Journal of Computations & Modelling, vol.1, no.2, 2011, 73-96
ISSN: 1792-7625 (print), 1792-8850 (online)
International Scientific Press, 2011

Four Moduli RNS Bases for Efficient Design of Modular Multiplication

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

Abstract

Residue Number System provides parallel and fast arithmetic operation by replacing large number computation with small moduli without carry propagation between moduli. RNS can be applied in application like public key cryptography in order to achieve more speed and less power consumption. Modular Multiplication is the main operation in this application. Selecting RNS moduli sets (bases) is the most important part in modular multiplication. In this work RNS bases in order to design efficient modular multiplication is presented. The proposed RNS bases in first basis employs the basis and multiplicative inverses with small hamming weight based on the work reported in literature and in second basis, well formed arithmetic unit RNS basis with efficient forward and reverse converter are employed. The proposed RNS bases are suitable for public key cryptography algorithm especially for Elliptic Curve Cryptography (ECC).

The results show that combination of these RNS basis has achieved noticeable improvement in hardware complexity and also less time delay.

---

[1] Department of Computer, Science and Research Branch, Islamic Azad University, Tehran, Iran, e-mail: {m.gerami, sh.rezaie}@srbiau.ac.ir

[2] Faculty of Electrical and Computer Engineering, Shahid Beheshti University, GC, Tehran, Iran , e-mail: {m_doust, navi, hashemipour}@sbu.ac.ir

Article Info: *Revised* : September 28, 2011. *Published online* : November 30, 2011

Keywords: Residue Number System, Modular Multiplication, Public Key Cryptography

1 Introduction

Residue Number System (RNS) perform addition, subtraction and multiplication in a fast manner because of its carry free nature [1]. Provided parallelism by RNS makes it suitable for application like digital image processing [2], digital signal processing (DSP) [3] and public key cryptography systems [4-7]. RNS consist of three main parts which include forward converter, reverse converter and arithmetic unit [8]. Efficiency of these three parts is depends on the number of moduli and its form. Therefore selecting RNS bases becomes more complicated with growth of application. The most popular RNS basis is $\{2^n-1, 2^n, 2^n+1\}$. This set benefits the balanced moduli and the best forward and reverse converter is reported in [9]. Moduli sets with higher dynamic ranges are reported like, $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$, $\{2^n-1, 2^n, 2^n+1, 2^{n-1}-1\}$ [10], [11] and $\{2^n-1, 2^n, 2^n+1, 2^{n-1}-1, 2^{n+1}-1\}$ [12]. Residue to binary converters for these moduli sets are consists of carry save adder (CSA) tree which results in hardware redundancy.

Other moduli sets like, $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$, $\{2^n-1, 2^n, 2^n+1, 2^{2n}+1\}$ are reported in [13-14] where simple structure of residue to binary converter make them suitable for high dynamic ranges application. Efficient reverse converters for these moduli sets based on new Chinese reminder theorem (CRT) are presented in [13] where the authors proposed memory less converter architecture and adder based. Among the RNS application, public key cryptography like RSA and ECC required more dynamic ranges. Growing up the key length in these algorithms caused to increase hardware complexity and time delay. Modular multiplication is basic operation in cryptosystems. Several algorithms in order to increase the

efficiency of modular multiplication were proposed. The most famous algorithm is Montgomery modular multiplication [15] where its RNS version is also designed for achieve higher performance [5-6]. RNS Montgomery multiplication performs modular multiplication by using auxiliary bases without any division. Selecting the proper RNS bases caused to achieve high performance of converters and arithmetic operation unit. In [16] in designing RNS bases, for first basis modulus in the form of $2^{k_j}-1$ are used and modulus in the form of $2^{k_i}+1$ are selected as second basis where $i, j = 0,1,2,\ldots,m$. The main disadvantage of this work is inefficient multiplicative reverses that caused to decrease efficiency of reverse converter. In [17] RNS bases in the form of $\{2^n-1,\ 2^{2^0 n}+1,\ 2^{2^1 n},...,2^{2^k n}+1\}$ for both bases are reported. The main disadvantages of these bases are unbalanced moduli that caused to decrease the efficiency of arithmetic unit. The best work that was done until now, is presented in [4]. In this work, RNS bases in form of $2^k - c_i$, where $0 \le c_i < 2^{k/2}$ are proposed. The main advantage of this report is to achieve efficient RNS to RNS conversion between two bases and simple multiplicative inverses, which are needed in the process of RNS Montgomery multiplication. In the proposed RNS bases, in one basis the four moduli sets in the form of $2^k - c_i$, where $0 \le c_i < 2^{k/2}$ [4] is applied. In second basis, moduli sets $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ and $\{2^n-1, 2^n, 2^n+1, 2^{2n}+1\}$ [13-14] are used. By employing these moduli sets in RNS Montgomery multiplication without losing arithmetic operation efficiency, faster RNS to RNS conversion is achieved which result to more efficient modular multiplication. Comparing to [4], more efficient arithmetic unit and converters is achieved.

This paper is organized as follows. RNS and modular multiplication background is presented in section 2 and section 3. The proposed RNS bases are presented in section 4. Reductions in the proposed RNS basis which is required in RNS to RNS conversion are detailed in section 5. In section 6, comparison with the other RNS bases are presented and finally section 7 concludes the paper.

