On the Design of RNS Bases for Modular Multiplication

Mohammad Esmaeildoust[1], Shirin Rezaei[2], Marzieh Gerami[2], and Keivan Navi[1]
(Corresponding author: Mohammad Esmaeildoust)

Faculty of Electrical and Computer Engineering, Shahid Beheshti University, GC, Tehran, Iran[1]
Microelectronic Laboratory of Shahid Beheshti University, Tehran, Iran[2]
{m_doust, navi}@sbu.ac.ir, {sh.rezaie, m.gerami}@srbiau.ac.ir

(Received Mar. 8, 2011; revised and accepted Jan. 27, 2012)

Abstract

Modular multiplication is the main and basic operation in public key cryptography algorithms like Elliptic Curve Cryptography (ECC). By using Residue Number System (RNS) in these algorithms, large number computation is replaced by computation over the small moduli without carry propagation between moduli. Moduli selection has an important role in the efficiency of modular multiplication. Therefore in this work the moduli sets for modular multiplication with aims of increasing the efficiency of arithmetic operation and speeding up the RNS to RNS conversions are presented. The proposed moduli sets are suitable for ECC. The results show a noticeable improvement in speed comparing to the state-of-the-art.

Keywords: Computer arithmetic, Modular multiplication, Residue Number System, Montgomery Multiplication.

1 Introduction

Carry free nature of residue number system makes it suitable for application which uses operations such as addition, subtraction and multiplication [29]. Public key cryptography [1, 2, 3, 4, 5, 6, 7, 14, 16, 17, 18, 21, 22, 26, 30, 31, 32, 33, 36], and digital signal processing (DSP) [10] are such applications. RNS mainly consists of three main parts which include arithmetic operation, forward and reverse conversion [28]. Moduli selection is one of the most important parts of RNS which can affect the efficiency of these three parts. Different moduli sets are presented for RNS. One of the most well known moduli set is {2^k, 2^k-1, 2^k+1} which is a balanced and well formed moduli set and the best reverse converter for this moduli set is reported in [38]. Forward conversion for the moduli 2^k, 2^k-1 and 2^k+1 can be done with simple process [11]. With growth of application especially for public key cryptography algorithms, more parallelism and dynamic ranges are needed. Therefore moduli sets with four and five moduli such as {2^k-1, 2^k, 2^k+1, 2^{k+1}-1}, {2^k-1, 2^k, 2^k+1, 2^{2k+1}-1}, {2^k-1, 2^k+1, 2^{2k}, 2^{2k}+1} and {2^k-1, 2^k, 2^k+1, 2^{k+1}-1, 2^{k-1}-1} are reported [8, 9, 23, 24]. The main part of public key cryptography algorithms like RSA [3, 4] and ECC [31, 32] is the modular multiplication. Montgomery modular multiplication [25] needs auxiliary moduli sets (basis) to perform modular multiplication without division. Selecting an efficient RNS bases can dramatically increase the performance of the modular multiplication. In [30] moduli in the form of 2^k±1 are considered and moduli set {2^m-1,$2^{2^0 m}$+1,$2^{2^1 m}$, ,$2^{2^k m}$+1} are proposed. The main disadvantage of this work is unbalanced moduli which yields inefficient arithmetic operation. In [20] for first and second basis moduli in the form of 2^{k_i}-1 and 2^{k_j}+1 where i, j =1,..., m are proposed, respectively. The same problem of unbalanced moduli and inefficiency of arithmetic operation still exist in this work. The other problem is the inefficient multiplicative inverses which lead to an increase in the delay of reverse converter. In [31] RNS implementation of multiplication for ECC is presented. Efficiency of arithmetic operation and conversion from weighted number to RNS and vice versa are not considered in their report. In [4] RNS bases in the form of 2^k-c$_i$ where $0 \leq$ c$_i$ $< 2^{k/2}$ are presented. In this approach exhaustive search for different sets are done and moduli sets with hamming weight equal to three are reported. Advantage of small hamming weight is in replacing multiplication with additions. Simple multiplicative inverses are another advantage of this work which relies on efficient conversion from RNS to its equivalent weighted number. This report provides the best solution to date considering the efficiency of arithmetic operation and also reverse and forward conversion. Since conversion from one basis to another is needed in modular multiplication process, efficiency of the moduli sets like {2^k-1, 2^k, 2^k+1, 2^{k-1}-1} [9, 23] and {2^k-1, 2^k, 2^k+1, 2^{k+1}-1, 2^{k-1}-1} [8] in the point of view of arithmetic unit, forward and reverse converter are not employed in [4]. Therefore in this paper efficient RNS bases for public key cryptography and specifically for ECC are presented. The main purpose is to increase the efficiency of arithmetic operation, and conversion from one basis to another. Therefore modular multiplication can be done with less delay. Designing the RNS bases with four and five moduli set are performed. In designing the first basis, exhaustive search are performed for moduli set in the form of 2^k-c$_i$ where $0 < c_i < 2^{k/2}$ with hamming weight equal to three. In designing the auxiliary basis, moduli sets $\{2^k\text{-}1, 2^k, 2^k\text{+}1, 2^{k-1}\text{-}1\}$ [9, 23] and $\{2^k\text{-}1, 2^k, 2^k\text{+}1, 2^{k+1}\text{-}1, 2^{k-1}\text{-}1\}$ [8] are used. With this approach in one of the basis, the advantages of arithmetic unit and efficient design of forward and reverse converter can be employed. Results shows a noticeable improvement of modular multiplication comparing to method proposed in [4].

This paper is organized as follows. First, RNS and modular multiplication background is presented in Section 2. The proposed RNS bases are presented in Section 3. Section 4 discusses the performance evaluation and comparison with the other RNS bases and finally Section 5 concludes the paper.

2 Related Background

2.1 RNS Background

An integer number X in residue number system is represented as X=(x_1, x_2,..., x_m) where x_i = (X mod P_i) and in order to avoid redundancy, RNS moduli must be pair wise relatively prime. An integer number X has unique representation between 0 and M-1 where M= $P_1 \times P_2 \times ... \times P_m$ is the dynamic range of the system. Residue to binary conversion can be performed by using two conventional algorithms, Mixed Radix Conversion (MRC) and Chinese Reminder Theorem (CRT). By MRC, the number X can be calculated from its residues by:

$$X = v_m \prod_{i=1}^{m-1} P_i + ... + v_3 P_2 P_1 + v_2 P_1 + v_1 \quad (1)$$

Where

$$v_1 = x_1 \quad (2)$$

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

And in the general manner:

$$v_m = \left| \begin{array}{c} (((x_m - v_1) \left| P_1^{-1} \right|_{P_m} - v_2) \left| P_2^{-1} \right|_{P_m} - \\ \cdots - v_{m-1}) \left| P_{m-1}^{-1} \right|_{P_m} \end{array} \right|_{P_m} \quad (4)$$

$| {P_i}^{-1} |_{P_j}$ is the multiplicative inverse of P_i in modulus P_j. Horner's scheme for MRC representation is [4]

$$X = v_1 + P_1(v_2 + P_2(v_3 + \cdots + P_{m-1} v_m) \cdots) \quad (5)$$

The v_i's in Equations (1) and (5) represents the intermediate system which called Mixed Radix System (MRS). By CRT, residue numbers can be converted into their equivalent in binary form as follow:

$$X = \left| \sum_{i=1}^{m} |x_i N_i|_{P_i} M_i \right|_M \quad (6)$$

Where $M = \prod_{i=1}^{m} P_i$, $M_i = \frac{M}{P_i}$ and $N_i = |\ {M_i}^{-1}|_{P_i}$ is the multiplicative inverse of M_i in modulo P_i. The calculation of X using CRT is implemented in parallel channel followed by a modulus M adder. This moduli adder is very large and this drawback leads to a very difficult VLSI design. On the other hand, MRC is a sequential algorithm that is not suitable for moduli set with more than four moduli. In some cases for the moduli set with more than four moduli, the combination of these two algorithms could be applied in order to achieve higher speed of reverse converter. Such implementations are reported in [9, 15, 23]. We present two lemmas which are employed in forward and reverse conversion in this paper.

