A Residue-to-Binary Converter for the Extended Four-Moduli Set $\{2^n - 1, 2^n + 1, 2^{2n} + 1, 2^{2n+p}\}$

Ahmad Hiasat

***Abstract*— This brief presents a residue-to-binary converter for the moduli set $\{2^n - 1, 2^n + 1, 2^{2n} + 1, 2^{2n+p}\}$, where n is a positive integer and $0 \le p \le n - 2$. The converter consists of three simplified $4n$-bit carry-save adders (CSAs) along with a modulo $(2^{4n} - 1)$ adder. The main contribution of this brief is reducing the requirements of the proposed CSA network, which has impacted the area, delay, power and energy. Compared with four-moduli and five-moduli sets that have the dynamic range $2^v(2^{4n} - 1)$, where $v = n$ or $2n$, the proposed converter resulted in the average area, delay, power, and energy reductions of 22.7%, 9.2%, 17.8%, and 24.5%, respectively. Moreover, the throughput rate per unit area has been improved by an average of 48.7%.**

***Index Terms*— Chinese remainder theorem, computer arithmetic, residue number system, residue-to-binary converter, VLSI implementation.**

I. Introduction

Residue number system (RNS) operations have no carry between different residue digits [1], [2]. Addition, subtraction, and multiplication run on a residue digit independently from computations on other digits [1], [2]. This independency reduces the time needed to perform residue-based arithmetic operations compared with binary-based operations. Therefore, RNS is used efficiently in some digital signal processing and cryptographic applications that require high-speed computations [2]–[6].

Selecting the proper moduli set is a very important step in utilizing RNS to serve the above-mentioned applications. The moduli set selection is determined by the dynamic range, the number of moduli, and the form of moduli targeted. While some applications can be well served by a three-moduli set, other applications require a wider dynamic range, a higher level of parallelism, and moduli forms that ease the residue-to-binary conversion and arithmetic operations within any RNS-based processor. The dynamic range is determined by the product of all moduli comprising the moduli set. Increasing the magnitude of a modulus within a set (i.e., vertical extension) achieves a wider dynamic range. Another way of achieving the same goal is by increasing the number of moduli (i.e., horizontal extension), which also leads to a higher level of parallelism required by specific applications like cryptography. The forms of moduli are also crucial in determining the speed of residue-to-binary conversion and residue-based arithmetic computations.

The popular three-moduli set, $\{2^n, 2^n - 1, 2^n + 1\}$, has a dynamic range of $3n$ bits, where n is a positive integer. This moduli set received considerable attention in terms of reverse conversion and arithmetic components' design [7], [8]. A vertical extension for this moduli set was introduced in [9]. However, to provide a horizontal extension, researchers introduced four-moduli sets [10]–[18], five-moduli sets [19]–[23], and other sets with more moduli [24]. The four-moduli sets can be divided into two main groups. The first group is of the form $\{2^n, 2^n - 1, 2^n + 1, 2^{n\pm 1} \pm 1\}$ or similar ones [10]–[13]. Such sets have balanced data paths; however, they provide $(4n \pm 1)$-bit dynamic ranges and have a complicated reverse conversion process in terms of area and time. The second group is of the form $\{2^v, 2^n - 1, 2^n + 1, 2^{2n} + 1\}$ or the form $\{2^v, 2^n - 1, 2^n + 1, 2^{2n+1} - 1\}$ [14]–[18], where $n \le v \le 2n$. The moduli in these sets of the group are unbalanced and they provide $5n$- to $6n$-bit or more dynamic ranges. Moduli sets of the second group like $\{2^v, 2^n - 1, 2^n + 1, 2^{2n+1} - 1\}$ are not only unbalanced, but also have a complicated conversion process [16], [18]. However, moduli sets of the second group of the form $\{2^v, 2^n - 1, 2^n + 1, 2^{2n} + 1\}$ (to which the moduli set considered in this brief belongs to), are also unbalanced. This unbalance affects adversely the throughput rate of RNS-based processors. Modulo $(2^{2n} + 1)$ arithmetic components have the longest path delay, whereas, $(2^n - 1)$ ones have the shortest. Nevertheless, moduli sets of this form enjoy easier conversion process [14]–[17]. The easiness of reverse conversion is due to the fact that the dynamic range is given by $2^v(2^{4n} - 1)$ [25].

On the other side, a five-moduli set like $\{2^{2n+p}, 2^n - 1, 2^n + 1, 2^n - 2^{((n+1)/2)} + 1, 2^n + 2^{((n+1)/2)} + 1\}$ provides a $(6n + p)$-bit dynamic range [22], [23]. However, the design of residue-based arithmetic components for the moduli of the form $(2^n \pm 2^{((n+1)/2)} + 1)$ is more demanding compared with moduli of the form $(2^n \pm 1)$ [26], [27].

The moduli set under consideration in this brief is the extended four-moduli set $\{2^n - 1, 2^n + 1, 2^{2n} + 1, 2^{2n+p}\}$, $(0 \le p \le n - 2)$, which provides a dynamic range of $(6n + p)$ bits. This set lies within the previously described second group of the four-moduli sets, which has a simpler conversion process. The moduli set considered in this brief is similar to the ones addressed in [14]–[17]. The five-moduli set introduced in [22] and [23] provides a similar dynamic range of $(6n + p)$ bits. In fact, the product of the last two moduli in the five-moduli set produces the third modulus in the four-moduli set under consideration.