2 Overview of RNS

An integer X in Residue Number System (RNS) represented as $(x_1, x_2, \ldots, x_m)$, where $x_i = X \bmod p_i$ that p_i is the modulo of the moduli set $S = \{p_1, p_2, \ldots, p_m\}$. In order to prevent redundancy, RNS moduli must be pair wise relatively prime. Each integer number in the range of 0 to $M-1$, where $M = \prod_{i=1}^{m} p_i$, has unique representation. Two most common algorithms to convert from RNS in to binary are CRT and MRC (Mixed Radix Conversion). These methods are described as fallow. In CRT the RNS integer number convert to its equivalent as below:

$$X = \left| \sum_{i=1}^{m} \left| x_i N_i \right|_{P_i} M_i \right|_M \tag{1}$$

Where $M = \prod_{i=1}^{m} p_i$, $M_i = \frac{M}{p_i}$ and $N_i = \left| M_i^{-1} \right|_{p_i}$ is the multiplicative inverse of M_i in modulo p_i. Another algorithm is MRC that weighted number X from its residues in RNS representation should be calculated as fallow:

$$X = v_n \prod_{i=1}^{v-1} P_i + \cdots + v_3 P_2 P_1 + v_2 P_1 + v_1 \tag{2}$$

Where

$$v_1 = x_1 \tag{3}$$

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

And in the general form is:

$$v_m = \left| (((x_m - v_1) \left| P_1^{-1} \right|_{P_m} - v_2) \left| P_2^{-1} \right|_{P_m} - \cdots - v_{m-1}) \left| P_{m-1}^{-1} \right|_{P_m} \right|_{P_m} \tag{5}$$

$\left| p_i^{-1} \right|_{p_j}$ is the multiplicative inverse of p_i in modulus p_j. Compare to MRC method which is the sequential algorithm, CRT is a parallel one. Depend on the

number and form of the moduli, one of these algorithms is used in design of reverse converter. In some cases for the moduli set with more than four moduli combination of these two algorithms could be applied to achieve high speed of the reverse converter.

3 Overview of RNS Montgomery multiplication

One of the most famous modular multiplication algorithms in Residue Number System is Montgomery modular multiplication. Montgomery algorithm computes the result of $X \times Y \times M^{-1}$ mod *T*, with an auxiliary base without any division. *X* and *Y*, are two large integer number with RNS representation $(x_1, x_2, \ldots, x_m)$ and $(y_1, y_2, \ldots, y_m)$ in first basis and in the second basis we consider X and Y as $(x'_1, x'_2, \ldots, x'_m)$ and $(y'_1, y'_2, \ldots, y'_m)$. M, M' as their dynamic range that $M = p_1 p_2 \cdots p_i$ and $M' = p'_1 p'_2 \cdots p'_i$, where p_i and p'_i are the modulo in moduli sets. Consider T which is $T < M < M'$, so that $\gcd(T, M) = \gcd(T, M') = \gcd(M, M') = 1$. As mentioned before, lack of division operation increase the speed of modular multiplication. In this section Montgomery modular multiplication algorithm that presented in [15] is described. 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 [4], RNS Montgomery multiplication algorithm consist of two conversions, one RNS product on the two basis, one RNS product on the first basis, two RNS products and one addition on the second basis. Besides forward and reverse converters is needed at the end of operations.

RNS Montgomery multiplication Algorithm
$1: D = X \times Y$ ($d_i = \lvert x_i \times y_i \rvert_{m_i}$ in base $\mathrm{B_n}$
$d'_i = \lvert x'_i \times y'_i \rvert_{m'_i}$ in base $\mathrm{B'_n}$)
$2: q_i = \left\lvert d_i \times \lvert T \rvert^{-1}_{m_i} \right\rvert_{m_i}$ in $\mathrm{B_n}$
$3: q_i$ in $\mathrm{B_n} \longrightarrow q'_i$ in $\mathrm{B'_n}$
$4: r' = (d'_i - q'_i \times N'_i) M^{-1}$ in $\mathrm{B'_n}$
$5: r$ in $\mathrm{B_n} \longleftarrow r'$ in $\mathrm{B'_n}$

4 Proposed RNS bases

In this section the proposed four moduli RNS bases is described. In Montgomery modular multiplication, an auxiliary basis to calculate the result is needed. For this basis four moduli sets $S_1 = \{2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1\}$ [13] and $S_2 = \{2^n - 1, 2^n, 2^n + 1, 2^{2n} + 1\}$ [14] with their efficient reverse converters are used. This dynamic RNS basis range is suitable for cryptography algorithms with large key size. For the first basis the RNS basis in the form of $2^k - c_i$, where $0 \le c_i < 2^{k/2}$ reported in [4] are employed. The advantage of this RNS basis is small hamming weight of moduli and multiplicative inverses that leads to implementing the faster modular multiplication. Additions and multiplications in RNS basis $S_1 = \{2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1\}$ [13] and $S_2 = \{2^n - 1, 2^n, 2^n + 1, 2^{2n} + 1\}$ [14], in second basis are done with more speed comparing to first basis. In Table 1, four moduli RNS bases for 160, 192 and 256 bit key length are reported. Notice that because of various moduli length bit in RNS basis, two different symbols for moduli length bit in first and second basis are used (k and n for first and second basis, respectively). Dynamic range of first basis is $4k$ and for second basis is

$5n$. As discuss in the RNS Montgomery multiplication algorithm, $4k$ must be less than $5n$. Therefore For an instance, for 256 key lengths, k considered as 64 bits in order to cover 256 bit dynamic ranges and for second basis n consider as 52 bits. Therefore $4k$ and $5n$ cover the required 256 bit dynamic ranges and also $4k < 5n$ is satisfied. Forward conversion in moduli in the form of $2^k - 1$ and $2^k + 1$ is straight forward and more simple logic circuits with less delay are required comparing to moduli in the form of $2^k - c_i$, where $0 \le c_i < 2^{k/2}$ [4].

