Selected RNS Bases for Efficient Design of
Montgomery Multiplication

Elham Javadnoori, Saeed Setayeshi, Mohammad Kheirandish

Department of Computer Engineering, Dezful Branch, Islamic Azad University, Dezful, Iran
JavadnooriElham@yahoo.com

Department of Physics and Nuclear Science, Amirkabir University, Tehran, Iran
Setayesh@aut.ac.ir

Department of Computer Engineering, Dezful Branch, Islamic Azad University, Dezful, Iran
Mkh_student@yahoo.com

Abstract
During the last years, different fast hardware implementations of RSA encryption have been done. In RSA algorithm the base operation is modular exponentiation. An efficient approach for modular exponentiation is Montgomery multiplication which computes the remainders without any division. Residue number system (RNS), replace large numbers with their small residues and does the mathematical operations on its small residues in parallel and without carry propagation. RNS Montgomery multiplication, similar to Montgomery multiplication has no division and moreover it computes addition, subtraction, and multiplication in RNS and therefore it is more efficient. Moduli selection has a great impact on RNS efficiency. In previous works optimal moduli sets are reported and in this work we propose a moduli set that is more efficient than the previous works.

Keywords: RSA, Montgomery Multiplication, RNS, Moduli Set

I. Introduction
RSA encryption algorithm is a public key cryptography algorithm so it has an encryption key and when the key length grows up, the RSA security will be better. In RSA, modular exponentiation is the base operation. In (Noorimehr et al., 2014) efficient algorithms for modular multiplication and modular exponentiation have been reported. Montgomery's algorithms compute the remainders without any division. With removing division from operations, an efficient residue computation will be achieved. But since RSA has very large operands so addition, subtraction and specially the multiplication are still very time consuming operations. (RNS) replaces large numbers with their small residues and compute the mathematic operations on its small residues in parallel and without carry propagation. RNS Montgomery multiplication is an RNS version of Montgomery multiplication in which the operations done in RNS form. And also the moduli selection has a great effect on the efficiency of RNS operations. Until now many moduli sets have been introduced for RNS that includes: \(\{2^n - 1, 2^n, 2^n + 1\}\), \(\{r^b - 1, r^a, r^c + 1\}\), \(\{2^n - 1, 2^n, 2^n + 1, 2^n+1 - 1\}\), \(\{2^n - 1, 2^n, 2^n + 1, 2^{n-1} - 1\}\), \(\{2^n - 1, 2^n, 2^n + 1, 2^n+1 - 1, 2^{n+1} - 1\}\), \(\{2^n - 1, 2^n, 2^n +\)
\(1, 2^{n+1} + 1\}, \{2^n - 1, 2^n, 2^n + 1, 2^{2n} + 1\}, \{2^{2n}, 2^n - 1, 2^{n+1} - 1\}, \{r^n - 2, r^n - 1, r^n\}, \{2^n, 2^n - 1, 2^n + 1\} \) and \(\{2^n, 2^n - 1, 2^n + 1\}(\text{Wanget al., 2002; Abdallah and Skavantzos, 2005; Caos et al., 2003; Caos et al., 2007; Caos et al., 2005; Haririet al., 2008; Hosseinzadehet al., 2007; Mohan and Premkumar, 2007; Molahosseiniet al., 2009}).

But among these only some are suitable for RNS Montgomery multiplication. As has been presented in (Montgomery, 1985), RNS Montgomery multiplication has two RNS bases. in (Manochehri Kalantari et al., 2004) moduli set in the form of \(2^{m_l} - 1\) has been used as first base and moduli set in the form of \(2^{m_l} + 1\) has been used for second base which . \(j = 0, 1, 2, \ldots, k\). Disadvantage of these moduli sets is the inefficient arithmetic unit and reverse converter. In (Bajard et al., 2009) RNS bases in the form of \(2^m - c_i\) where \(0 \leq c_i < 2^{m/2}\) have been presented for both first and second bases. The main advantage of these bases is their small hamming weight because the efficiency of the first RNS bases depends on its moduli's hamming weight. In (Geramiet al., 2011) similar to (Bajard et al., 2009) a four-moduli set with elements in the form of \(2^m - c_i\) that \(0 \leq c_i < 2^{m/2}\) has been used as the first base and for second base the moduli sets \(\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}\) and \(\{2^n, 2^n, 2^n + 1, 2^n + 1\}(\text{Molahosseiniet al., 2010; Caos et al., 2003})\) have been used. In compare with previous works it has more efficient arithmetic unit and converters. And also in (Esmaeilidoust et al., 2014) similar to (Bajard et al., 2009) the moduli set in the form of \(2^m - c_i\) has been used as the first RNS base and \(\{2^m - 1, 2^m, 2^m + 1, 2^{m+1} - 1\}\) and \(\{2^m - 1, 2^m, 2^m + 1, 2^{m+1} - 1\}\) have been used as the second base. The arithmetic unit and the RNS to RNS converter are more efficient with these RNS bases.

