Reverse Converter for the Moduli Set \( \{2^n-1, 2^n, 2^n+1\} \) Base on Grouping Number

Saeid Banhanfar\(^1\), Nadali Zarei\(^2\)

\(^1\)Dept. of engineering, Imam Housein University, Tehran, Iran

\(^2\)Dept. of engineering, Imam Housein University, Tehran, Iran

Abstract
The Residue Number System (RNS) is a non-weighted system that is very efficient in digital signal processing and communicational applications. The previous proposed methods for the residue to binary (R/B) conversions are based on the Chinese Reminder Theorem (CRT) or Mixed Radix Conversion (MRC). These theorems are difficult to implement. In this paper, we present a new high-speed ROM-less residue to binary converter for the three moduli set of \( \{2^n-1, 2^n, 2^n+1\} \). Our technique unlike previous methods uses the grouping numbers in dynamic representation range \( M \) which its delay is much less than other converters.

**Keywords:** residue number system, reverse converter, moduli set \( \{2^n-1, 2^n, 2^n+1\} \), grouping number.

1. Introduction
Residue Number System is an unconventional system. In this system, an integer \( X \) is represented by its reminder modulo a number of different bases. These residue numbers are smaller than the original number in the conventional system, so computations can be done with more speed and low power [1]. The advantages of RNS for implementing digital signal processors for certain applications such as FIR filtering are well-known [2-5]. Some of the more recent applications have been for 1-D filtering [6-8], 2-D filtering [9], video filtering [10], RSA cryptography [11-14], Elliptic curve cryptography [15], m-ary orthogonal keyed communication scheme [16], general purpose RISC DSP [17] and Image processing [18].

The RNS is determined using a set of relative pair wise prime integers positive co-prime integers \( \{m_1, m_2, ..., m_n\} \) as moduli set. The dynamic range \( M \) of that system is given as a product of the moduli \( m_i \) where

\[
M = \prod_{i=1}^{n} m_i.
\]  

Any integer \( X \) between 0 and \( M - 1 \) can be uniquely represented as \( (x_1, x_2, ..., x_n) \). The residues \( x_i = \lfloor X/m_i \rfloor \), also called residue digits, are defined as

\[
(2) \ x_i = X \mod m_i, \quad 0 \leq x_i < m_i.
\]

The two most important issues for the residue arithmetic are the choice of moduli sets and the conversion of residue to binary numbers. The choice of moduli set in RNS is of continuing interest. Early designs of RNS-based processors were largely based on ROMs which used small set of mutually prime integers to realize a large dynamic range. However, the R/B converters for the general moduli sets are hardware intensive and implemented based on LUTs (Look-up tables). The access time of the LUTs and the need to read these iteratively have made the implementations inefficient for ASIC realization for RNS with large dynamic range. Hence, the more recent trend has been to use moduli sets which can help to eliminate the ROMs. These are known as power-of-two related moduli sets or “conversion-friendly” moduli sets [19].

In residue-based processors, designing an efficient R/B converter is very important. Considerable emphasis has been put on the popular moduli sets like \( \{2^n-1, 2^n, 2^n+1\} \), \( \{2^n-1, 2^n, 2^n+1\} \), and \( \{2^n-1, 2^n, 2^{n+1}-1\} \) [3, 20-25]. The arithmetic processors of these sets and the corresponding residue to binary converters make them very attractive compared with other sets [26]. The powers-of-two related moduli sets have the advantage that all operations required in digital signal processing applications such as modulo addition, modulo subtraction, modulo multiplication, scaling and FIR filtering can be efficiently performed due to the attractive arithmetic properties of these moduli [19].

The three moduli set \( \{2^n-1, 2^n, 2^n+1\} \) is of special interest because several operations in this system can be performed efficiently with limited amount or even without ROM. The periodicity properties exhibited by three moduli of this RNS result in superb performance of the binary to residue converter and modulo addition even for large \( n \) [27].
Section 2 describes how the conversion of RNS to binary system using the new approach. Section 3 presents the hardware implementation and in section 4, the proposed design is compared with other reported converters.

