A New Four-Moduli Set \(\{2^{2n}, 2^{n^2} - 1, 2^{n^2} + 1, 2^{n+1} - 1\}\) with an Efficient Residue to Binary Converter

Mohammad Reza Noori Mehr, Mehdi Hosseinzadeh, Hamid Reza Hosseini

Abstract: In this paper a new four-moduli set \(\{2^{2n}, 2^{n^2} - 1, 2^{n^2} + 1, 2^{n+1} - 1\}\) for even \(n\) is introduced. This moduli set has \(4n\)-bit dynamic range and well-formed moduli which can result in efficient implementation of the residue to binary converter as well as internal RNS arithmetic circuits. Then, an efficient residue to binary converter for proposed moduli set is presented. The converter for this moduli set is implemented in two-level structure which is designed based on Chinese remainder theorem (CRT) and the new CRT-I methods. The presented residue to binary converter has lower hardware cost and results in a significant reduction in the conversion delay compared to the residue to binary converter of the latest introduced four-moduli set \(\{2n - 1, 2n, 2n + 1, 2n+1 - 1\}\) that has the same dynamic range as the proposed four-moduli set.

Keywords: residue number system (RNS); reverse converter; New Chinese remainder theorem 1 (CRT-I); computer arithmetic.

1. Introduction

Residue number system (RNS) is a non-weighted system and uses residues of a number in particular modulus for its representation. One of the most important characteristics of the RNS is the limited propagation of the carry-out digit among modulus in arithmetic. Instead of performing arithmetic on a large number, calculations are done on its corresponding residues in parallel. This feature significantly increases the calculation speed and decreases the consumed power [1,2]. Considering the characteristics of the RNS, this system has been applied on many arithmetic applications such as digital signal processing (DSP) systems [3], digital filters [4,5], image processing [6], distributed storing and retrieval of data [7], RSA cryptography algorithms [8,9], digital communications [10] and generally in applications in which addition, subtraction and multiplication operations are repeated in a specific range of numbers. In addition, RNS architectures are essentially error-resilient and facilitate error detection and correction in other systems [11,12].

Selecting appropriate moduli set plays a significant role in designing a RNS system, because the speed of internal RNS arithmetic circuits as well as the speed and complexity of the residue to binary converter have a large dependency on the form and number of the selected moduli [1]. The moduli of the form \(2^k\) and \(2^k \pm 1\) have the most use in RNS moduli sets. Because, RNS systems based on these moduli can be efficiently implemented by using usual binary hardware [13,14].

Up to now, many moduli sets have been presented with various dynamic ranges (DR) such as \(\{2^{n+1} - 1, 2^n, 2^n - 1\}\) [15], \(\{2^n - 1, 2^n, 2^{2n+1} - 1\}\) [16] and \(\{2^n, 2^{2n} - 1, 2^{2n} + 1\}\) [17], which have dynamic ranges equal to 3n, 4n and 5n-bits respectively. Some applications require large dynamic ranges with high parallelism. Therefore, four-moduli sets \(\{2^n - 1, 2^n, 2^n + 1,\)
A New Four-Moduli Set \(\{2^n, 2^{n/2} - 1, 2^{n/2} + 1, 2^{n+1} - 1\}\) with an Efficient Residue …

In this paper, the new four-moduli set \(\{2^n, 2^{n/2} - 1, 2^{n/2} + 1, 2^{n+1} - 1\}\) for even \(n\) has been proposed, which has an RNS arithmetic unit with lower complexity and higher speed due to using of well-formed moduli. Next, a two-level design of residue to binary converter for the proposed moduli set based on Chinese remainder theorem (CRT) and New CRT-I is presented. The converter is ROM-free and based on carry save adders and modular adders which can be efficiently implemented by VLSI circuits. In comparison to the residue to binary of the four-moduli set \(\{2^n - 1, 2^n, 2^{n+1}, 2^{n+1} - 1\}\), the proposed converter has better performance in terms of hardware requirements and conversion delay. In the rest of this paper, in Section 2, a background of residue numbers systems has been presented. In Section 3, the design of the residue to binary converter for the proposed moduli set has been discussed and in Section 4, the performed evaluation has been reviewed.

