SUT-RNS Residue-to-Binary Converters Design

Evangelos Vassalos, Dimitris Bakalis
Electronics Laboratory, Dept. of Physics
University of Patras
Patras, Greece
vassalos@upatras.gr, bakalis@physics.upatras.gr

Haridimos T. Vergos
Dept. of Computer Engineering & Informatics
University of Patras
Patras, Greece
vergos@ceid.upatras.gr

***Abstract*—The Stored Unibit Transfer (SUT) encoding has been recently proposed as a redundant high-radix encoding for each of the channels of a Residue Number System (RNS) that can improve the efficiency of Binary Signed Digit (BSD)-encoded RNS. However, a residue-to-binary (reverse) converter for it has not yet been reported in the open literature. In this paper we introduce SUT-RNS reverse converters for two different moduli sets, that is, for the 3-moduli $\{2^n\text{-}1, 2^n, 2^n\text{+}1\}$ and for the 4-moduli $\{2^n\text{-}1, 2^n, 2^n\text{+}1, 2^{2n}\text{+}1\}$ sets. The area and delay costs of the proposed converters are shown to be less than those required by the corresponding RNS converters for the BSD encoding. In the 4-moduli set case, the converters' costs are shown to be close to those of the corresponding converters for the binary encoding.**

Keywords-Redundant arithmetic; residue number system; reverse converter; stored unibit transfer;

I. INTRODUCTION

The Residue Number System (RNS) [1] [2] is a carry-limited number system commonly adopted for speeding-up computations in digital signal processing [3] [4], cryptography [5] [6] and communication units [7]. An RNS is characterized by a set $\{m_1, m_2, \ldots, m_p\}$ of p moduli that are pair-wise relatively prime. An integer A, with $0 \le A < M$, where $M = m_1 \times m_2 \times \ldots \times m_p$, has a unique representation in the RNS given by the set of residues $\{a_1, a_2, \ldots, a_p\}$, where a_i is the least non-negative remainder of the division of A by m_i, with $i = 1, \ldots, p$. An arithmetic operation $\otimes$ on two RNS operands A and B is defined as $Z=A \otimes B \leftrightarrow (z_1, \ldots, z_p) = (a_1, \ldots, a_p) \otimes (b_1, \ldots, b_p)$, where $z_i = \left| a_i \otimes b_i \right|_{m_i}$ is the residue of $a_i \otimes b_i$ taken modulo m_i. This means that all arithmetic operations are performed on narrow residues instead of wide operands, while also being carried out in parallel in separate arithmetic units known as channels. Furthermore, no carry propagation exists between any two channels, making high-speed parallel arithmetic processing possible. RNSs built on moduli of the $2^n \pm 1$ forms have received significant attention due to the efficient architectures that have been proposed for the design of the respective arithmetic components.

Several contemporary applications require high dynamic ranges making the carry propagation in the computations a speed-limiting factor. A typical example is cryptosystems. Although RNS can be used for speeding-up computations in those applications (see for example [5] [6] for asymmetric ciphers), the remaining intra-channel carry propagation can still be large. In order to reduce the intra-channel carry propagation in RNS, redundant encodings have been proposed for the residues inside each channel. The Binary Signed Digit (BSD) encoding [8], which uses the digit set {-1, 0, +1}, was one of the first considered [9-11]. Due to its redundancy, the length of carry propagation chain is limited to only one digit position, instead of the entire operand length. Thus, BSD offers constant-time addition independently of the operands' width. On the other hand, each BSD digit requires two bits for its encoding and dedicated building blocks have to be used in the design of arithmetic circuits, leading to large area and routing overheads. In order to decrease the added costs related to BSD, hybrid redundant encodings, such as those with weighted two-valued digit sets have been alternatively proposed [12] [13]. These encodings offer a tradeoff between the area and routing overheads and the carry propagation time. Furthermore, they can utilize standard building blocks such as full or half adders instead of custom ones, resulting that way to more efficient implementations. A hybrid redundant encoding that has been recently proposed for the modulo $2^n \pm 1$ channels' operands, along with efficient architectures for modulo $2^n \pm 1$ addition, subtraction, multiplication and encoding conversion, is the SUT-RNS encoding [14-16], which is briefly reviewed in section II.

In every RNS system, the operands must be converted from their binary representation to their corresponding residues and vice versa. The reverse (residue-to-binary) conversion is a complex process and should be efficiently realized so as to prevent performance degradation of the overall RNS system. An extensive amount of research has been done on the design of efficient RNS reverse converters for various moduli sets assuming either a binary [17-22] or a BSD encoding [10] [11] at the converters' inputs. All these converters are based on the Chinese Remainder Theorem (CRT), the New Chinese Remainder Theorems (New-CRT I and New-CRT II) or the Mixed-Radix Conversion (MRC) algorithm. However, besides a preliminary attempt to convert a single modulo $2^n \pm 1$ operand from/to its binary-RNS to/from its SUT-RNS encoding [16], no reverse converter of any moduli set has been presented so far for the SUT-RNS encoding.

In this paper we deal with the problem of designing efficient RNS reverse converters for the SUT-RNS encoding. We consider two different moduli sets: (a) the well-known 3-moduli set $\{2^n\text{-}1, 2^n, 2^n\text{+}1\}$ which has a dynamic range approximately equal to $3n$ bits, and (b) the 4-moduli set $\{2^n\text{-}1, 2^n, 2^n\text{+}1, 2^{2n}\text{+}1\}$ which has a larger dynamic range approximately equal to $5n$ bits. Efficient reverse converters for them exist both for a binary [18] [20] and a BSD [11]

978-0-7695-4798-5/12 $26.00 © 2012 IEEE
DOI 10.1109/DSD.2012.121

encoding at the converters' inputs, which enables a comparison with the proposed converters. Their evaluation reveals that they are far more efficient, in terms of area and delay, than the corresponding converters for the BSD encoding while in the 4-moduli set case the overhead of the proposed converters, compared to the corresponding converters for the binary encoding, is very small.

The rest of the paper is organized as follows. The next section briefly reviews the SUT-RNS encoding. Section III presents some properties that are useful in the reverse conversion and introduces the converters for the two moduli sets. Section IV compares the proposed converters against the corresponding converters for the binary and the BSD encoding. The last section concludes.

II. Redundant High-Radix SUT-RNS Encoding

SUT-RNS has been proposed as a redundant, high-radix encoding for RNS-based systems with moduli of the $2^n \pm 1$ forms. Architectures for the design of modulo $2^n \pm 1$ adders, subtractors and multipliers according to this encoding have been recently presented [14] [15]. The SUT-RNS can be also used for the modulo 2^n channel of an RNS and modulo 2^n arithmetic units can be easily derived based on [12] and [13]. The main advantage of the SUT-RNS encoding, against the BSD, is its ability to use standard building blocks (full adders, half adders) in the design of the corresponding arithmetic units. Furthermore, compared to the BSD encoding, the SUT-RNS encoding reduces significantly the area and routing overhead at the cost of a small increase in delay, while the selection the radix value in SUT-RNS can be used to trade-off between these two parameters. Hence, the SUT-RNS encoding is perfectly suited for applications having large dynamic ranges and strict area and delay constraints.