2. Proposed Method

For residue to binary conversion in moduli set \(\{2^n-1, 2^n, 2^n+1\}\), we distribute the numbers in dynamic representation range \(M\) into several groups and subgroups which a part of this novel idea is presented in [28]. Since, residue representation of \(X\) in above moduli set is corresponding with \((x_1, x_2, x_3)\), then the three residues denotes as:

\[
\begin{align*}
x_1 &= X \mod 2^n - 1 = x_{1,n-1}x_{1,n-2} \cdots x_{1,0} \\
x_2 &= X \mod 2^n = x_{2,n-1}x_{2,n-2} \cdots x_{2,0} \\
x_3 &= X \mod 2^n + 1 = x_{3,n}x_{3,n-1} \cdots x_{3,0}
\end{align*}
\]

So, the group number of any residue number in the considered moduli set obtains according to Figure 1.

For example, the value of \(\beta\) for numbers in first group with range \([0, 2^{2n}+2^n]\) is shown in table 1.

<table>
<thead>
<tr>
<th>Number</th>
<th>Subgroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 \rightarrow 2^n</td>
<td>0</td>
</tr>
<tr>
<td>(2^n+1 \rightarrow 2(2^n+1) - 1)</td>
<td>1</td>
</tr>
<tr>
<td>(2^n(2^n+1) \rightarrow 3(2^n+1) - 1)</td>
<td>2</td>
</tr>
<tr>
<td>(\vdots)</td>
<td>(\vdots)</td>
</tr>
<tr>
<td>((2^n-1)(2^n+1) \rightarrow 2^n(2^n+1) - 1)</td>
<td>(2^n-1)</td>
</tr>
</tbody>
</table>

For determination of group number of any residue number, first should be get the value of \(\omega\) For clarity, we have exhibited it in range \([0, 2^{2n}+2^n]\) as follows:

\[
\omega = |x_1 - x_3|_{2^{-1}} \Rightarrow \begin{cases} 
0 \leq X < 2^n+1, & \omega = 0 \\
2^n+1 \leq X < 2(2^n+1), & \omega = 2 \\
2(2^n+1) \leq X < 3(2^n+1), & \omega = 4 \\
\vdots & \vdots \\
(2^n-1)(2^n+1) \leq X < 2^n(2^n+1), & \omega = 2^{2n-2} \\
2^n(2^n+1) \leq X < (2^n+1)(2^n+1), & \omega = 2^{2n-1} \\
\vdots & \vdots \\
(2^n-1)(2^n+1) \leq X < 2^n(2^n+1), & \omega = 2^{2n-3} \\
(2^n-1)(2^n+1) \leq X < 2^n(2^n+1), & \omega = 0 
\end{cases}
\]

According to (7) and with regard to the product result from moduli subtraction in each group be appeared first, odd values and afterward even respectively. Since, in order to accomplishment of arithmetic operations should be arranged the \(\omega\) values increasingly, so it is achievable through one bit right rotate. Therefore, if assume \(\omega = 0, 2, 4, 6, \ldots, 2^n-2, 1, 3, \ldots, 2^n-3, 2^n-2\), after 1-bit right rotate, we get \(\omega' = 0, 1, 2, \ldots, 2^n-3, 2^n-2\).

Thus, by having the values of \(\beta\) and \(\omega'\), the group number of any residue number in RNS (counting from 0) is defined as

\[
\alpha = |\omega' - \beta|_{2^{-1}}, \quad 0 \leq \alpha \leq 2^n - 2.
\]

Table 2 shows the distribution of numbers in dynamic range \([0, 2^{2n} - 2^n]\) which is given as a product of the \(m_i\)’s in moduli set \([2^n-1, 2^n, 2^n+1]\).

<table>
<thead>
<tr>
<th>Number</th>
<th>Group</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 \rightarrow 2^n \rightarrow \cdots \rightarrow 2^n(2^n+1) - 1</td>
<td>0</td>
</tr>
<tr>
<td>2^n(2^n+1) \rightarrow 2[2^n(2^n+1)] - 1</td>
<td>1</td>
</tr>
<tr>
<td>\vdots</td>
<td>\vdots</td>
</tr>
<tr>
<td>(2^n-2)[2^n(2^n+1)] \rightarrow (2^n-1)[2^n(2^n+1)] - 1</td>
<td>(2^n-2)</td>
</tr>
</tbody>
</table>

Therefore, after the determination of group and subgroup numbers of any number in RNS, its
corresponding number in the binary system is achievable according to equation

\[ X = \alpha \cdot 2^n (2^n + 1) + \beta (2^n + 1) + x_3 \quad (9) \]