2. Background

An RNS is defined in terms of a relatively-prime moduli set \(\{m_1, m_2, \ldots, m_n\}\) where gcd \((m_i, m_j) = 1\) for \(i \neq j\), and gcd \((a, b)\) denotes the greatest common divisor of \(a\) and \(b\). A weighted number \(X\) can be represented as \(X = (x_1, x_2, \ldots, x_n)\), where

\[
x_i = X \mod m_i = \lfloor \frac{X}{m_i} \rfloor, \quad 0 \leq x_i < m_i.
\]

Such a presentation is unique for any integer \(X\) in the range \([0, M - 1]\), where \(M = m_1 m_2 \ldots m_n\) is the DR of the moduli set \(\{m_1, m_2, \ldots, m_n\}\) [21].

**Chinese remainder theorem**: By CRT [22], the reverse conversion can be done as

\[
Y = \left[\sum_{i=1}^{n} \tilde{m}_i k_i x_i \right]_M
\]

where

\[
M = \prod_{i=1}^{n} m_i, \quad \tilde{m}_i = \frac{M}{m_i}, \quad k_i = \tilde{m}_i^{-1}, \quad \left[ k_i \times \tilde{m}_i \right]_{m_i} = 1, \quad x_i = \left[ X \right]_{m_i}.
\]

**New Chinese remainder theorem 1**: By CRT-I [23], the RNS number \((x_1, x_2, \ldots, x_n)\) can be converted into its equivalent weighted number as

\[
X = x_1 + m_1 k_1 (x_2 - x_1) + m_2 k_2 (x_3 - x_2) + \cdots + m_{n-1} k_{n-1} m_n (x_n - x_{n-1}) \left[ m_{n-1} \right]_{m_n \ldots m_2 m_1}
\]

where

\[
\left[ k_1 \times m_1 \right]_{m_2 \ldots m_n} = 1, \quad \left[ k_2 \times m_1 \times m_2 \right]_{m_3 \ldots m_1} = 1, \quad \ldots, \quad \left[ k_{n-1} \times m_1 \times m_2 \times \cdots \times m_{n-1} \right]_{m_n} = 1.
\]

Where \(k_1, k_2, \ldots, k_{n-1}\) are the multiplicative inverses.

3. Residue to binary converter

To achieve an efficient residue to binary converter for the moduli set \(\{2^n, 2^{n/2} - 1, 2^{n/2} + 1, 2^{n+1} - 1\}\) with corresponding residues \((x_1, x_2, x_3, x_4)\), two-level design is employed. In the first level, the equivalent weighted number of the residues \(x_1, x_2\) and \(x_3\) is obtained by using CRT
method based on the subset \(\{2^{2n}, 2^{n/2} - 1, 2^{n/2} + 1\}\). Next, the result of the first level and \(x_4\) are combined by using New CRT-I with respect to the set \(\{2^{2n}(2^{n/2} - 1)(2^{n/2} + 1), 2^{n+1} - 1\}\).

### 3.1 Converter for the moduli set \(\{2^{2n}, 2^{n/2} - 1, 2^{n/2} + 1\}\) based on CRT

In this section, the CRT is employed for designing an efficient reverse conversion algorithm. The following theorem and properties are needed for the derivation of conversion algorithm.

**Theorem:** The moduli \(2^{2n}, 2^{n/2} - 1, 2^{n/2} + 1\) are pairwise relatively prime.

**Proof:** Since \(2^{n/2} - 1\) and \(2^{n/2} + 1\) are two odd numbers and \(2^{2n}\) is an even number, it is clear that \(2^{2n}\) is relatively prime to the moduli \(2^{n/2} - 1\) and \(2^{n/2} + 1\). Also, the moduli \(2^{n/2} - 1\) and \(2^{n/2} + 1\) are relatively prime, they were previously used in RNS moduli set \(\{2^{n}, 2^{n/2} - 1, 2^{n/2} + 1, 2^{n+1}, 2^{2n+1} - 1\}\) [24].