The design of efficient reverse converters is an essential part in improving the performance of all application-specific RNS-based processors. Therefore, this brief is an additional effort to reduce the hardware complexity of a residue-to-binary converter and reduce the overall complexity of a RNS-based processor, for the extended four-moduli set $\{2^n - 1, 2^n + 1, 2^{2n} + 1, 2^{2n+p}\}$.

The notational conventions used in this brief are as follows.

1) $\{m_1, m_2, \ldots\ldots, m_N\}$, moduli set of N pairwise relatively prime positive integers.
2) $M = \prod_{i=1}^{N} m_i$, dynamic range.
3) For any integer $X \in [0, M)$, the residue representation of X is $X \overset{RNS}{\rightarrow} (R_1, R_2, \ldots\ldots, R_N)$, where $R_i = \langle X \rangle_{m_i}$ is the least non-negative remainder when dividing X by m_i.
4) $\hat{m}_i = (M/m_i)$.
5) $\langle\ (1/\hat{m}_i)\rangle_{m_i}$ is the multiplicative inverse of $\hat{m}_i$ (i.e., $\langle\langle \hat{m}_i \rangle_{m_i} (1/\hat{m}_i)\rangle_{m_i} = 1$).
6) $\lfloor u \rfloor$ is the floor value of any number u.

Manuscript received November 13, 2016; revised January 29, 2017; accepted March 9, 2017. Date of publication March 28, 2017; date of current version June 23, 2017.

The author is with the Department of Computer Engineering, Princess Sumaya University for Technology, Amman 11941, Jordan (e-mail: a.hiasat@psut.edu.jo).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TVLSI.2017.2681758

II. Residue-to-Binary Conversion Algorithm

In the past three decades, the residue-to-binary converters proposed in the literature used the Chinese remainder theorem (CRT), the mixed-radix conversion, or the new CRT [1], [28] to decode RNS to a weighted representation. This also applies to the converters cited in this brief. The progress achieved in proposing structurally simpler and faster converters was based on manipulating the different terms resulting from applying the CRT or the new CRT. In this section, the CRT is also used to produce a simpler converter structure for the four-moduli set $\{m_1, m_2, m_3, m_4\}$, where $m_1 = 2^n - 1$, $m_2 = 2^n + 1$, $m_3 = 2^{2n} + 1$, and $m_4 = 2^{2n+p}$. The general form of the CRT is given by [1]

$$X = \sum_{i=1}^{4} \hat{m}_i \left\langle \frac{1}{\hat{m}_i} \right\rangle_{m_i} R_i - MA(X) \tag{1}$$

where $A(X)$ is an integer that depends on the value of X. The residue representation of $X \in [0, 2^{2n+p}(2^{4n} - 1))$ is given by: (R_1, R_2, R_3, R_4). The binary formats of these four digits are as follows:

1) $R_1 = \sum_{i=0}^{n-1} r_{1(i)} 2^i = r_{1(n-1)} \cdots\cdots r_{1(0)}$;
2) $R_2 = \sum_{i=0}^{n} r_{2(i)} 2^i = r_{2(n)} \cdots\cdots r_{2(0)}$;
3) $R_3 = \sum_{i=0}^{2n} r_{3(i)} 2^i = r_{3(2n)} \cdots\cdots r_{3(0)}$;
4) $R_4 = \sum_{i=0}^{2n+p-1} r_{4(i)} 2^i = r_{4(2n+p-1)} \cdots\cdots r_{4(0)}$.

For this set, $\hat{m}_1 = (2^n + 1)(2^{2n} + 1)2^{2n+p}$, $\hat{m}_2 = (2^n - 1)(2^{2n} + 1)\, 2^{2n+p}$, $\hat{m}_3 = (2^{2n} - 1)2^{2n+p}$, and $\hat{m}_4 = (2^{4n} - 1)$. This brief introduces the multiplicative inverses for this extended four-moduli set. For the cases $(0 \le p \le n - 2)$, it can be easily proved that these multiplicative inverses are given by $\langle (1/\hat{m}_1) \rangle_{m_1} = 2^{n-p-2}$, $\langle (1/\hat{m}_2) \rangle_{m_2} = 2^{n-p-2}$, $\langle (1/\hat{m}_3) \rangle_{m_3} = \langle -2^{2n-p-1} \rangle_{m_3}$, and $\langle (1/\hat{m}_4) \rangle_{m_4} = \langle -1 \rangle_{m_4}$.

