ON THE USE OF DOUBLE-LSB AND SIGNED-LSB ENCODINGS FOR RNS

Evangelos Vassalos[1], *Dimitris Bakalis*[1] *and Haridimos T. Vergos*[2]

[1] Electronics Laboratory, Physics Department
University of Patras
Patras, Greece
{vassalos@upatras.gr, bakalis@physics.upatras.gr}

[2] Computer Engineering and Informatics Department
University of Patras
Patras, Greece
{vergos@ceid.upatras.gr}

ABSTRACT

Double-LSB and signed-LSB have been recently proposed as efficient encodings for addition and multiplication in the modulo 2^n+1 and 2^n±1 channels respectively of a Residue Number System (RNS). In this paper we introduce reverse converters for two 4-moduli sets that adopt these encodings and analyze their efficiency. The theoretical and experimental results that are derived indicate that the area and delay costs of the reverse conversion in such RNSs are comparable to those that use the ordinary binary encoding and hence both the double-LSB and the signed-LSB encodings are well suited to RNSs.

Index Terms**—**Residue Number Systems, residue-to-binary conversion, reverse conversion, signed-LSB encoding, double-LSB encoding, modulo 2^n±1 arithmetic.

1. INTRODUCTION

The Residue Number System (RNS) is a carry-free number system frequently adopted in the implementation of digital signal processing (DSP) algorithms where mainly additions, subtractions and multiplications are required [1] [11].

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 $0 \leq A < M$, 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 smaller residues instead of large operands, while also being carried out in parallel in separate arithmetic units known as channels. Furthermore, no carry propagation exists between the channels of an RNS, thereby high-speed parallel arithmetic processing is possible.

RNSs built on moduli of the 2^n±1 forms have received significant attention, mainly due to the efficient architectures that have been proposed for the design of the respective arithmetic units [3] [10]. A moduli set that has been thoroughly examined in the past is the 3-moduli set $\{2^n-1, 2^n, 2^n+1\}$ which has a dynamic range approximately equal to $3n$ bits. However, in order to increase the parallelism and hence the speed of the arithmetic processing as well as the dynamic range, moduli sets with more than 3 moduli and dynamic ranges larger than $3n$ bits have been proposed in the last few years for contemporary digital signal processing applications [10].

In every RNS, the operands must be converted from their binary representations to their corresponding residues and vice versa. While forward conversion is usually a simple process, the reverse (residue-to-binary) conversion is more complex and must be efficiently realized so as to prevent performance degradation of the overall RNS. An extensive amount of research has been done on the design of efficient RNS reverse converters for various moduli sets [2] [9] [14-16].

Almost all RNSs use a modulo of the 2^n+1 form. Every operand in this modulo requires n+1 bits for its encoding while 2^n-1 combinations of them are not used. Leibowitz [8] introduced the diminished-one encoding for modulo 2^n+1 operands, in which each operand is represented decreased by one compared to its normal binary encoding. All arithmetic operations are separately considered for zero operands and results of arithmetic units. The diminished-one encoding has the advantage that the computations in the modulo 2^n+1 channel are restricted to n bits. However, special circuitry for zero operand handling is required and decrementers/incrementers have to be used in the forward/reverse conversion. In order to overcome these deficiencies, alternative encodings have been explored in the last few years for the modulo 2^n+1 channel along with efficient arithmetic units (adders/multipliers). The double-LSB [4] [6] and the signed-LSB [5] encodings are such examples.

In this paper we examine the reverse conversion in RNSs that utilize the double-LSB or signed-LSB encodings. Although some initial remarks have been made in [4] [5] on this subject, no thorough analysis has been reported. We consider two 4-moduli sets, (a) the $\{2^n-1, 2^n, 2^n+1, 2^{2n}+1\}$ set which has a dynamic range approximately equal to $5n$ bits and (b) the $\{2^n-1, 2^{2n}, 2^n+1, 2^{2n}+1\}$ set which has a dynamic range approximately equal to $6n$ bits, and we formally introduce efficient residue-to-binary converters for them according to the New CRT I theorem [14]. These moduli sets provide high dynamic ranges and are therefore suitable for demanding DSP applications. The converters evaluation reveals that their costs, in terms of area and delay, are comparable to those of the corresponding converters for the binary encoding. Hence, the double-LSB and signed-LSB encodings are well suited to RNSs.

The remaining of the paper is organized as follows. The next section briefly reviews the double-LSB and signed-LSB encodings and presents some properties that are useful in the reverse conversion. Section 3 introduces the converters for the two moduli sets while Section 4 compares the converters for the two above-mentioned encodings against the corresponding converters for the binary encoding.

