *13th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools*, 2010.

[14] R. Zimmermann, “Efficient vlsi implementation of modulo (2n ± 1) addition and multiplication,” in *14th IEEE Symposium on Computer Arithmetic*, 1999.
[15] S. B. R.A. Patel, M. Benaissa and N. Powell, “Power-delay-area efficient modulo $2^n + 1$ adder architecture for rns,” *Electronics Letters*, 2005.
[16] R. Zimmermann, “Binary adder architectures for cell-based VLSI and their synthesis,” Ph.D. dissertation, Swiss Federal Institute of Technology (ETH) Zurich, Hartung-Gorre Verlag, 1998.
[17] R. Chaves and L. Sousa, “Improved 2n + 1 multipliers,” *Technical Report RT/*14/2003*, INESC-ID,* July
2003.
[18] R. Chaves and L. Sousa, “$2^n + 1$, 2^n+k, $2^n - 1$: A new RNS moduli set extension,” *in EUROMICRO Systems on Digital System Design*, 2004.
[19] P. M. Matutino, R. Chaves, and L. Sousa, “Binary-to-RNS conversion units for moduli {2n±3},” in *14th EUROMICRO Conference on Digital System Design: Architectures, Methods and Tools*, 2011.
[20] F. J. Taylor, “Residue arithmetic: A tutorial with examples,” *Computer*,vol. 17, pp. 50–62, May 1984.
[21] M. Esmaeildoust, A. Kaabi, “High Speed Reverse Converter for the Five Moduli Set {2^n, 2^n-1, 2^n+1, 2^n-3, $^{2n-1}$-1}”, *TJMCS*, pp. 438-450, 2009.
[22] K. Navi, A. S. Molahosseini, and M. Esmaeildoust, “How to teach residue number system to computer scientists and engineers?” *IEEETrans. Edu*., vol. 54, no. 1, pp. 156–163, Feb. 2011.
[23] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Design*.Oxford, U.K.: Oxford Univ. Press, 2000.
[24] B. Cao, C. H. Chang, and T. Srikanthan, “Adder based residue to binary converters for a new balanced 4-moduli set,” in *Proc. 3rd IEEE Symp.Image, Signal Process. Anal.*, 2003, vol. 2, pp. 820–825.
[25] N Keivan, M Esmaeildoust, AS Molahosseini, “A General Reverse Converter Architecture with Low Complexity and High Performance”, *IEICE TRANSACTIONS on Information and Systems* 94 (2), (2011) 264-273.
[26] B. Cao, T. Srikanthan, C.H. Chang, “Efficient reverse converters for the four-moduli sets {2n–1, 2n, 2n+1, 2n+1–1} and {2n–1, 2n, 2n+1, 2n–1–1},” IEE Proc. Comput. Digit.Tech., vol. 152, 687–96, 2005.
[27] K. Hwang, “Computer Arithmetic: Principle, Architecture and Design”, New York: Wiley, 1979.
[28] S.J. Piestrak, “Design of residue generators and multioperand modular adders using carry-save adders”, IEEE Trans. Comput., Vol. 423, pp. 68–77, 1994.
[29] S.J. Piestrak, “A high speed realization of a residue to binary converter”, *IEEE Trans. Circuits Syst*. II, Analog.Digit. Signal Process., Vol. 42, pp. 661–663, Oct. 1995.
[30] D.Younes, P.Steffan, “Novel Modulo 2^n+1 Subtractor and Multiplier ”, *ICONS: The Sixth International Conference on Systems,* pp.36-38, 2011.
[31] Reto Zimmermann, “Lecture notes on Computer Arithmetic: Principles, Architectures and VLSI Design,” Integrated System Laboratory, Swiss Federal Institute of Technology (ETH) Zurich, Mar, 16, 1999.
[32] 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.