Table 1: Proposed RNS bases

Proposed RNS bases	256 key length		192 key length		160 key length	
	First Base	Second Base	First Base	Second Base	First Base	Second Base
4 moduli RNS bases s1	$2^{64}-2^{10}-1$	2^{52}	$2^{48}-2^{10}-1$	2^{39}	$2^{40}-2^{10}-1$	2^{32}
	$2^{64}-2^{16}-1$	$2^{52}-1$	$2^{48}-2^{16}-1$	$2^{39}-1$	$2^{40}-2^{16}-1$	$2^{32}-1$
	$2^{64}-2^{19}-1$	$2^{52}+1$	$2^{48}-2^{19}-1$	$2^{39}+1$	$2^{40}-2^{19}-1$	$2^{32}+1$
	$2^{64}-2^{20}-1$	$2^{105}-1$	$2^{48}-2^{20}-1$	$2^{79}-1$	$2^{40}-2^{20}-1$	$2^{65}-1$
4 moduli RNS bases s2	$2^{64}-2^{8}-1$	2^{52}	$2^{48}-2^{10}-1$	2^{39}	$2^{40}-2^{10}-1$	2^{32}
	$2^{64}-2^{22}-1$	$2^{52}-1$	$2^{48}-2^{16}-1$	$2^{39}-1$	$2^{40}-2^{16}-1$	$2^{32}-1$
	$2^{64}-2^{15}-1$	$2^{52}+1$	$2^{48}-2^{19}-1$	$2^{39}+1$	$2^{40}-2^{19}-1$	$2^{32}+1$
	$2^{64}-2^{16}-1$	$2^{104}+1$	$2^{48}-2^{20}-1$	$2^{78}+1$	$2^{40}-2^{20}-1$	$2^{64}+1$

Required steps for RNS to RNS conversion in RNS Montgomery multiplication is shown in Figure 1. Notice that the complexity of algorithm is in line 3 and 5 of RNS Montgomery multiplication Algorithm [4].

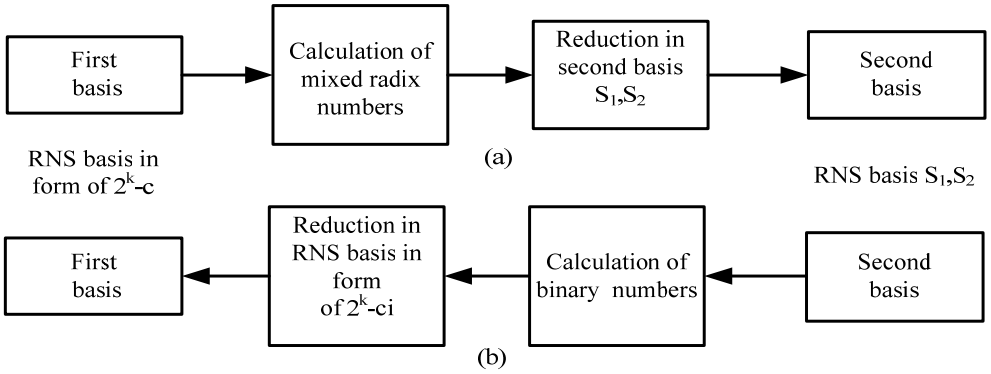

Figure 1: RNS Montgomery Multiplication Algorithm, a) conversion from first basis to the second basis (line 3), b) conversion from second basis to the first basis (line 5)

4.1 RNS to RNS conversion from first to second basis

As shown in Figure 1, RNS to RNS conversion from first basis to second basis consists of two steps: RNS to MRS conversion in first basis and MRS to RNS from first to second basis. Therefore we have:

$$Delay_{RNS-RNS} = Delay_{RNS-MRS} + Delay_{MRS-RNS} \quad (6)$$

Since the first RNS basis in this work in the moduli in the form of $2^k - c_i$ where $0 \le c_i < 2^{k/2}$, efficient RNS to MRS conversion are reported in [4] which is used in this work. Based on [4], cost of RNS to MRS conversion is shown in Eq. 7.

$$Delay_{RNS-MRS} = \left(\sum_{i=1}^{m-1} \max_{j=2,k;i<j} (\omega(m^{-1}{}_{i,j}) + 2\omega(c_j) + 4) \right) kD_{FA} \quad (7)$$

In Eq. 7, $w(k)$ is the hamming weight of k and m is the number of moduli. In order to convert MRS to RNS form first to second basis, the critical moduli in second basis must be considered. In [13] comparisons of critical moduli for different RNS basis are done. Considering new modulo $2^n + 1$ adder proposed in [25], comparison of critical moduli for the proposed RNS bases is shown in table 2. Comparison between modulo $2^{2k+1} - 1$ and $2^k + 1$ shows that the unit gate delay for both is same. Therefore reductions for both moduli are calculated in

section 5. Because of different moduli bit length in the proposed RNS bases (smaller bit length in second RNS basis), the unit gate delays that are shown in Table 2 for second basis is less than first basis. Therefore more speed of arithmetic operation with advantages of efficient converters is achieved.

Table 2: Comparison of critical moduli for different RNS bases

Critical moduli	Unit gate delay
$2^k - 2^t - 1$	2log(k-1) +7
$2^{2k} + 1$	2log(k) +7
$2^{2k+1} - 1$	2log (k)+5
$2^k + 1$	2log (k)+5

According to the Eq. 7, conversion delays from RNS to MRS based on their key length are shown in Table 3.

Table 3: Cost of RNS to MRS conversion

Size of key length	Delay of RNS to MRS conversion for S1	Delay of RNS to MRS conversion for S2
160	52 kD_{FA}	52 kD_{FA}
192	65 kD_{FA}	65 kD_{FA}
256	60 kD_{FA}	39 kD_{FA}

In this work two different basis for auxiliary basis were proposed, so we have two various delay of MRS to RNS conversion. As proofed in section 5, delay of MRS to RNS conversion for critical moduli set S_1 and S_2 are shown in Eq. 8 and Eq. 9 respectively.