Every SUT-encoded number is composed of SUT digits. Each SUT digit consists of several two-valued digits (twits) of three types: posibits {0, +1}, negabits {−1, 0}, and unibits {−1, +1} [12]. A posibit has a lower value equal to 0 whereas a negabit and a unibit have a lower value equal to −1. All three types of twits require one bit for their representation and use bias encoding, that is, their lower value is encoded in binary with the logical 0 whereas their upper value is encoded in binary with logical 1. The dot and symbolic notations along with the binary encodings of the twits are listed in Table I. Hereafter, the symbolic notation of a twit will be used to denote its logical value and not its arithmetic value.

Every SUT-RNS-encoded number consists of k radix-2^h SUT digits. The symbolic and dot notation, of a k-digit ($D_{k-1}\ldots D_0$) radix-2^h SUT-RNS-encoded number X, are shown in Fig. 1. The negabits and the posibits represent the main part of X whereas the unibits represent the transfer part of X. Each radix-2^h SUT digit consists of (h+1) twits, that is, (h−1) posibits, 1 negabit and 1 unibit. In every SUT digit, the negabit along with the (h−1) posibits, each having a distinct weight equal to a power of 2, represent a number in the $[-2^{h-1}, +2^{h-1}-1]$ range. Since the minimum and maximum values of D_i, $0 \le i < k$, are equal to $-(2^{h-1}+1)$ and $+2^{h-1}$, respectively,

TABLE I. Binary Encoding, Dot and Symbolic Notation of Twits

Twit	*Dot notation*	*Symbolic notation*	*Lower value*	*Upper value*
Negabit	○	X_i	0 (−1)	1 (0)
Posibit	●	x_i	0 (0)	1 (+1)
Unibit	■	x'_i	0 (−1)	1 (+1)

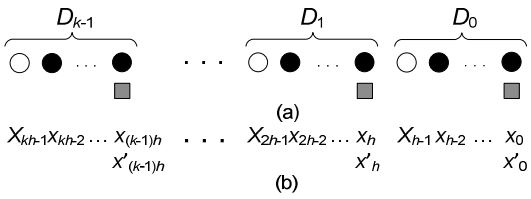

Figure 1. (a) Dot and (b) symbolic notation of a k-digit radix-2^h SUT-RNS number X.

and since each D_i has a weight equal to 2^{ih}, the maximum representable number is equal to $X_{MAX} = +2^{h-1} \cdot R$, whereas the minimum representable number is equal to $X_{MIN} = -(2^{h-1}+1) \cdot R$, where $R = (2^{kh}-1)/(2^h-1)$.

A modulo 2^n+1 (2^n-1) number X, $X \in [0, 2^n]$ ($X \in [0, 2^n-1]$ assuming a double representation for zero), can be encoded in SUT-RNS by selecting the appropriate values of k and h so that $n=k\times h$ and by utilizing the positive range of the SUT-RNS encoding for some values of X and the negative range for the remaining values.

Example 1

When k=2 and h=3 (n=6), an SUT-RNS encoding can represent all numbers between -45 and +36. A single radix-8 SUT digit in this case can represent all numbers between -5 and +4. The value 7 can be encoded with two SUT digits as $+7 = +1\times 8^1 - 1\times 8^0$. Since +1 and −1 can be written as $+1 = 0\times 2^2 + 1\times 2^1 + 0\times 2^0 - 1\times 2^0$ and $-1 = 0\times 2^2 + 0\times 2^1 + 0\times 2^0 - 1\times 2^0$, a possible SUT-RNS encoding of 7 can be $\left\langle \begin{matrix} 110 & 100 \\ 0 & 0 \end{matrix} \right\rangle$ in both moduli 2^6-1 and 2^6+1 cases. Furthermore, it holds that $|37|_{65} = |-28|_{65}$ and $|37|_{63} = |-26|_{63}$. Hence, the value 37 can be encoded in SUT-RNS as $-28 = -3\times 8^1 - 4\times 8^0$ in modulo 2^6+1 and can be encoded as $-26 = -3\times 8^1 - 2\times 8^0$ in modulo 2^6-1 arithmetic. Since values −3, −4 and −2 can be written as $-3 = -1\times 2^2 + 0\times 2^1 + 0\times 2^0 + 1\times 2^0$, $-4 = -1\times 2^2 + 0\times 2^1 + 1\times 2^0 - 1\times 2^0$, and $-2 = -1\times 2^2 + 1\times 2^1 + 1\times 2^0 - 1\times 2^0$, two possible SUT-RNS encodings of 37 in modulo 2^6+1 and 2^6-1 can be $\left\langle \begin{matrix} 000 & 001 \\ 1 & 0 \end{matrix} \right\rangle$ and $\left\langle \begin{matrix} 000 & 011 \\ 1 & 0 \end{matrix} \right\rangle$ respectively. ■

III. Residue-to-Binary Converters

Before introducing the reverse converters for the two moduli sets, we present some properties that are useful in their design.

Let $X = \left\langle \begin{matrix} X_{kh-1}x_{kh-2}\cdots x_{(k-1)h} & \cdots & X_{h-1}x_{h-2}\cdots x_0 \\ x'_{(k-1)h} & & x'_0 \end{matrix} \right\rangle$ denote a k digit radix-2^h SUT-RNS encoded number (see Fig. 1). X is composed of posibits, negabits and unibits. Let also $Y = y_{n-1}\cdots y_0$ and $Z = z_{n-1}\cdots z_0$ denote two n-bit vectors, where $y_{ih-1} = X_{ih-1}$, $0 < i \le k$, $y_{ih-j} = x_{ih-j}$, $0 < i \le k$, $2 \le j \le h$, and $z_{ih+1} = x'_{ih}$, $0 \le i < k$, while all the remaining bits of Z are equal to 0. In other words, Y contains the logical values of the main part of X, while Z contains the logical values of the transfer part (unibits) of X shifted one position to the left.

Property I: Considering the arithmetic values of X, Y and Z, it holds that: $X = Y + Z + X_{MIN}$

Proof: Since −1 = 0−1 and 0 = 1−1, every negabit can be treated as a posibit with the same logical value as long as we decrease its value by one. Similarly, since −1=2×0−1 and +1=2×1−1, every unibit can be treated as a doublebit (posibit with double weight) as long as a correction equal to −1 is also taken into account. Hence:

$$X = Y + Z - \sum_{i=1}^{k} 2^{ih-1} - \sum_{i=1}^{k} 2^{(i-1)h}$$
$$= Y + Z - (2^{h-1}+1)\cdot R = Y + Z + X_{MIN} \quad \blacksquare$$

Property I indicates that we can split an SUT-RNS encoded number in two binary vectors as long as we include a correction term. Furthermore, it shows that if we add those two vectors, Y and Z, with the constant X_{MIN}, then we can get the value of X in binary.

Consider that m denotes an integer greater than or equal to zero.