The new CRT proposed in [28] could have been used. However, the original form of the CRT [1] is used instead since this work introduces the new multiplicative inverses of the moduli under consideration. Therefore, (1) can be rewritten as

$$\begin{aligned} X = {} & (2^n + 1)(2^{2n} + 1)2^{2n+p}\, 2^{n-p-2} R_1 \\ & + (2^n - 1)(2^{2n} + 1)2^{2n+p}\, 2^{n-p-2} R_2 \\ & - (2^{2n} - 1)2^{2n+p}\, 2^{2n-p-1} R_3 \\ & - (2^{4n} - 1)R_4 - MA(X). \end{aligned} \tag{2}$$

Dividing (2) by 2^{2n+p}, taking the floor value, applying modulus $(2^{4n} - 1)$, and taking 2^{n-p-2} as a common factor, then (2) can be expressed as

$$\left\lfloor \frac{X}{2^{2n+p}} \right\rfloor = \langle 2^{n-p-2}(\ell_1 + \ell_2 + \ell_3 + \ell_4) \rangle_{(2^{4n}-1)} \tag{3}$$

where

$$\ell_1 = \langle (2^{2n} + 1)(2^n + 1)R_1 \rangle_{(2^{4n}-1)} \tag{4}$$

$$\ell_2 = \langle (2^{2n} + 1)(2^n - 1)R_2 \rangle_{(2^{4n}-1)} \tag{5}$$

$$\ell_3 = \langle -2^{n+1}(2^{2n} - 1)R_3 \rangle_{(2^{4n}-1)} \tag{6}$$

$$\ell_4 = \langle -2^{n+2} R_4 \rangle_{(2^{4n}-1)}. \tag{7}$$

Accordingly, $X = \lfloor X/(2^{2n+p}) \rfloor 2^{2n+p} + R_4$ is evaluated as a simple concatenation process [1].

When simplifying (4)–(7), the following modulo $(2^j - 1)$ properties will be used extensively [1], particularly in simplifying ℓ_2, ℓ_3, and ℓ_4. Property (1) states that modulo $(2^j - 1)$ multiplication of a residue number by 2^q, where j and q are positive integers, is equivalent to a q-bit circular left shift. Property (2) states that modulo $(2^j - 1)$ of a negative number is the one's complement of this number.

1) Simplifying (4) leads to

$$\begin{aligned} \ell_1 &= (2^{3n} + 2^{2n} + 2^n + 1)R_1 \\ &= \overbrace{R_1 \star R_1 \star R_1 \star R_1}^{4n} \\ &= \widetilde{R}_1 \end{aligned} \tag{8}$$

where $\star$ designates a concatenation process and where

$$\widetilde{R}_1 = R_1 \star R_1 \star R_1 \star R_1. \tag{9}$$

2) Simplifying (5) leads to

$$\ell_2 = \langle (2^{3n} - 2^{2n} + 2^n - 1)R_2 \rangle_{(2^{4n}-1)}. \tag{10}$$

Since $R_2 = 2^n r_{2(n)} + \hat{R}_2$, where $\hat{R}_2 = r_{2(n-1)} \cdots r_{2(0)}$, (10) can be rewritten as

$$\begin{aligned} \ell_2 &= \langle (2^{3n} - 2^{2n} + 2^n - 1)(2^n r_{2(n)} + \hat{R}_2) \rangle_{(2^{4n}-1)} \\ &= \left\langle (-2^{3n} + 2^{2n} - 2^n + 1) r_{2(n)} + \overbrace{\hat{R}_2 \star \overline{\hat{R}_2} \star \hat{R}_2 \star \overline{\hat{R}_2}}^{4n} \right. \\ &\qquad \left. + (2^{4n} - 2^{3n} + 2^{2n} - 2^n) \right\rangle_{(2^{4n}-1)} \\ &= \left\langle \overbrace{\hat{R}_2 \star \overline{\hat{R}_2} \star \hat{R}_2 \star \overline{\hat{R}_2}}^{4n} + (-2^{3n} + 2^{2n} - 2^n + 1) \right. \\ &\qquad \left. \times (1 + r_{2(n)}) \right\rangle_{(2^{4n}-1)} \\ &= \langle \widetilde{R}_2 + k_2 \rangle_{(2^{4n}-1)} \end{aligned} \tag{11}$$

where

$$\widetilde{R}_2 = \overbrace{\hat{R}_2 \star \overline{\hat{R}_2} \star \hat{R}_2 \star \overline{\hat{R}_2}}^{4n} \tag{12}$$

$$k_2 = \langle (-2^{3n} + 2^{2n} - 2^n + 1)(1 + r_{2(n)}) \rangle_{(2^{4n}-1)}. \tag{13}$$

3) Simplifying (6), since $R_3 = 2^{2n} r_{3(2n)} + \hat{R}_3$, where $\hat{R}_3 = r_{3(2n-1)} \cdots r_{3(0)}$, we have

$$\begin{aligned} \ell_3 &= \langle -2^{n+1}(2^{2n} - 1)(2^{2n} r_{3(2n)} + \hat{R}_3) \rangle_{(2^{4n}-1)} \\ &= \langle 2^{n+1}((2^{2n} - 1) r_{3(2n)} + (1 - 2^{2n})\hat{R}_3) \rangle_{(2^{4n}-1)} \\ &= \left\langle 2^{n+1}(\overbrace{\overline{\hat{R}_3} \star \hat{R}_3}^{4n} + (2^{2n} - 1)(1 + r_{3(2n)})) \right\rangle_{(2^{4n}-1)} \\ &= \langle \widetilde{R}_3 + k_3 \rangle_{(2^{4n}-1)} \end{aligned} \tag{14}$$

where

$$\widetilde{R}_3 = \left\langle 2^{n+1}(\overbrace{\overline{\hat{R}_3} \star \hat{R}_3}^{4n}) \right\rangle_{(2^{4n}-1)} \tag{15}$$

$$k_3 = \langle 2^{n+1}(2^{2n} - 1)(1 + r_{3(2n)}) \rangle_{(2^{4n}-1)}. \tag{16}$$