**Property 1:** The residue of a negative residue number \((-v)\) in modulo \((2^n - 1)\) is the one’s complement of \(v\), where \(0 \leq v < 2^n - 1\) [17].

**Property 2:** The multiplication of a residue number \(v\) by \(2^k\) in modulo \((2^n - 1)\) is carried out by \(k\) bit circular left shift, where \(k\) is a natural number [17].

According to (2) and by assuming \(m_1 = 2^{2n}, m_2 = 2^{n/2} - 1\) and \(m_3 = 2^{n/2} + 1\) we have

\[
\hat{m}_1 = (2^n - 1), \quad \hat{m}_2 = 2^{2n}(2^{n/2} + 1), \quad \hat{m}_3 = 2^{2n}(2^{n/2} - 1), \quad M = 2^{2n}(2^n - 1).
\] (6)

Considering (6), the required multiplicative inverses for (2) are computed as follows:

\[
|k_1 \times (2^n - 1)|_{2^{2n}} = 1 \rightarrow k_1 = -(2^n + 1)
\] (7)

\[
|k_2 \times 2^{2n}(2^{n/2} + 1)|_{2^{2n} - 1} = 1 \rightarrow k_2 = 2^{(n/2)-1}
\] (8)

\[
|k_3 \times 2^{2n}(2^{n/2} - 1)|_{2^{2n} + 1} = 1 \rightarrow k_3 = 2^{(n/2)-1}.
\] (9)

The binary vectors \(x_1, x_2\) and \(x_3\) can be represented in bit-level as \(x_1 = (x_{1,2n} \ldots x_{1,1} x_{1,0})\), \(x_2 = (x_{2,2n} \ldots x_{2,1} x_{2,0})\) and \(x_3 = (x_{3,2n} \ldots x_{3,1} x_{3,0})\). Now, (2) can be rewritten as

\[
Y = \sum_{i=1}^{n} \hat{m}_i |k_i |_{m_i} x_i - M \times l.
\] (10)

Where \(l\) is an integer number and depends on the value of \(Y\). By replacing (6)-(9) in (10) we have

\[
Y = \left\{ \begin{array}{l}
-(2^{2n} - 1) \times x_1 \\
+ 2^{2n} \times (2^{n/2} + 1) \times 2^{(n/2)-1} \times x_2 \\
+ 2^{2n} \times (2^{n/2} - 1) \times 2^{(n/2)-1} \times x_3 \\
\end{array} \right.
\] (11)

\[
- 2^{2n} \times (2^n - 1) \times l
\]

Both sides of (11) are divided by \(2^{2n}\) as

\[
\frac{Y}{2^{2n}} = \left( \begin{array}{l}
-1 + 2^{2n} \times x_1 + (2^{n/2} + 1) \times 2^{(n/2)-1} \times x_2 \\
+ (2^{n/2} - 1) \times 2^{(n/2)-1} \times x_3 \\
\end{array} \right) \times (2^n - 1) \times l.
\] (12)

and calculating the floor values in modulo \((2^n - 1)\) results in the following

\[
|\frac{Y}{2^{2n}}| = \left| \begin{array}{l}
-1 \times x_1 \times (2^n - 1) + (2^{n/2} + 1) \times 2^{(n/2)-1} \times x_2 \times (2^n - 1) \\
+ (2^{n/2} - 1) \times 2^{(n/2)-1} \times x_3 \times (2^n - 1) \\
\end{array} \right|_{(2^n - 1)}.
\] (13)
Now, the number $Y$ can be calculated by the following

$$Y = x_1 + 2^{2n} \times V$$

$$V = \left\lfloor \frac{Y}{2^{2n}} \right\rfloor.$$  \hfill (15)

Eq. (13) can be rewritten as

$$V = S_1 + S_2 + S_{3_1} + S_{3_2}(2^{n-1})$$ \hfill (16)

where

$$S_i = \left\lfloor x_1(2^{n-1}) \right\rfloor = \left(\frac{x_{i,2n-1} \cdots x_{i,1}x_{i,0}}{2^n}\right)_{(2^n-1)} = \left(\frac{x_{i,2n-1} \cdots x_{i,n+1}x_{i,n}}{2^n} + x_{i,n-1} \cdots x_{i,1}x_{i,0}\right)_{2^n-1}. \hfill (17)$$