2. PRELIMINARIES

In this section, we firstly review the double-LSB and the signed-LSB encodings and then present some useful properties of theirs.

Definition 1 *(double-LSB encoding)* [4]*:* A double-LSB encoded number X consists of a k-bit binary number accompanied by an extra bit in the least significant position.

In the following, we denote a double-LSB encoded number X as $X = \left\langle \begin{matrix} x_{k-1} \cdots x_0 \\ X_0 \end{matrix} \right\rangle$, where X_0 denotes the extra bit in the least significant position. The double-LSB encoding can cover all numbers in the $[0, 2^k]$ range and hence can be effectively used for encoding the operands in a modulo 2^k+1 channel of an RNS.

Definition 2 *(negabit):* A negabit is a two-valued digit with a lower value equal to –1 and an upper value equal to 0. A negabit uses bias encoding, that is, its lower value is encoded in binary with logical 0 and its upper value is encoded in binary with logical 1.

Definition 3 *(signed-LSB encoding)* [5]*:* A signed-LSB encoded number X consists of a k-bit binary number accompanied by a negabit in the least significant position.

In the following, we denote a signed-LSB encoded number X as $X = \left\langle \begin{matrix} x_{k-1} \cdots x_0 \\ X_0 \end{matrix} \right\rangle$, where X_0 in this case denotes the negabit in the least significant position. The signed-LSB encoding covers all numbers in the $[-1, 2^k-1]$ range and hence can be effectively used in a modulo 2^k+1 channel of an RNS by representing the 2^k value as -1, which is congruent to $\left|2^k\right|_{2^k+1}$. The signed-LSB encoding can also be adopted in the modulo 2^k-1 channel [5].

We now present some properties of the binary, the double-LSB and the signed-LSB encodings that are useful for the reverse conversion process.

Property 1*:* Assume a bit z. For every $k > i \geq 0$, it holds that $\left|z2^{i+k}\right|_{2^k-1} = \left|z2^i\right|_{2^k-1}$.

Proof:

$$\left|z2^{i+k}\right|_{2^k-1} = \left|z2^i(2^k-1) + z2^i\right|_{2^k-1} = \left|z2^i\right|_{2^k-1}$$

■

Property 1 implies that all bits with weights greater than or equal to 2^k can be considered as having weights less than 2^k.

Property 2*:* Every negabit can be treated equivalently as a bit provided that a correction term equal to –1 is considered.

Proof:

The proof is straightforward since –1 = 0–1 and 0 = 1–1.

■

Property 2 implies that the negabit which exists in the signed-LSB encoding can be treated as a bit.

Property 3*:* Assume a k-bit unsigned binary number $X=x_{k-1}\ldots x_0$. If we denote as $\overline{X}$ its bit-wise complement, then it holds that $-X = \overline{X} - (2^k-1)$.

Proof:

Since for every bit z it holds that $-z = \overline{z} - 1$, we have that:

$$-X = -\sum_{i=0}^{k-1} x_i 2^i = \sum_{i=0}^{k-1} (\overline{x}_i - 1) 2^i = \overline{X} - (2^k-1).$$

■

Property 4*:* Assume a k-bit double-LSB encoded number $X = \left\langle \begin{matrix} x_{k-1} \cdots x_0 \\ X_0 \end{matrix} \right\rangle$. If we denote as $\overline{X}$ the number that results from inverting the value of every bit of X, that is, $\overline{X} = \left\langle \begin{matrix} \overline{x}_{k-1} \cdots \overline{x}_0 \\ \overline{X}_0 \end{matrix} \right\rangle$, then it holds that $-X = \overline{X} - 2^k$.

Proof:

$$\begin{aligned} -X &= -\left(\sum_{i=0}^{k-1} x_i 2^i + X_0 2^0\right) = \sum_{i=0}^{k-1} (\overline{x}_i - 1) 2^i + \overline{X}_0 - 1 \\ &= \overline{X} - (2^k - 1) - 1 = \overline{X} - 2^k \end{aligned}$$

■

Property 5: Assume a k-bit signed-LSB encoded number $X = \left\langle \begin{matrix} x_{k-1} \cdots x_0 \\ X_0 \end{matrix} \right\rangle$. If we denote as $\overline{X}$ the number that results from inverting the value of every bit of X, that is, $\overline{X} = \left\langle \begin{matrix} \overline{x}_{k-1} \cdots \overline{x}_0 \\ \overline{X}_0 \end{matrix} \right\rangle$, then it holds that $-X = \overline{X} - (2^k - 1)$.