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

$$Delay_{MRS-RNS} = \left(\sum_{i=1}^{m-1} (MA(2^{2n} + 1) + 3) \right) D_{FA} \quad (9)$$

MA in this equation denotes the modular addition. By using modulo 2^n-1 and 2^n+1 adder (*MA*) reported in [23] and considering *n*-bit delay of full adder (D_{FA}) and 2*n*-bit delay of *FA* for *MA* (2^n-1) and 2^n+1 respectively, we have:

$$Delay_{MRS-RNS} = (6n+9)D_{FA} \tag{10}$$

$$Delay_{MRS-RNS} = (12n+9)D_{FA} \tag{11}$$

Overall delays of RNS to RNS conversion for the proposed RNS bases are shown in Table 4.

Table 4: Total cost of RNS to RNS conversion from first to second basis

Size of key length	RNS to RNS for S1	RNS to RNS for S2
160	$(52k+6n+9)D_{FA}$	$(52k+12n+9)D_{FA}$
192	$(65k+6n+9)D_{FA}$	$(65k+12n+9)D_{FA}$
256	$(60k+6n+9)D_{FA}$	$(39k+12n+9)D_{FA}$

4.2 RNS to RNS conversion from second basis to the first basis

In order to achieve RNS to RNS conversion from second to first basis according to Figure 1, we have

$$Delay_{RNS-RNS} = Delay_{RNS-weighted} + Delay_{weighted-RNS} \tag{12}$$

Delays of RNS to weighted conversion from second to first basis are shown in Table 5. Weighted to RNS conversion in first basis is proved in section 5, therefore considering critical moduli $2^{2n+1}-1$ in S_1 and $2^{2n}+1$ in S_2 we have:

$$Delay_{RNS-Weighted} = \begin{cases} (12n+5)D_{FA} & for\ s_1 \\ (8n+3)D_{FA} & for\ s_2 \end{cases} \tag{13}$$

$$Delay_{Weighted-RNS} = \left(\sum_{i=1}^{m-1}\left(2\omega(c_j')+2\right)\right)kD_{FA} = 18kD_{FA} \tag{14}$$

Table 5: Shows the area and delay costs of proposed moduli sets

Proposed moduli	Converter	Moduli Set	Area	Delay
S_1:4 moduli proposed 1	[13]	$\{2^n, 2^n-1, 2^n+1, 2^{2n+1}-1\}$	$(8m+2)\ A_{FA} + (m-1)A_{XOR} + (m-1)A_{AND} + (4m+1)A_{OR} + (4m+1)A_{XNOR}\ mA_{2:1MUX} + (7m+1)\ A_{NOT}$	$(12n+5)D_{FA}$
S_2:4 moduli proposed 2	[14]	$\{2^n, 2^n-1, 2^n+1, 2^{2n}+1\}$	$(11m+6)\ A_{FA} + (2m-1)A_{XOR} + (2m-1)A_{AND} + (4m)A_{OR} + (4m)A_{XNOR} + (5m+3)\ A_{NOT}$	$(8n+3)D_{FA}$

Total delay of RNS to RNS from second basis to first basis for 160,192 and 256 bit key length is represented in Table 6.

Table 6: Total delay of conversion from second basis to first basis

Proposed moduli sets	RNS to RNS delay
S_1	$18k+12n+5$
S_2	$18k+8n+3$

Finally we should calculate total delay of RNS Montgomery Multiplication Algorithm. Delay is obtained from summation of RNS to RNS conversion from first basis to second basis and vice versa. Notice that key length must be considered in calculation of overall delay. The results are shown in Table 7.

Table 7: Total cost of RNS to RNS conversions

key length	S_1	S_2
160	$(70k+18n+14)D_{FA}$	$(70k+20n+12)D_{FA}$
192	$(83k+18n+14)D_{FA}$	$(83n+20n+12)D_{FA}$
256	$(78k+18n+14)D_{FA}$	$(57k+20n+12)D_{FA}$

5 Reduction in moduli 2^n+1, $2^{2n}+1$ and $2^{2n+1}-1$

As shown in Figure 1, after calculation of MRS, conversion to RNS in second basis is needed. This section describes MRS to RNS reduction in moduli 2^n+1, $2^{2n+1}-1$ and $2^{2n}+1$. Considering MRC we have:

$$x_i = \left| v_1 + p_1(v_2 + p_2(v_3 + ... + p_{m-1}v_m)...) \right|_{p_j} \tag{15}$$

Where p_i is the moduli in the form $2^k - 2^t - 1$ and p_j is 2^n, 2^n-1, 2^n+1, $2^{2n+1}-1$ and $2^{2n}+1$. Therefore

$$x_i = \left| v_1 + (2^k - 2^{t_1} - 1)(v_2 + (2^k - 2^{t_2} - 1)(\underbrace{v_3 + (2^k - 2^{t_3} - 1)v_4}_{L})) \right|_{p_j} \tag{16}$$

L is considered as basic operation in Eq. 16. Since the second RNS bases are the moduli sets $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ and $\{2^n-1, 2^n, 2^n+1, 2^{2n}+1\}$, MRS to RNS conversion in this moduli sets must be calculated. Two moduli 2^n and 2^n-1 are especial case of $2^k - 2^t - 1$. In [4] reduction in moduli $2^k - 2^t - 1$ is presented. Therefore calculating MRS to RNS conversion in other moduli 2^n+1, $2^{2n+1}-1$ and $2^{2n}+1$ are considered in the following.