4) Simplifying (7), since $R_4 = \overbrace{r_{4(2n+p-1)} \cdots\cdots r_{4(0)}}^{2n+p}$, we have

$$\begin{aligned}
\ell_4 &= \langle -2^{n+2} R_4 \rangle_{(2^{4n}-1)} \\
&= \Big\langle -\overbrace{0\cdots 0}^{n-p-2}\,\overbrace{r_{4(2n+p-1)}\cdots r_{4(0)}}^{2n+p}\,\overbrace{0\cdots 0}^{n+2} \Big\rangle_{(2^{4n}-1)} \\
&= \Big\langle \overbrace{1\cdots 1}^{n-p-2}\,\overbrace{\overline{r}_{4(2n+p-1)}\cdots \overline{r}_{4(0)}}^{2n+p}\,\overbrace{1\cdots 1}^{n+2} \Big\rangle \\
&= \Big\langle \overbrace{1\cdots 1}^{n-p-2}\,\overbrace{\overline{r}_{4(2n+p-1)}\cdots \overline{r}_{4(0)}}^{2n+p}\,\overbrace{0\cdots 0}^{n+2} + (2^{n+2}-1) \Big\rangle_{(2^{4n}-1)} \\
&= \langle \widetilde{R}_4 + k_4 \rangle_{(2^{4n}-1)} \qquad (17)
\end{aligned}$$

where

$$\widetilde{R}_4 = \overbrace{1\cdots 1}^{n-p-2}\,\overbrace{\overline{r}_{4(2n+p-1)}\cdots \overline{r}_{4(0)}}^{2n+p}\,\overbrace{0\cdots 0}^{n+2} \qquad (18)$$

$$k_4 = 2^{n+2} - 1. \qquad (19)$$

Defining $k = k_2 + k_3 + k_4$ and substituting (8), (11), (14), and (17) into (3) produce

$$\left\lfloor \frac{X}{2^{n+p}} \right\rfloor = \langle 2^{n-p-2}(\widetilde{R}_1 + \widetilde{R}_2 + \widetilde{R}_3 + \widetilde{R}_4 + k)\rangle_{(2^{4n}-1)}. \qquad (20)$$

Following a similar approach to that adopted in [23], the value of k can be expressed in terms of different combinations of $r_{2(n)}$ and $r_{3(2n)}$ using (13), (16), and (19) as follows:

$$k = \begin{cases} 2^{3n} + 2^{2n} + 2^{n}, & r_{2(n)} r_{3(2n)} = 00 \\ 2^{3n+1} + 2^{3n} + 2^{2n} - 2^{n}, & r_{2(n)} r_{3(2n)} = 01 \\ 2^{2n+1} + 1, & r_{2(n)} r_{3(2n)} = 10 \\ 2^{3n+1} + 2^{2n+1} - 2^{n+1} + 1, & r_{2(n)} r_{3(2n)} = 11. \end{cases} \qquad (21)$$

Using binary notation, (21) can be written as

$$k = \begin{cases} \overbrace{0\cdot\cdot 001}^{n}\,\overbrace{0\cdot\cdot 001}^{n}\,\overbrace{0\cdot\cdot 001}^{n}\,\overbrace{0\cdot\cdot 00}^{n}, & r_{2(n)} r_{3(2n)} = 00 \\ \overbrace{0\cdot\cdot 011}^{n}\,\overbrace{0\cdot\cdot 000}^{n}\,\overbrace{1\cdot\cdot 111}^{n}\,\overbrace{0\cdot\cdot 00}^{n}, & r_{2(n)} r_{3(2n)} = 01 \\ \overbrace{0\cdot\cdot 000}^{n}\,\overbrace{0\cdot\cdot 010}^{n}\,\overbrace{0\cdot\cdot 000}^{n}\,\overbrace{0\cdot\cdot 01}^{n}, & r_{2(n)} r_{3(2n)} = 10 \\ \overbrace{0\cdot\cdot 010}^{n}\,\overbrace{0\cdot\cdot 001}^{n}\,\overbrace{1\cdot\cdot 110}^{n}\,\overbrace{0\cdot\cdot 01}^{n}, & r_{2(n)} r_{3(2n)} = 11. \end{cases} \qquad (22)$$

The four different combinations in (22) can be combined into one expression as follows:

$$k = \overbrace{0\cdots 0 r_{3(2n)} \overline{r}_{2(n)}}^{n}\,\overbrace{0\cdots 0 b_1 b_2}^{n}\,\overbrace{r_{3(2n)}\cdots r_{3(2n)} \overline{r}_{2(n)}}^{n}\,\overbrace{0\cdots 0 r_{2(n)}}^{n} \qquad (23)$$

where $b_1 = r_{2(n)} \wedge \overline{r}_{3(2n)}$ and $b_2 = r_{2(n)} \odot r_{3(2n)}$.

For the cases ($p = n-1$ or $p = n$), the multiplicative inverses are $\langle 1/\hat{m}_1 \rangle_{m_1} = 2^{2n-p-2}$ and $\langle 1/\hat{m}_2 \rangle_{m_2} = \langle -2^{2n-p-2} \rangle_{m_2}$, while $\langle 1/\hat{m}_3 \rangle_{m_3}$ and $\langle 1/\hat{m}_4 \rangle_{m_4}$ remain the same. Hence, the same approach introduced above is also applicable.

