A Fully Parallel Reverse Converter

Mehdi Hosseinzadeh, Amir Sabbagh Molahosseini, Keivan Navi

Abstract—The residue number system (RNS) is popular in high performance computation applications because of its carry-free nature. The challenges of RNS systems design lie in the moduli set selection and in the reverse conversion from residue representation to weighted representation. In this paper, we proposed a fully parallel reverse conversion algorithm for the moduli set \( \{r^i - 2, r^i - 1, r^i\} \), based on simple mathematical relationships. Also an efficient hardware realization of this algorithm is presented. Our proposed converter is very faster and results to hardware savings, compared to the other reverse converters.

Keywords—Reverse converter, residue to weighted converter, residue number system, multiple-valued logic, computer arithmetic.

I. INTRODUCTION

The conventional arithmetic carries propagation based on a weighted number system is the reason for performance degradation in hardware computing systems. The residue number system is a non-weighted number system which speeds up arithmetic operations by dividing them into smaller parallel operations [1], [2]. Since the arithmetic operations in each moduli are independent of the others, there is no carry arithmetic propagation among them and so RNS leads to carry-free addition, multiplication and borrow-free subtraction [3]. RNS is one of the most popular techniques for reducing the power dissipation and the computation load in VLSI systems design [4]. Some applications of the RNS are real-time processing, digital filters [5], [7], digital signal processing (DSP) [8], [9], the RSA encoding algorithm [12] and digital communication [13]. The architecture of the RNS is naturally fault tolerant and consequently, it is used to error detection, error correction and fault tolerance [15].

Despite binary logic in which logical levels are restricted to two possible states, there exists an alternative named multiple-valued logic. In MVL, the number of discrete signal values or logic states extends beyond two. Arithmetic units implemented with MVL achieve more efficient use of silicon resource and circuit interconnections [20]. There is a clear mathematical attraction of using multiple-valued number representation in RNS. The modular arithmetic that is inherent in MVL can be match with modular arithmetic needed in RNS.

The concept of MVL-RNS was introduced by Soderstrand et al. [17] to design a high speed FIR digital filter. The reverse converter proposed in [17] is based on chinese remainder theorem (CRT) and implemented with read-only memories (ROM’s). This converter is practical to implement small and medium RNS dynamic ranges and it is not appropriate for large dynamic ranges. In [18], new RNS systems based on the moduli of forms \( r^i - 2, r^i - 1 \) and \( r^i + 1 \) are presented. Abdallah et al. in [18] developed a systematic framework utilizing high-radix arithmetic for efficient MVL-RNS implementations and proposed many radix-r moduli sets. This moduli sets are not pairwise relatively prime that resulting in reduced dynamic ranges and unbalanced moduli. The reverse converter presented in [18] is based on CRT and because of the scale-down factors that used to makes moduli pairwise relatively prime, conversion delay and cost are increased. In [19], a new moduli set \( \{r^i - 2, r^i - 1, r^i\} \) where \( r = 2k+1, k=1,2,\ldots \) for MVL-RNS was proposed. This moduli set includes pairwise relatively prime and balanced moduli that offers large dynamic range and simple realization of related circuits. The reverse converter presented in [19] is based on CRT and requires many multiplications and reductions, so its area and delay complexities have been increased.

In this paper, a fully parallel reverse conversion algorithm dedicated to the moduli set \( \{r^i - 2, r^i - 1, r^i\} \) where \( r = 2k+1, k=1,2,\ldots \) and an efficient hardware realization are presented. Our proposed converter is faster and has lower hardware cost than the other MVL-based reverse converters for three-modulus sets.

The rest of paper is organized as follows. In section II we introduce the necessary background. The conversion algorithm and its hardware realization are presented in section III. Section IV makes comparison and section V is conclusion.

II. BACKGROUND

Residue Number System: A residue number system is defined in terms of a relatively-prime moduli set \( \{P_1, P_2, \ldots, P_n\} \) that is \(\gcd(P_i, P_j)=1 \) for \( i \neq j \) and \( i, j=1,2,\ldots,n \). A weighted number \( X \) can be represented as \( X=(x_1, x_2, \ldots, x_n) \), where

\[
x_i = X \mod P_i = [X]_{P_i}, 0 \leq x_i < P_i
\]

Such a representation is unique for any integer \( X \) in the range \( [0,M-1] \), where \( M=P_1P_2\ldots P_n \) is the dynamic range of the moduli set \( \{P_1, P_2, \ldots, P_n\} \).