5.1 Reduction in modulo 2^n+1

Based on Eq. 16 we have:

$$x_i = \left| v_1 + (2^k - 2^{t_1} - 1)(v_2 + (2^k - 2^{t_2} - 1)(\underbrace{v_3 + (2^k - 2^{t_3} - 1)v_4}_{L})) \right|_{2^n+1} \tag{17}$$

The value L, is the basic operation in Eq. 17. So L should be calculated as fallow:

$$\begin{aligned} L &= \left| v_i + (2^k - 2^{t_i} - 1)v_{i+1} \right|_{2^n+1} = \left| v_i + 2^k \times v_{i+1} - 2^{t_i} \times v_{i+1} - v_{i+1} \right|_{2^n+1} \\ &= \left| v_i + v_{i+1}\underbrace{00...0}_{k} - v_{i+1}\underbrace{00...0}_{t_i} - v_{i+1} \right|_{2^n+1} \end{aligned} \tag{18}$$

In order to compute the result of Eq. 18 in modulo 2^n+1, $(n+1)$ bit separation of values more than $(n+1)$ bit is needed. Therefore

$$L=\left|v^1{}_i+v^3{}_{i+1}+v'_{i+1}-v^4{}_{i+1}-v^5{}_{i+1}-v^6{}_{i+1}-v^7{}_{i+1}\right|_{2^n+1} \tag{19}$$

Where

$$\left.\begin{aligned} v^2{}_i &= \underbrace{00\ldots0}_{2n-k+2} v_{k-1}..v_{n+1} \\ v^2{}_{i+1} &= v_{2n-k+1}..v_0 \underbrace{00\ldots0}_{k-n-1} \end{aligned}\right\} v'_{i+1}=v_{i+1,2n-k+1}..v_{i+1,0}v_{i,k-1}..v_{i,n+1}$$

$$v^1{}_i = v_{i,n}..v_{i,0}$$

$$v^1{}_{i+1} = \underbrace{00\ldots0}_{n+1}$$

$$v^3{}_{i+1} = \underbrace{00\ldots0}_{3n-2k+3} v_{k-1}..v_{2n-k+2}$$

$$v^4{}_{i+1} = v_{n-t}..v_0 \underbrace{00\ldots0}_{t_i}$$

$$v^5{}_{i+1} = \underbrace{00\ldots0}_{2n-k-t_i+2} v_{k-1}..v_{n-t+1}$$

$$v^6{}_{i+1} = v_{i+1,n}..v_{i+1,0}$$

$$v^7{}_{i+1} = \underbrace{00\ldots0}_{2n-k+2} v_{k-1}..v_{n+1}$$

Notice that negative numbers in modulo 2^n+1 can be calculated as:

$$\left|-v\right|_{2^n+1}=\left|2^n+1-v\right|_{2^n+1}=\left|\bar{v}+2\right|_{2^n+1} \tag{20}$$

Thus

$$L=\left|v^1{}_i+v^3{}_{i+1}+v'_{i+1}+\bar{v}^4_{i+1}+\bar{v}^5_{i+1}+\bar{v}^6_{i+1}+\bar{v}^7_{i+1}+8\right|_{2^n+1} \tag{21}$$

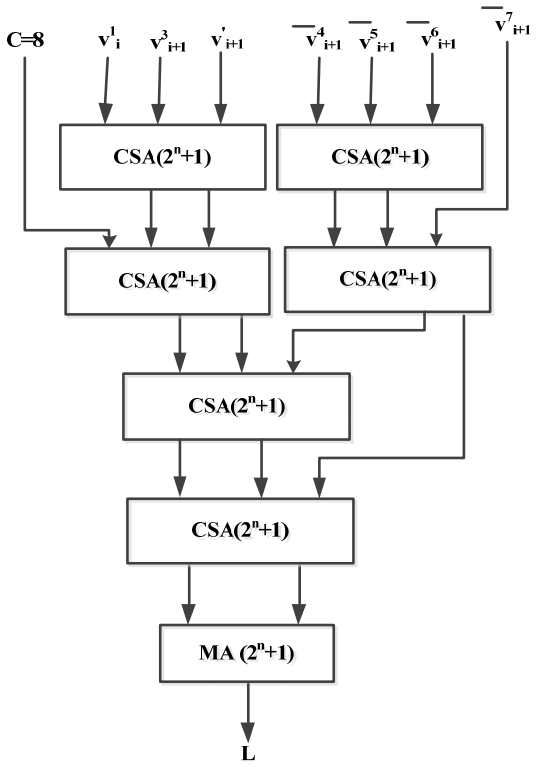

Figure 2: Hardware implementation of value L in modulo 2^n+1

Hardware implementation of Eq. 21 is shown in Figure 2. The result of $H = v_i + (2^k - 2^{t_i} - 1) \times L$ should be calculated in the next step:

$$\begin{aligned} H &= \left| v_i + \left(2^k - 2^{t_i} - 1\right) L \right|_{2^n+1} = \left| v_i + 2^k \times L - 2^{t_i} \times L - L \right|_{2^n+1} \\ &= \left| v_i + L\underbrace{00\ldots0}_{k} - L\underbrace{00\ldots0}_{t_i} - L \right|_{2^n+1} = \left| v^1{}_i + v^2{}_i + \overline{v}'_{i+1} + \overline{v}''_{i+1} + \overline{L} + 6 \right|_{2^n+1} \end{aligned} \quad (22)$$

According to Eq. 22, value H should be separated in four parts as bellow. Therefore CSA inputs in Figure 3, changed to

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

$$\left.\begin{aligned} v^{4}_{i+1} &= L_{n-t_i}...L_{0}\,\underbrace{00...0}_{t_i} \\ v^{5}_{i+1} &= \underbrace{00...0}_{n-t_i+1} L_{n}...L_{n-t+1} \end{aligned}\right\} v''_{i+1} = L_{n-t_i}...L_{0}L_{n}...L_{n-t+1}$$

Hardware implementation of H is shown in Figure 3. According to Figure 2, the delay of conversion from MRS to RNS in worst case can be calculated as:

$$Delay_{MRS-RNS} = \left(\sum_{i=1}^{m-1} \left(MA(2^{n}+1)+4 \right) \right) D_{FA} \qquad (23)$$