The above equation can be parsed as follow:

$$S_{11} = \left\lfloor \left(\frac{x_{1,n-1} \cdots x_{1,1}x_{1,0}}{n}\right)_{2^n-1} \right\rfloor = \overline{x}_{1,n-1} \cdots \overline{x}_{1,1} \overline{x}_{1,0} \hfill (18)$$

$$S_{12} = \left\lfloor -2^n \left(\frac{x_{1,2n-1} \cdots x_{1,n+1}x_{1,n}}{n}\right)_{2^n-1} \right\rfloor = \overline{x}_{1,2n-1} \cdots \overline{x}_{1,n+1} \overline{x}_{1,n} \hfill (19)$$

$$S_2 = \left\lfloor (2^{n-1} + 2^{(n/2)-1}) \times x_2 \right\rfloor_{2^n-1} = \left\lfloor \left(\frac{0 \cdots 00x_{2,(n/2)-1} \cdots x_{2,1}x_{2,0}}{n/2}\right)_{2^n-1} \right\rfloor = x_{2,0}x_{2,(n/2)-1} \cdots x_{2,2}x_{2,0}x_{2,(n/2)-1} \cdots x_{2,1} \hfill (20)$$

$$S_{31} = \left\lfloor 2^{n-1} \times x_3 \right\rfloor_{2^n-1} = \left\lfloor \left(\frac{0 \cdots 00x_{3,(n/2)-1} \cdots x_{3,1}x_{3,0}}{(n/2)-1}\right)_{(n/2)+1} \right\rfloor = x_{3,0}0 \cdots 00x_{3,(n/2)-1} \cdots x_{3,1} \hfill (21)$$

$$S_{32} = \left\lfloor -2^{(n/2)-1} \times x_3 \right\rfloor_{2^n-1} = \left\lfloor \left(\frac{0 \cdots 00x_{3,n/2} \cdots x_{3,1}x_{3,0}}{(n/2)-1}\right)_{(n/2)+1} \right\rfloor = \overline{x}_{3,n/2} \cdots \overline{x}_{3,1} \overline{x}_{3,0}1 \cdots 11 \hfill (22)$$

Therefore, (16) becomes as follow:

$$V = \left\lfloor S_{11} + S_{12} + S_2 + S_{31} + S_{32} \right\rfloor_{(2^n-1)}. \hfill (23)$$

Finally, since $x_1$ is a $2n$-bit number, $Y$ in (14) can be obtained as

$$Y = x_1 + 2^{2n} \times V$$

$$= \frac{V_{n-1} \cdots V_1 V_0 x_{1,2n-1} \cdots x_{1,1}x_{1,0}}{2^n}. \hfill (24)$$
By calculating $Y$ and having the residue of $x_2$, the residue to binary converter for the moduli set $\{2^{2n}(2^n - 1), 2^{n+1} - 1\}$ is designed.

### 3.2 Converter for the moduli set $\{2^{2n}(2^n - 1), 2^{n+1} - 1\}$ based on CRT-I

The CRT-I for these two moduli requires only one multiplicative inverse as

$$k \times 2^{2n}(2^n - 1) \mod 2^{n+1} - 1 = 1 \rightarrow k = -2^3. \quad \text{(25)}$$

The $X = (Y, x_4)$ can be obtained by substituting the value of $k$, and moduli $m_1 = 2^{2n}(2^n - 1)$, $m_2 = 2^{n+1} - 1$ in (4) as shown below

$$X = Y + 2^{2n}(2^n - 1) - 2^3(x_4 - Y) \mod 2^{n+1} - 1. \quad \text{(26)}$$

The binary vectors $Y$ and $x_4$ can be represented in bit-level as $Y = (Y_3 \ldots Y_1 Y_0)$ and $x_4 = (x_{4n} \ldots x_{4,1} x_{4,0})$.