III. Hardware Implementation, Evaluation, Comparison, and VLSI Realization

Fig. 1 shows the hardware implementation of the proposed four-moduli reverse converter. It consists of one modulo $(2^{4n}-1)$ adder and three carry-save adders (CSAs), namely, CSA-1, CSA-2, and CSA3. CSA-1 adds $\widetilde{R}_1$, $\widetilde{R}_2$, and $\widetilde{R}_3$ as follows:

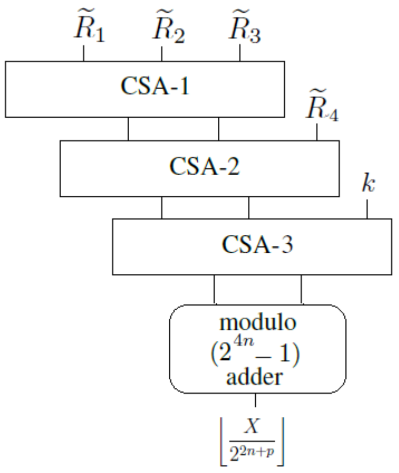

Fig. 1. Hardware structure of the proposed residue-to-binary converter.

$$\begin{aligned}
\widetilde{R}_1 + \widetilde{R}_2 + \widetilde{R}_3 &= R_1 \star R_1 \star R_1 \star R_1 \\
&\quad + \hat{R}_2 \star \overline{\hat{R}_2} \star \hat{R}_2 \star \overline{\hat{R}_2} \\
&\quad + \langle 2^{n+1}(\overline{\hat{R}_3} \star \hat{R}_3) \rangle_{(2^{4n}-1)}. \qquad (24)
\end{aligned}$$

As can be concluded from (24), the operands of $\widetilde{R}_1$, $\widetilde{R}_2$, and $\widetilde{R}_3$ have repeated patterns. These repeated patterns are used to reduce hardware requirements. Recalling that for the three single-bit variables a, b and c: $a \oplus b \oplus c = a \oplus \overline{b} \oplus c = a \oplus \overline{b} \oplus \overline{c}$, the result of evaluating (24) using CSA-1 can be expressed in binary form as a $4n$-bit sum vector ($s_{4n-1} \cdots\cdots s_0$) and a $4n$-bit carry vector ($c_{4n} \cdots\cdots c_1$) as follows:

$$\begin{array}{r}
\overbrace{r_{1(n-1)}\cdots r_{1(0)}}^{n}\,\overbrace{r_{1(n-1)}\cdots r_{1(0)}}^{n}\,\overbrace{r_{1(n-1)}\cdots r_{1(0)}}^{n}\,\overbrace{r_{1(n-1)}\cdots r_{1(0)}}^{n} \\
+\,\overbrace{r_{2(n-1)}\cdots r_{2(0)}}^{n}\,\overbrace{\overline{r}_{2(n-1)}\cdots \overline{r}_{2(0)}}^{n}\,\overbrace{r_{2(n-1)}\cdots r_{2(0)}}^{n}\,\overbrace{\overline{r}_{2(n-1)}\cdots \overline{r}_{2(0)}}^{n} \\
+\,\overbrace{\overline{r}_{3(n-2)}\cdots \overline{r}_{3(0)}}^{n-1}\,\overbrace{r_{3(2n-1)}\cdots r_{3(0)}}^{2n}\,\overbrace{\overline{r}_{3(2n-1)}\cdots \overline{r}_{3(n-1)}}^{n+1} \\
\hline
\overbrace{s_{4n-1}\cdots s_{3n}}^{n}\,\overbrace{s_{3n-1}\cdots s_{2n}}^{n}\,\overbrace{s_{2n-1}\cdots s_{n}}^{n}\,\overbrace{s_{n-1}\cdots s_{0}}^{n} \\
+\,\overbrace{c_{4n}\cdots c_{3n+1}}^{n}\,\overbrace{c_{3n}\cdots c_{2n+1}}^{n}\,\overbrace{c_{2n}\cdots c_{n+1}}^{n}\,\overbrace{c_{n}\cdots c_{1}}^{n}
\end{array}. \qquad (25)$$

Fig. 2(a) and (b) shows the hardware realization of the sum vector and the carry vector, respectively. Figs. 2(a) and (b) constitutes CSA-1. Thus, CSA-1 requires $3n$ XOR gates and $10n$ AND/OR gates. CSA-2 adds $\widetilde{R}_4$ to the outputs of CSA-1. $\widetilde{R}_4$ consists of $4n$ bits as follows: $(2n+p)$ variable bits and $(2n-p)$ zeros/ones. However, CSA-3 adds k to the outputs of CSA-2, where k consists of $(3n-5)$ ones/zeros and $(n+5)$ variable bits. Hence, exploiting the repeated binary patterns in (24) and (25) on one side and utilizing the relation between the two logic operations (XOR and XNOR) on the other side led to a significant gate count reduction. This gate count reduction and specifically in CSA-1 resulted in significant area, time, and power reductions. The newly introduced simple closed-form multiplicative inverses also produced simplified realizations.