Lemma 1 *The residue of a negative residue number $(-v)$ in modulo (2^k - 1) is the one's complement of v, where $0 \leq v < 2^k$ - 1 [27].*

Lemma 2 *The multiplication of a residue number v by 2^P in modulo (2^k - 1) is carried out by P bit circular left shift, where P is a natural number [27].*

2.2 Modular Multiplication in RNS

In this section, Montgomery modular multiplication which is introduced in [25] will be discussed. Consider X and Y as two large numbers (public key cryptography nature), first we consider RNS with two bases $B_m = \{P_1,..., P_m\}$ and $B'_m = \{P'_1,..., P'_m\}$ with $M = \prod_{i=1}^{m} P_i$ and $M' = \prod_{i=1}^{m} P'_i$, respectively. RNS representation of A and B in the first basis is equal to (a_1,..., a_m) and (b_1,..., b_m). In the auxiliary basis B'_m, RNS representation is equal to (a'_1,..., a'_m) and (b'_1,...,b'_m). We consider T such that $T < M < M'$ and gcd (T, M) = gcd (T, M') = gcd (M, M') = 1. The modular multiplication computing $X \times Y \times M^{-1}$ (mod T) is as follows:

```
RNS Montgomery Multiplication:
1: D = $X \times Y$
($d_i = |x_i \times y_i|_{P_i}$ in Basis $B_m$ and $d'_i = |x'_i \times y'_i|_{P'_i}$ in basis $B'_m$)
2: $q_i = \left| d_i \times |T|_{P_i}^{-1} \right|_{P_i}$ in B$_m$
3: $q_i$ in $B_m \rightarrow q'_i$ in $B'_m$
      RNS to RNS conversion from first to auxiliary basis
4: $r' = (d'_i - q'_i \times N'_i)M^{-1}$ in $B'$m
5: $r$ in $B_m \leftarrow r'$ in $B'_m$
      RNS to RNS conversion from auxiliary to first basis
```

The basic operation for RNS Montgomery multiplication consists of two conversions, one RNS product on the two bases, one RNS product on the first basis, two RNS products and one addition on the second basis [4]. Therefore intelligent selection of RNS moduli sets can yield to speed up the modular addition and multiplication in each moduli. For an instance, modular multiplication and addition in moduli 2^k-1 and 2^k+1 can be implemented with more efficient hardware architecture and speed up comparing to general moduli [13, 34, 37]. By this point of view, it is desired that the reverse and forward conversion can be done with more speed as much as possible. With targeting these aims in the following the proposed RNS bases will be discussed.

3 Moduli Selection for Modular Multiplication

Selecting the efficient RNS bases is crucial for achieving efficient modular multiplication. As mentioned before, in [4] the RNS bases in the form of 2^k-c_i where $0 \leq c_i < 2^{k/2}$ is presented. The aim of this selection is that reduction in modulo 2^k-c_i is easy and efficient arithmetic operation will be achieved by this form of moduli. In their approach an exhaustive search for moduli set with hamming weight equal to three with simple multiplicative inverses is carried out. The aim of this work is that with small hamming weight for multiplicative inverses, multiplication can be replaced by simple shift and some additions. Therefore modular multiplication can be done in a fast and efficient way. Reduction method in [4] in moduli 2^k-c_i for the value less than 2^{2k} is reported as $2\omega(c_i)$+2 addition of k bit words where $\omega(c_i)$ is the hamming weight of c_i. As mentioned before, using moduli sets with efficient arithmetic units and reverse converters has significant importance in modular multiplication. Therefore in proposing RNS bases, the sets like $\{2^k, 2^{k+1}-1, 2^k-1, 2^k+1\}$ [9, 23] and $\{2^k, 2^{k+1}-1, 2^k-1, 2^k+1, 2^{k-1}-1\}$ [8] are employed for second basis. For first basis like the method proposed in [4] the exhaustive search for the moduli in the form of 2^k-c_i where $0 < c_i < 2^{k/2}$ are carried out. Based on multiplicative inverses reported in [19], the first basis enjoys simple multiplicative inverses which lead to replacing the multiplication needed in MRS by simple shift and some addition. Despite all the advantages of the first basis which are selected from moduli in the form of 2^k-c_i, the second basis enjoys more efficient arithmetic operation, forward and reverse conversion. Figure 1 shows the process of conversion from one basis to another which is needed in line 3 and 5 of modular multiplication described in Section 2.2.

3.1 RNS to RNS Conversion from first to second Basis

In this work c_i's are considered as $2^{t_i} \pm 1$ where $0 < t_i < k/2$. Conversion delay from first to second basis which is shown in Figure 1 can be calculated as

$$Delay_{RNS\text{–}RNS} = Delay_{RNS\text{–}MRS} + Delay_{MRS\text{–}RNS} \quad (7)$$

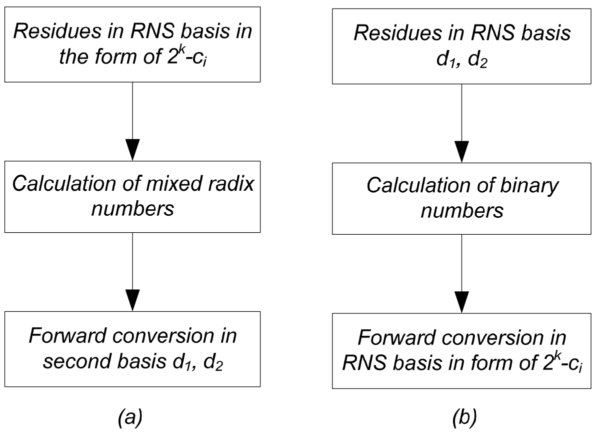
Residues in RNS basis in the form of 2^k-c_i
Calculation of mixed radix numbers
Forward conversion in second basis d_1, d_2
(a)
Residues in RNS basis d_1, d_2
Calculation of binary numbers
Forward conversion in RNS basis in form of 2^k-c_i
(b)

Figure 1: RNS to RNS conversion, (a) conversion from first to second basis (B_m to B'_m), (b) conversion from second to first basis (B'_m to B_m), d_1 and d_2 are the moduli sets $\{2^k, 2^{k+1}-1, 2^k-1, 2^k+1\}$ and $\{2^k, 2^{k+1}-1, 2^k-1, 2^k+1, 2^{k-1}-1\}$, respectively.