Property II: It holds that:

$$\left|2^m \cdot X\right|_{2^n-1} = \left\| 2^m \cdot Y \right|_{2^n-1} + \left|2^m \cdot Z\right|_{2^n-1} + \left|2^m \cdot X_{MIN}\right|_{2^n-1} \Big|_{2^n-1}$$

Proof: The proof is straightforward and is therefore omitted. ■

Property II implies that the multiplication of an SUT-RNS encoded number X by 2^m taken modulo 2^n-1 can be performed by: (a) rotating the bits of the two vectors Y and Z m positions to the left and (b) considering a correction term.

Let $\overline{Y}$ and $\overline{Z}$ denote the vectors that result from Y and Z by inverting the logical values of all their bits except the 0s of Z.

Property III: Considering the arithmetic values of X, $\overline{Y}$ and $\overline{Z}$, it holds that: $-X = \overline{Y} + \overline{Z} - X_{MAX}$

Proof: Consider a posibit $p \in \{0, +1\}$, a negabit $n \in \{-1, 0\}$ and a unibit $u \in \{-1, +1\}$. If we denote as $\overline{p}$, $\overline{n}$, and $\overline{u}$ the complements of the logical values of p, n, and u, respectively, it then holds that $-p = \overline{p} - 1$, $-n = \overline{n}$, and $-u = 2\overline{u} - 1$. Hence $-X$ is equal to the sum of $\overline{Y}$ and $\overline{Z}$ as long as we consider a correction of −1 for every posibit and unibit. The total correction that is then required is equal to:

$$-2^0 - 2^0 - 2^1 - \ldots - 2^{h-2} - \cdots - 2^{(k-1)h} - 2^{(k-1)h} - 2^{(k-1)h+1} - \ldots - 2^{kh-2} =$$
$$= -\sum_{i=1}^{k} 2^{ih-1} = -2^{h-1}\frac{2^{kh}-1}{2^h-1} = -2^{h-1}\cdot R = -X_{MAX} \quad \blacksquare$$

Property III introduces a way to negate an SUT-RNS encoded number.

Property IV: It holds that:

$$\left|-2^m \cdot X\right|_{2^n-1} = \left\| 2^m \cdot \overline{Y} \right|_{2^n-1} + \left|2^m \cdot \overline{Z}\right|_{2^n-1} - \left|2^m \cdot X_{MAX}\right|_{2^n-1} \Big|_{2^n-1}$$

Proof:The proof is straightforward and is therefore omitted. ■

Property IV implies that the computation of $\left|-2^m \cdot X_{SUT}\right|_{2^n-1}$ can be performed by: (a) rotating the bits of $\overline{Y}$ and $\overline{Z}$ m positions to the left, and (b) considering a correction term.

Example 2

Let X denote a 2-digit radix-2^4 SUT-RNS-encoded number (n=k×h=2×4=8) which has a value equal to $X = \left\langle \begin{matrix} 1011 & 0011 \\ 0 & 0 \end{matrix} \right\rangle = 2\times 16^1 - 6\times 16^0 = 26.$ According to Property I we can compute the value of X if we add Y=10110011=179, Z=0, and the correction X_{MIN} = −(2^3+1)(2^8+1)/(2^4−1)=−9×17, that is, X=179+0−153=26. According to Property II, $\left|2^2 \cdot X\right|_{2^8-1}$ can be computed by rotating each bit of Y and Z two positions to the left (11001110=206 and 00000000=0) and adding them along with a correction equal to $-4\times 9\times 17$. Hence, $\left|2^2 \cdot X\right|_{2^8-1} = \left|206 - 4\times 9\times 17\right|_{255} = 104$. According to Property III, in order to compute –X, we derive $\overline{Y}$ =01001100=76 and $\overline{Z}$ =00100010=34, add them and decrease the resulting binary value by a constant equal to X_{MAX}=2^3×17=136. Hence –X = 110−136 = –26. Finally, according to Property IV, $\left|-2^2 \cdot X\right|_{2^8-1}$ can be computed by rotating every bit of $\overline{Y}$ and $\overline{Z}$ two positions to the left (00110001=49 and 10001000=136) and decreasing their sum by the constant correction 4×2^3×17. Hence, $\left|-2^2 \cdot X\right|_{2^8-1} = \left|185 - 4\times 2^3\times 17\right|_{255} = \left|-104\right|_{255} = 151$. ■

In the following, we utilize the CRT and the New CRT-I theorem in order to derive reverse converters for the SUT-RNS encoding for two different moduli sets, that is, the 3-moduli set {2^n-1, 2^n, 2^n+1} which has a dynamic range approximately equal to 3n bits and the 4-moduli set {2^n-1, 2^n, 2^n+1, 2^{2n}+1} which has a dynamic range approximately equal to 5n bits, respectively. Efficient reverse converters for them for both the binary and the BSD encodings can be found in [18], [20] and [11]. Hence, a comparison against the proposed converters for the SUT-RNS encoding is feasible and will be given in the next section.

A. Reverse Converter for the {2^n-1, 2^n, 2^n+1} RNS

Consider an RNS with the three-moduli set {2^n-1, 2^n, 2^n+1} and a 3n-bit number $X \in [0, 2^{3n}-2^n)$. X can be uniquely represented in RNS by {x_1, x_2, x_3}, where $x_1 = \left|X\right|_{2^n-1}$,

$x_2 = |X|_{2^n}$, $x_3 = |X|_{2^n+1}$. Utilizing the CRT, X can be computed from $\{x_1, x_2, x_3\}$ as [17] [18]:

$X = x_2 + 2^n |T_1 + T_2 + T_3|_{2^{2n}-1}$, where

$T_1 = |(2^{n-1} + 2^{2n-1})x_1|_{2^{2n}-1}$, $T_2 = |-2^n x_2|_{2^{2n}-1}$,

$T_3 = |(2^{n-1} - 2^{2n-1})x_3|_{2^{2n}-1}$.

Hence, X can be reconstructed by concatenating the n bits of x_2 with the $2n$ bits derived by the sum of the T_i terms taken modulo 2^{2n}-1.

Assume that x_1, x_2, x_3 are SUT-RNS encoded according to Fig. 1. x_1, x_2, x_3 consist of posibits, negabits and unibits. Then, the n bits of the binary encoding of x_2 can be derived according to Property I. The derivation of each T_i term along with its corresponding correction is based on Properties I-IV. For example, when $n=k\times h=2\times4=8$, the binary vectors along with the correction term of each T_i term are shown in Fig. 2. Specifically, the correction for the term T_1 is derived by applying Property II, that is

$|(2^{n-1} + 2^{2n-1})X_{MIN}|_{2^{2n}-1} = |-(2^{n-1} + 2^{2n-1})(2^{h-1} + 1)R|_{2^{2n}-1}$.

Property IV is applied in order to derive the correction for term T_2, that is $|-2^n X_{MAX}|_{2^{2n}-1} = |-2^n 2^{h-1} R|_{2^{2n}-1}$. Accordingly, by employing again Properties II and IV, we derive the correction for term T_3. The closed forms that are given for the correction terms hold for every combination of n, k and h. We can unify all required corrections for the T_i terms into a single $2n$-bit correction term equal to:

$C_3 = |-(2^n + 2^{2n-1})R - (2^{n+1} + 1)2^{h-1}R|_{2^{2n}-1}$

The last row of Fig. 2 presents the binary value of the C_3 correction term when $n=k\times h=2\times4=8$. A Dadda adder tree composed of full and half adders can be used to compress the bits of the terms T_1, T_2, T_3 and C_3 of Fig. 2 in two $2n$-bit vectors that can then be driven to a parallel modulo 2^{2n}-1 adder in order to derive the $2n$ most significant bits of X. The constant bits of the C_3 term can be utilized for simplifying the implementation of the converter circuit.