The most relevant and competitive works are the ones in [14], [16], [17], and [23]. The converter introduced in [23] deals with an extended five-moduli set with a dynamic range of $(6n+p)$ bits, where p is limited to the range: $0 \leq p \leq ((n-5)/2)$.

TABLE I
HARDWARE AND TIME REQUIREMENTS OF DIFFERENT RESIDUE-TO-BINARY CONVERTERS

Converter	No. of Moduli	Dynamic Range bits	Full Adder (FA)	XOR/ XNOR	AND/OR	$(2^{4n}-1)$ MA	Delay
Proposed	4	$6n+p$	$3n+p+5$	$8n-p-4$	$15n-p-4$	1	$3t_{FA}+t_{MA(4n)}$
[14]	4	$5n$	$11n+6$	$6n-1$	$6n-1$	1	$3t_{FA}+t_{MA(4n)}$
[16]	4	$6n$	$10n+6$	$6n-6$	$6n-6$	1	$3t_{FA}+t_{MA(4n)}$
[17]	4	$5n+p$	$5n+p+6$	$6n-p-6$	$14n-p-15$	1	$3t_{FA}+t_{MA(4n)}$
[23]	5	$6n+p$	$16n+p+4$	$14n-2p-8$	$7n-p-4$	1	$4t_{FA}+t_{MA(4n)}$

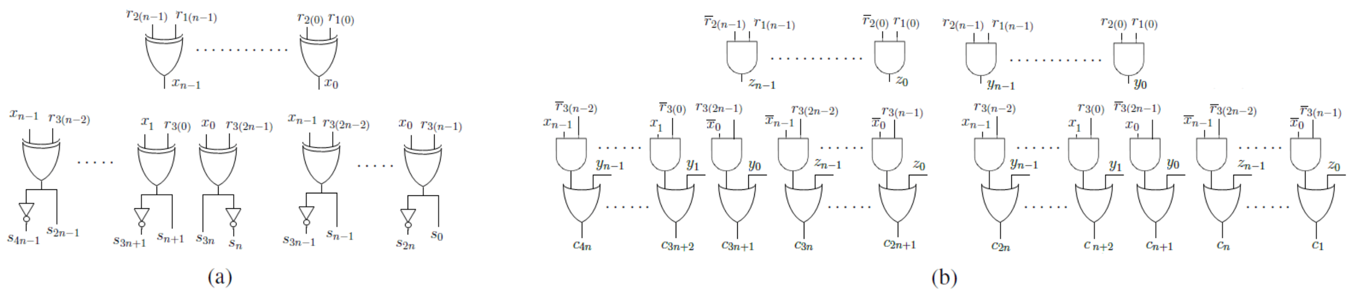

Fig. 2. Hardware needed to produce the (a) sum vector and (b) carry vector of CSA-1 of Fig. 1.

TABLE II
COMPARISON BETWEEN DIFFERENT CONVERTERS USING GE AND VLSI SYNTHESIS RESULTS OF AREA (A), DELAY (D), POWER (P), ENERGY (EN), AND EFFICIENCY (EF)

	n	M bits	GE #	A μm^2	D ps	P μW	En fJ	Ef $\frac{MCps}{m^2}$
New	7	42	947	4152	798	98	79	301
	11	66	1607	6113	835	111	93	195
	15	90	2183	9375	858	120	103	124
[14]	7	35	1264	5381	852	112	96	218
	11	55	2096	7796	887	121	108	144
	15	75	2844	11591	916	132	121	94
[16]	7	42	1200	5198	834	111	93	230
	11	66	2004	7556	879	119	105	150
	15	90	2724	11247	907	130	119	98
[17]	7	42	1030	4629	833	108	91	259
	11	66	1742	7003	880	123	109	162
	15	90	2370	10428	904	132	119	106
[23]	7	42	1597	6539	969	161	156	157
	11	66	2637	10319	1085	181	197	89
	15	90	3593	16238	1106	260	288	55

TABLE III
RELATIVE PERFORMANCE INDICATORS OF THE NEW CONVERTER: GE REDUCTION (GER), AREA REDUC. (AR), DELAY REDUC. (DR), POWER REDUC. (PR), ENERGY REDUC. (ENR), EFFICIENCY IMPROVEMENT (EFI), AND THE AVERAGE OF VALUES (AVR)

	n	% GER	% AR	% DR	% PR	% EnR	% EfI
[14]	7	25.1	22.8	6.3	12.2	17.7	38.1
	11	23.3	21.6	5.9	8.5	13.9	35.4
	15	23.2	19.1	6.3	9.0	14.9	31.9
[16]	7	21.1	20.1	4.3	11.0	15.1	30.8
	11	19.8	19.1	5.0	6.5	11.4	30.0
	15	19.9	16.6	5.4	7.9	13.4	27.8
[17]	7	8.1	10.3	4.2	9.1	13.2	16.2
	11	7.7	12.7	5.2	10.2	14.7	20.4
	15	7.9	10.1	5.1	8.8	13.4	17.0
[23]	7	40.7	36.5	17.6	38.7	49.4	91.7
	11	39.1	40.8	23.0	38.6	52.8	119.1
	15	39.2	42.3	22.4	53.6	64.2	125.5
AVR			22.7	9.2	17.8	24.5	48.7

Table I summarizes the hardware and time requirements of the proposed converter and the ones in [14], [16], [17], and [23], where the inverters are ignored.