Where

$$Delay_{RNS\text{–}MRS} = \left(\sum_{i=1}^{m-1} \max_{j=2,m;i<j} (\omega(|P_i^{-1}|_{P_j}) + 2\omega(c_j) + 4)\right) kD_{FA}$$

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

Where m is the number of moduli, $\omega(k)$ is the Hamming weight of k and MA(2^k+1) is the Modulo(2^k+1)adder. Based on [4] delay of RNS to MRS in first basis is

$$Delay_{RNS-MRS} = \left(\sum_{i=1}^{m-1} \max_{j=2,m;i<j} (\omega(|P_i^{-1}|_{P_j}) + 2\omega(c_j) + 4)\right) nD_{FA} \quad (8)$$

Equation (8) is the delay for calculation of mixed radix number v_i's in Equation (1). In order to calculate the delay of conversion from MRS to RNS, delay of slowest moduli is considered. As proved in Section 3.3, the worst case delay of MRS to RNS conversion in critical moduli 2^k+1 when all the moduli are in the form of 2^k-2^{t_i}-1 is

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

Different implementation for MA(2^k+1) with varied area and delay are reported, such as using carry propagate adder (CPA) or parallel prefix adder [39]. Therefore in MA(2^k+1) needed in Equation (9) one of these implementation can be used.

3.2 RNS to RNS Conversion from second to the first Basis

As depicted in Figure 1, conversion from second to first basis is needed in RNS Montgomery modular multiplication. The second basis is the four and five moduli set

$\{2^k, 2^{k+1}\text{-}1, 2^k\text{-}1, 2^k{+}1\}$ [9, 23] and $\{2^k, 2^{k+1}\text{-}1, 2^k\text{-}1, 2^k{+}1, 2^{k-1}\text{-}1\}$ [8]. Delay and area of conversion from RNS to weighted number using the assumption reported in [9, 23, 24] are shown in Table 1. Note that several implementations with different delay and area for the reverse conversion are proposed for four moduli set $\{2^k, 2^{k+1}\text{-}1, 2^k\text{-}1, 2^k{+}1\}$, but for simplicity, one design is included in Table 1. RNS to RNS conversion from second to first basis can be calculated as

$$\begin{aligned} &Delay_{RNS\text{–}RNS} = \\ &\quad Delay_{RNS\text{–}Weighted} + Delay_{Weighted\text{–}RNS} \end{aligned} \tag{10}$$

Where

$$\begin{aligned} &Delay_{RNS-Weighted} = \begin{cases} (23k+12)/2D_{\text{FA}} & \text{d}_1 \\ (18k+L+7)D_{\text{FA}} & \text{d}_2 \end{cases} \\ &Delay_{Weighted\text{–}RNS} = \left(\sum_{i=1}^{m-1}(2\omega(c'_j)+2)\right)kD_{FA} \end{aligned} \tag{11}$$

The delays of RNS to weighted number are included in Table 1 and conversion delay of weighted number to its equivalent residue number in first basis are proven in Section 3.4. It is worth mentioning that by using these four and five moduli sets, arithmetic operation in one basis can be done in a fast and efficient way. Therefore both RNS to RNS conversion and arithmetic operation are improved and the Montgomery modular multiplication can be done more rapidly. In Table 2, four and five moduli bases for k equal to 64 is reported.

3.3 Reduction of Mixed Radix Numbers in Moduli 2^k+1, 2^k-1, 2^{k-1}-1 and 2^{k+1}-1

Reduction of mixed radix numbers v_i's into residues in moduli 2^k, 2^k-1, 2^k+1, 2^{k+1}-1 and 2^{k-1}-1 are needed in the process of RNS to RNS conversion from first to second basis. Based on Equation (5) we have

$$x_i = |v_1 + P_1(v_2 + P_2(v_3 + \cdots + P_{m-1}v_m)\cdots)|_{P_j} \tag{12}$$

Where P_1, P_2,..., P_{m-1} are the moduli in the form of 2^k-$2^{t_i}\pm$ 1 and P_j is equal to 2^k, 2^k-1, 2^k+1, 2^{k+1}-1 and 2^{k-1}-1. Therefore Equation (12) can be rewritten as

$$x_i = \left| \begin{aligned} &v_1 + (2^k - 2^{t_1} \pm 1)(v_2 + \\ &(2^k - 2^{t_2} \pm 1)(\underbrace{v_3 + (2^k - 2^{t_3} \pm 1)v_4}_{J} + \cdots)\cdots) \end{aligned} \right|_{P_j} \tag{13}$$

Reduction in modulo 2^k can be simply done by considering k-bit LSB of the result of summation. Therefore the reduction in moduli 2^k-1, 2^k+1, 2^{k+1}-1 and 2^{k-1}-1 will be discussed in the following.

3.3.1 Reduction in Modulo 2^k+1

Based on Equation (13) we have

$$x_i = \left| \begin{aligned} &v_1 + (2^k - 2^{t_1} \pm 1)(v_2 + \\ &(2^k - 2^{t_2} \pm 1)(\underbrace{v_3 + (2^k - 2^{t_3} \pm 1)v_4}_{J} + \cdots)\cdots) \end{aligned} \right|_{2^k+1} \tag{14}$$

The value of 2^k-$2^{t_i}\pm$1 in modulo 2^k+1 is equal to -(2^{t_i}+2) and -2^{t_i} for 2^k-2^{t_i}-1 and 2^k-2^{t_i}+1, respectively. Basic operation in Equation (14) is calculation of J. Calculation of J can be done as

$$J = \begin{cases} |v_i - (2^{t_i}+2)v_{i+1}|_{2^k+1} & \text{if} \quad P_i = 2^k - 2^{t_i} - 1 \\ |v_i - 2^{t_i}v_{i+1}|_{2^k+1} & \text{if} \quad P_i = 2^k - 2^{t_i} + 1 \end{cases} \tag{15}$$

$$J = \begin{cases} \left| v_i - v_{i+1}\underbrace{0\cdots0}_{t_i} - v_{i+1}0 \right|_{2^k+1} & \text{if} \quad P_i = 2^k - 2^{t_i} - 1 \\ \left| v_i - v_{i+1}\underbrace{0\cdots0}_{t_i} \right|_{2^k+1} & \text{if} \quad P_i = 2^k - 2^{t_i} + 1 \end{cases} \tag{16}$$

(k+1)-bit separation results in

$$J = \begin{cases} |v_i - v^1_{i+1} - v^2_{i+1} - v^3_{i+1}|_{2^k+1} & \text{if} \quad P_i = 2^k - 2^{t_i} - 1 \\ |v_i - v^2_{i+1} - v^3_{i+1}|_{2^k+1} & \text{if} \quad P_i = 2^k - 2^{t_i} + 1 \end{cases} \tag{17}$$

Where

$$\begin{aligned} v^1_{i+1} &= v_{i+1,k-2}\cdots v_{i+1,0}0 \\ v^2_{i+1} &= v_{i+1,k-t_i}\cdots v_{i+1,0}\underbrace{0\cdots0}_{t_i} \\ v^3_{i+1} &= \underbrace{0\cdots00}_{\text{k}-\text{t}+2_{\text{i}}} v_{i+1,k-1}\cdots v_{i+1,k-t_i+1} \end{aligned}$$

The computation of Equation (17) can be implemented only using additions, given that negative number in modulo 2^k+1 can be expressed as

$$|-v|_{2^k+1} = \left|2^k + 1 - v\right|_{2^k+1} = |\bar{v} + 2|_{2^k+1} \tag{18}$$