B. Reverse Converter for the $\{2^n$-1, 2^n, 2^n+1, 2^{2n}+1$\}$ RNS

Consider an RNS with the four-moduli set $\{2^n$-1, 2^n, 2^n+1, 2^{2n}+1$\}$ and a $5n$-bit number $X \in [0, 2^{5n}-2^n)$. X can be uniquely represented in RNS by $\{x_1, x_2, x_3, x_4\}$, where $x_1 = |X|_{2^n-1}$, $x_2 = |X|_{2^n}$, $x_3 = |X|_{2^n+1}$, $x_4 = |X|_{2^{2n}+1}$. According to the New CRT-I theorem, X can be computed from $\{x_1, x_2, x_3, x_4\}$ as [11] [20]:

$X = x_2 + 2^n |T_1 + T_2 + T_3 + T_4|_{2^{4n}-1}$, where

$T_1 = |2^{n-2}(2^{2n} + 1)(2^n + 1)x_1|_{2^{4n}-1}$, $T_2 = |-2^{3n} x_2|_{2^{4n}-1}$,

$T_3 = |(2^{2n} + 1)(2^{n-2} - 2^{2n-2})x_3|_{2^{4n}-1}$ and

$T_4 = |(2^{3n-1} - 2^{n-1})x_4|_{2^{4n}-1}$.

Hence, X can be reconstructed by concatenating the n bits of x_2 with the $4n$ bits derived by the sum of the T_i terms taken modulo 2^{4n}-1.

Assume that x_1, x_2, x_3, x_4 are SUT-RNS encoded. Obviously, x_4 has twice SUT digits compared to the other three residues. The n bits of the binary encoding of x_2 can be derived according to Property I. The derivation of each T_i term can be based on Properties I-IV similarly to the previous moduli set case. For example, when $n=k\times h=2\times3=6$, the binary vectors along with the correction term of each T_i term are shown in Fig. 3. The closed forms that are given for the correction terms also hold for every combination of n, k and h. We can unify all required corrections for the T_i terms into a single $4n$-bit correction term equal to:

$$C_4 = |-(2^{4n-1} + 2^{3n} + 2^{3n-1} + 2^{2n-1} + 2^{n-1})2^{h-1}R - (2^{4n-2} + 2^{3n-1} + 2^{2n-2} + 2^{n-1})R - (2^{3n-1} + 2^{n-1})2^{h-1}R^* - 2^{3n-1}R^*|_{2^{4n}-1}$$

where $R^* = \frac{2^{2kh}-1}{2^h-1}$. The last row of Fig. 3 presents the binary value of the C_4 correction term when $n=k\times h=2\times3=6$. A Dadda adder tree can be used to compress the bits of the terms T_1-T_4 and C_4 of Fig. 3 in two $4n$-bit vectors. Then, a parallel modulo 2^{4n}-1 adder can derive the $4n$ most significant bits of X. The constant bits of the C_4 term can be utilized for simplifying the implementation of the converter circuit.

C. Sign Detection

An SUT-RNS encoded number can have a positive or a negative value (see Example 1). The CRT and the New CRT-I are valid for either positive or negative values of the x_is. However, an issue arises with the modulo 2^n channel.

TERM	*BINARY VECTOR*																*CORRECTION*
	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	
T_1	$x_{1,0}$	$X_{1,7}$	$x_{1,6}$	$x_{1,5}$ $x'_{1,4}$	$x_{1,4}$	$X_{1,3}$	$x_{1,2}$	$x_{1,1}$ $x'_{1,0}$	$x_{1,0}$	$X_{1,7}$	$x_{1,6}$	$x_{1,5}$ $x'_{1,4}$	$x_{1,4}$	$X_{1,3}$	$x_{1,2}$	$x_{1,1}$ $x'_{1,0}$	$\left\|-(2^{n-1} + 2^{2n-1})(2^{h-1} + 1)R\right\|_{2^{2n}-1}$
T_2	$\overline{X}_{2,7}$	$\overline{x}_{2,6}$	$\overline{x}_{2,5}$ $\overline{x}'_{2,4}$	$\overline{x}_{2,4}$	$\overline{X}_{2,3}$	$\overline{x}_{2,2}$	$\overline{x}_{2,1}$ $\overline{x}'_{2,0}$	$\overline{x}_{2,0}$									$\left\|-2^n 2^{h-1} R\right\|_{2^{2n}-1}$
T_3	$\overline{x}_{3,0}$	$X_{3,7}$	$x_{3,6}$	$x_{3,5}$ $x'_{3,4}$	$x_{3,4}$	$X_{3,3}$	$x_{3,2}$	$x_{3,1}$ $x'_{3,0}$	$x_{3,0}$	$\overline{X}_{3,7}$	$\overline{x}_{3,6}$	$\overline{x}_{3,5}$ $\overline{x}'_{3,4}$	$\overline{x}_{3,4}$	$\overline{X}_{3,3}$	$\overline{x}_{3,2}$	$\overline{x}_{3,1}$ $\overline{x}'_{3,0}$	$\left\|-2^{n-1}(2^{h-1} + 1)R - 2^{2n-1}2^{h-1}R\right\|_{2^{2n}-1}$
C_3	0	1	0	1	1	1	1	0	0	1	1	0	1	1	1	0	$\left\|-(2^n + 2^{2n-1})R - (2^{n+1} + 1)2^{h-1}R\right\|_{2^{2n}-1}$

Figure 2. Binary vectors and correction terms for the $\{2^n$-1, 2^n, 2^n +1$\}$ reverse converter when n=8 (k=2, h=4)