Addition, subtraction and multiplication on residues can be performed in parallel without any carry propagation among the residue digits. Hence, by converting the arithmetic of large numbers to a set of the parallel arithmetic of smaller numbers, the RNS representation yields significant speed up.

The challenges of the RNS system design lie in the moduli set selection and in the conversion of the residues to the
equivalent weighted number. The importance of the moduli selection is due to the fact that the dynamic range, the speed as well as the hardware complexity of RNS systems depend on the forms as well as the number of the moduli chosen. Transforming a weighted integer number to the residue representation is called forward conversion and getting back to the weighted representation is referred to as reverse conversion. Forward conversion is conceivably simple and consists of several modulo adders but the reverse conversion involves considerable degree of complexity and includes a lot of modulo operations. The algorithms of reverse conversion are based primarily on chinese remainder theorem and mixed-radix conversion (MRC).

Chinese Remainder Theorem: by CRT, the number \( N \) is calculated from residues by:

\[
X = \left( \sum_{i=1}^{n} x_i N_i \right) M_i M^{-1}
\]

where \( M_i = M / P_i \) and \( N_i = M_i^{-1} / P_i \) is the multiplicative inverse of \( M_i \) modulo \( P_i \).

Mixed-Radix Conversion: the number \( X \) can be computed by the formula:

\[
X = a_n P_n + \ldots + a_2 P_2 \bar{P}_1 + a_1 \bar{P}_1 + a_0
\]

where \( a_n \) are called the mixed-radix coefficients and they can be obtained from the residues by:

\[
a_n = \left( \frac{x_n - a_{n-1} P_{n-1} \bar{P}_{n-2} - \ldots - a_2 P_2 \bar{P}_1}{P_n} \right)
\]

where \( n>1 \) and \( a_0=x_0 \).

The CRT requires modular additions and multiplications and it is not efficient for the implementation. The MRC is a sequential algorithm and requires modular multiplications and subtractions that is not suitable for efficient hardware realization. In the next section we proposed a novel conversion algorithm which depends on simple mathematical relationships without using CRT or MRC.

The RNS with Moduli Set \( \{r^n-2, r^{n-1}, r^n\} \): In [19], a new moduli set \( \{r^n-2, r^{n-1}, r^n\} \) was introduced for RNS. This moduli set contains pairwise relatively prime moduli for all different values of \( n \), so the dynamic range is greater than the dynamic range of similar moduli sets in [19]. Because of using of high radix (\( r>2 \)), this RNS can be simply realized in MVL and can provides very large dynamic range for different radix-r values.

Forward conversion in this RNS is performed as follows. Suppose a \( 3n \)-digit radix-\( r \) number \( Y \):

\[
Y = (y_{3n-1} \ldots y_{3n-2} \ldots y_0)
\]

We must compute the residues \( x_1, x_2, x_3 \) corresponding to the residue set \( \{r^n-2, r^{n-1}, r^n\} \). First we calculate the residue in moduli \( r \). So we have:

\[
x_1 = Y \mod r^n = (y_{3n-1} \ldots y_1 y_0) \mod r^n
\]

\[
= ((y_{3n-1} \ldots y_1 y_0) r^n + (y_{n-1} \ldots y_1 y_0)) \mod r^n
\]

Therefore, it is enough to consider the right most \( n \) digits and the rest of digits will be ignored as they are multiplies of \( r^n \).