Thus

$$J = \begin{cases} |v_i + \bar{v}^1_{i+1} + \bar{v}^2_{i+1} + \bar{v}^3_{i+1} + 6|_{2^k+1} & \text{if} \quad P_i = 2^k - 2^{t_i} - 1 \\ |v_i + \bar{v}^2_{i+1} + \bar{v}^3_{i+1} + 4|_{2^k+1} & \text{if} \quad P_i = 2^k - 2^{t_i} + 1 \end{cases} \tag{19}$$

Hardware implementations of Equation (19) are shown in Figure 2. Correspondence value of J according to moduli 2^k-2^{t_i}-1 or 2^k-2^{t_i}+1 is inserted in Equation (14) with (k+1)-bit in binary representation. Therefore $J' = v_i + (2^k - 2^{t_i} \pm 1) \times J$ must be calculated in next step. Calculation of J' is as follows.

$$J' = \begin{cases} |v_i - (2^{t_i}+2)J|_{2^k+1} & \text{if} \quad P_i = 2^k - 2^{t_i} - 1 \\ |v_i - 2^{t_i}J|_{2^k+1} & \text{if} \quad P_i = 2^k - 2^{t_i} + 1 \end{cases} \tag{20}$$

$$J' = \begin{cases} |v_i + \bar{J}^1 + \bar{J}^2 + \bar{J}^3 + \bar{J}^4 + 8|_{2^k+1} & \text{if } P_i = 2^k - 2^{t_i} - 1 \\ |v_i + \bar{J}^3 + \bar{J}^4 + 4|_{2^k+1} & \text{if } P_i = 2^k - 2^{t_i} + 1 \end{cases} \tag{21}$$

Where

$$\begin{aligned} J^1 &= J_{k-1}\dots J_0 0 \\ J^2 &= \underbrace{0\cdots0}_{k} J_n \\ J^3 &= J_{k-t_i}\dots J_0\underbrace{0\cdots0}_{t_i} \\ J^4 &= \underbrace{0\cdots00}_{k-t_i+1} J_n \dots J_{k-t_i+1} \end{aligned}$$

Table 1: Delay and area of reverse conversion in second RNS basis

RNS Basis	Delay of Conversion from RNS to Weighted	Area of Conversion from RNS to Weighted
$\{2^k$, 2^{k+1}-1, 2^k-1, 2^k+1$\}$ [23]	$((23k+12)/2)D_{FA}$	$((k\text{-}4)(k+1)/2 + 9k+5)A_{FA}$ $+2kA_{OR}+2kA_{XNOR}+(6k+1)A_{NOT}$
$\{2^k$, 2^{k+1}-1, 2^k-1, 2^k+1, 2^{k-1}-1$\}$ [8]	$(18k+L+7)\ D^*_{FA}$	$((5k^2+43k+m)/6+16k\text{-}1)A_{FA}$ $+\ (6k+1)A^*_{NOT}$

* m=k-4, 9k-12 and 5k-8 for k=6h-2, 6h and 6h+2 (h= 1, 2,...), respectively, and L is the number of the levels of the CSA tree with ((k/2)+1) inputs.

Table 2: Proposed RNS bases for k=64

RNS bases	First Basis B_m	Auxiliary Basis B'_m
Four moduli RNS bases	2^{64}-2^8-1, 2^{64}-2^{22}-1, 2^{64}-2^{15}-1, 2^{64}-2^{16}-1	2^{64}, 2^{65}-1, 2^{64}-1, 2^{64}+1
Five 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^{64}, 2^{65}-1, 2^{64}-1, 2^{64}+1, 2^{63}-1

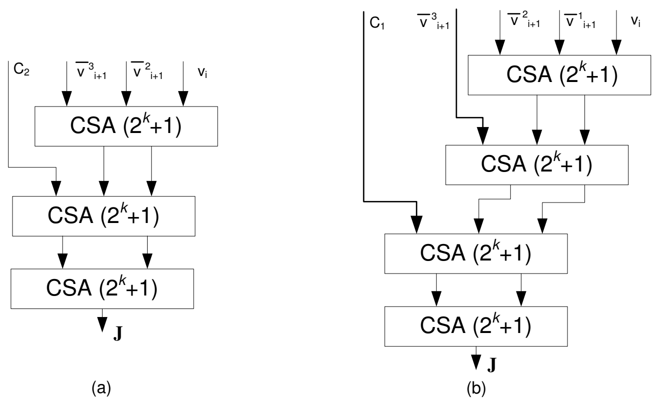

Figure 2: Hardware implementation of reduction of J in modulo 2^k+1. C_1 is equal to 3 and C_2 is equal to 2. (a) when 2^k-2^{t_i}+1, (b) when 2^k-2^{t_i}-1.

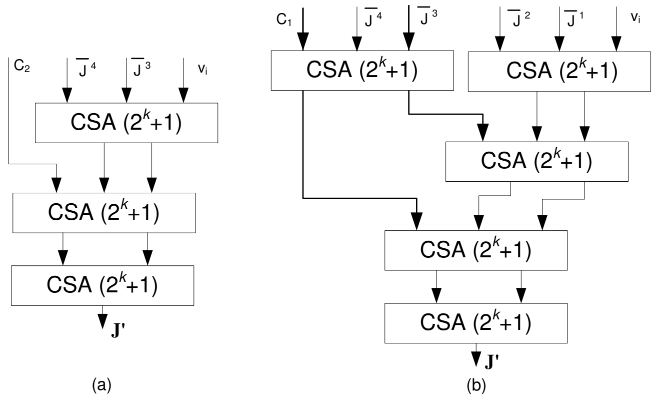

Figure 3: Hardware implementation of reduction of J' in modulo 2^k+1. C_1 is equal to 4 and C_2 is equal to 2. (a) when 2^k-2^{t_i}+1, (b) when 2^k-2^{t_i}-1.

Hardware implementations of J' are shown in Figure 3. Carry save adder (CSA) in modulo 2^k+1 add the input values with an extra unit [11]. Therefore in each CSA shows in Figures 2 and 3 the one unit is decreased from constant values in Equation (19) and 21. The remaining values of constants are included as the input of CSA in Figures 2 and 3. Based on hardware implementations in Figures 2 and 3, the delay and area of conversion from MRS to RNS in the worse case can be calculated as

$$Delay_{MRS-RNS} = \left(\sum_{i=1}^{m-1}\left(MA(2^k+1)+3\right)\right)D_{FA} \quad (22)$$

$$Area_{MRS\text{–}RNS} = \begin{pmatrix} MA(2^k+1)+3CSA(2^k+1) \\ \sum_{i=2}^{m-1}\left(MA(2^k+1)+4CSA(2^k+1)\right)\end{pmatrix} A_{FA} \quad (23)$$

3.3.2 Reduction in Modulo 2^{k+1}-1

Based on Equation (13) we have

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

Calculation of I as the basic operation is

$$I = \left|v_i + (2^k - 2^{t_i} \pm 1)v_{i+1}\right|_{2^{k+1}-1} \quad (25)$$

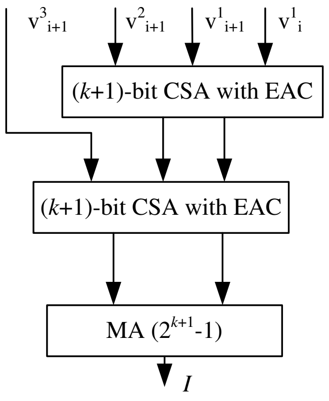

Figure 4: Hardware implementation of reduction of I in modulo 2^{k+1}-1.