TERM	2^{23}	2^{22}	2^{21}	2^{20}	2^{19}	2^{18}	2^{17}	2^{16}	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}	CORRECTION
T_1	$x_{1,1}$	$x_{1,0}$	$X_{1,5}$	$x_{1,4}$	$x_{1,3}$	$X_{1,2}$	$x_{1,1}$	$x_{1,0}$	$X_{1,5}$	$x_{1,4}$	$x_{1,3}$	$X_{1,2}$	$x_{1,1}$	$x_{1,0}$	$X_{1,5}$	$x_{1,4}$	$x_{1,3}$	$X_{1,2}$	$x_{1,1}$	$x_{1,0}$	$X_{1,5}$	$x_{1,4}$	$x_{1,3}$	$X_{1,2}$	$\left\| -(2^{3n-2}+2^{n-2})R -(2^n+1)(2^{h-1}+1)R \right\|_{2^{4n}-1}$
	$x'_{1,0}$			$x'_{1,3}$			$x'_{1,0}$			$x'_{1,3}$			$x'_{1,0}$			$x'_{1,3}$			$x'_{1,0}$			$x'_{1,3}$			
T_2	$\overline{X}_{2,5}$	$\overline{x}_{2,4}$	$\overline{x}_{2,3}$	$\overline{X}_{2,2}$	$\overline{x}_{2,1}$	$\overline{x}_{2,0}$																			$\left\| -2^{3n}2^{h-1}R \right\|_{2^{4n}-1}$
		$\overline{x}'_{2,3}$			$\overline{x}'_{2,0}$																				
T_3	$\overline{x}_{3,1}$	$\overline{x}_{3,0}$	$X_{3,5}$	$x_{3,4}$	$x_{3,3}$	$X_{3,2}$	$x_{3,1}$	$x_{3,0}$	$\overline{X}_{3,5}$	$\overline{x}_{3,4}$	$\overline{x}_{3,3}$	$\overline{X}_{3,2}$	$\overline{x}_{3,1}$	$\overline{x}_{3,0}$	$X_{3,5}$	$x_{3,4}$	$x_{3,3}$	$X_{3,2}$	$x_{3,1}$	$x_{3,0}$	$\overline{X}_{3,5}$	$\overline{x}_{3,4}$	$\overline{x}_{3,3}$	$\overline{X}_{3,2}$	$\left\| -2^{2n-2}(2^{2n}+1)2^{h-1}R -2^{n-2}(2^{2n}+1)(2^{h-1}+1)R \right\|_{2^{4n}-1}$
	$\overline{x}'_{3,0}$			$x'_{3,3}$			$x'_{3,0}$			$\overline{x}'_{3,3}$			$\overline{x}'_{3,0}$			$x'_{3,3}$			$x'_{3,0}$			$\overline{x}'_{3,3}$			
T_4	$x_{4,6}$	$X_{4,5}$	$x_{4,4}$	$x_{4,3}$	$X_{4,2}$	$x_{4,1}$	$x_{4,0}$	$\overline{X}_{4,11}$	$\overline{x}_{4,10}$	$\overline{x}_{4,9}$	$\overline{X}_{4,8}$	$\overline{x}_{4,7}$	$\overline{x}_{4,6}$	$\overline{X}_{4,5}$	$\overline{x}_{4,4}$	$\overline{x}_{4,3}$	$\overline{X}_{4,2}$	$\overline{x}_{4,1}$	$\overline{x}_{4,0}$	$X_{4,11}$	$x_{4,10}$	$x_{4,9}$	$X_{4,8}$	$x_{4,7}$	$\left\| -2^{3n-1}(2^{h-1}+1)R^{*} -2^{n-1}2^{h-1}R^{*} \right\|_{2^{4n}-1}$
			$x'_{4,3}$			$x'_{4,0}$			$\overline{x}'_{4,9}$			$\overline{x}'_{4,6}$			$\overline{x}'_{4,3}$			$\overline{x}'_{4,0}$			$x'_{4,9}$			$x'_{4,6}$	
C_4	1	1	1	1	1	0	0	1	1	0	0	1	0	0	0	1	1	0	1	1	0	0	1	1	$\left\| -(2^{4n-1}+2^{3n}+2^{3n-1})2^{h-1}R -(2^{2n-1}+2^{n-1})2^{h-1}R -(2^{4n-2}+2^{3n-1}+2^{2n-2}+2^{n-1})R -(2^{3n-1}+2^{n-1})2^{h-1}R^{*}-2^{3n-1}R^{*} \right\|_{2^{4n}-1}$

Figure 3. Binary vectors and correction terms for the {2^n-1, 2^n, 2^n+1, 2^{2n}+1} reverse converter when n=6 (k=2, h=3).

Let us first consider the 3-moduli set {2^n-1, 2^n, 2^n+1}. According to the CRT theorem, $X = x_2 + 2^n \left| T_1 + T_2 + T_3 \right|_{2^{2n}-1}$.

Assume that x_2 is SUT-RNS encoded in the negative value range. Then, its corresponding positive value is equal to 2^n+x_2 and $\left|T_2\right|_{2^{2n}-1} = \left|-2^n(2^n + x_2)\right|_{2^{2n}-1} = \left|-2^n x_2 - 1\right|_{2^{2n}-1}$. Although the CRT theorem is still valid, in hardware, an adder is required in order to propagate the carry from the n least significant bits to the $2n$ most significant positions resulting in a significant overhead in the converter. To avoid this, we can simply detect the sign of x_2 and decrease the value of C_3 by one when x_2 is less than 0, while deriving the n least significant bits of the output of the reverse converter utilizing Property I.

The same holds for the 4-moduli set {2^n-1, 2^n, 2^n+1, 2^{2n}+1}. According to the New CRT-I theorem, $X = x_2 + 2^n \left| T_1 + T_2 + T_3 + T_4 \right|_{2^{4n}-1}$. Assume that x_2 is SUT-RNS encoded in the negative value range. Then, its corresponding positive value is equal to 2^n+x_2 and $\left|T_2\right|_{2^{4n}-1} = \left|-2^{3n}(2^n + x_2)\right|_{2^{4n}-1} = \left|-2^{3n} x_2 - 1\right|_{2^{4n}-1}$ .Hence, similarly to the 3-moduli set case, we can simply detect the sign of x_2 and decrease the value of C_4 by one when x_2 is less than 0, while deriving the n least significant bits of the output of the reverse converter utilizing Property I.

Consider an SUT-RNS encoded number X with k SUT digits (D_{k-1}...D_0) as the one given in Fig. 1. The sign of X can be determined by the sign of the first non-zero digit with the maximum weight. Zero is represented in an SUT digit as $\left\langle \begin{matrix} 01\cdots11 \\ 1 \end{matrix} \right\rangle$ or $\left\langle \begin{matrix} 10\cdots01 \\ 0 \end{matrix} \right\rangle$. If we denote as $ZD_i = \begin{cases} 0 & D_i \neq 0 \\ 1 & D_i = 0 \end{cases}$ the zero indication of the i-th SUT digit, $0 \le i < k$, then its value is given by the logic equation:

$$ZD_i = (\overline{X}_{(i+1)h-1} \wedge x_{(i+1)h-2} \wedge \cdots \wedge x_{ih+1} \wedge x_{ih} \wedge x'_{ih}) \vee (X_{(i+1)h-1} \wedge \overline{x}_{(i+1)h-2} \wedge \cdots \wedge \overline{x}_{ih+1} \wedge x_{ih} \wedge \overline{x}'_{ih})$$

where $\vee$ and $\wedge$ denote logical OR and AND, and can be realized with two (h+1)-input AND gates and one two-input OR gate.