Proof:

According to Property 2, we can treat the negabit in the least significant position as a bit as long as we add –1. Hence:

$$-X = -(\sum_{i=0}^{k-1} x_i 2^i + X_0 - 1) = \sum_{i=0}^{k-1} (\overline{x}_i - 1) 2^i + \overline{X}_0$$
$$= \overline{X} - (2^k - 1)$$

■

Properties 3-5 imply that the negation of a binary encoded, a double-LSB encoded and a signed-LSB encoded number can be accomplished by inverting every bit of the number, treating the negabit in the latter case as bit and considering a constant correction term.

3. RESIDUE-TO-BINARY CONVERTERS

Consider an RNS with the p-moduli set $\{m_1, m_2, \ldots, m_p\}$ and a number $X \in [0, m_1 \times m_2 \times \ldots \times m_p)$. X can be uniquely represented in RNS as $\{x_1, x_2, \ldots, x_p\}$, where $x_1 = |X|_{m_1}$, $x_2 = |X|_{m_2}$, …, $x_p = |X|_{m_p}$.

The New CRT-I theorem [14] states that, given the residues $\{x_1, x_2, \ldots, x_p\}$, the binary number X can be computed as:

$$X = \left| \begin{matrix} x_1 + k_1 m_1 (x_2 - x_1) + k_2 m_1 m_2 (x_3 - x_2) \\ + \cdots + k_{p-1} m_1 m_2 \ldots m_{p-1} (x_p - x_{p-1}) \end{matrix} \right|_{m_1 m_2 \ldots m_p},$$

or equivalently [16] as:

$$X = x_1 + m_1 \left| \begin{matrix} k_1 (x_2 - x_1) + k_2 m_2 (x_3 - x_2) \\ + \cdots + k_{p-1} m_2 \ldots m_{p-1} (x_p - x_{p-1}) \end{matrix} \right|_{m_2 \ldots m_p}$$

where:

$$|k_1 m_1|_{m_2 \ldots m_p} = 1,\ |k_2 m_1 m_2|_{m_3 \ldots m_p} = 1,\ \ldots,$$
$$|k_{p-1} m_1 m_2 \ldots m_{p-1}|_{m_p} = 1 \cdot$$

In the following, we utilize the New CRT-I theorem in order to derive reverse converters for two different moduli sets, that is, the $\{2^n\text{-}1, 2^n, 2^n\text{+}1, 2^{2n}\text{+}1\}$ and the $\{2^n\text{-}1, 2^{2n}, 2^n\text{+}1, 2^{2n}\text{+}1\}$ 4-moduli sets which have high dynamic ranges of approximately $5n$ and $6n$ bits, respectively.

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

Consider the 4-moduli $\{2^n\text{-}1, 2^n, 2^n\text{+}1, 2^{2n}\text{+}1\}$ RNS and a $5n$-bit number $X \in [0, 2^{5n}\text{-}2^n)$. X can be uniquely represented in this RNS by $\{x_1, x_2, x_3, x_4\}$, where $x_1 = |X|_{2^n-1}$, $x_2 = |X|_{2^n}$, $x_3 = |X|_{2^n+1}$ and $x_4 = |X|_{2^{2n}+1}$. Obviously, x_4 is wider than the other three residues. According to the New CRT-I theorem, X can be computed from $\{x_1, x_2, x_3, x_4\}$ as [12]:

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

where

$$T_1 = |2^{n-2}(2^{2n}+1)(2^n+1)x_1|_{2^{4n}-1},$$
$$T_2 = |-2^{3n} x_2|_{2^{4n}-1},$$
$$T_3 = |(2^{2n}+1)(2^{n-2} - 2^{2n-2}) x_3|_{2^{4n}-1}, \text{ and}$$
$$T_4 = |(2^{3n-1} - 2^{n-1}) x_4|_{2^{4n}-1}.$$

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 two different cases: (a) an RNS that uses the double-LSB encoding for the modulo 2^n+1 and 2^{2n}+1 channels, that is, x_3 and x_4 are double-LSB encoded, and (b) an RNS that uses the signed-LSB encoding for the modulo 2^n-1, 2^n+1 and 2^{2n}+1 channels, that is, x_1, x_3 and x_4 are signed-LSB encoded. The derivation of each T_i term for each of the two different RNS cases results from the Properties 1-5 given in the previous section. For example, when n=4, the binary vector along with the correction term of each T_i term are shown in Fig. 1 (the shaded extra bits in the T_1 term exist only in the signed-LSB RNS case). Notation $x_{i,j}$ is used to indicate the j-th bit of residue x_i. The closed forms that are given for the correction terms hold for every value of n. We can unify all required corrections for the T_i terms into a single $4n$-bit correction term C equal to $C_{d-LSB} = |2^{3n-2} - 2^{n-2} - 1|_{2^{4n}-1}$ and $C_{s-LSB} = |-3 \cdot 2^{3n-2} - 2^{n-2} - 1|_{2^{4n}-1}$ in the cases of the double-LSB-based and signed-LSB-based RNS, respectively. When n=4, $C_{d-LSB} = 1019$ and $C_{s-LSB} = 62458$.