In this work we propose two RNS bases and prove that they are more efficient than the other RNS bases that have been presented so far in the literature. The RNS base that we use for first base is in the form of \(2^m - c_i\) where \(0 \leq c_i < 2^{m/2}\) and for second base we use the moduli set \(\{2^n - 1, 2^n, 2^n + 1, 2^{2n} - 1\}(\text{Noorimehret al., 2014}).

The organization of the paper is as the following. Firstly, we introduce the background of RNS and Montgomery multiplication in section 2. In section 3 we propose our moduli set and evaluate the performance of RNS Montgomery multiplication with our proposed moduli set. Then in section 4 we compare our moduli set with the previous works. Finally we conclude the paper with a summary and outlook for further research in section 5.

II. Related Background

A. Modular Exponentiation and Montgomery Multiplication

Montgomery's methods are efficient methods (Montgomery, 1985) for modular multiplication and modular exponentiation that computes the remainder without any division. If \(n\) is a k-bit number then \(2^{k-1} \leq n < 2^k\). It is assumed that \(r = 2^k\) then division by \(r\) can be done by simple shifts. In Montgomery's methods \(r, n\) must be relatively prime to each other, in other word \(GCD(r, n) = GCD(2^k, n) = 1\). If \(n\) is odd this assumption will be satisfied.
Function ModExp(M,e,n)

Output: $x = M^e \mod n$

1: Compute $n'$ using Euclid's algorithm
2: $\bar{M} := M \cdot r \mod n$
3: $\bar{x} := 1 \cdot r \mod n$
4: For $i = l - 1$ downto 0 do
5: $\bar{x} := \text{MonMul}(\bar{x}, \bar{x})$
6: If $e = 1$ then $\bar{x} := \text{MonMul}(\bar{M}, \bar{x})$
7: $\bar{x} := \text{MonMul}(\bar{x}, 1)$
8: Return $x$

Function MonMul($\bar{a}, \bar{b}$)

Output: $u = \bar{a} \cdot \bar{b} \cdot r^{-1} \mod n$

$u = a \cdot b \cdot r \mod n$

1: $t := \bar{a} \cdot \bar{b}$
2: $m := t \cdot n \mod r$
3: $u := (t + m \cdot n) / r$
4: If $u \geq n$ then return $u - n$
Else return $u$

Figure 1. Modular exponentiation using Montgomery multiplication

B. Residue Number System

Residue Number System (RNS) is an efficient parallel technique for performing arithmetic operations (addition, subtraction and multiplication). In (Bajard et al., 2004) the RNS system have been studied and used in Montgomery's algorithms. In RNS, large numbers replace with their residues and the operations perform on these residues in parallel and without carry propagation.

In the RNS there is a set of moduli $B = (m_1, m_2, ..., m_k)$ that is called RNS base, where $m_i$ s are relatively prime to each other. For each positive integer $x$, such that $0 \leq x < M$ where $M = \prod_{i=1}^{k} m_i$, there is a unique sequence $(x_1, x_2, ..., x_k)$ of positive integers, where $x_i = x \mod m_i$, this sequence is called the RNS representation of $x$. If $(y_1, y_2, ..., y_k)$ are RNS representation of $X$ and $Y$, Eq. 1 shows that how arithmetic operations perform in the RNS system.