Using Lemma1 and 2 result in

$$I = \left|v_i^1 + v_{i+1}^1 + v_{i+1}^2 + v_{i+1}^3\right|_{2^{k+1}-1} \quad (26)$$

Where

$$\begin{aligned}
v_i^1 &= 0v_i \\
v_{i+1}^1 &= v_{i+1,0}0v_{i+1,k-1}\cdots v_{i+1,1} \\
v_{i+1}^2 &= \bar{v}_{i+1,k-t_i}\cdots\bar{v}_{i+1,0}1\bar{v}_{i+1,k-1}\cdots\bar{v}i_{i+1,k-t_i+1} \\
v_{i+1}^3 &= \begin{cases} 1\bar{v}_{i+1} & \text{if} \quad \mathrm{p_i}= 2^{\mathrm{n}}-2^{\mathrm{t_i}}-1 \\ 0v_{i+1} & \text{if} \quad \mathrm{p_i}= 2^{\mathrm{n}}-2^{\mathrm{t_i}}+1 \end{cases}
\end{aligned}$$

By replacing I in Equation (24) with (k+1)-bit in binary form, v_i + (2^k - 2^{t_i}±1)×I must be calculated. Therefore v_i^1, v_{i+1}^1, v_{i+1}^2 and v_{i+1}^3 in Equation (26) changed to

$$\begin{aligned}
v_i^1 &= 0v_i \\
v_{i+1}^1 &= I_0I_{k-1}\cdots I_1 \\
v_{i+1}^2 &= \bar{I}_{k-t_i}\cdots\bar{I}_0\bar{I}_k\cdots\bar{I}_{k-t_i+1} \\
v_{i+1}^3 &= \begin{cases} \bar{I} & \text{if} \quad \mathrm{P_i}= 2^{\mathrm{k}}-2^{\mathrm{t_i}}-1 \\ I & \text{if} \quad \mathrm{P_i}= 2^{\mathrm{k}}-2^{\mathrm{t_i}}+1 \end{cases}
\end{aligned}$$

Hardware implementation of Equation (26) is shown in Figure 4.

Based on Figure 3, hardware requirements for calculation of I consists of two levels of CSA with end around carry (EAC) followed by the MA(2^{k+1}-1). Therefore delay and area of conversion from MRS to RNS can be calculated as

$$Delay_{MRS–RNS} = \left(\sum_{i=1}^{m-1}\left(MA(2^{k+1}-1)+2\right)\right)D_{FA} \quad (27)$$

$$Area_{MRS–RNS} = \left(\sum_{i=1}^{m-1}\left(MA(2^{k+1}-1)+2CSA(2^{k+1}-1)\right)\right)A_{FA} \quad (28)$$

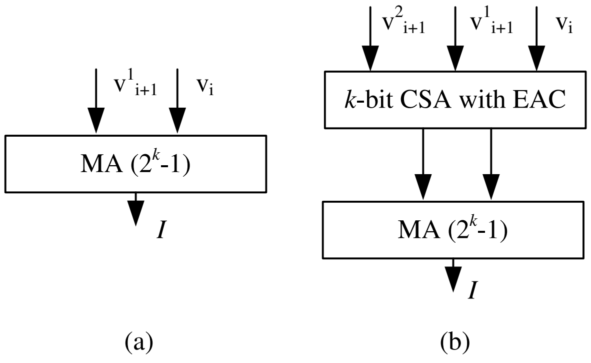

Figure 5: Hardware implementation of reduction of I in modulo 2^k-1, (a) when P_i = $2^k-2^{t_i}+1$, (b) when P_i= $2^k-2^{t_i}-1$.

3.3.3 Reduction in modulo 2^k-1

Based on Equation (13) we have

$$x_i = \left| \begin{array}{l} v_1+(2^k-2^{t_1}\pm 1)(v_2+ \\ (2^n-2^{t_2}\pm 1)(\underbrace{v_3+(2^n-2^{t_3}\pm 1)v_4}_{I}+\cdots)\cdots) \end{array} \right|_{2^k-1} \quad (29)$$

Calculation of I as the basic operation is

$$I = \left|v_i + (2^k - 2^{t_i} \pm 1)v_{i+1}\right|_{2^k-1} \quad (30)$$

The values of $2^k-2^{t_i}-1$ and $2^k-2^{t_i}+1$ in modulo 2^k-1 are equal to -2^{t_i} and $-2^{t_i}+2$, respectively. Therefore Equation (30) can be rewritten as

$$I = \begin{cases} |v_i - 2^{t_i}v_{i+1}|_{2^k-1} & \text{if} \quad P_i = 2^k-2^{t_i}-1 \\ |v_i + (-2^{t_i}+2)v_{i+1}|_{2^k-1} & \text{if} \quad P_i = 2^k-2^{t_i}+1 \end{cases} \quad (31)$$

Using Lemma1 and 2 result in

$$I = \begin{cases} \left|v_i + v_{i+1}^1\right|_{2^k-1} & \text{if} \quad P_i = 2^k-2^{t_i}-1 \\ \left|v_i + v_{i+1}^1 + v_{i+1}^2\right|_{2^k-1} & \text{if} \quad P_i = 2^k-2^{t_i}+1 \end{cases} \quad (32)$$

Where

$$\begin{aligned}
v_{i+1}^1 &= \bar{v}_{i+1,k-t_i-1}\cdots\bar{v}_{i+1,0}\bar{v}_{i+1,k-1}\cdots\bar{v}_{i+1,k-t_i} \\
v_{i+1}^2 &= v_{i+1,k-2}\cdots v_{i+1,0}v_{i+1,k-1}
\end{aligned}$$

Hardware implementation of Equation (32) is shown in Figure 5. Worse case delay and area of conversion from MRS to RNS in modulo 2^k-1 can be calculated as

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

$$Area_{MRS-RNS} = \left(\sum_{i=1}^{m-1}\left(MA(2^k-1)+CSA(2^k-1)\right)\right)A_{FA} \quad (34)$$

3.3.4 Reduction in Modulo 2^{k-1}-1

Based on Equation (13) we have

$$x_i = \left| \begin{array}{l} v_1 + (2^k - 2^{t_1} \pm 1)(v_2 + \\ (2^k - 2^{t_2} \pm 1)(\underbrace{v_3 + (2^k - 2^{t_3} \pm 1)v_4}_{I} + \cdots) \cdots) \end{array} \right|_{2^{k-1}-1} \quad (35)$$

The value of $2^k - 2^{t_i} - 1$ and $2^k - 2^{t_i} + 1$ in modulo 2^{k-1}-1 are equal to $-2^{t_i} + 1$ and $-2^{t_i} + 3$, respectively. Therefore Equation (35) can be rewritten as

$$I = \begin{cases} |v_i + (-2^{t_i} + 1)v_{i+1}|_{2^{k-1}-1} & \text{if} \quad P_i = 2^k - 2^{t_i} - 1 \\ |v_i + (-2^{t_i} + 3)v_{i+1}|_{2^{k-1}-1} & \text{if} \quad P_i = 2^k - 2^{t_i} + 1 \end{cases} \quad (36)$$

Using Lemma1 and 2 result in

$$I = \begin{cases} \left| \begin{array}{l} v_i^1 + v_i^2 + v_{i+1}^1 + v_{i+1}^2 + \\ v_{i+1}^3 + v_{i+1}^4 \end{array} \right|_{2^{k-1}-1} & \text{if } P_i = 2^k - 2^{t_i} - 1 \\ \left| \begin{array}{l} v_i^1 + v_i^2 + v_{i+1}^1 + v_{i+1}^2 + \\ v_{i+1}^3 + v_{i+1}^4 + v_{i+1}^5 + v_{i+1}^6 \end{array} \right|_{2^{k-1}-1} & \text{if } P_i = 2^k - 2^{t_i} + 1 \end{cases} \quad (37)$$