Now we investigate the moduli \( r^n-1 \). The residue of \( Y \) in moduli \( r^n-1 \) can be calculated as follow:

\[
x_2 = Y \mod r^n-1 = (y_{3n-1} \ldots y_1 y_0) \mod r^n-1
\]

\[
= ((y_{3n-1} \ldots y_{2n+1} y_{2n}) r^{2n} + (y_{2n-1} \ldots y_{n+1} y_n) r^n + (y_{n-1} \ldots y_1 y_0)) \mod r^n-1
\]

\[
= \left( (y_{3n-1} \ldots y_{2n}) + (y_{2n-1} \ldots y_n) + (y_{n-1} \ldots y_0) \right) \mod r^n-1
\]

So, the number \( Y \) is partitioned into consecutive \( n \)-digit blocks and then we must sum these blocks by a modular adder. The residue of \( Y \) in moduli \( r^n-2 \) is obtained as follow:

\[
x_3 = Y \mod r^n-2 = (y_{3n-1} \ldots y_1 y_0) \mod r^n-2
\]

\[
= ((y_{3n-1} \ldots y_{2n+1} y_{2n}) r^{2n} + (y_{2n-1} \ldots y_{n+1} y_n) r^n + (y_{n-1} \ldots y_1 y_0)) \mod r^n-2
\]

The conversion method is based on simple mathematical relationships without using CRT or MRC.

Theorem: Given the moduli set \( \{r^n-2, r^{n-1}, r^n\} \), the residue number \( (x_1, x_2, x_3) \) is converted into the radix-\( r \) weighted number by:

\[
X = \left( c' \times r^{2n} + (b' \times r^n) + x_1 \right) \mod M
\]

where:

\[
c' = \begin{cases}
\frac{x_1 + x_3 - 2x_2 + 3r^n - 10}{2} & \text{if } (x_1 + x_3) \text{ is odd} \\
\frac{x_1 + x_3 - 2x_2 + 2(r^n - 4)}{2} & \text{if } (x_1 + x_3) \text{ is even}
\end{cases}
\]

\[
b' = \begin{cases}
\frac{-3x_1 + 4x_2 - x_3 + r^n + 6}{2} & \text{if } (x_1 + x_3) \text{ is odd} \\
\frac{-3x_1 + 4x_2 - x_3 + 2(r^n + 2)}{2} & \text{if } (x_1 + x_3) \text{ is even}
\end{cases}
\]

\[
M = (r^n - 2)(r^{n-1})r^n
\]

Proof: Suppose the \( 3n \)-digit radix-\( r \) number \( X \) that is partitioned into consecutive \( n \)-digit blocks as:

\[
X = y_{3n-1} \ldots y_{2n+1} y_{2n} y_{2n-1} \ldots y_{n+1} y_n y_{n-1} \ldots y_0 \mod c\]

\[
b\]

\[
a
\]

We know from the previous section that forward conversion is performed by the following equations:

\[
X = y_{3n-1} \ldots y_{2n+1} y_{2n} y_{2n-1} \ldots y_{n+1} y_n y_{n-1} \ldots y_0 \mod c\]

\[
b\]

\[
a
\]

We know from the previous section that forward conversion is performed by the following equations:

\[
X = y_{3n-1} \ldots y_{2n+1} y_{2n} y_{2n-1} \ldots y_{n+1} y_n y_{n-1} \ldots y_0 \mod c\]

\[
b\]

\[
a
\]
Considering the maximum value of residues, the values of $M$ and $X$ can be rewritten as

$$x_2 = a + b + c - (\alpha \times (r^n - 1))$$

$$x_3 = a + b + c - (\beta \times (r^n - 2))$$

By substituting the value of $a$ from (14), we have

$$b + c = x_2 - x_1 + (\alpha \times (r^n - 1))$$

$$2b + 4c = x_3 - x_1 + (\beta \times (r^n - 2))$$

Therefore, we have

$$c = \frac{x_1 + x_2 - 2x_3 + (\alpha \times (r^n - 2)) - (\alpha \times 2(r^n - 1))}{2}$$

$$b = \frac{-x_3 - 3x_1 - 4x_2 + (\beta \times (r^n - 2)) + (\alpha \times 4(r^n - 1))}{2}$$