\[
X \pm Y = \left( |(x_1 \pm y_1)|_{m_1}, |(x_2 \pm y_2)|_{m_2}, \ldots, |(x_k \pm y_k)|_{m_k} \right)
\]
\[
X \times Y = \left( |(x_1 \times y_1)|_{m_1}, |(x_2 \times y_2)|_{m_2}, \ldots, |(x_k \times y_k)|_{m_k} \right)
\]  
(1)

Converting $X$ to its RNS representation is called forward conversion and converting $(x_1, x_2, ..., x_k)$ back to its binary representation is called reverse conversion. Forward conversion is done by reduction in modulo $m_i$ s (Esmaeildoust et al., 2014; Gerami et al., 2011). Reverse conversion can perform in two manners: CRT and MRC methods. Eq. 2 shows reverse conversion with CRT method where $M = \prod_{i=1}^{k} m_i$, $M_i = M / m_i$, $N_i = |M_i^{-1}|_{m_i}$ is multiplicative inverse of $M_i$ in modulo $P_i$.

\[
X = \left\lfloor \sum_{i=1}^{k} x_i N_i |m_i M_i \right\rfloor \mod M
\]  
(2)

Eq. 3, 4 shows two MRC method for reverse conversion, where $|m_i^{-1}|_{m_j}$ is multiplicative inverse of $m_i$ in modulo $m_j$. 

12
\[ X = a_n \prod_{i=1}^{a-1} m_i + \cdots + a_2 m_2 m_1 + a_1 \quad (3) \]

\[ a_1 = x_1 \]
\[ a_2 = \left| (x_2 - a_1) | m_1^{-1} | m_2 \right| m_2 \]
\[ \vdots \]
\[ a_k = \left| \left( (x_k - a_1) | m_1^{-1} | m_k - a_2 \right) | m_2^{-1} | m_k - \cdots - a_{k-1} \right| m_{k-1}^{-1} | m_k \right| m_k \]

Horner’s MRC algorithm for reverse conversion is shown in Eq.4:
\[ X = a_1 + m_1(a_2 + m_2(a_3 + \cdots + m_{k-1}a_k) \ldots) \quad (4) \]

C. RNS Montgomery Multiplication

RNS Montgomery multiplication (Bajard et al., 2009) which is shown in Fig.2 this computes the remainders without any divisions similar to Montgomery multiplication and furthermore it performs additions, subtractions and multiplication of large numbers in RNS system, therefore it is more efficient than the Montgomery multiplication.

RNS Montgomery multiplication has two RNS bases; first RNS base \( B = (m_1, m_2, \ldots, m_k) \) and second RNS base \( B’ = (m_1’, m_2’, \ldots, m_k’) \). The RNS representation of \( X \) and \( Y \) in first RNS base are \( (x_1, x_2, \ldots, x_k) \) and \( (y_1, y_2, \ldots, y_k) \), and in second base are \( (x_1’, x_2’, \ldots, x_k’) \) and \( (y_1’, y_2’, \ldots, y_k’) \). \( GCD(T, M) = GCD(T, M’) = GCD(M, M’) = 1 \) and \( T < M < M’ \).

\[
\begin{align*}
\text{RNS Montgomery Multiplication} \\
1: & \quad D = X \times Y \left( d_i = \left| x_i \times y_i \right| m_i \text{ in base } B, \quad d_i’ = \left| x_i’ \times y_i’ \right| m_i’ \text{ in base } B’ \right) \\
2: & \quad q_i = \left| d_i \times T^{-1} \right| m_i \text{ in } B \\
3: & \quad q_i \text{ in } B \rightarrow q_i’ \text{ in } B’ \\
4: & \quad r’ = (d_i’ - q_i’ \times N_i’) M^{-1} \text{ in } B’ \\
5: & \quad r \text{ in } B \leftarrow r’ \text{ in } B’ 
\end{align*}
\]

Figure.2. RNS Montgomery multiplication algorithm

The time complexity of RNS Montgomery multiplication is in lines 3, 5 because the lines 1, 2, 4 are completely RNS operations and perform in parallel. This time complexity as is reported in (Bajard et al., 2009; Geramiet al., 2011; Esmaeildoust et al., 2014) is shown in Eq. 5.

\[
\text{Latency}_{\text{MonMult}} = \frac{\text{Latency}_{\text{RNS–RNS}}}{\text{from first to second RNS base}} + \frac{\text{Latency}_{\text{RNS–RNS}}}{\text{from second to first RNS base}}
\]

\[
\text{Latency}_{\text{RNS–MRS}} + \text{Latency}_{\text{MRS–RNS}}
\]

\[
\text{Latency}_{\text{RNS–weighted}} + \text{Latency}_{\text{weighted–RNS}}
\]
III. Proposed Moduli sets

The previous RNS bases for RNS Montgomery multiplication and our proposed RNS bases are shown in Table 1.