Where

$$\begin{array}{l} v_i^1 = v_{i,k-2} \cdots v_{i,0} \\ v_i^2 = \underbrace{0 \cdots 0}_{k-2} v_{i,k-1} \\ v_{i+1}^1 = \underbrace{1 \cdots 1}_{k-t_i-2} \bar{v}_{i+1,k-1} \cdots \bar{v}_{i+1,k-t_i-1} \\ v_{i+1}^2 = \bar{v}_{i+1,k-t-2} \cdots \bar{v}_{i+1,0} \underbrace{1 \cdots 1}_{t_i} \\ v_{i+1}^3 = v_{i+1,k-2} \cdots v_{i+1,0} \\ v_{i+1}^4 = \underbrace{0 \cdots 0}_{k-2} v_{i+1,k-1} \\ v_{i+1}^5 = v_{i+1,k-3} \cdots v_{i+1,0} 0 \\ v_{i+1}^6 = \underbrace{0 \cdots 0}_{k-3} v_{i+1,k-1} v_{i+1,k-3} \end{array}$$

Hardware implementation of Equation (37) is shown in Figure 6. After calculation of I as (k-1)-bit in binary form in Equation (39), $I' = v_i + (2^k - 2^{t_i} \pm 1) \times I$ must be calculated. Therefore variables in Equation (37) can be rewritten as

$$I' = \left| v_i^1 + v_i^2 + I^1 + I^2 + I^3 \right|_{2^{k-1}-1} \quad (38)$$

Where

$$\begin{array}{l} v_i^1 = v_{i,k-2} \cdots v_{i,0} \\ v_i^2 = \underbrace{0 \cdots 0}_{k-2} v_{i,k-1} \\ I^1 = I_{k-3} \cdots I_0 I_{k-2} \\ I^2 = \bar{I}_{k-t_i-3} \cdots \bar{I}_0 \bar{I}_{k-2} \cdots \bar{I}_{k-t_i-2} \\ I^3 = \begin{cases} \bar{I} & \text{if } \mathrm{P_i = 2^k - 2^{t_i} - 1} \\ I & \text{if } \mathrm{P_i = 2^k - 2^{t_i} + 1} \end{cases} \end{array}$$

Hardware implementation of I' is shown in Figure 6.c. The delay and area of conversion from MRS to RNS in modulo 2^{k-1}-1 in worse case can be calculated as

$$Delay_{MRS-RNS} = \left(MA(2^{k-1} - 1) + 4 + \sum_{i=2}^{m-1} (MA(2^{k-1} - 1) + 3) \right) D_{FA} \quad (39)$$

$$Area_{MRS-RNS} = \left(\begin{array}{l} MA(2^{k-1} - 1) + 6CSA(2^{k-1} - 1) + \\ \sum_{i=2}^{m-1} (MA(2^{k-1} - 1) + 3CSA(2^{k-1} - 1)) \end{array} \right) A_{FA} \quad (40)$$

3.4 Reduction of Weighted Number in Moduli 2^k-c$_i$

As mentioned before in [4] moduli in the form of 2^k-$2^{t_i} \pm$ 1 are used in order to achieve small hamming weight for both moduli and multiplicative inverses. Since for one bases in this paper such as proposed in [4] the modulus in the form of 2^k-$2^{t_i} \pm$ 1 where 0 < t$_i$ < $k/2$ are used to design efficient RNS bases, the binary to residue converter for this moduli is presented. In [4] to calculate the residue numbers from MRS in five moduli set based on Equation (5) the following operation must be considered:

$$x_j = \left| v_1 + p_1(v_2 + P_2(v_3 + P_3 \underbrace{(v_4 + P_4 v_5)}_{H})) \right|_{2^k - 2^{t_j} \pm 1} \quad (41)$$

The delay of H is addition of n bit words where $\omega(c_i)$ and $\omega(c'_j)$ is the hamming weight of 2^k-2^{t_i}±1 and 2^k-2^{t_j}±1, respectively. In [4] total delay for m moduli is reported as

$$Delay_{MRS-RNS} = \sum_{i=1}^{m-1} \max_{j=1,m} (\omega(c_i) + 2\omega(c'_j) + 2)) k D_{FA} \quad (42)$$

For designing binary to residue converter in modulo 2^k-2^{t_j}±1 for M-bit dynamic range we have

$$X = \sum_{i=0}^{M/k-1} 2^{ik} x_i \quad (43)$$

In this work, M=4k and 5k-bit dynamic ranges are considered. Therefore in the worse case we have

$$X = \sum_{i=0}^{4} 2^{ik} x_i = 2^{4k} x_4 + 2^{3k} x_3 + 2^{2k} x_2 + 2^k x_1 + x_0 \quad (44)$$

The above equation can be rewritten as

$$X = 2^k (2^k (2^k \underbrace{(2^k x_4 + x_3)}_{z} + x_2) + x_1) + x_0 \quad (45)$$

Unlike the calculation needed in Equation (42), the hamming weight of c$_i$ in calculation of z as the basic operation in Equation (45) is equal to zero. Therefore the delay of reduction in moduli 2^k-2^{t_j}±1 reported in Equation (42) for the proposed RNS bases is changed to

$$Delay_{Weighted-RNS} = (\sum_{i=1}^{m-1} (2\omega(c'_j) + 2)) k D_{FA} \quad (46)$$

Therefore less addition is needed in the process of forward converter.

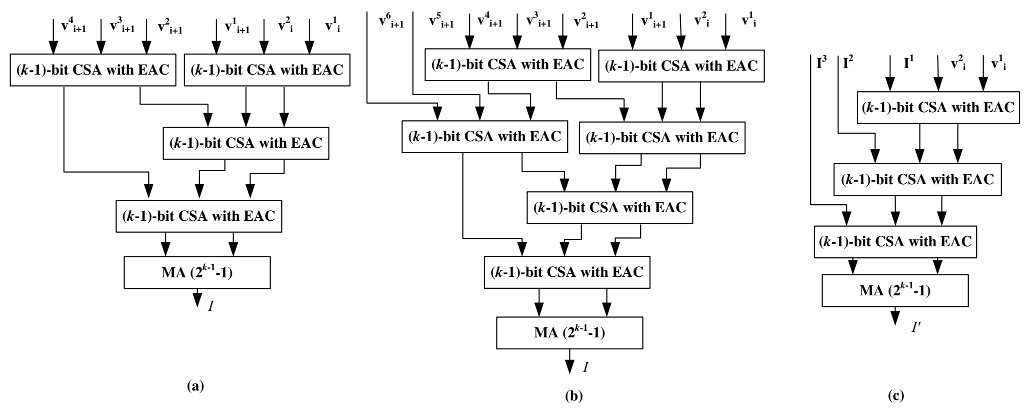

Figure 6: Hardware implementation of reduction of I in modulo 2^{n-1}-1, (a) when $P_i = 2^k - 2^{t_i} + 1$, (b) when $P_i = 2^k - 2^{t_i} - 1$, (c). Calculation of I'.