Additionally, an implementation using VLSI tools was conducted, where the (2^n-1) modular adder proposed in [29] was used in all structures. All the converters were modeled using Verilog HDL for different values of n. The modeled designs were synthesized using Synopsys Design Compiler (G-2012.06). The synthesized structures were also mapped to 65-nm Synopsys DesignWare Digital Logic Libraries. Synopsis Power Complier was used to accurately estimate the power consumption. The five structures were optimized and functionally checked. Table II lists the results of VLSI implementation after place and route phase for area, time delay, power, energy, and efficiency (i.e., $=((\text{throughput})/(\text{area}))$ in million conversions per second per m^2). Table II also lists the number of gate equivalents (GEs) for each design using unit gate model [8]. The reductions achieved are listed in Table III. Compared with [14], [16], [17], and [23], Table III indicated that the proposed converter reduced the area by an average of 22.7%, delay by an average of 9.2%, power by an average of 17.8%, and energy by an average of 24.5%. The new proposed converter has improved the efficiency by an average of 48.7%.

Although the above results are based on a 65-nm application-specific integrated circuit (ASIC) technology, the proposed converter would still outperform competitive converters even when selecting other ASIC technologies. This is due to the fact that the new design requires less GE, where the GE is a technology-independent indicator. On the other hand, field programmable gate arrays (FPGAs) are also suitable for implementing logic-intensive circuits like the reverse converters considered in this brief. However, experimental results showed that the delay of FPGA-based converters implemented using the 65-nm Virtex-5 FPGA (e.g., XC5VLX220) from Xilinx is an order of magnitude larger than the corresponding ASIC-based implementations [11]. The regularity of the basic components of the converters works best for an ASIC-based implementation when considering the "place and route phase." Thorough information on implementing four-moduli set converters using ASICs and FPGAs is found in [18] and [30].

IV. Conclusion

Using the newly introduced compact forms of the multiplicative inverses, this brief proposed a residue-to-binary converter for the extended moduli set $\{2^n - 1, 2^n + 1, 2^{2n} + 1, 2^{2n+p}\}$. Compared with the earlier work, the proposed structure reduced the hardware requirements in the CSA network of the converter. This hardware reduction reflected positively on the area, time, power, and energy of the proposed work when synthesized and compared with other published works. It also improved the efficiency considerably in terms of the throughout rate per unit area.

References

[1] N. S. Szabó and R. Tanaka, *Residue Arithmetic and Its Applications to Computer Technology*. New York, NY, USA: McGraw-Hill, 1967.

[2] M. A. Soderstrand, W. K. Jenkins, G. A. Jullien, and F. J. Taylor, Eds., *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*. New York, NY, USA: IEEE Press, 1986.

[3] C. B. Dutta, P. Garai, and A. Sinha, "Design of a reconfigurable DSP processor with bit efficient residue number system," *Int. J. VLSI Design Commun. Syst.*, vol. 3, no. 5, pp. 175–189, Oct. 2012.

[4] M. Esmaeildoust, D. Schinianakis, H. Javashi, T. Stouraitis, and K. Navi, "Efficient RNS implementation of elliptic curve point multiplication over GF(p)," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 8, pp. 1545–1549, Aug. 2013.

[5] J. C. Bajard, N. Meloni, and T. Plantard, "Efficient RNS bases for cryptography," in *Proc. 17th IMACS World Congr., Sci. Comput., Appl. Math. Simulation*, Paris, France, Jul. 2005, pp. 1–7.

[6] S. Antão, J.-C. Bajard, and L. Sousa, "Elliptic curve point multiplication on GPUs," in *Proc. 21st IEEE Int. Conf. Appl.-Specific Syst., Archit. Processors*, Jul. 2010, pp. 192–199.

[7] W. Wang, M. N. S. Swamy, M. O. Ahmad, and Y. Wang, "A study of the residue-to-binary converters for the three-moduli sets," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 50, no. 2, pp. 235–243, Feb. 2003.

[8] R. Zimmermann, "Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication," in *Proc. 14th IEEE Symp. Comput. Arithmetic*, Apr. 1999, pp. 158–167.

[9] L. Sousa and R. Chaves, "$\{2^n + 1, 2^{n+k}, 2^n - 1\}$: A new RNS moduli set extension," in *Proc. EUROMICRO Syst. Digit. Syst. Design (DSD)*, Aug. 2004, pp. 210–217.

[10] P. V. A. Mohan and A. B. Premkumar, "RNS-to-binary converters for two four-moduli sets $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ and $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1\}$," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 6, pp. 1245–1254, Jun. 2007.

[11] L. Sousa, S. Antão, and R. Chaves, "On the design of RNS reverse converters for the four-moduli set $\{2^n + 1, 2^n - 1, 2^n, 2^{n+1} + 1\}$," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 21, no. 10, pp. 1945–1949, Oct. 2013.

[12] P. Patronik and S. J. Piestrak, "Design of reverse converters for general RNS moduli sets $\{2^k, 2^n - 1, 2^n + 1, 2^{n+1} - 1\}$ and $\{2^k, 2^n - 1, 2^n + 1, 2^{n-1} - 1\}$ (n even)," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 6, pp. 1687–1700, Jun. 2014.