Since, for multiplying one \( n \)-bit number in 2 is sufficient to do one bit left cyclic shift of number, then the value of \( \alpha \cdot 2^n \) is computed as the \( n \)-bit left cyclic shift of \( \alpha \). In other words, we have

\[ x_{n-1} \cdot x_{n-2} \cdot x_{n-3} \ldots x_1 \cdot x_0 \times 2^n = x_{n-1} \cdot x_{n-2} \cdot x_{n-3} \ldots x_1 \cdot x_0 \cdot 000 \ldots 0 \]

(10)

Also, in order to multiplication of \( n \)-bit number in \( 2^n+1 \), we get

\[ x_{n-1} \cdot x_{n-2} \cdot x_{n-3} \ldots x_1 \cdot x_0 \times (2^n + 1) = x_{n-1} \cdot x_{n-2} \cdot x_{n-3} \ldots x_1 \cdot x_0 \cdot 000 \ldots 0 \times 2^n \]

(11)

Therefore, to achieve the product of a \( n \)-bit number by \( 2^n \cdot (2^n+1) \) in (9), will be got:

\[ x_{n-1} \cdot x_{n-2} \cdot x_{n-3} \ldots x_1 \cdot x_0 \times 2^n \cdot (2^n + 1) = x_{n-1} \cdot x_{n-2} \cdot x_{n-3} \ldots x_1 \cdot x_0 \cdot 000 \ldots 0 \]

(12)

Now equation (9) can be rewritten as

\[ X = \alpha x_0 \cdot 000 \ldots 0 + \beta x_0 \cdot x_1 \ldots x_{n-1} \cdot x_3 \]

(13)

As be seen, \( X \) is the \( 3n \) bits number. For computation of the lower \( n \) bits of \( X \), we can defined \( \theta \) according to

\[ \theta = \beta + x_0 \cdot x_1 \ldots x_{n-1} \cdot x_3 \]

(14)

In the other hand, because the lower \( n \) bits of \( X \) in binary system is equal to the division reminder of number modulo \( 2^n \), then can be concluded in the considered moduli set, no need to computation of \( \theta \), due to the lower \( n \) bits of \( \theta \) is achievable through \( x_2 \),

\[ \theta_{n-1}\theta_{n-2}\ldots\theta_0 = x_2 \]

(15)

So, the most significant bit (MSB) of \( \theta \) is required to computed only and it denotes the carry bit, namely

\[ C_{out\theta} = \theta_n = G_{0,n-1} = C1 \]

(16)

As we know, the \( x_3 \) is the residue of the binary number mod \( 2^n+1 \) and the largest value of \( x_3 \) is equal to \( 000 \ldots 0 \).

In this case, in addition of \( \beta + x_3 \) which \( \beta \) is a \( n \) bit number, we will be \( C_{out\beta} = 0 \). Notice that, only while \( x_3 = 000 \ldots 0 \), we can be had \( C_{out\beta} = 1 \). Hence, the two bits of \( C_{out\beta} \) and \( x_{3,n} \) are not one simultaneously. So we get

\[ \delta = C_{out\beta} + x_{3,n} \]

(17)

Therefore, according to the all mentioned issues, can be said

\[ X_{n-1,0} = x_2 = x_{2,n-1} \cdot x_{2,n-2} \cdot x_{2,0} \]

\[ C1 = C_{out\theta} \]

\[ X_{2n-1,n} = \alpha + \beta + C_{out\theta} + x_{3,n} = \alpha + \beta + \delta \]

\[ C2 = C_{out\theta} = MSB(\alpha + \beta + \delta) \]

(18)

\[ X_{3n-1,2n} = \alpha + C2 \]

Due to, the most significant higher \( 2n \) bits of \( X \) gives as

\[ X_{3n-1,2n} = \alpha + \beta + \delta \]

(19)

Therefore, the final form of (7) can be rewritten as

\[ X = X_{3n-1,2n} \cdot x_2 \]

(20)

Thus, (20) is implemented as a simple concatenation of \( X_{n-1,0} \), \( X_{2n-1,n} \) and also \( X_{3n-1,2n} \). Implementation of equation (20) is shown in Figure 2.