Now, (26) can be simplified as follow:

$$X = Y + 2^{2n}(2^n - 1)H \quad \text{(27)}$$

where

$$H = \left| S_4 + S_5 \right|_{2^{n+1} - 1} \quad \text{(28)}$$

$$S_4 = \left| -2^3 \times x_4 \right|_{2^{n+1} - 1} = \left| -2^3(x_{4,n} \ldots x_{4,1} x_{4,0}) \right|_{2^{n+1} - 1} = \bar{x}_{4,n-3} \ldots \bar{x}_{4,0} \bar{x}_{4,n-1} \bar{x}_{4,n-2} \quad \text{(29)}$$

$$S_5 = \left| 2^3 \times Y \right|_{2^{n+1} - 1} = \left| 2^3(Y_{3n-1} \ldots Y_0) \right|_{2^{n+1} - 1} = 2^3(Y_{3n-1} \ldots Y_{2n+2} Y_{2n+3} \ldots Y_{n+2} Y_{n+1}) \times 2^{n+2} + Y_{n+1} \ldots Y_{0} \times 2^{n+1} \quad \text{(30)}$$

Now, (30) can be divided in three parts as below

$$S_{51} = \left| 2^3(Y_{n-1} \ldots Y_0) \right|_{2^{n+1} - 1} = Y_{n-3} \ldots Y_0 Y_n Y_{n-1} Y_{n-2} \quad \text{(31)}$$

$$S_{52} = \left| 2^{n+1}(Y_{2n+1} \ldots Y_{n+2} Y_{n+1}) \right|_{2^{n+1} - 1} = Y_{2n-2} \ldots Y_{n+1} Y_{2n+1} Y_{2n} Y_{2n-1} \quad \text{(32)}$$

$$S_{53} = \left| 2^{2n+2}(000Y_{3n-1} \ldots Y_{2n+3} Y_{2n+2}) \right|_{2^{n+1} - 1} = Y_{3n-1} \ldots Y_{2n+2} 000 \quad \text{(33)}$$

Therefore, (28) can be rewritten as

$$H = \left| S_4 + S_{51} + S_{52} + S_{53} \right|_{2^{n+1} - 1}. \quad \text{(34)}$$

Finally, (27) can be simplified as follows:

$$X = Z - T \quad \text{(35)}$$
where
\[
Z = Y + 2^{3n} H = H_n \cdots H_1 H_0 Y_{3n-1} \cdots Y_0 
\]
\[
T = 2^{2n} H = 0 \cdots 00 H_n \cdots H_1 H_0 0 \cdots 00.
\]

Example: The four-moduli set \(\{2^{2n}, 2^{n^2-1}, 2^{n^2}+1, 2^{n+1}-1\}\) for \(n=4\) can be chosen as \(\{m_1, m_2, m_3, m_4\} = \{256, 3, 5, 31\}\).

Two weighted numbers \(A\) and \(B\) can be chosen as
\[
A = (1001011010)_2 = 602 \\
B = (1101000100)_2 = 836
\]

The RNS representation of two weighted numbers \(A\) and \(B\) for the moduli set \(\{256, 3, 5, 31\}\) is as follows:
\[
A_1 = (1011010)_2 = 90 \\
A_2 = (10)_2 = 2 \\
A_3 = (10)_2 = 2 \\
A_4 = (1101)_2 = 13
\]
\[
B_1 = (1000100)_2 = 68 \\
B_2 = (10)_2 = 2 \\
B_3 = (01)_2 = 1 \\
B_4 = (11110)_2 = 30
\]

Now for calculating \(X = A + B\) in RNS system, we have
\[
x_1 = \left| A_1 + B_1 \right|_{m_1} = \left| 90 + 68 \right|_{256} = 158 = (10011110)_2 \\
x_2 = \left| A_2 + B_2 \right|_{m_2} = \left| 2 + 2 \right|_3 = 1 = (01)_2 \\
x_3 = \left| A_3 + B_3 \right|_{m_3} = \left| 2 + 1 \right|_3 = 3 = (11)_2 \\
x_4 = \left| A_4 + B_4 \right|_{m_4} = \left| 13 + 30 \right|_{31} = 12 = (1100)_2
\]