All SUT digits with negabits equal to logic 0 represent non-positive numbers. Furthermore, when a negabit has a logic value equal to 1 then it represents a non-negative number. The only exception is the SUT digit $\left\langle \begin{matrix} 10\cdots00 \\ 0 \end{matrix} \right\rangle$ which, although has a negabit with a logic value equal to 1, represents a negative number (–1). If we denote as $SD_i = \begin{cases} 0 & D_i > 0 \\ 1 & D_i < 0 \end{cases}$ the sign indication of the i-th SUT digit, $0 \le i < k$, then its value is given by the logic equation $SD_i = (\overline{x}_{(i+1)h-2} \wedge \cdots \wedge \overline{x}_{ih+1} \wedge \overline{x}'_{ih}) \vee \overline{X}_{(i+1)h-1}$ and can be realized with a (h-1)-input AND gate and a two-input OR gate. Hence, the sign of the SUT-RNS encoded number X, $S_X = \begin{cases} 0 & X \ge 0 \\ 1 & X < 0 \end{cases}$, is given by the logic equation:

$$S_X = (\overline{ZD}_{k-1} \wedge SD_{k-1}) \vee (ZD_{k-1} \wedge \overline{ZD}_{k-2} \wedge SD_{k-2} \vee \cdots \vee (ZD_{k-1} \wedge \cdots \wedge ZD_1 \wedge \overline{ZD}_0 \wedge SD_0)$$

The corresponding circuit is graphically presented in Fig. 4.

Once the sign of x_2, $\overline{S}_{x_2}$, is derived, it can be utilized to decrease the value of C_3 or C_4 by one when x_2<0. Since, for every bit S it holds that $-S = \overline{S} - 1$, an efficient way to deal with S_{x_2} is to decrease the value of C_3 or C_4 by one and add $\overline{S}_{x_2}$ in the least significant bit position in the Dadda adder tree. Fig. 5 presents the complete architectures of the proposed reverse converters for the two moduli sets.

It should be noted that the computation of S_{x_2} is not expected to delay the addition of the required bits in the Dadda

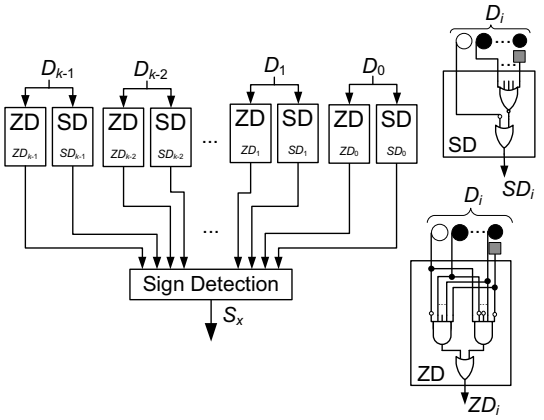

Figure 4. Block diagram of the sign detection circuit for X.

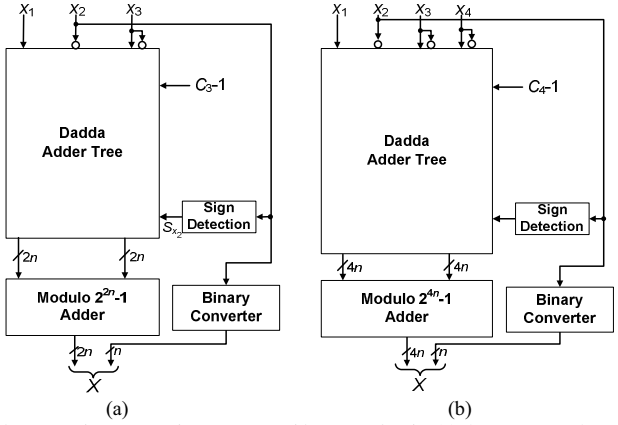

Figure 5. The proposed converter architectures for the (a) $\{2^n\text{-}1, 2^n, 2^n\text{+}1\}$ moduli set and (b) $\{2^n\text{-}1, 2^n, 2^n\text{+}1, 2^{2n}\text{+}1\}$ moduli set.

adder tree of the reverse converters. This is due to the following: (a) x_2 corresponds to the modulo 2^n channel and this channel is usually faster than the rest channels of the 2^n-1, 2^n+1, 2^{2n}+1 forms. It is therefore expected that the delay of the sign detection of x_2 will be hidden within the increased delay of the other channels. (b) The $\overline{S}_{x_2}$ bit can be always driven to the last level of the adder tree and therefore part of the delay for its derivation can be hidden within the delay of the full adders and half adders of the preceding levels of the adder tree.

IV. Evaluation and Comparisons

In this section we evaluate the proposed residue-to-binary converters for the SUT-RNS encoding and compare them against the corresponding converters that use the BSD encoding at their inputs, while comparison with the binary encoded converters is also given for completeness. For the two latter cases we assume the architectures proposed in [11], [18] and [20] which are considered the current state-of-the-art.

While the binary encoded and the proposed SUT-RNS encoded reverse converters make use of conventional arithmetic units, such as full adders and half adders, this is not the case for the BSD encoded ones that utilize dedicated building cells. Thus, comparisons based on the number of full adders, half adders and carry-propagate adders used cannot be given. An area-delay comparison of the various converters can be based on the unit gate model [23]. The unit gate model assumes that each monotonic gate counts as one gate equivalent for both area and delay, while two-input XOR and XNOR gates count for two gate equivalents for both area and delay.

Table II presents the total area and delay estimations of the proposed converters for the 3-moduli set $\{2^n\text{-}1, 2^n, 2^n\text{+}1\}$ according to the unit-gate model. We assume that h is greater than 2. The n least significant bits of the output can be derived according to Property I by a two-operand binary parallel-prefix adder along with k inverters while the $2n$ most significant bits of the output can be derived by a Dadda adder tree, a modulo 2^{2n}-1 parallel-prefix adder with single zero representation at the output and the sign detection circuit for x_2. The delay of the Dadda adder tree is equal to the delay of two full adders and one half adder (or a simplified full adder with one of its inputs connected to logic 1) since the maximum number of bits with the same weight that have to be added (see for example Fig. 2) is equal to 6 while one of them has a constant value. Considering the delay, we provide in Table II both an optimistic estimate that does not consider the delay of the x_2 sign detection, assuming that the x_2 input is available earlier than the other inputs, as well as a pessimistic one that assumes that all inputs are available at the same time. Table II also presents the total area and delay estimates of the binary and BSD-encoded reverse converters for the 3-moduli set $\{2^n\text{-}1, 2^n, 2^n\text{+}1\}$ assuming the architectures presented in [18] and [11] respectively. In all converters, the modulo 2^n parallel-prefix adders assume a Kogge-Stone architecture whereas all modulo 2^{2n}-1 parallel-prefix adders assume the architecture of [24] along with $2n$ NOR gates at the output for providing a single zero representation.