A Dadda reduction tree can be used to compress the bits of Fig. 1 in two $4n$-bit vectors. Then, a parallel modulo 2^{4n}-1 adder can derive the $4n$ most significant bits of X. A block diagram of the proposed architecture is shown in Fig. 3(a). The constant bits of the correction term can be utilized for simplifying the implementation of the reverse converter.

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

Consider an RNS with the 4-moduli set $\{2^n\text{-}1, 2^{2n}, 2^n\text{+}1, 2^{2n}\text{+}1\}$ and a $6n$-bit number $X \in [0, 2^{6n}\text{-}2^{2n})$. X can be uniquely represented in this RNS by $\{x_1, x_2, x_3, x_4\}$, where $x_1 = |X|_{2^n-1}$, $x_2 = |X|_{2^{2n}}$, $x_3 = |X|_{2^n+1}$ and $x_4 = |X|_{2^{2n}+1}$. Obviously, x_2 and x_4 are wider than the other two residues. According to the New CRT-I theorem, X can be computed from $\{x_1, x_2, x_3, x_4\}$ as [9]:

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

where

TERM	BINARY VECTOR																CORRECTION	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}	*double-LSB*	*signed-LSB*
T_1	$x_{1,1}$	$x_{1,0}$ $X_{1,0}$	$x_{1,3}$	$x_{1,2}$	$x_{1,1}$	$x_{1,0}$ $X_{1,0}$	$x_{1,3}$	$x_{1,2}$	$x_{1,1}$	$x_{1,0}$ $X_{1,0}$	$x_{1,3}$	$x_{1,2}$	$x_{1,1}$	$x_{1,0}$ $X_{1,0}$	$x_{1,3}$	$x_{1,2}$	0	$\left\|-2^{n-2}(2^{3n}+2^{2n}+2^{n}+1)\right\|_{2^{4n}-1}$
T_2	$\overline{x}_{2,3}$	$\overline{x}_{2,2}$	$\overline{x}_{2,1}$	$\overline{x}_{2,0}$													$\left\|2^{3n}-1\right\|_{2^{4n}-1}$	$\left\|2^{3n}-1\right\|_{2^{4n}-1}$
T_3	$\overline{x}_{3,1}$	$\overline{x}_{3,0}$ $\overline{X}_{3,0}$	$x_{3,3}$	$x_{3,2}$	$x_{3,1}$	$x_{3,0}$ $X_{3,0}$	$\overline{x}_{3,3}$	$\overline{x}_{3,2}$	$\overline{x}_{3,1}$	$\overline{x}_{3,0}$ $\overline{X}_{3,0}$	$x_{3,3}$	$x_{3,2}$	$x_{3,1}$	$x_{3,0}$ $X_{3,0}$	$\overline{x}_{3,3}$	$\overline{x}_{3,2}$	$\left\|-2^{3n-2}(2^{2n}+1)\right\|_{2^{4n}-1}$	$\left\|-2^{n-2}(2^{2n}+1)(2^{2n}-2^{n}+1)\right\|_{2^{4n}-1}$
T_4	$x_{4,4}$	$x_{4,3}$	$x_{4,2}$	$x_{4,1}$	$x_{4,0}$ $X_{4,0}$	$\overline{x}_{4,7}$	$\overline{x}_{4,6}$	$\overline{x}_{4,5}$	$\overline{x}_{4,4}$	$\overline{x}_{4,3}$	$\overline{x}_{4,2}$	$\overline{x}_{4,1}$	$\overline{x}_{4,0}$ $\overline{X}_{4,0}$	$x_{4,7}$	$x_{4,6}$	$x_{4,5}$	$\left\|-2^{3n-1}\right\|_{2^{4n}-1}$	$\left\|-2^{3n}+2^{n-1}\right\|_{2^{4n}-1}$
C	c_{15}	c_{14}	c_{13}	c_{12}	c_{11}	c_{10}	c_{9}	c_{8}	c_{7}	c_{6}	c_{5}	c_{4}	c_{3}	c_{2}	c_{1}	c_{0}	$\left\|2^{3n-2}-2^{n-2}-1\right\|_{2^{4n}-1}$	$\left\|-3\cdot 2^{3n-2}-2^{n-2}-1\right\|_{2^{4n}-1}$

Fig. 1. Binary vectors and correction terms for the {2^n-1, 2^n, 2^n+1, 2^{2n}+1} residue-to-binary converters.