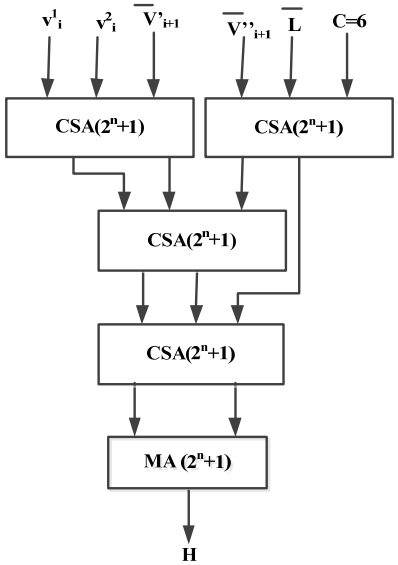

Figure 3: Hardware implementation of value H in modulo $2^{n}+1$

5.2 Reduction in modulo $2^{2n}+1$

According to Eq. 16 we can calculate the reduction in modulo $2^{2n}+1$ as fallow:

$$x_i = \left| v_1 + (2^k - 2^{t_1} \pm 1)(v_2 + (2^k - 2^{t_2} \pm 1)(\underbrace{v_3 + (2^k - 2^{t_3} \pm 1)v_4}_{I})) \right|_{2^{2n}+1} \quad (24)$$

Negative numbers in modulo $2^{2n}+1$ can be calculated as:

$$\left|-v\right|_{2^{2n}+1} = \left|2^{2n}+1-v\right|_{2^{2n}+1} = \left|(2^{2n}-1-v)+2\right|_{2^{2n}+1} = \left|\,\overline{v}+2\right|_{2^{2n}+1} \quad (25)$$

For calculation of I in Eq. 24 we have

$$I = \left|v_i + (2^k - 2^{t_i} - 1)v_{i+1}\right|_{2^{2n}+1} = \left|v_i + 2^k v_{i+1} - 2^{t_i} v_{i+1} - v_{i+1}\right|_{2^{2n}+1} \quad (26)$$

By considering $(2n+1)$ bit separation and considering negative numbers such as modulo $2^{2n}+1$ we have

$$I = \left|v^2_{i+1} + v'_{i+1} + \overline{v}^3_{i+1} + \overline{v}^4_{i+1} + 4\right|_{2^{2n}+1} \quad (27)$$

Where

$$\left.\begin{aligned} v^1{}_i &= \underbrace{00..0}_{2n-k+1} v_{k-1}..v_0 \\ v^1{}_{i+1} &= v_{2n-k}..v_0 \underbrace{00..0}_{k} \end{aligned}\right\} v'_{i+1} = v_{i+1,2n-k}..v_{i+1,0} v_{i,k-1}..v_{i,0}$$

$$v^2{}_{i+1} = \underbrace{00..0}_{3n-2k+2} v_{k-1}..v_{2n-k+1}$$

$$v^3{}_{i+1} = \underbrace{00..0}_{2n-k-t_i+1} v_{i+1} \underbrace{00..0}_{t_i}$$

$$v^4{}_{i+1} = \underbrace{00..0}_{2n-k+1} v_{i+1}$$

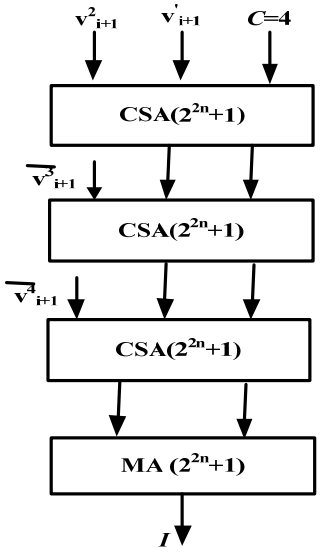

Figure 4: Hardware implementation of I in modulo $2^{2n}+1$

Hardware implementation of I is shown in Figure 4. In this step the result of $F = \nu_i + (2^k - 2^t - 1) \times I$ must be calculated. Therefore inputs in of Figure 4, changes to

$$F = \left| v^{1}_{i} + \overline{v}'_{i+1} + \overline{v}''_{i+1} + \overline{I} + 6 \right|$$

$$v^{1}_{i} = \underbrace{00...0}_{2n-k+1} v_i$$

$$\left.\begin{aligned} v^{1}_{i+1} &= I_{2n-k} ... I_0 \underbrace{00...0}_{k} \\ v^{2}_{i+1} &= \underbrace{00...0}_{2n-k+1} I_{2n} ... I_{2n-k+1} \end{aligned}\right\} v'_{i+1} = I_{2n-k} ... I_0 I_{2n} ... I_{2n-k+1}$$

$$\left.\begin{aligned} v^{3}_{i+1} &= I_{2n-t_i} ... I_0 \underbrace{00...0}_{t_i} \\ v^{4}_{i+1} &= \underbrace{00...0}_{2n-t_i+1} I_{2n} ... I_{2n-t_i+1} \end{aligned}\right\} v''_{i+1} = I_{2n-t_i} ... I_0 I_{2n} ... I_{2n-t_i+1}$$

Hardware implementation of F is shown in Figure 4. According to Figure 4, delay of conversion from MRS to RNS in worst case can be calculated as:

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

5.3 Reduction in modulo 2^{2n+1}-1

According to Eq. 16, reduction in modulo $2^{2n+1}-1$, can be computed as bellow:

$$x_i = \left| v_1 + (2^k - 2^{t_1} - 1)(v_2 + (2^k - 2^{t_2} - 1)(\underbrace{v_3 + (2^k - 2^{t_3} - 1)v_4}_{Z})) \right|_{2^{2n+1}-1} \tag{29}$$

Negative numbers in modulo $2^{2n+1}-1$ should be calculated as:

$$\left|-v\right|_{2^{2n+1}-1} = \left|2^{2n+1} - 1 - v\right|_{2^{2n+1}-1} = \left|\overline{v}\right|_{2^{2n+1}-1} \tag{30}$$