Considering the maximum value of residues, the values of $\alpha$ and $\beta$ are obtained as below

$$\alpha = 2, \beta = 7 \quad \text{if} \quad (x_1 + x_3) \quad \text{is odd}$$

$$\alpha = 2, \beta = 6 \quad \text{if} \quad (x_1 + x_3) \quad \text{is even}$$

By substituting the values of $\alpha$ and $\beta$ in (21) and (22), we obtained the following equations

$$c' = \begin{cases} 
\frac{x_1 + x_2 - 2x_3 + 3r^n - 10}{2} & \text{if} \quad (x_1 + x_3) \quad \text{is odd} \\
\frac{x_1 + x_2 - 2x_3 + 2(r^n - 4)}{2} & \text{if} \quad (x_1 + x_3) \quad \text{is even} \\
\frac{-3x_1 + 4x_2 - x_3 + r^n + 6}{2} & \text{if} \quad (x_1 + x_3) \quad \text{is odd} \\
\frac{-3x_1 + 4x_2 - x_3 + 2(r^n + 2)}{2} & \text{if} \quad (x_1 + x_3) \quad \text{is even}
\end{cases}$$

Thus, the values of $b'$ and $c'$ are calculated and then the result of concatenating $x_1$, $b'$ and $c'$ should be reduced in order to placed in the dynamic range. So

$$X = \left[ (c' \times 2^n) + (b' \times r^n) + x_1 \right]_M$$

The following example clarifies the conversion method.

**Example:** Given the moduli set \{r^2 - 2, r^2 - 1, r^n\} where $r=3$ and $n=3$, the residue number $(x_3, x_2, x_1) = (0, 12, 14)$ is converted to the weighted number $X$ by this way

Moduli Set: \{3^3 - 2, 3^3 - 1, 3^3\} = [25, 26, 27]

$M = 25 \times 26 \times 27 = 17550$

$(x_3 x_2 x_1)$ is even, so

$$c' = \frac{14 + 0 - 24 + 46}{2} = 18$$

and

$$b' = \frac{-42 + 48 - 0 + 58}{2} = 32$$

Therefore,

$$X = \left[ 18 \times 3^6 + 32 \times 3^3 + 14 \right]_{17550} = 14000$$

It is easy to see that Theorem 1 is very simpler than CRT or MRC. Also Theorem 1 enables us to implement a fully parallel reverse converter for the moduli set \{r^n - 2, r^n - 1, r^n\}. Hardware realization of the proposed reverse converter is based on equations (9)-(11). The computation of (10) requires an $n$-digit radix-$r$ carry save adder (CSA) followed by two $n$-digit radix-$r$ adder and a multiplexer. We check whether $(x_1 + x_3)$ is odd or even by using radix-$r$ XOR gates. Instead of direct division by 2, we used the multiplication by $2^{-1}$ and as noted in [18], this multiplication can be performed by existing radix-$r$ multiplier belonging to the RNS processing hardware. Equation (11) has a similar realization and can be implemented with a CSA tree, two adders, one multiplexer and one multiplier for performing multiplication by $2^{-1}$. Since $x_1$ is an $n$-digit radix-$r$ number, no additional hardware is needed to compute $(r^bd + x_3)$. The desired result is the concatenation of $x_1$ and $b$. Finally, a $n$-digit radix-$r$ modular adder is used to perform the addition of $r^b x_3$ and $(r^b x_3)$. Fig. 1 shows the hardware architecture of the proposed reverse converter.