By letting the values of residues \(x_1, x_2, x_3\) and \(n = 4\) in (18)-(24) we have
\[
S_{11} = (0001)_2 = 1 \\
S_{12} = (0110)_2 = 6 \\
S_{2} = (1010)_2 = 10 \\
S_{31} = (1001)_2 = 9 \\
S_{32} = (1001)_2 = 9 \\
V = \left| 1 + 6 + 10 + 9 \right|_{13} = 5 = (0101)_2 \\
Y = \left(010110111110\right)_2 = 1438
\]

Then, the required values should be substituted in (29) and (31)-(34) So
Finally, by letting values of $H$ and $Y$ in (36) and (37), $X$ can be computed as follows:

$$Z = \left(\frac{00000010110011110}{12}\right)_5 = 1438$$

$$T = (0 \ldots 00)_2 = 0$$

$$X = Z - T = \left(\frac{00000010110011110}{12}\right)_5 = 1438$$

Now if we calculate the addition of $X = A + B$ in the weighted number system, the result is

$$X = A + B = 602 + 836 = 1438.$$  

As it can be observed, the obtained value for $X = A + B$ in weighted number and residue number system is equal.

---

**Fig. 1.** The proposed residue to binary converter: (a) second level (b) first Level

*Hardware Implementation:* hardware architecture of the proposed residue to binary converter for the four-moduli set \{\(2^{2n}, 2^{n^2} -1, 2^{n^2} +1, 2^{n+1} -1\}\} is shown in Fig 1. The
hardware includes \((2.5n+1)\) NOT gates in operand preparation unit (OPU) 1 for calculating the expressions \((18), (19)\) and \((20)\). Therefore, the delay for OPU1 equals to \(t_{\text{FA}}\). To implement \((23)\) a 5-operand modulo \((2^n–1)\) adder \([25]\) is required which consists of three \(n\)-bit carry-save adder (CSA) with end-around carry (EAC) following by a modulo \((2^n–1)\) adder. In this paper for implementing the modulo \((2^n–1)\) adder, \(n\)-bit carry propagate adder (CPA) with EAC has been applied \([26]\). Each CSA with EAC has the delay of one full adder (FA), and the delay of a CPA with EAC is twice the delay of a regular CPA. In OPU2, \((n+1)\) NOT gates are used for calculating \((29)\) and for calculating \((34)\) a 4-operand modulo \((2^n+1–1)\) adder is used which consists of two \((n+1)\)-bit CSA with EAC following by a \((n+1)\)-bit CPA with EAC. Since equations \((21), (22)\) and \((33)\) have some bits with constant values of 0 or 1 so some of the full adders (FAs) in CSA2, CSA3 and CSA5 are reduced to pairs of XNOR/OR or XOR/AND gates. Finally, implementation of \((35)\) requires a \((4n+1)\)-bit regular binary subtracter. This subtracter can be realized by \((n+1)\) NOT gates, \((n+1)\) FAs and \(3n\) pairs of XNOR/OR gates. It should be noted that realization of \((24)\) and \((36)\) rely on simple concatenation without the use of any computational hardware.

4. PERFORMANCE EVALUATION

In Table I, the performance of the proposed residue to binary converter for the moduli set \(\{2^{2n}, 2^{n/2}–1, 2^{n/2}+1, 2^{n+1}–1\}\) has been compared with converters in \([18,19]\) and \([20]\) from both hardware cost and delay viewpoints. For a better comparison, the unit gate model is considered to obtain total area and delay estimations. Based on this model, each two-input monotonic gate counts as one gate in area and delay, an XOR/XNOR gate counts as two gates in area and delay and a FA has area of seven gates and delay of four gates. The corresponding total unit gate area and delay are presented in Table I. Comparisons in Table 1 show that the proposed residue to binary converter for moduli set \(\{2^{2n}, 2^{n/2}–1, 2^{n/2}+1, 2^{n+1}–1\}\) has less hardware cost and delay in compare with residue to binary converters presented in \([18,19]\) and \([20]\).