Based on Eq. 29 calculation of Z shown as:

$$Z = \left|v_i + (2^k - 2^{t_i} \pm 1)v_{i+1}\right|_{2^{2n+1}-1} = \left|v_i + 2^k v_{i+1} - 2^{t_i} v_{i+1} - v_{i+1}\right|_{2^{2n+1}-1} \tag{31}$$

$$Z = \left|v^1_i + v^1_{i+1} + \overline{v}^2_{i+1} + \overline{v}^3_{i+1}\right|_{2^{2n+1}-1} \tag{32}$$

Where:

$$v^1_i = \underbrace{00..0}_{2n-k+1} v_i$$

$$v^1_{i+1} = v_{2n-k}...v_0 \underbrace{00..0}_{2n-k+1} v_{k-1} v_{k-2}...v_{2n-k+1}$$

$$v^2_{i+1} = \underbrace{00..0}_{2n-k-t_i+1} v_i \underbrace{00..0}_{t}$$

$$v^3_{i+1} = \underbrace{00..0}_{2n-k+1} v_{i+1}$$

Hardware implementation of Z is represented in Figure 5. In this step the result of $E = v_i + (2^k - 2^t - 1) \times Z$ should be calculated. Therefore inputs in Figure 5, changes to

$$E = \left|v^1_i + v^1_{i+1} + \overline{v}^2_{i+1} + \overline{Z}\right|$$

$$v^1_i = \underbrace{00...0}_{2n-k+1} v_i$$

$$v^1_{i+1} = Z_{2n-k}...Z_0 Z_{2n}...Z_{2n-k+1}$$

$$v^2_{i+1} = Z_{2n-t_i}...Z_0 Z_{2n}...Z_{2n-t_i+1}$$

Hardware implementation of E is shown in Figure 5. Therefore delay of conversion from MRS to RNS in worst case can be calculated as:

$$Delay_{MRS-RNS} = \left(\sum_{i=1}^{m-1} \left(MA\left(2^{2n+1}-1\right)+2 \right) \right) D_{FA} \tag{33}$$

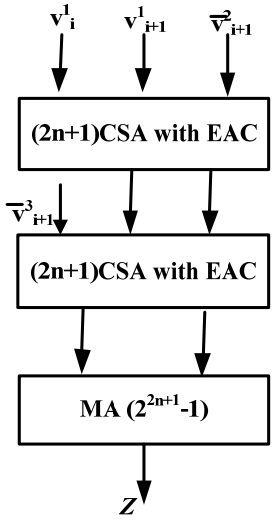

Figure 5: Hardware implementation of Z in modulo $2^{2n+1}-1$

5.4 Reduction in modulo 2^k-2^t-1

Reduction of weighted number to its RNS representation in moduli 2^k-2^t-1 is proved in this section. Weighted to RNS conversion in moduli in the form of 2^k-2^t-1 can be done as

$$\begin{aligned} |X|_{2^k-2^{ti}-1} &= \left| \sum_{i=0}^{4} 2^{ik} x_i = x_0 + 2^k x_1 + 2^{2k} x_2 + 2^{3k} x_3 + 2^{4k} x_4 \right|_{2^k-2^{ti}-1} \\ &= \left| 2^k (2^k (2^k (\underbrace{2^k x_4 + x_3}_{T}) + x_2) + x_1) + x_0 \right|_{2^k-2^{ti}-1} \end{aligned} \tag{34}$$

Cost of conversion of MRS to RNS in [4] is reported as

$$Delay_{MRS-RNS} = \left(\sum_{i=1}^{m-1} \max_{j=1,m} (\omega(c_i) + 2\omega(c'_j) + 2) \right) kD_{FA} \quad (35)$$

According to the Eq. 34 hamming weight of 2^k is equal to one. Therefore T in Eq. 34 could be calculated with one shift and addition. In Eq. 35, $\omega(c_i)$ is the hamming weight of c_i which is equal to zero, therefore cost of weighted to RNS conversion is

$$Delay_{weighted-RNS} = \left(\sum_{i=1}^{m-1} 2\omega(c'_j) + 2) \right) kD_{FA} \quad (36)$$

6 Comparison

As mentioned in [13] and [24], the unit gate delays of addition reported in [18] and [19] are, $2\log_2^{(k)} + 6$ and $2\log_2^{(k-1)} + 7$, respectively. In this work by using the unit gate delay of modular adder [20] and [18], the delay of modulo $2^{2n+1}-1$ and $2^{2n}+1$ are obtained as $2\log_2^{(n+0.5)} + 5$ and $2\log_2^{(n)} + 5$ [25]. It is obvious from Table 2 that the moduli set $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$, is faster than the moduli set $\{2^n-1, 2^n, 2^n+1, 2^{2n}+1\}$, moreover moduli sets $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ and $\{2^n-1, 2^n, 2^n+1, 2^{2n}+1\}$ can result in better tradeoffs between the RNS arithmetic unit delay and reverse converter performance.

In moduli set $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$, modulo $2^{2n+1}-1$ is the critical moduli and also in moduli set $\{2^n-1, 2^n, 2^n+1, 2^{2n}+1\}$, modulo $2^{2n}+1$ has the worst delay in addition and therefore in MRS to RNS conversion. Considering fast implementation of modular adder such as parallel prefix adders [21-22] modulo 2^k-1 implementation in much faster than modulo 2^k-2^{ti}-1 adder. Therefore unbalanced modulus $2^{2n+1}-1$ and $2^{2n}+1$ does not decrease the efficiency of proposed RNS bases. It is worth mentioning that, the unbalanced RNS basis

results in efficient residue to binary conversion in second basis. The first basis is the four moduli RNS basis in the form of $2^k - c_i$ where $0 \le c_i < 2^{k/2}$ as proposed in [4]. Table 8, shows the comparison between four moduli sets which are proposed in this work and RNS basis that reported in [4]. Notice that this comparison also depends on dynamic range, on the other hand based on key length of cryptography, different cost of RNS to RNS conversion are shown in Table 8. As shown in this table, noticeable improvement in speed of the RNS to RNS conversion compared to [4] is achieved. Note that in [4] four moduli RNS comparison are propose d for 256 bit key length, therefore for fair comparison, cost of RNS to RNS conversion for 256 bit key size are done in Table 8. Since one of important part of RNS Montgomery multiplication is the RNS to RNS conversion, therefore RNS Montgomery multiplication could be executed in faster design by the proposed bases.