4 Complexity of Modular Multiplication and Comparison

The efficiency of arithmetic operation is the main advantages of this work. In the first line of RNS Montgomery modular multiplication algorithm presented in Section 2.2, the operations $d_i = |a_i \times b_i|_{P_i}$ and $d'_i = |a'_i \times b'_i|_{P'_i}$ for i = 1,..., m and in the fourth line computation of $r'_i = \left|(d'_i - q'_i \times N'_i)\,|M^{-1}|_{P'_i}\right|_{P'_i}$ are required. In [4] these operations in the first line have the cost of 2m products of k-bit words and $2m$ reductions with cost of $\sum_{i=1}^{m}(2\omega(c_i)+2) + \sum_{j=1}^{m}(2\omega(c'_j)+2)$ additions of k-bit words. The operations in fourth line have the cost of $2m$ products of k-bit words and $2m$ reductions with cost of $2\sum_{j=1}^{m}(2\omega(c'_j)+2)$ additions of k-bit words. Since the second basis of our approach are the moduli sets {2^k-1, 2^k, 2^k+1, 2^{k-1}-1} and {2^k-1, 2^k, 2^k+1, 2^{k+1}-1, 2^{k-1}-1}, the reduction in these moduli sets can be implemented with more simple process. For an instance for numbers in the dynamic ranges of $4k$ and $5k$ bits, several levels of CSA followed by MA($2^k\pm$ 1) is needed [11]. Therefore the delay in reduction $\sum_{j=1}^{m}(2\omega(c'_j)+2)$ can be eliminated and replaced by delay of some levels of CSA and MA($2^k\pm$ 1). It is worth mentioning that in the second basis the required product can be produced more efficiently than the RNS basis with moduli in the form of 2^k-c$_i$ [13, 34, 37]. The delay of RNS to RNS conversion in [4] assuming m, k-bits arithmetic unit is

$$Delay_{RNS\text{–}RNS} = Delay_{RNS\text{–}MRS} + Delay_{MRS\text{–}RNS} \quad (47)$$

Where

$$Delay_{RNS-MRS} = \\ (\sum_{i=1}^{m-1} \max_{j=2,m;i<j} (\omega(|P_i^{-1}|_{P_j}) + 2\omega(c_j) + 4))kD_{FA}$$

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

With using the moduli sets {2^k-1, 2^k, 2^k+1, 2^{k-1}-1} and {2^k-1, 2^k, 2^k+1, 2^{k+1}-1, 2^{k-1}-1} in second basis, the delay of MRS to RNS conversion in second basis is removed in Equation (47) and replaced by

$$Delay_{MRS-RNS} = \left(\sum_{i=1}^{m-1}(MA(2^k+1)+3)\right)D_{FA} \quad (48)$$

$$Area_{MRS\text{–}RNS} = \left(\sum_{i=1}^{m-1}(MA(2^k+1)+4n)\right)A_{FA} \quad (49)$$

In conversion from second to first basis, the delay of RNS to MRS is removed and replaced by delays reported in Table 1 for four and five moduli sets. Conversion from weighted number to RNS in next basis is improved by $(\sum_{i=1}^{m-1}(2\omega(c'_j)+2))kD_{FA}$ computation instead of the computation required for MRS to RNS in Equation (47). Therefore noticeable increase in speed in Montgomery modular multiplication is achieved. In [4], performance comparison is based on cost of RNS to RNS conversion which is based on the number of required 64-bit FA. In the proposed RNS bases, the moduli sets {2^k-1, 2^k, 2^k+1, 2^{k-1}-1} and {2^k-1, 2^k, 2^k+1, 2^{k+1}-1, 2^{k-1}-1} with their reverse converters are employed as the second basis. Reverse converters structure of these moduli sets mainly consists of CSA tree followed by MA(2^k-1). Therefore comparison based on the number of required 64-bit FA would not be a fair comparison. In order to achieve a fair comparison, critical path delay is considered with the following assumptions. Calculation of mixed radix numbers is considered with serial process (Equation (47)). In order to compute residue in next basis, bearing in mind that the forward conversion in each moduli is independent from the other moduli, the delay of critical and slowest moduli must be considered for this part. Table 3 shows the comparison delay of RNS to RNS conversion of the proposed RNS bases comparing to [4]. As shown in Table 3, the proposed RNS basis has achieved noticeable improvement in delay of RNS to RNS conversion required in modular multiplication and therefore the overall performance of the modular multiplication is improved.

Table 3: Comparison delay of different RNS bases

RNS bases	**Total delay**	**Improvement (%)**
4 moduli bases (P-S) [4]	(6976)D_{FA}	-
4 moduli -Proposed	(4938)D_{FA}	(P-S) 29%
5 moduli bases (P) [4]	(13848)D_{FA}	-
5 moduli bases (S) [4]	(11840)D_{FA}	-
5 moduli -Proposed	(9512)D_{FA}	(P) 31%
		(S) 20%

In [4] to show the efficiency of modular multiplication with its proposed RNS bases, comparison with standard Montgomery algorithms [12, 25] and its RNS versions [2, 35] are done and shown to be more efficient. Also comparison with Karatsuba implementation [12] is done and has achieved the same efficiency with more straightforward architecture. Since the proposed RNS bases has achieved noticeable increase in speed in modular multiplication comparing to [4], we can conclude that comparing to [2, 35] and [12] more efficient design of modular multiplication is achieved by the proposed RNS bases.

5 Conclusions

This paper has proposed the RNS bases for public key cryptography and especially for ECC. With using arithmetic friendly moduli sets in the second basis, multiplication in each moduli can be done with more speed. Fast conversion from one basis to another leads to an increase in the overall performance of the system. Comparison with the best RNS bases in literature achieved 29% improvement in delay of RNS to RNS conversion for four moduli RNS bases and 20% improvement in speed of RNS to RNS conversion in five moduli RNS bases. Therefore noticeable improvement in speed of the modular multiplication is achieved by the proposed RNS bases.

References

[1] J. Bajard, L. Didier, and P. Kornerup, "An rns montgomery's modular multiplication algorithm," *IEEE Transactions on Computers*, vol. 47, no. 2, pp. 167–178, 1998.

[2] J. Bajard, L. Didier, and P. Kornerup, "Modular multiplication and base extensions in residue number systems," in *Proceedings of the 15th IEEE Symposium on Computer Arithmetic (ARITH '01)*, pp. 59–65, 2001.

[3] J. C. Bajard and L. Imbert, "A full rns implementation of rsa," *IEEE Transactions on Computers*, vol. 53, no. 6, pp. 769–774, 2004.

[4] J. C. Bajard, M. Kaihara, and T. Plantard, "Selected rns bases for modular multiplication," in *19th IEEE International Symposium on Computer Arithmetic (Arith '09)*, pp. 25–32, 2009.

[5] S. Basu, "A new parallel window-based implementation of the elliptic curve point multiplication in multi-core architectures," *International Journal of Network Security*, vol. 13, no. 3, pp. 234–241, 2011.

[6] T. Blum and C. Paar, "High-radix montgomery modular exponentiation on reconfigurable hardware," *IEEE Transactions on Computers*, vol. 50, pp. 759–764, 2001.

[7] V. Bunimov, M. Schimmler, and B. Tolg, "A complexity-effective version of montgomery's algorithm," in *Proceedings of the 29th Annual International Symposium Computer Architecture (ISCA '02) Workshop Complexity Effective Designs*, pp. 1–7, May 2002.