Similarly to the 3-moduli set case, Table III presents the total area and delay estimations of the proposed converters for the 4-moduli set $\{2^n\text{-}1, 2^n, 2^n\text{+}1, 2^{2n}\text{+}1\}$ according to the unit-gate model. We again assume that h is greater than 2. The n least significant bits of the output can be derived as in the previous case while the $4n$ most significant bits of the output can be derived by a Dadda adder tree, a modulo 2^{4n}-1 parallel-prefix adder with single zero representation at the output and

TABLE II. Unit-Gate Area and Delay Closed Forms for the $\{2^N\text{-}1, 2^N, 2^N\text{+}1\}$ Reverse Converter

Converter	*Delay (eq. gates)*	*Area (eq. gates)*
Proposed	MIN: $2\log n + 16$ MAX: $2\log n + 10 + \max\{\log[2k(k+1)(h+1)], 6\}$	$9n\log n + 33n + k/2(k+73) + 14$
[18]	$2\log n + 11$	$6n\log n + 31n$
[11]	$2\log n + 22$	$15n\log n + 113n + 24$

TABLE III. Unit-Gate Area and Delay Closed Forms for the $\{2^N\text{-}1, 2^N, 2^N\text{+}1, 2^{2N}\text{+}1\}$ Reverse Converter

Converter	*Delay (eq. gates)*	*Area (eq. gates)*
Proposed	MIN: $2\log n + 22$ MAX: $2\log n + 12 + \max\{\log[2k(k+1)(h+1)], 10\}$	$15n\log n + 95n + k/2(k+185) + 14$
[20]	$2\log n + 20$	$12n\log n + 111n + 39$
[11]	$2\log n + 24$	$27n\log n + 333n + 24$

the sign detection circuit for x_2. The delay of the Dadda adder tree is equal to the delay of three full adders and one half adder (or a simplified full adder) since, in this moduli set case, the maximum number of bits with the same weight that have to be added (see for example Fig. 3) is equal to 7 with one of them being constant. We again provide in Table III both an optimistic and a pessimistic estimate regarding the delay. Table III also presents the total area and delay estimates of the binary and BSD-encoded reverse converters for the 4-moduli set $\{2^n\text{-}1,\ 2^n,\ 2^n\text{+}1,\ 2^{2n}\text{+}1\}$ assuming the architectures presented in [20] and [11] respectively. We make the same assumptions as before for the modulo 2^n parallel-prefix adders and the modulo 2^{4n}-1 parallel-prefix adders.

Fig. 6 (Fig. 7) graphically compares the unit-gate area and delay of the three 3-moduli set (4-moduli set) reverse converters for values of n up to 64 covering dynamic ranges up to 320 bits. We consider four different values of h. We observe that the delay of the proposed converters lies between those of the reverse converters for the binary and the BSD encoding while the area of the proposed converters is close to the area required by the converters for the binary encoding and is significantly smaller than that required by the converters for the BSD encoding, especially for large values of n. Furthermore, for a given value of n, increasing h reduces the area and the delay of the corresponding proposed converters. This is justified by the fact that a large value of h leads to a small number of SUT digits. Finally, the differences between the area and the delay of the proposed converters and the binary-encoded ones are much smaller in the 4-moduli set case than in the 3-moduli set case.

The unit-gate model estimations for area and delay can only be considered as indicative. Furthermore, the unit-gate model does not consider power dissipation. To attain realistic results, reverse converters for three values of n were described in HDL. For the SUT-RNS case we assumed a value of h equal to four. The delay of the sign detection circuit for x_2 was taken into account in all cases. After simulating the resulting descriptions, the converters were synthesized and mapped to a 90 nm power-characterized CMOS implementation technology [25]. The Synopsys Design Compiler tool in the topographical mode was used for the synthesis and mapping of the converters. In this mode, for achieving faster timing closure, the tool performs floorplanning in parallel with synthesis and mapping and the design is annotated with wiring lengths and fan-out and parasitic capacitances coming directly from the floorplan of the design and not from a wire load model. We assumed that each converter's input and output is driven by the output of a D flip flop and drives the input of a D flip flop of the same implementation library, respectively. A typical corner (1.2V, 25^{o}C) was considered. Each converter was recursively optimized for speed using a bottom-up approach. A final area recovery step was then applied. For obtaining power data, we assumed an operating 400MHz frequency at each design and equiprobable inputs and measured the average power dissipation. Table IV presents the attained area, delay and power dissipation results. The results validate the conclusions that were previously reported regarding the area and the delay of the various converters. Furthermore, the results indicate that the average power dissipation of the proposed reverse converters is also significantly smaller than that of the converters for the BSD encoding.

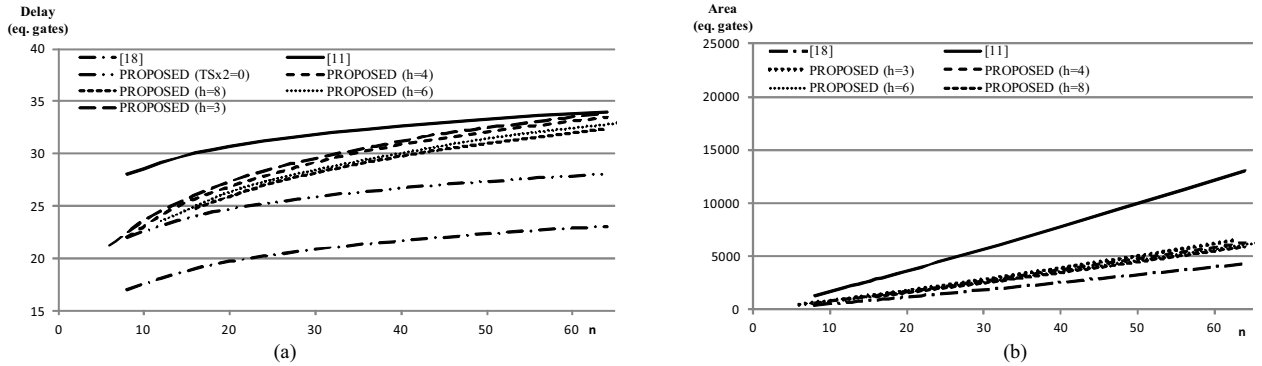

Figure 6. Unit-gate (a) delay and (b) area estimations for the 3-moduli set reverse converter.

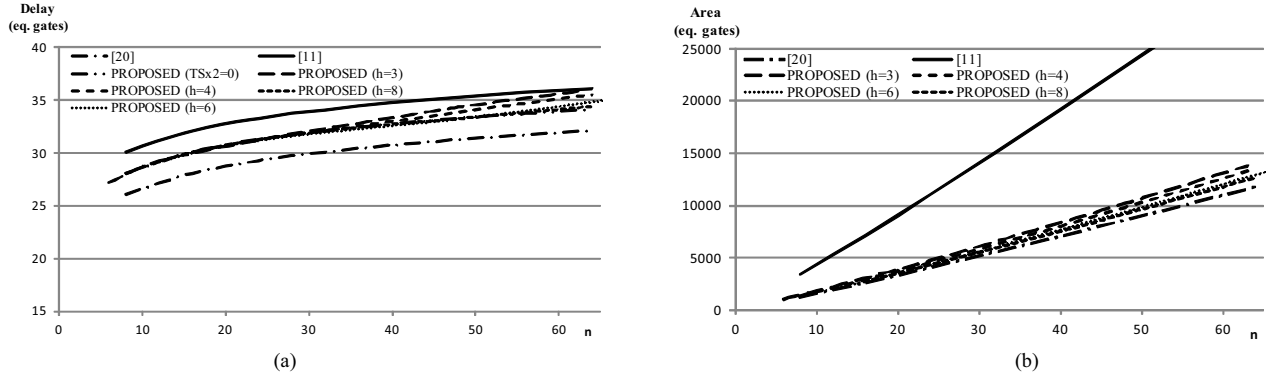

Figure 7. Unit-gate (a) delay and (b) area estimations for the 4-moduli set reverse converter.