As is shown in Table 1, previous works use 4-moduli set with moduli in the form of $2^m - c_i$ where $0 \leq c_i < 2^{m/2}$ as first RNS base. Because of its small hamming weight we also use this moduli set as first RNS base in this work. In (Noorimhe et al., 2014) an efficient reverse convertor with RNS base $\{2^{n-1}, 2^n, 2^n + 1, 2^{2n} - 1\}$ is reported. We use this RNS base as second RNS base and use its reverse convertor in RNS Montgomery multiplication and we show that with use of these RNS bases, the efficiency of RNS Montgomery multiplication will be better than the previous works.

### Table I

<table>
<thead>
<tr>
<th></th>
<th>Previous RNS Base</th>
<th>Proposed RNS Base</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$ (Bajardet al., 2009)</td>
<td>${2^m - c_1, 2^m - c_2, 2^m - c_3, 2^m - c_4}$</td>
<td>${2^m - c_5, 2^m - c_6, 2^m - c_7, 2^m - c_8}$, $0 \leq c_i &lt; 2^{m/2}$</td>
</tr>
<tr>
<td>$s_2$ (Geramiet al., 2011)</td>
<td>${2^m - c_1, 2^m - c_2, 2^m - c_3, 2^m - c_4}$</td>
<td>${2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1}$</td>
</tr>
<tr>
<td>$s_3$ (Geramiet al., 2011)</td>
<td>${2^m - c_1, 2^m - c_2, 2^m - c_3, 2^m - c_4}$</td>
<td>${2^n - 1, 2^n, 2^n + 1, 2^{2n+1} + 1}$</td>
</tr>
<tr>
<td>$s_4$ (Esmaeildoust et al., 2014)</td>
<td>${2^m - c_1, 2^m - c_2, 2^m - c_3, 2^m - c_4}$</td>
<td>${2^n - 1, 2^n, 2^n + 1, 2^{2n+1} + 1}$</td>
</tr>
<tr>
<td>$s_5$ Prop. MS</td>
<td>${2^m - c_1, 2^m - c_2, 2^m - c_3, 2^m - c_4}$</td>
<td>${2^n - 1, 2^n, 2^n + 1, 2^{2n+1} + 1}$</td>
</tr>
</tbody>
</table>

The dynamic range of proposed first and second moduli set is $4m$ and $5n$ respectively. We choose $m$ and $n$ such that $4m < 5n$. This was one of the Montgomery multiplication prerequisite. $keyLength \leq 5n$ and $keyLength \leq 4m$, so that the encryption key has a valid RNS representation with these RNS bases. According to these conditions, for key length of 256 bit, $m=64$ and $n=52$. Table 2 shows our proposed RNS bases for key length of 160, 192 and 256.

### Table II

<table>
<thead>
<tr>
<th></th>
<th>Proposed Moduli Sets for Key Length of 160, 192 and 256 Bit</th>
</tr>
</thead>
<tbody>
<tr>
<td>160 key length</td>
<td>192 key length</td>
</tr>
<tr>
<td></td>
<td>First Base</td>
</tr>
<tr>
<td>2^{40} - 2^1 - 1</td>
<td>2^{32}</td>
</tr>
<tr>
<td>2^{40} - 2^{29} - 1</td>
<td>2^{32} - 1</td>
</tr>
<tr>
<td>2^{40} - 2^{37} - 1</td>
<td>2^{32} + 1</td>
</tr>
<tr>
<td>2^{40} - 2^{39} - 1</td>
<td>2^{63} - 1</td>
</tr>
</tbody>
</table>