[13] P. Patronik and S. J. Piestrak, "Design of reverse converters for the new RNS moduli set $\{2^n + 1, 2^n - 1, 2^n, 2^{n-1} + 1\}$ (n odd)," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 61, no. 12, pp. 3436–3449, Dec. 2014.

[14] B. Cao, C.-H. Chang, and T. Srikanthan, "An efficient reverse converter for the 4-moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{2n} + 1\}$ based on the new Chinese remainder theorem," *IEEE Trans. Circuits Syst. I, Fundam. Theory Appl.*, vol. 50, no. 10, pp. 1296–1303, Oct. 2003.

[15] Y.-C. Kuo, S.-H. Lin, M.-H. Sheu, J.-Y. Wu, and P.-S. Wang, "Efficient VLSI design of a reverse RNS converter for new flexible 4-moduli set $(2^{\mathrm{p+k}}, 2^{\mathrm{p}} + 1, 2^{\mathrm{p}} - 1, 2^{2\mathrm{p}} + 1)$," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2009, pp. 437–440.

[16] A. S. Molahosseini, K. Navi, C. Dadkhah, O. Kavehei, and S. Timarchi, "Efficient reverse converter designs for the new 4-moduli sets $\{2^n - 1, 2^n, 2^n + 1, 2^{2\mathrm{n}} + 1 - 1\}$ and $\{2^n - 1, 2^n + 1, 2^{2n}, 2^{2n} + 1\}$ based on new CRTs," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 57, no. 4, pp. 823–835, Apr. 2010.

[17] M.-H. Sheu, Y.-C. Kuo, S.-H. Lin, and S.-M. Siao, "Efficient reverse converter design for new adaptable four-moduli set $\{2^{n+k}, 2^n + 1, 2^n - 1, 2^{2n} + 1\}$," *IEICE Trans. Fundam. Electron., Commun. Comput. Sci.*, vol. E96-A, no. 7, pp. 1571–1578, Jul. 2013.

[18] L. Sousa and S. Antão, "MRC-based RNS reverse converters for the four-moduli sets $\{2^n + 1, 2^n - 1, 2^n, 2^{2n+1} - 1\}$ and $\{2^n + 1, 2^n - 1, 2^{2n}, 2^{2n+1} - 1\}$," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 59, no. 4, pp. 244–248, Apr. 2012.

[19] A. A. Hiasat, "VLSI implementation of new arithmetic residue to binary decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 1, pp. 153–158, Jan. 2005.

[20] B. Cao, C.-H. Chang, and T. Srikanthan, "A residue-to-binary converter for a new five-moduli set," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 5, pp. 1041–1049, May 2007.

[21] H. Ahmadifar and G. Jaberipur, "A new residue number system with 5-moduli set: $\{2^{2q}, 2^q \pm 3, 2^q \pm 1\}$," *Comput. J.*, vol. 58, no. 7, pp. 1548–1565, Jul. 2015.

[22] H. Pettenghi, R. Chaves, and L. Sousa, "RNS reverse converters for moduli sets with dynamic ranges up to $(8n+1)$-bit," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 60, no. 6, pp. 1487–1500, Jun. 2013.

[23] A. Hiasat, "A reverse converter and sign detectors for an extended RNS five-moduli set," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 1, pp. 111–121, Jan. 2017.

[24] H. Pettenghi, R. Chaves, and L. Sousa, "Method to design general RNS reverse converters for extended moduli sets," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 60, no. 12, pp. 877–881, Dec. 2013.

[25] A. A. E. Zarandi, A. S. Molahosseini, L. Sousa, and M. Hosseinzadeh, "An efficient component for designing signed reverse converters for a class of RNS moduli sets of composite form $\{2^k, 2^P - 1\}$," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 25, no. 1, pp. 48–59, Jan. 2017.

[26] A. A. Hiasat, "A suggestion for a fast residue multiplier for a family of moduli of the form $(2^n - (2^p \pm 1))$," *Comput. J.*, vol. 47, no. 1, pp. 93–102, Jan. 2004.

[27] P. M. Matutino, H. Pettenghi, R. Chaves, and L. Sousa, "RNS arithmetic units for modulo $\{2^n \pm k\}$," in *Proc. Euromicro Conf. Digit. Syst. Design*, Sep. 2012, pp. 795–802.

[28] Y. Wang, "New Chinese remainder theorems," in *Proc. 32nd Asilomar Conf. Signals, Syst., Comput.*, vol. 1. Nov. 1998, pp. 165–171.

[29] L. Kalampoukas, D. Nikolos, C. Efstathiou, H. T. Vergos, and J. Kalamatianos, "High-speed parallel-prefix module $2^n - 1$ adders," *IEEE Trans. Comput.*, vol. 49, no. 7, pp. 673–680, Jul. 2000.

[30] A. A. E. Zarandi, A. S. Molahosseini, and L. Sousa, "ASIC and FPGA implementations of modern 4-moduli RNS reverse converters using distinct configurations," Inst. Engenharia Sistemas Computadores, Lisbon, Portugal, INESC-ID Tech. Rep. 8, Mar. 2015.