<table>
<thead>
<tr>
<th>( \alpha + 1 )</th>
<th>( \alpha )</th>
<th>( \beta )</th>
<th>( \delta )</th>
<th>( x_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Mux</td>
<td>EAC-PPA</td>
<td>C2</td>
<td>( X_{3n-1,2n} )</td>
<td>( X_{2n-1,n} )</td>
</tr>
</tbody>
</table>

**Fig. 2 reverse conversion unit**

### 3. Hardware Structure

The group detection function is determined by Eq.(8) as

\[ \alpha = [\alpha']^{2n-1} \]

According to [28], since \( \alpha \) is computed as a residue modulo \( 2^n - 1 \), then, instead of subtracting \( \beta^{2^n-1} \) we can add its additive inverse modulo \( 2^n - 1 \). An additive inverse modulo \( 2^n - 1 \) is simply a negation of
binary representation. For simplification reasons the additive inverse of \( |\beta|_{2^n-1} \) is denoted as
\[
\hat{\beta} = -|\beta|_{2^n-1}.
\]  
(21)

So that, the binary form of (21) is \( \hat{\beta} = \bar{\beta}_{n-1} \ldots \bar{\beta}_1 \bar{\beta}_0 \).

Thus (8) can be rewritten as the sum
\[
\alpha = |\omega' + \hat{\beta}|_{2^n-1}.
\]  
(22)

From [29], an addition modulo \( 2^n - 1 \) with redundant zero elimination can be expressed as
\[
[a + b]_{2^n-1} = [a + b + c_{out} + p]_{2^n}.
\]  
(23)

where \( c_{out} \) is a carry bit of \( a + b \) addition and \( p = 1 \) for \( a + b = 11 \ldots 1_2 \). The sum \( c_{out} + p = 0 \) for \( a + b < 2^n - 1 \) and \( 1 \) for \( a + b \geq 2^n - 1 \) [30]. By assuming that \( C_{in} = c_{out} + p \), the final form of (22) is then
\[
\alpha = |\omega' + \hat{\beta} + C_{in}|_{2^n}.
\]  
(24)

Also, the values of \( \omega \) and \( \beta \) is given using this way. Notice that, in computing of \( \omega = x_1 - x_3 \) \( 1_{2^n-1} \), because \( x_j \) is a residue number modulo \( 2^n +1 \) and \( x_3 \leq 2^n \) then \( x_3 \) \( 1_{2^n-1} \) is given by OR-ing the least and the most significant bits of \( x_j \). Therefore, binary form of \( x_3 \) \( 1_{2^n-1} \) is \( x_{3,0} + x_{3,n-1} \ldots x_{3,1} \).

Proposed method for the numbers conversion from residue system to binary system is implemented with parallel prefix structure including parallel-prefix adder and end-around-carry prefix adder, both of which are introduced in [29]. A parallel prefix adder and also parallel prefix adder with end-around-carry are built from elements shown in Figure 3.

The signals \( G_{i,j} \) and \( P_{i,j} \) are the carry generation and propagation functions from the position \( i \) to \( j \). For an addition of two binary vectors \( a_{n-1} \ldots a_0 \) and \( b_{n-1} \ldots b_0 \) and for \( i < k < j \), these functions can be expressed by logic equations
\[
G_{i,j} = a_i \cdot b_i \\
P_{i,j} = a_i \oplus b_i \\
G_{i,j} = G_{i,k} \cdot P_{k+1,j} + G_{k+1,j} \\
P_{i,j} = P_{k,j} \cdot P_{k+1,j}.
\]  
(25)

The carry signals \( c_j \) are equal to \( G_{0,j-1} \) and the bits \( s_j \) of a final sum are \( s_j = P_{j,j} \oplus c_j \). An addition advantage of prefix structures is that the end-around carry can be added in the last stage with a delay cost of two logic levels [30]. The detailed description of this idea is presented in [29].

In this paper, the standard unit-gate model [29] used to estimate the area and time (AT) characteristics of proposed design in order to reverse conversion of numbers by moduli set \( \{2^n-1, 2^n, 2^n+1\} \). In this model, each two-input monotonic gate (e.g. AND, OR, NAND, NOR), and also XOR/XNOR gate counts as one and two gates respectively.