A. The latency of RNS to MRS conversion

We use the moduli set $\{2^m - 2^{t_1} - 1, 2^m - 2^{t_2} - 1, 2^m - 2^{t_3} - 1, 2^m - 2^{t_4} - 1\}$ where $t_i < m/2$, as the first base. The elements of this moduli set are in the form of $2^m - c_i$, where $0 \leq c_i < 2^{m/2}$, so according to (Bajardet al., 2009) the latency of RNS to MRS is shown in Eq. 6.
Where $\omega(m_{i,j}^{-1})$ is the hamming weight of $m_{i,j}^{-1}$ and $\omega(m_{i,j}^{-1}) = \frac{m-i}{|t_i-t_j|}$. For selection of $t = \{t_1, t_2, t_3, t_4\}$ an exhaustive search on $t_i$ s are done and all possible permutations in the relative positions of the elements have been considered and finally the elements that has the below conditions are selected.

- $1 \leq t_i < m/2$
- $t_1 \neq t_2 \neq t_3, \neq t_4$
- $((m - t_i)/|t_i - t_j|) \in N$
- $Latency_{RNS-MRS}$ be minimum

For example for key length of 256, $Latency_{RNS-MRS} = 41 \cdot T_{FA}$

### B. The latency of MRS to RNS conversion

Converting the MRS form into RNS form is done by computing the residues in modulo $2^n - 1$, $2^n + 1$, and $2^{2n} + 1$. These operations are done in parallel so the latency of MRS to RNS is equal the worst latency of these moduli. The modulus that has the worst latency is called the critical modulus. The residue in modulo $2^n$ is just the $n$ lowest bits, so it isn’t the critical modulus. We can find out the critical modulus by comparing the unit gate delay of these moduli. According to (Jaberipourand Parhami, 2009) these unit gate delay are computed and are shown in Table 3.

### TABLE III

<table>
<thead>
<tr>
<th>Modulus</th>
<th>Unit Gate Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>$2^n - 1$</td>
<td>$2 \log(n) + 3$</td>
</tr>
<tr>
<td>$2^n + 1$</td>
<td>$2 \log(n) + 5$</td>
</tr>
<tr>
<td>$2^{2n} - 1$</td>
<td>$2 \log(n - 0.5) + 5$</td>
</tr>
</tbody>
</table>

As we see in Table 3 the critical modulus is $2^n + 1$. In (Esmaeildoust et al., 2014; Gerami et al., 2011) the latency of MRS to RNS for the modulus $2^n + 1$ is computed and here is shown in Eq. 7.

$$Latency_{MRS-RNS} = \left(\sum_{i=1}^{k-1} MA(2^n + 1) + 4\right)T_{FA}$$

Considering the latency of $MA(2^n + 1)$ as $2n T_{FA}$ (Bayoumi et al., 1987), Eq. 7 can be rewritten in Eq. 8.

$$Latency_{MRS-RNS} = (6n + 12)T_{FA}$$
C. The latency of RNS to Weighted conversion

RNS to weighted conversion denotes the reverse conversion. We use the reverse convertor that has been reported in (Noorimehet al., 2014). The delay of this reverse convertor has been reported in (Noorimehet al., 2014) as \((7n + 3)T_{FA}\) so the latency of RNS to Weighted conversion is:

\[
\text{Latency}_{\text{RNS-weighted}} = (7n + 3)T_{FA}
\]  

(9)

D. The latency of Weighted to RNS conversion

Converting the weighted form into RNS form means to compute the residue of \(X\) in modulo \(2^m - 2^{i_t} - 1\). That is denoted by \(|X|_{2^m - 2^{i_t} - 1}\). As proved in (Geramiet al., 2011; Esmaeildoustet al., 2014) the latency of computing \(|X|_{2^m - 2^{i_t} - 1}\) is shown in Eq. 10

\[
\text{Latency}_{\text{weighted-RNS}} = \left( \sum_{i=1}^{k-1} (2\omega(c'_j) + 2) \right) mT_{FA} = 18mT_{FA}
\]  

(10)

IV. Comparison

The total latency of RNS Montgomery multiplication is as below:

\[
T_{\text{Total}} = \text{Latency}_{\text{MonMul}} = \text{Latency}_{\text{RNS-MRS}} + \text{Latency}_{\text{MRS-RNS}} + \text{Latency}_{\text{RNS-weighted}} + \text{Latency}_{\text{weighted-RNS}}
\]  