TERM	BINARY VECTOR																CORRECTION	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}	*double-LSB*	*signed-LSB*
T_1	$x_{1,1}$	$x_{1,0}$ $X_{1,0}$	$x_{1,3}$	$x_{1,2}$	$x_{1,1}$	$x_{1,0}$ $X_{1,0}$	$x_{1,3}$	$x_{1,2}$	$x_{1,1}$	$x_{1,0}$ $X_{1,0}$	$x_{1,3}$	$x_{1,2}$	$x_{1,1}$	$x_{1,0}$ $X_{1,0}$	$x_{1,3}$	$x_{1,2}$	0	$\left\|-2^{n-2}(2^{3n}+2^{2n}+2^{n}+1)\right\|_{2^{4n}-1}$
T_2	$\overline{x}_{2,7}$	$\overline{x}_{2,6}$	$\overline{x}_{2,5}$	$\overline{x}_{2,4}$	$\overline{x}_{2,3}$	$\overline{x}_{2,2}$	$\overline{x}_{2,1}$	$\overline{x}_{2,0}$									$\left\|2^{2n}-1\right\|_{2^{4n}-1}$	$\left\|2^{2n}-1\right\|_{2^{4n}-1}$
T_3	$x_{3,1}$	$x_{3,0}$ $X_{3,0}$	$\overline{x}_{3,3}$	$\overline{x}_{3,2}$	$\overline{x}_{3,1}$	$\overline{x}_{3,0}$ $\overline{X}_{3,0}$	$x_{3,3}$	$x_{3,2}$	$x_{3,1}$	$x_{3,0}$ $X_{3,0}$	$\overline{x}_{3,3}$	$\overline{x}_{3,2}$	$\overline{x}_{3,1}$	$\overline{x}_{3,0}$ $\overline{X}_{3,0}$	$x_{3,3}$	$x_{3,2}$	$\left\|-2^{4n-2}-2^{2n-2}\right\|_{2^{4n}-1}$	$\left\|-2^{4n-1}-2^{2n-1}+2^{3n-2}+2^{n-2}\right\|_{2^{4n}-1}$
T_4	$\overline{x}_{4,0}$ $\overline{X}_{4,0}$	$x_{4,7}$	$x_{4,6}$	$x_{4,5}$	$x_{4,4}$	$x_{4,3}$	$x_{4,2}$	$x_{4,1}$	$x_{4,0}$ $X_{4,0}$	$\overline{x}_{4,7}$	$\overline{x}_{4,6}$	$\overline{x}_{4,5}$	$\overline{x}_{4,4}$	$\overline{x}_{4,3}$	$\overline{x}_{4,2}$	$\overline{x}_{4,1}$	$\left\|-2^{2n-1}\right\|_{2^{4n}-1}$	$\left\|-2^{2n}+2^{4n-1}\right\|_{2^{4n}-1}$
C	c_{15}	c_{14}	c_{13}	c_{12}	c_{11}	c_{10}	c_{9}	c_{8}	c_{7}	c_{6}	c_{5}	c_{4}	c_{3}	c_{2}	c_{1}	c_{0}	$\left\|-2^{4n-2}+2^{2n-2}-1\right\|_{2^{4n}-1}$	$\left\|-2^{4n-2}-3\cdot 2^{2n-2}-1\right\|_{2^{4n}-1}$

Fig. 2. Binary vectors and correction terms for the {2^n-1, 2^{2n}, 2^n+1, 2^{2n}+1} residue-to-binary converters.

$$T_1 = \left|2^{n-2}(2^{2n}+1)(2^{n}+1)x_1\right|_{2^{4n}-1},$$
$$T_2 = \left|-2^{2n}x_2\right|_{2^{4n}-1},$$
$$T_3 = \left|(2^{2n}+1)(2^{2n-2}-2^{n-2})x_3\right|_{2^{4n}-1}, \text{ and}$$
$$T_4 = \left|(2^{2n-1}-2^{4n-1})x_4\right|_{2^{4n}-1}.$$