Table 8: Comparison RNS to RNS conversion for key length 256 bit

Moduli sets	Cost of RNS to RNS Conv. (D_{FA})	Improvement
Proposed (S_1)	5942	%12
Proposed (S_2)	4700	%30
[4]	6784	---

7 Conclusion

With growing up key length in cryptography systems, needs for speed, security and less hardware redundancy are sensible. Therefore this paper proposed RNS bases to satisfy the key length of cryptosystem and also have proper speed and efficient hardware implementation. Four moduli RNS bases for public key cryptography algorithm such as ECC are proposed. Whit using these RNS bases, less complexity in hardware and more speed in arithmetic unit compare to the best

work in literature has been achieved. The advantage of moduli sets which are employed for second basis are efficient forward and reverse converter and efficient arithmetic unit. Therefore noticeable improvement in time delay with 160,192 and 256 bits key length that are used in cryptography algorithm is achieved.

References

[1] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Design*, Oxford University Press, 2000.

[2] W. Wei *et al.*, RNS application for digital image processing, in *Proc. 4th IEEE Int. Workshop Syst.-on-Chip Real-Time Appl.*, (2004), 77-80.

[3] 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).

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

[5] J.C. Bajard and L. Imbert, A Full RNS Implementation of RSA, *IEEE Transactions on Computers*, **53**(6), (2004), 769-774.

[6] J. Bajard, L. Didier and P. Kornerup, An RNS Montgomery's Modular Multiplication Algorithm, *IEEE Trans. Computers*, **47**(2), (February, 1998), 167-178.

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

[8] K. Navi, A.S. Molahosseini and M. Esmaeildoust, How to Teach Residue Number System to Computer Scientists and Engineers, *IEEE Transactions on Education*, **54**(1), (2011), 156-163.

[9] 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*, **50**(7), (2002), 1772-1779.

[10] P.V.A. Mohan and A.B. Premkumar, RNS-to-Binary Converters for Two Four-Moduli Set $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ and $\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$, *IEEE Transactions on Circuits and Systems-I*, **54**(6), (2007), 1245-1254.

[11] B. Cao, T. Srikanthan and C.H. Chang, Efficient reverse converters for the four-moduli sets $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ and $\{2^n-1, 2^n, 2^n+1, 2^{n-1}-1\}$, *IEEE Proc. Comput. Digit. Tech.*, **152**, (2005), 687-696.

[12] B. Cao, C.H. Chang and T. Srikanthan, A Residue-to-Binary Converter for a New Five-Moduli Set, *IEEE Transactions on Circuits and Systems-I*, **54**(5), (2007), 1041-1049.

[13] A.S. Molahosseini, K. Navi, C. Dadkhah, O. Kavehei and S. Timarchi, Efficient Reverse Converter Designs for the new 4-Moduli Set $\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ 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.

[14] 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.

[15] P. Montgomery, Modular Multiplication without Trial Division, *Mathematics of Computation*, **44**(170), (April, 1985), 519-521.

[16] K. Manochehri Kalantari, S. Pour Mozafari and B. Sadeghiyan, Improved RNS for RSA Hardware Implementation, **2**(2&4- b), (2004), 31-39.

[17] F. Pourbigharaz and H. M. Yassine, A signed digit architecture for residue to binary transformation, *IEEE Transactions on Computers*, **46**(10), (1997), 1146-1150.

[18] Leonel Sousa and Ricardo Chaves, A Universal Architecture for Designing Efficient Modulo 2^n+1 Multipliers, *IEEE Transactions on Circuits and Systems-I*, **52**(6), (June, 2005).

[19] H.T. Vergos and C. Efstathiou, Design of efficient modulo 2^n+1 multipliers, *IET Comput. Digit. Tech.* **1**(1), (2007), 49-57.

[20] M. Esmaeildoust, K. Navi and M. R. Taheri, High speed reverse converter for new five-moduli set $\{2^n, 2^{2n+1}-1, 2^{n/2}-1, 2^{n/2}+1, 2^n+1\}$, *IEICE Electron. Express*, **7**(3), (2010), 118-125.

[21] R.A. Patel, M. Benaissa and S. Boussakta, Efficient new approach for modulo 2^n-1 addition in RNS, *IEEE Proc.-Comput. Digit. Tech.*, **153**(6), (2006), 399-405.

[22] C. Efstathiou, H.T. Vergos and D. Nikolos, Modified Booth Modulo 2^n-1 Multipliers, *IEEE Transactions on Computers*, **53**(3), (2004), 370-374.

[23] M.A. Bayoumi, G.A. Jullien and W.C. Miller, A VLSI implementation of residue adder, *IEEE Transactions on Circuits and Systems*, **34**(3), (1987), 284-288.

[24] K. Navi, M. Esmaeildoust and A.S. Molahosseini, A General Reverse Converter Architecture with Low Complexity and High Performance, *IEICE TRANSACTIONS on Information and Systems*, **E94-D**(2), (2011), 264-273.

[25] G. Jaberipur and B. Parhami, Unified Approach to the Design of Modulo-$(2^n \pm 1)$ Adders Based on Signed-LSB Representation of Residues, *19th IEEE International Symposium on Computer Arithmetic*, (2009), 57-64.