TABLE IV. CMOS VLSI AREA, DELAY AND AVERAGE POWER DISSIPATION RESULTS

			Binary-RNS			BSD-RNS			SUT-RNS		
n	*k*	*h*	*Delay (ns)*	*Area (μm^2)*	*Power (mW)*	*Delay (ns)*	*Area (μm^2)*	*Power (mW)*	*Delay (ns)*	*Area (μm^2)*	*Power (mW)*
{2^n-1, 2^n, 2^n+1} reverse converters											
8	2	4	1.00	6365	1.65	1.78	19771	8.40	1.27	8810	2.52
16	4	4	1.15	14919	3.87	1.95	44123	18.75	1.42	20749	6.14
32	8	4	1.33	34231	8.88	2.05	99216	41.44	1.57	48123	14.34
{2^n-1, 2^n, 2^n+1, 2^{2n}+1} reverse converters											
8	2	4	1.49	17385	6.57	1.95	50641	23.21	1.60	19538	7.51
16	4	4	1.63	39159	14.84	2.09	110974	51.30	1.74	44145	16.99
32	8	4	1.82	87289	33.84	2.31	240411	113.60	1.89	99570	34.35

V. CONCLUSIONS

Redundant encodings can be used to reduce the carry propagation and the delay inside each channel of an RNS. SUT has been proposed as a redundant high-radix encoding for RNS that can significantly improve the area and routing costs compared to the BSD-encoded RNS with only a small increase in delay. In this paper, we have introduced residue-to-binary converters for the SUT-RNS encoding for two representative moduli sets, that is, for the $\{2^n\text{-}1, 2^n, 2^n\text{+}1\}$ and the $\{2^n\text{-}1, 2^n, 2^n\text{+}1, 2^{2n}\text{+}1\}$ sets. The evaluation of the proposed converters has shown that they are more efficient, both in terms of area and delay, than the corresponding converters that utilize the BSD encoding. Hence, the SUT-RNS encoding can be effectively utilized in RNS-based applications that require high dynamic ranges with strict area and delay constraints. The methodology that has been introduced for the proposed reverse converters can be easily applied to the design of reverse converters for other moduli sets as well such as those reported in [21].

ACKNOWLEDGMENT

This research was supported by the Caratheodory Programme of the University of Patras (D.178).

REFERENCES

[1] P. V. Ananda Mohan, *Residue number systems: algorithms and architectures*. Norwell, MA: Kluwer Academic, 2002.

[2] A. Omondi and B. Premkumar, *Residue number systems: theory and implementation*. London: Imperial College Press, 2007.

[3] R. Chaves and L. Sousa, "RDSP: A RISC DSP based on Residue Number System," In *Proc. 6th Euromicro Symp. Digital System Design*, 2003, pp. 128–135.

[4] G. C. Cardarilli, A. Nannarelli and A. Re, "Residue Number System for low-power DSP applications," In *Proc. 41st Asilomar Conf. Signals, Systems and Computers*, 2007, pp. 1412–1416.

[5] J. C. Bajard and L. Imbert, "A full RNS implementation of RSA," *IEEE Trans. Comput.*, vol. 53, no. 6, pp. 769–774, 2004.

[6] D. Schinianakis, A. Fournaris, H. Michail, A. Kakarountas and T. Stouraitis, "An RNS implementation of an F_p elliptic curve point multiplier," *IEEE Trans. Circuits Syst. I*, vol. 56 no. 6, pp. 1202–1213, 2009.

[7] A. Madhukumar and F. Chin, "Enhanced architecture for Residue Number System-based CDMA for high-rate data transmission," *IEEE Trans. Wireless Commun.*, vol. 3, no. 5, pp. 1363–1368, 2004.

[8] A. Avizienis, "Signed-digit number representation for fast parallel arithmetic," *IRE Trans. Electronic Comput.*, vol. EC-10, no.3, pp. 389–400, 1964

[9] S. Wei and K. Shimizu, "A novel residue arithmetic hardware algorithm using a Signed-Digit number representation," *IEICE Trans. Inform. Systems*, vol. E83-D, no. 12, pp. 2056–2064, 2000.

[10] A. Lindstrom, M. Nordseth, L. Bengtsson and A. Omondi, "Arithmetic circuits combining residue and signed-digit representations," In *Proc. Asia-Pacific Comput. Systems Archit. Conf.*, 2003, pp. 246–257.

[11] A. Persson and L. Bengtsson, "Forward and reverse converters and moduli set selection in signed-digit Residue Number Systems," *J. Signal Process. Syst.*, vol. 56, no. 1, pp. 1–15, 2009.

[12] G. Jaberipur, B. Parhami and M. Ghodsi, "Weighted two-valued digit-set encodings: unifying efficient hardware representation schemes for redundant number systems," *IEEE Trans. Circuits Syst. I*, vol. 52, no. 7, pp. 1348–1357, 2005.

[13] G. Jaberipur and B. Parhami, "Stored-transfer representations with weighted digit-set encodings for ultrahigh-speed arithmetic," *IET Circuits Devices Syst.*, vol. 1, no. 1, pp. 102–110, 2007.

[14] S. Timarchi and K. Navi, "Arithmetic circuits of redundant SUT-RNS," *IEEE Trans. Instrum. Meas.*, vol. 58, no. 9, pp. 2959-2968, 2009.

[15] S. Timarchi and M. Fazlali, "An efficient power-area-delay modulo 2^n-1 multiplier," In *Proc. 15th CSI Int. Symp. Comput. Archit. and Digital Systems*, 2010, pp. 157–160.

[16] E. Vassalos, D. Bakalis and H. T. Vergos, "SUT-RNS forward and reverse converters," In *Proc. IEEE Comput. Society Annual Symp. VLSI*, 2010, pp. 11–16.

[17] S. Andraos and H. Ahmed, "A new efficient nemoryless residue to binary converter," *IEEE Trans. Circuits Syst.*, vol. 35, no. 11, pp. 1441–1444, 1988.

[18] Z. Wang, G. Jullien and W. C. Miller, An improved residue-to-binary converter, *IEEE Trans. Circuits Syst. I*, vol. 47, no. 9, pp. 1437–1440, 2000.

[19] Y. Wang, X. Song, M. Aboulhamid and H. Shen, "Adder based residue to binary number converters for $(2^n\text{-}1, 2^n, 2^n\text{+}1)$," *IEEE Trans. Signal Process.*, vol. 50, no. 7, pp. 1772–1779, 2002.

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

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

[22] K. Gbolagade, R. Chaves, L. Sousa and S. Cotofana, "An improved RNS reverse converter for the $\{2^{2n+1}\text{-}1, 2^n, 2^n\text{-}1\}$ moduli set," In *Proc. Int. Symp. Circuits Syst.*, 2010, pp. 2103–2106.

[23] A. Tyagi, "A reduced-area scheme for carry-select adders," *IEEE Trans. Comput.*, vol. 42, no. 10, pp. 1163–1170, 1993.

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

[25] Synopsys Inc., SAED 90nm EDK, 2011, https://www.synopsys.com/apps/protected/university/members.html.