Hence, X can be reconstructed by concatenating the $2n$ bits of x_2 with the $4n$ bits derived by the sum of the T_i terms taken modulo 2^{4n}-1. Assuming the same RNS cases as in the previous moduli set, the derivation of each T_i term for each of the two different RNS cases results from Properties 1-5. For example, when n=4, the binary vector along with the correction term of each T_i term are shown in Fig. 2 (the shaded extra bits in the T_1 term exist only in the signed-LSB RNS case). Unifying all required corrections for the T_i terms into a single $4n$-bit correction term C, we have for the two RNS cases that $C_{d-LSB} = \left|-2^{4n-2}+2^{2n-2}-1\right|_{2^{4n}-1}$ and $C_{s-LSB} = \left|-2^{4n-2}-3\cdot 2^{2n-2}-1\right|_{2^{4n}-1}$. When n=4, $C_{d-LSB} = 49214$ and $C_{s-LSB} = 48958$.

A Dadda reduction tree can be used in this moduli set as well to compress the bits of Fig. 2 in two $4n$-bit vectors. Then, a parallel modulo 2^{4n}-1 adder can derive the $4n$ most significant bits of X. A block diagram of the proposed architecture is shown in Fig. 3(b). The constant bits of the correction term can be also utilized for simplifying the implementation of the reverse converter.

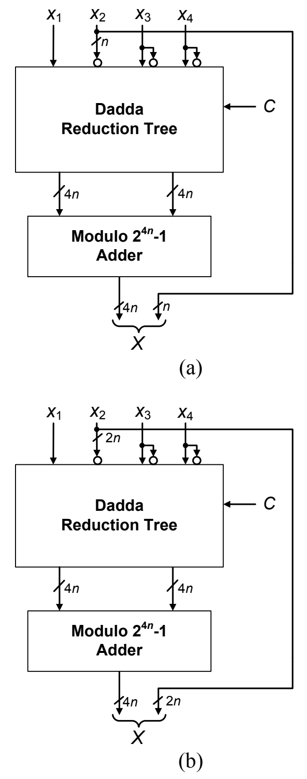

Fig. 3. The proposed converter architectures for the (a) {2^n-1, 2^n, 2^n+1, 2^{2n}+1} and (b) {2^n-1, 2^{2n}, 2^n+1, 2^{2n}+1} moduli sets.

4. EVALUATION AND COMPARISONS

At first we evaluate the area and the delay of the proposed residue-to-binary converters for the two moduli sets and the two different encodings under consideration. All converters utilize the same fast modulo 2^{4n}-1 adder which can follow any desired architecture as long as it provides a single representation of zero at its output.

Regarding the delay of the Dadda reduction tree, the double-LSB-based converters for both moduli sets are based on a vector matrix with a maximum height equal to 6 (see Figs. 1-2). Since one of the 6 bits has a constant value and the vector matrix does not have the same number of bits in every column, the Dadda reduction tree can be designed to have a delay equal to that of 1 half adder (HA) and 2 full adders (FAs). The vector matrix of the signed-LSB-based converters has a maximum height equal to 7. However, since one of the 7 bits has a constant value and its neighboring columns have fewer bits, the maximum height can be reduced at no cost to 6 and hence the delay of the Dadda reduction tree is equal to the delay of 3 FAs.

Regarding the area of the Dadda reduction tree, the vector matrix of the double-LSB-based converters for the $\{2^n\text{-}1, 2^n, 2^n+1, 2^{2n}+1\}$ moduli set consists of $13n$+6 bits and $4n$ constant bits while the vector matrix of the double-LSB-based converters for the $\{2^n\text{-}1, 2^{2n}, 2^n+1, 2^{2n}+1\}$ moduli set consists of $14n$+6 bits and $4n$ constant bits. Hence, in order to be reduced to the $8n$ bits that will be fed to the modulo 2^{4n}-1 adder, $5n$+6 FAs and $4n$ HAs or simplified FAs (that is, full adders with one input connected to logic 1) or $6n$+6 FAs and $4n$ HAs or simplified FAs are required for the two moduli set cases, respectively. The corresponding converters for the signed-LSB encoding have 4 more bits in each modulo case and therefore require 4 additional FAs.

Table I summarizes the area and delay requirements of the reverse converters, in equivalent gates, according to the unit-gate model [13]. We consider all three different encodings, that is, the binary, the double-LSB and the signed-LSB one. For completeness, besides the two 4-moduli sets of Section 3, we also consider the 3-moduli set $\{2^n\text{-}1, 2^n, 2^n+1\}$ and the converters that were reported in [4] and [5]. For the binary case, the architectures of [2] [9] and [15] are considered for the three moduli sets, which are the currently most efficient ones. In all converters, for the modulo 2^{2n}-1 or modulo 2^{4n}-1 adders we assume the parallel-prefix-based architecture of [7] along with $2n$ or $4n$ NOR gates at the output for achieving a single zero representation.