(11)

For example, for key length of 256 bit the total latency of our proposed moduli set and the previous moduli sets are shown in Eq. 12 – 15.

\[
T_{\text{Total}}(S_1) = 60mT_{FA} + (6n + 9)T_{FA} + (12n + 5)T_{FA} + 18mT_{FA}
\]  

(12)

\[
= (78m + 18n + 14)T_{FA}
\]

\[
T_{\text{Total}}(S_2) = 39mT_{FA} + (12n + 9)T_{FA} + (8n + 3)T_{FA} + 18mT_{FA}
\]  

(13)

\[
= (57m + 20n + 12)T_{FA}
\]

\[
T_{\text{Total}}(S_3) = 41mT_{FA} + (6m + 9)T_{FA} + ((23m + 12)/2)T_{FA} + 18mT_{FA}
\]  

(14)

\[
= (65m + (23m/2) + 15)T_{FA}
\]

\[
T_{\text{Total}}(S_4) = 41mT_{FA} + (6n + 12)T_{FA} + (7n + 3)T_{FA} + 18mT_{FA}
\]  

(15)

\[
= (59m + 13n + 15)T_{FA}
\]

The total latency for key length of 60 to 260 for our proposed moduli set and other previous moduli sets are shown in Fig. 3. As we see, in all key length the total latency of Montgomery multiplication with our proposed moduli set is less than its total latency with previous moduli sets.
In (Esmaeildoust et al., 2014; Gerami et al., 2011) their RNS bases are compared with (Bajard et al., 2009), and they show that their RNS bases are more efficient than (Bajard et al., 2009). As we demonstrated above, our proposed RNS bases are more efficient than (Esmaeildoust et al., 2014; Gerami et al., 2011) so we can conclude that in compare with (Bajard et al., 2009) our RNS bases are more efficient.

![Figure.3. Total latency of RNS to RNS conversion](image)

V. Conclusions and future remarks
This article proposes RNS bases for use in the RNS version of RSA encryption method. A moduli set with advantage of small hamming weight are employed as first base and a moduli set with efficient reverse converter are used as second base. Then the latency of RNS Montgomery multiplication are computed and compared for $S_1, S_2, S_3, S_4$ and our proposed moduli set. We demonstrated that in all key length with our proposed RNS base, the latency of RNS to RNS conversion decreased and therefore an efficient RNS Montgomery multiplication is achieved.

Also in future we can make the RNS Montgomery multiplication better and decrease its latency with these following suggestions:

- Introducing a moduli set with less hamming weight for first moduli set.
- Introducing a moduli set with more numbers of elements (e.g. five or six-moduli set) that cause to higher dynamic range and thus higher key length will be covered.
- Improving the reverse converter
References


vii. Cao B, Srikanthan T and Chang CH (2005). "Efficient Reverse Converter for the 4-Moduli Sets \( \{ 2^n - 1, 2^n + 1, 2^{2n+1} - 1 \} \) and \( \{ 2^n - 1, 2^n, 2^{2n+1} + 1 \} \). IEEE Transactions on Circuits and Systems, Vol(152), pp 687-696.


xiv. Mohan PVA and Premkumar AB (2007). "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), pp 1245-1254.

xv. Molahosseini AS, Dadkhah C, Navi K, Esghighi M (2009). "Efficient MRC-Based Residue to Binary Converters for the New Moduli Sets \( \{ 2^{2n}, 2^n - 1, 2^{n+1} - 1 \} \) and \( \{ 2^{2n}, 2^n - 1, 2^{n+1} - 1 \} \). IEEE TRANSACTIONS ON INFORMATION AND SYSTEMS, Vol (E92-D), pp 1628-1638.

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

xviii. Noorimehr MR, Hoseinzade M, Navi K (2014). "Efficient Reverse Converters for 4-Moduli Sets \( \{2^{2n-1} - 1, 2^n, 2^n + 1, 2^n - 1\} \) and \( \{2^{2n-1}, 2^{2n-1} - 1, 2^n + 1, 2^n - 1\} \) Based on CRTs Algorithm". *Springer: Circuits Systems and Signal Processing*, Vol (33), pp 3145-3163.