<table>
<thead>
<tr>
<th>Converter</th>
<th>FA</th>
<th>XNOR/OR pairs</th>
<th>XOR/AND pairs</th>
<th>NOT</th>
<th>Other</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>([18])-C1</td>
<td>(9n+5+k)</td>
<td>(2n)</td>
<td>–</td>
<td>(6n+1)</td>
<td>–</td>
<td>(((23n+12)/2)t) (FA)</td>
</tr>
<tr>
<td>([18])-C1 ROM</td>
<td>(8n+4)</td>
<td>(2n)</td>
<td>–</td>
<td>(6n+1)</td>
<td>ROM</td>
<td>((9n+6)t) (FA)</td>
</tr>
<tr>
<td>[20]</td>
<td>(10n+6+k)</td>
<td>(6n+2)</td>
<td>–</td>
<td>(7n+2)</td>
<td>MU X</td>
<td>(((15n+22)/2)t) (FA)</td>
</tr>
<tr>
<td>[19]</td>
<td>(26n+8)</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>ROM</td>
<td>((7n+8)t) (FA)+2 (t) ROM</td>
</tr>
<tr>
<td>proposed</td>
<td>(9n+4)</td>
<td>(3.5n–1)</td>
<td>(0.5n+2)</td>
<td>(4.5n+3)</td>
<td>–</td>
<td>((5n+8)t) (FA)+((3n)t) (NOT) XNOR/OR+3 (t) NOT</td>
</tr>
</tbody>
</table>

\(k=(n–4)(n+1)/2\)
Another important issue in designing a RNS system is the speed of internal RNS arithmetic which depends on the size and forms of modulus of a moduli set. In [27-29] unit gate delays of parallel-prefix modular adders for moduli $2^k-1$, $2^k+1$ and $2^k+3$ have been determined which are equal to $2\log_2(k_1)+3$, $2\log_2(k_2)+6$ and $2\log_2(k_3+1)+7$ respectively can be used for estimating the speed of moduli sets with different moduli. The modulo with the form of $2^k$ has the addition delay equal to the modulo $2^k-1$ adder. In each moduli set the modulo with most unit gate delay determines the overall speed of RNS arithmetic unit based on that moduli set. In proposed moduli set the modulo $2^{2n}$ and in moduli sets $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ and $\{2^n-3, 2^n-1, 2^{n+1}, 2^{n+3}\}$ the moduli $2^n+1$ and $2^n+3$ determine the speed of RNS arithmetic unit based on their corresponding moduli sets respectively. By letting $k_1=2n$, $k_2=n$ and $k_3=n$ the overall speed of each moduli set is specified. The results of these calculations have been shown in Table 2. As it can be observed, the proposed moduli set has more execution speed in comparison to the other similar moduli sets.

Table 2. Comparison the Speed of the Different Moduli Sets

<table>
<thead>
<tr>
<th>Moduli set</th>
<th>Critical modulo</th>
<th>Delay</th>
</tr>
</thead>
<tbody>
<tr>
<td>${2^{2n}, 2^{n+2}-1, 2^{n+2}+1, 2^{n+1}-1}$</td>
<td>$2^{2n}$</td>
<td>$2\log_2(n)+5$</td>
</tr>
<tr>
<td>${2^n-1, 2^n, 2^n+1, 2^{n+1}-1}$</td>
<td>$2^n+1$</td>
<td>$2\log_2(n)+6$</td>
</tr>
<tr>
<td>${2^n-3, 2^n-1, 2^{n+1}, 2^{n+3}}$</td>
<td>$2^n+3$</td>
<td>$2\log_2(n-1)+7$</td>
</tr>
</tbody>
</table>

Conclusion

In this paper an efficient residue to binary converter for the new four-moduli set $\{2^{2n}, 2^{n+2}-1, 2^{n+2}+1, 2^{n+1}-1\}$ for even $n$ is proposed, which has low hardware cost and delay time. In addition, this moduli set has a lower delay for RNS arithmetic unit due to applying well-formed moduli. With respect to obtained results, the new proposed residue to binary converter has significant improvements from both delay and hardware cost viewpoints in comparison to the residue to binary converter of the latest introduced four-moduli set $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$.

REFERENCES

A New Four-Moduli Set \(\{2^n, 2^n - 1, 2^{n/2} + 1, 2^{n+1} - 1\}\) with an Efficient Residue …