To also attain more realistic area and delay results, the above-mentioned converters were described in HDL for three values of n (4, 8 and 16). After validating the correct operation of the converters by simulation, each design was synthesized and mapped to a 90nm CMOS standard cell library, assuming typical process parameters. Table II presents the attained area and delay results.

Considering the examined 4-moduli sets, the delay of the proposed converters for the signed-LSB encoding is practically equal to that required by the binary-encoded converters while the delay of the double-LSB based converters is a little smaller. However, in the 3-moduli set case, the converters for the double-LSB and signed-LSB encodings are slower than the converters for the binary encoding, due to the high efficiency of the converters proposed in [15]. Furthermore, the area of the proposed converters in the $\{2^n\text{-}1, 2^n, 2^n+1, 2^{2n}+1\}$ 4-moduli set, is smaller than that required by the binary-encoded reverse converters. This is due to the fact that the binary-encoded converters have more bits in their vector matrices. In the other two moduli set cases, the area of the converters is practically the same.

Taking into consideration that the area and delay costs of the forward conversion of the double-LSB and signed-LSB encodings are also practically equal to the corresponding costs of the binary encoding [4] [5], we can conclude that both the double-LSB and the signed-LSB encodings are well suited to RNSs.

TABLE I
UNIT-GATE AREA AND DELAY ESTIMATIONS OF THE VARIOUS CONVERTERS (IN EQUIVALENT GATES)

Converter	*Binary*		*double-LSB*		*signed-LSB*	
	Delay	*Area*	*Delay*	*Area*	*Delay*	*Area*
3-*Moduli Set* $\{2^n–1, 2^n, 2^n+1\}$						
OR gates	1	n	–	–	–	–
Dadda reduction tree	4	$14n$	8	$13n+14$	10	$13n+28$
Modulo $2^{2n}–1$ adder	$2\log n+6$	$6n\log n+16n$	$2\log n+6$	$6n\log n+16n$	$2\log n+6$	$6n\log n+16n$
Total	$2\log n+11$	$6n\log n+31n$	$2\log n+14$	$6n\log n+29n+14$	$2\log n+16$	$6n\log n+29n+28$
4-*Moduli Set* $\{2^n–1, 2^n, 2^n+1, 2^{2n}+1\}$						
Dadda reduction tree	12	$67n+39$	10	$47n+42$	12	$47n+70$
Modulo $2^{4n}–1$ adder	$2\log n+8$	$12n\log n+44n$	$2\log n+8$	$12n\log n+44n$	$2\log n+8$	$12n\log n+44n$
Total	$2\log n+20$	$12n\log n+111n+39$	$2\log n+18$	$12n\log n+91n+42$	$2\log n+20$	$12n\log n+91n+70$
4-*Moduli Set* $\{2^n–1, 2^{2n}, 2^n+1, 2^{2n}+1\}$						
Dadda reduction tree	12	$60n+24$	10	$54n+42$	12	$54n+70$
Modulo $2^{4n}–1$ adder	$2\log n+8$	$12n\log n+44n$	$2\log n+8$	$12n\log n+44n$	$2\log n+8$	$12n\log n+44n$
Total	$2\log n+20$	$12n\log n+104n+24$	$2\log n+18$	$12n\log n+98n+42$	$2\log n+20$	$12n\log n+98n+70$

TABLE II
CMOS VLSI AREA AND DELAY RESULTS OF THE VARIOUS CONVERTERS

Converter	*Binary*		*double-LSB*		*signed-LSB*	
n	*Delay (ns)*	*Area (μm^2)*	*Delay (ns)*	*Area (μm^2)*	*Delay (ns)*	*Area (μm^2)*
			3-*Moduli Set* {2^n–1, 2^n, 2^n+1}			
4	0.37	2434	0.50	2441	0.55	2614
8	0.44	5584	0.57	5593	0.62	5631
16	0.52	12253	0.64	12118	0.69	12290
			4-*Moduli Set* {2^n–1, 2^n, 2^n+1, 2^{2n}+1}			
4	0.67	7640	0.63	7013	0.68	7443
8	0.74	15919	0.70	15272	0.73	15701
16	0.84	34821	0.78	33330	0.81	33483
			4-*Moduli Set* {2^n–1, 2^{2n}, 2^n+1, 2^{2n}+1}			
4	0.64	7302	0.66	7432	0.65	8057
8	0.71	15581	0.71	16154	0.72	16771
16	0.79	33247	0.80	34531	0.82	35233