**IV. PERFORMANCE EVALUATION**

The reverse converter proposed in this paper is a novel converter dedicated to the moduli set \{r^n - 2, r^n - 1, r^n\}. Therefore, to verify the performance of this converter, it has to be compared with other proposed converters which can convert residue numbers to their equivalent radix-$r$ (r > 2) weighted representation. Such reverse converters are presented in [19] and [18]. In [19] a CRT-based reverse converter for the moduli set \{r^n - 2, r^n - 1, r^n\} was presented. In [18] four three-moduli sets $s_{r_1} = \{r^{r+1} + 1, r^{r+1} + 2, r^{r+1}\}, s_{r_2} = \{r^{r+1} + 1, r^{r+1} + 2, r^{r+1} + 1\}, s_{r_3} = \{r^{r+1} - 1, r^{r+1} - 2, r^{r+1}\}$ and $s_{r_4} = \{r^{r+1} - 1, r^{r+1} - 2, r^{r+1} - 1\}$ are proposed and also a conversion algorithm based on CRT with scale-down factors is presented. Since the hardware requirements and conversion speed for the moduli sets $s_1$ and $s_3$ are the same as the moduli sets $s_2$ and $s_4$, respectively, we confine our comparison to the reverse converter for moduli sets $s_1$ and $s_3$. All the hardwares listed in Table I, are radix-$r$ hardware. An overview of MVL circuits’ implementation can be found in [20], [21]. It should be noted the radix-$r$ full adder (FA) cell is a $r$-input adder cell. Therefore, a radix-$r$ CSA tree includes $r$-to-2 CSA’s. Since the CSA tree used in Fig. 1 has 8 inputs, the maximum level of $r$-to-2 CSA in CSA tree is two. So, the total delay of the CSA tree used in Fig. 1 is the delay of two radix-$r$ FA. Also for $r=3$, this CSA tree consists of three 4-to-2 radix-3 CSA and for greater values of $r$ (r>3), it consists of two $r$-to-2 CSA. For comparison with other converters, we consider the worst case ($r=3$) for CSA tree. The hardware requirements and conversion delays of the reverse converters listed in Table I and II, respectively. It should be noted that we used some radix-$r$ NOT and XOR gates for calculating negative numbers and checking the value of $(x_1 + x_3)$ for odd or even, respectively. Since these gates are not on the critical delay path and their cost are small, we don't taking into
account these gates in Table I and II. It has been assumed that $T_{FA}, T_{MUX}, T_{MUL}$, and $T_{MOA}$ refer to the delays of the radix-$r$ full adder, multiplexer, multiplier and mod $M$ modular adder, respectively.

It is clear from Table I and II that the area and delay of the proposed converter are considerably better than the other converters. Our proposed converter is very faster and results to hardware savings when compared with the converters of [18] and [19]. It must be noted that for a specified dynamic system, the value of $n$ for our reverse converter is smaller than the $n$ for reverse converters of [18].

V. CONCLUSION

In this paper, we proposed a fully parallel reverse conversion algorithm for the moduli set $\{r^n - 2, r^n - 1, r^p\}$, based on simple mathematical relationships. Our proposed conversion algorithm is very simple and resulted to an efficient hardware realization of the reverse converter. In comparison with other reverse converters for MVL based RNS systems, the presented converter has a superior area-time complexity.

REFERENCES


Fig. 1 Block diagram of the proposed reverse converter.

TABLE I COMPARISON OF HARDWARE REQUIREMENTS

<table>
<thead>
<tr>
<th>converter</th>
<th>Moduli Set</th>
<th>CSA Adder</th>
<th>Subtractor</th>
<th>Total FA's</th>
<th>Modular Adder (3n)</th>
<th>Multiplier (n)</th>
<th>Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>[19]</td>
<td>{r^n - 2, r^n - 1, r^n}</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>12n</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>[18]</td>
<td>{r^n+1, r^n+1, r^n+1}</td>
<td>1</td>
<td>4</td>
<td>-</td>
<td>13n</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>[18]</td>
<td>{r^{n-1}, r^{n-1}, r^{n-1}}</td>
<td>--</td>
<td>3</td>
<td>3</td>
<td>21n</td>
<td>1</td>
<td>3</td>
</tr>
<tr>
<td>Proposed</td>
<td>{r^n - 2, r^n - 1, r^n}</td>
<td>4</td>
<td>4</td>
<td>--</td>
<td>8n</td>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

TABLE II COMPARISON OF CONVERSION DELAYS

<table>
<thead>
<tr>
<th>converter</th>
<th>Moduli Set</th>
<th>Conversion Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>[19]</td>
<td>{r^n - 2, r^n - 1, r^n}</td>
<td>(5n)T_{FA}+T_{MUL}+T_{MOA}</td>
</tr>
<tr>
<td>[18]</td>
<td>{r^{n+1}, r^{n+1}, r^{n+1}}</td>
<td>(6n)T_{FA}+T_{MUL}+T_{MOA}</td>
</tr>
<tr>
<td>[18]</td>
<td>{r^{n-1}, r^{n-1}, r^{n-1}}</td>
<td>(6n)T_{FA}+T_{MUL}+T_{MOA}</td>
</tr>
<tr>
<td>Proposed</td>
<td>{r^n - 2, r^n - 1, r^n}</td>
<td>(n+2)T_{FA}+T_{MUL}+T_{MOA}+T_{MUX}</td>
</tr>
</tbody>
</table>