[8] 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*, vol. 54, no. 5, pp. 1041–1049, 2007.

[9] 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}," *IEE Proceeding on Computers and Digital Techniques '05*, vol. 152, pp. 687–696, 2005.

[10] G. C. Cardarilli, A. Nannarelli, and M. Re, "Residue number system for low-power dsp applications," in *Proceedings of the 41nd IEEE Asilomar Conference on Signals, Systems, and Computers '07*, pp. 1412–1416, 2007.

[11] R. Chaves and L. Sousa, "Improving residue number system multiplication with more balanced moduli sets and enhanced modular arithmetic structures," *IET Computers and Digital Techniques*, vol. 1, no. 5, pp. 472–480, 2007.

[12] J. P. David, K. Kalach, and N. Tittley, "Hardware complexity of modular multiplication and exponentiation," *IEEE Transactions on Computers*, vol. 56, no. 10, pp. 1308–1319, 2007.

[13] C. Efstathiou, H. T. Vergos, and D. Nikolos, "Modified booth modulo 2^n-1 multipliers," *IEEE Transactions on Computers*, vol. 53, no. 2, pp. 370–374, 2004.

[14] S. E. Eldridge and C. D. Walter, "Hardware implementation of montgomery's modular multiplication algorithm," *IEEE Transactions on Computers*, vol. 42, no. 6, pp. 693–699, 1993.

[15] M. Esmaeildoust, K. Navi, and M. 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 Electronics Express*, vol. 7, no. 3, pp. 118–125, 2010.

[16] M. Hedabou, "A frobenius map approach for an efficient and secure multiplication on koblitz curves," *International Journal of Network Security*, vol. 3, no. 3, pp. 233–237, 2006.

[17] M. S. Hwang and C. Y. Liu, "Authenticated encryption schemes: Current status and key issues," *International Journal of Network Security*, vol. 1, no. 2, pp. 61–73, 2005.

[18] M. S. Hwang and P. C. Sung, "A study of micropayment based on one-way hash chain," *International Journal of Network Security*, vol. 2, no. 2, pp. 81–90, 2006.

[19] M. Kaihara J. C. Bajard and T. Plantard. "Report: Selected rns bases for modular multiplication,". tech. rep., http://www.lirmm.fr/ bajard/ MesPublis/BKP09Report.pdf.

[20] K. M. Kalantari, S. P. Mozafari, and B. Sadeghiyan, "Improved RNS for RSA hardware implementation," *Journal on Computer Science and Engineering*, vol. 2, no. 2&4 (b), pp. 31–39, 2004.

[21] C. McIvor, M. McLoone, A. Daly J. V. McCanny, and W. Marnane, "Fast montgomery modular multiplication and rsa cryptographic processor architectures," in *Proceedings of the 37th Annual Asilomar Conferences on Signals, Systems, and Computers '03*, pp. 379–384, 2003.

[22] C. McIvor, M. McLoone, and J. V. McCanny, "Modified montgomery modular multiplication and rsa exponentiation," *IEE Proceeding on Computers and Digital Techniques'04*, vol. 151, pp. 402–408, 2004.

[23] 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*, vol. 54, no. 6, pp. 1245–1254, 2007.

[24] A. S. Molahosseini, K. Navi, O. Kavehei C. Dadkhah, 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*, vol. 57, no. 4, pp. 823–835, 2010.

[25] P. Montgomery.

[26] S. Moon, "A binary redundant scalar point multiplication in secure elliptic curve cryptosystems," *International Journal of Network Security*, vol. 3, no. 2, pp. 132–137, 2006.

[27] 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*, vol. E94-D, no. 2, pp. 264–273, 2011.

[28] K. Navi, A. S. Molahosseini, and M. Esmaeildoust, "How to teach residue number system to computer scientists and engineers," *IEEE Transactions on Education*, vol. 53, no. 3, pp. 156–163, 2010.

[29] B. Parhami. "Computer arithmetic: Algorithms and hardware design,". tech. rep., Oxford University Press, 2000.

[30] F. Pourbigharaz and H. M. Yassine, "A signed digit architecture for residue to binary transformation," *IEEE Transactions on Computers*, vol. 46, no. 10, pp. 1146–1150, 1997.

[31] 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 Syatems-I*, vol. 56, no. 6, pp. 1206–1213, 2009.

[32] D. M. Schinianakis, A. P. Kakarountas, and T. Stouraitis, "A new approach to elliptic curve cryptography: An rns architecture," in *Proceedings of the IEEE Mediterranean Electrotech*, pp. 1241–1245, May 2006.

[33] A. Skavantzos and M. Abdallah, "Implementation issues of two level residue number system with pairs of conjugate moduli," *IEEE Transactions on Signal Processing*, vol. 47, no. 3, pp. 826–838, 1999.

[34] L. Sousa and R. Chaves, "A universal architecture for designing efficient modulo $2^n + 1$ multipliers," *IEEE Transactions on Circuits and Systems-I*, vol. 52, no. 6, pp. 1166–1178, 2005.

[35] R. Szerwinski and T. Gneysu, "Exploiting the power of gpus for asymmetric cryptography," in *Proceedings of the 10th International Workshop on Cryptographic Hardware and Embeded Systems (CHES '08)*, vol. LNCS 5154.

[36] A. F. Tenca and C. K. Koc, "A scalable architecture for modular multiplication based on montgomery's algorithm," *IEEE Transactions on Computers*, vol. 52, no. 9, pp. 1215–1221, 2003.

[37] H. T. Vergos and C. Efstathiou, "Design of efficient modulo 2^n+1 multipliers," *IET Computers and Digital Techniques*, vol. 1, no. 1, pp. 49–57, 2007.

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

[39] R. Zimmerman, "Efficient VLSI implementation of modulo ($2^n \pm 1$)addition and multiplication," in *Proceedings of the 14th IEEE Symposium on Computer Arithmetic (Arith '99)*, pp. 158–167, Apr. 1999.

Mohammad Esmaeildoust is Ph.D. candidate in Computer architecture at Shahid Beheshti University of Technology (Tehran, Iran). He received his M.Sc. degree in Computer architecture at Shahid Beheshti University of Technology (Tehran, Iran) in 2008. He received his B.Sc. degree in 2006 from shahed University in Hardware Engineering. His research interests include reconfigurable computing, public key cryptography and computer arithmetic especially on residue number system.

Shirin Rezaei 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 and the M.Sc. degree in Computer Architecture at IAU, Science and Research Branch in 2011. Her research interests include computer arithmetic especially on residue number system.

Marzieh Gerami was born in Shahr_e_kord, Iran, in 1985. She received the B.Sc. degree from Shahid Bahonar University Of Kerman, Iran in 2007 and the M.Sc. degree in Computer Architecture at IAU, Science and Research Branch in 2011. Her research interests include public key cryptography with emphasis on elliptic curve cryptography.

Keivan Navi received the B.Sc. and M.Sc. degrees in computer hardware engineering from Beheshti University, Tehran, Iran, in 1987 and Sharif University of Technology, Tehran, Iran, in 1990, respectively. He also received the Ph.D. degree in computer architecture from Paris XI University, Paris, France, in 1995. He is currently Associate Professor in faculty of electrical and computer engineering of Beheshti University. His research interests include VLSI design, single electron transistors (SET), carbon nano tube, computer arithmetic, interconnection network and quantum computing. He has published over 50 ISI and research journal papers and over 70 IEEE, international and national conference paper.