5. CONCLUSIONS

In the last few years, two novel encodings, the double-LSB and the signed-LSB, along with efficient arithmetic units (adders and multipliers) for them have been proposed for the modulo 2^n+1 or 2^n±1 channels of an RNS. We have introduced efficient reverse converters for RNSs that utilize such encodings in their channels. The {2^n-1, 2^n, 2^n+1, 2^{2n}+1} and the {2^n-1, 2^{2n}, 2^n+1, 2^{2n}+1} 4-moduli sets have been considered. The analysis of their area and delay using both the unit-gate theoretical model as well as implementations in full static CMOS has shown that the converters costs in RNSs that use the double-LSB or the signed-LSB encodings are comparable to those in RNSs that use the binary encoding. Hence, both encodings can be efficiently utilized in RNSs.

ACKNOWLEDGEMENT

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*, Kluwer Academic, Norwell, MA, 2002.

[2] 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}," *IEEE Trans. Circuits and Systems I*, vol. 50, no. 10, pp. 1296-1303, October 2003.

[3] R. Conway and J. Nelson, "Improved RNS FIR Filter Architectures," *IEEE Trans. Circuits and Systems II*, vol. 51, no. 1, pp. 26-28, January 2004.

[4] G. Jaberipur, "A One-Step Modulo 2^n+1 Adder Based on Double-lsb Representation of Residues," *CSI Journal Computer Science and Engineering*, vol. 4, no. 2-4, pp. 10-16, 2006.

[5] G. Jaberipur and B. Parhami, "Unified Approach to the Design of Modulo-(2^n±1) Adders Based on Signed-LSB Representation of Residues," *Proc. of 19th IEEE Int. Symp. on Computer Arithmetic*, pp. 57-64, 2009.

[6] G. Jaberipur and H. Alavi, "A Modulo 2^n+1 Multiplier with Double-LSB Encoding of Residues," *Proc. of CSI Int. Symp. on Computer Architecture and Digital Systems*, pp. 147-150, 2010.

[7] L. Kalampoukas, D. Nikolos, C. Efstathiou, H. T. Vergos and J. Kalamatianos, "High-Speed Parallel-Prefix Modulo 2^n-1 Adders," *IEEE Trans. Computers*, vol. 49, no. 7, pp. 673-680, July 2000.

[8] L. Leibowitz, "A Simplified Binary Arithmetic for the Fermat Number Transform," *IEEE Trans. Acoustics, Speech and Signal Processing*, vol. ASSP-24, no. 5, pp. 356-359, October 1976.

[9] A. Molahosseini, K. Navi, C. Dadkhah, O. Kavelei and S. Timarchi, "Efficient Reverse Converter Designs for the New 4-Moduli Sets {2^n-1, 2^n, 2^n+1, 2^{2n+1}-1} and {2^n-1, 2^n+1, 2^{2n}, 2^{2n}+1} Based on New CRTs," *IEEE Trans. Circuits and Systems I*, vol. 57, no. 4, pp. 823-835, April 2010.

[10] K. Navi, A. Molahosseini and M. Esmaeildoust, "How to Teach Residue Number System to Computer Scientists and Engineers," *IEEE Trans. Education*, vol. 54, no.1, pp. 156-163, February 2011.

[11] A. Omondi and B. Premkumar, *Residue Number Systems: Theory and Implementation*, Imperial College Press, London, 2007.

[12] A. Persson and L. Bengtsson, "Forward and Reverse Converters and Moduli Set Selection in Signed-Digit Residue Number Systems," *Journal of Signal Processing Systems*, vol. 56, no. 1, pp. 1-15, July 2009.

[13] A. Tyagi, "A Reduced-Area Scheme for Carry-Select Adders," *IEEE Trans. Computer*, vol. 42, no. 10, pp. 1163-1170, October 1993.

[14] Y. Wang, "Residue-to-Binary Converters based on New Chinese Remainder Theorems," *IEEE Trans. Circuits and Systems II*, vol. 47, no. 3, pp. 197-205, March 2000.

[15] Z. Wang, G. A. Jullien and W. C. Miller, "An Improved Residue-to-Binary Converter," *IEEE Trans. Circuits and Systems I*, vol. 47, no. 9, pp. 1437-1440, September 2000.

[16] Y. Wang, X. Song, M. Aboulhamid and H. Shen, "Adder based Residue to Binary Number Converters for (2^n-1, 2^n, 2^n+1)," *IEEE Trans. Signal Processing*, vol. 50, no. 7, pp. 1772-1779, July 2002