The Group Number Detection (GND) unit of shown in Fig. 1 comprises three main adders: one modulo \( 2^n \) adder and two adder mod \( 2^n -1 \). For calculation of \( \beta \) modulo \( 2^n \), we used the parallel prefix adder from [29] by \( A = 5n + (3/2)n \log_2 n + 4 \) and \( T = 2 \log_2 n + 6 \). Also, for determination of values \( \omega \) and \( \alpha \), applied the end-around-carry prefix adder which its hardware and delay are \( A = 8n + (3/2)n \log_2 n - 3 \) and \( T = 2 \log_2 n + 6 \). Consequently, area and time of GND unit are:

\[
A_t = 21n + \frac{9}{2}n \log_2 n - 2 \\
T_t = 4 \log_2 n + 12.
\]  
(26)

As we mentioned already, since lower \( n \) bits of the decoded number are available directly as residue corresponding to modulus \( 2^n \), then without need to the hardware components, can be efficiently implemented. C1, the product carry bit from \( \beta + x_3 \) addition be computed using the carry generation unit which is shown in Figure 4.

---

Fig. 3 Blocks of prefix adder structure [30].
The carry generation unit uses \((n-1)\) black nodes and \(n\) input nodes (denoted as square). Its area and time can be expressed as

\[
A_2 = 3n + 3(n-1) = 6n - 3 \\
T_2 = 2\log_2 n + 2 \tag{27}
\]

For the calculation of middle \(n\) bits from the position \(X_n\) to \(X_{2n-1}\), applied the new and modified structure of end-around-carry parallel-prefix adder namely (EAC-PPA) which the output carry signal is determined by \(C_2\). Figure 5 is used to compute the value of \(\alpha + \beta + \delta\) and \(C_2 = C_{out}\).

So, the required hardware and time for generation of bits \(n\) to \(2n-1\) of the binary number can be expressed as

\[
A_4 = 29n + 6n \log_2 n \\
T_4 = 6\log_2 n + 20 \tag{29}
\]

It is shown in EAC-PPA circuit that delay of generation of \(C2\) is equal to \(6\log_2 n + 17\).

According to Fig. 2, the most significant bits of \(X\) in binary system is given by \(\alpha\) or \(\alpha + 1\). The output carry signal from the circuit shown in Fig. 5 be called \(C2\) is selecting line of multiplexer which determine whether \(\alpha\) be directed to output or \(\alpha + 1\). In order to computation of \(\alpha + 1\), is sufficient after the value determination of \(\alpha\), be add with 1.

Therefore, the circuit \(\alpha + 1\) perform the function of adding 1 to a \(n\)-bit input number. Consider

\[
\alpha = \alpha_0, \alpha_1, \alpha_2, ..., \alpha_{n} \alpha_{n+1} \alpha_{n+2} ..., \alpha_{2n-1} = \epsilon_{n}, \epsilon_{n+1}, ..., \epsilon_{2n-1}
\]

We have the following equation, which imply that the circuit plus 1 requires \(n-1\) XOR gates and \(n\) AND gates plus 1 inverter.

\[
\epsilon_0 = \epsilon_{0, \alpha_0}, \epsilon_{1, \alpha_0} \oplus \epsilon_{1, \alpha_1}, \epsilon_{n, \alpha_0} \oplus \epsilon_{n, \alpha_n}, \epsilon_{n+1, \alpha_0} \oplus \epsilon_{n+1, \alpha_{n+1}, ..., \epsilon_{2n-1, \alpha_{2n-1}}}
\]

The circuit requires the hardware of \(3n\) and delay of 2 logic levels. Thus, the sum of them consist of

\[
A_5 = 23n + \frac{9}{2} n \log_2 n - 2 \tag{30}
\]

\[
T_5 = 4\log_2 n + 14.
\]

Since, \(AT\) parameters of \(n\) bits Mux are \(3n\) and 2 respectively, then area and delay of generation of bits from position \(2n\) to \(3n-1\) are

\[
A_6 = 32n + 6n \log_2 n \\
T_6 = 6\log_2 n + 19 \tag{31}
\]

Total delay of the circuit is determined by a path consisting one unit of group detection, \(C2\) generation unit and multiplexer. The total area and delay of the designed reverse converter circuit are

\[
A_{tot} = A_2 + A_3 + A_5 + A_{OR} + A_{max} = 4\ln + 6n \log_2 n - 1 \\
T_{tot} = T_2 + T_3 + T_5 + T_{max} = 6\log_2 n + 19 \tag{32}
\]

4. Conclusions

Reverse converter is one of the most important issues in residue number system. In this paper, a novel and fast algorithm for the conversion of numbers given in RNS \(\{2^n-1, 2^n, 2^n+1\}\) is presented. Our proposed technique is based on grouping numbers which has significant reduction in delay, compared to other methods. Furthermore, it accomplishes reverse conversion without applying generic approaches such as CRT and MRC.
References


