SUT-RNS Forward and Reverse Converters

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

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

Abstract—**Stored Unibit Transfer (SUT) has been recently proposed as a redundant high-radix encoding for the channels of a Residue Number System (RNS) that can improve the efficiency of conventional redundant RNS. In this paper we propose modulo 2^n±1 forward and reverse converters for the SUT-RNS encoding. The proposed converters are based on parallel-prefix binary or modulo adders and are therefore very efficient.**

Keywords-Redundant arithmetic; modulo arithmetic; residue number system; modulo 2^n±1; stored-unibit-trasfer; forward converter; reverse converter

I. INTRODUCTION

Residue Number System (RNS) [1] [2] is a number system commonly adopted for speeding up computations in digital signal processing [3] [4] [5] [6], cryptography [7] and telecommunication applications [8] [9]. A non-positional RNS is defined by a set of L moduli, suppose $\{m_1, \ldots, m_L\}$ that are pair-wise relatively prime. Assume that $|A|_M$ denotes the modulo M residue of an integer A, that is, the least non-negative remainder of the division of A by M. A has a unique representation in the RNS, given by the set $\{a_1, \ldots, a_L\}$ of residues, where $a_i = |A|_{m_i}$. An operation $\otimes$ over an RNS is defined as $(z_1, \ldots, z_L) = (a_1, \ldots, a_L) \otimes (b_1, \ldots, b_L)$, where $z_i = |a_i \otimes b_i|_{m_i}$. The computation of z_i only depends on a_i, b_i, and m_i and each z_i is computed in parallel in a separate arithmetic unit often called a channel. Since each channel deals with narrow residues instead of wide numbers and since all channels operate in parallel, significant speedup over the binary may be achieved. RNSs built on 2^n±1 moduli have received significant attention due to the efficient arithmetic circuits that have been proposed for them. Any carry propagation in an RNS is restricted inside each channel.

Binary Signed-Digit (BSD) [10] has been proposed as a redundant, carry-free, number system where addition can be performed in constant time. The BSD number system represents each number with a set of digits in {-1, 0, +1}. Each digit requires two bits for its representation leading to a significant overhead in storage, processing and interconnection requirements. Hybrid redundant number systems, such as those with weighted two-valued digit set encodings [11] [12], have been proposed as an alternative that can limit the maximum length of carry propagation chains to any desired value and can lead to a wide representation range without the added costs associated with BSD. Furthermore, they can utilize conventional components such as full/half adders and can therefore produce highly efficient circuit implementations. Examples of such encodings are the Stored-Unibit Transfer (SUT) encoding [11] [12] and the Signed-LSB encoding [13].

Several attempts have been made to combine the parallel nature of RNS with the carry-free or carry-limited nature of redundant number systems. [14] [15] and [16] are among the most recent works that deal with the use of BSD inside each channel of an RNS in order to eliminate the intra-channel carry propagation. They propose efficient arithmetic circuits, such as adders and multipliers, for the modulo 2^n±1 cases. The authors of [13] propose modulo 2^n±1 adders based on the Signed-LSB encoding. In order to trade-off the area overhead of the BSD with the delay, [17] [18] propose the use of the SUT encoding for the modulo 2^n±1 RNS channels and present SUT-RNS addition, subtraction and multiplication circuits. However, to the best of our knowledge, no architecture has been reported so far for converting a binary modulo 2^n±1 number from/to its corresponding SUT-RNS encoding, making the arithmetic circuits proposed for SUT-RNS in [17] [18] inapplicable.

In this paper we fill this gap by presenting forward and reverse converters for modulo 2^n±1 SUT-RNS encoding. The proposed converters are based on parallel-prefix binary or modulo 2^n±1 adders and some extra simple logic and are very efficient.

The remaining of the paper is organized as follows. The next section presents an overview of the SUT-RNS encoding. Forward and reverse converters for modulo 2^n±1 SUT-RNS channels are given in Sections III and IV, respectively. Section V evaluates the proposed circuits and presents some experimental results. Section VI concludes the paper.

II. REDUNDANT HIGH-RADIX SUT-RNS

Every SUT-encoded number is composed of SUT digits. Each SUT digit consists of two-valued digits (twits) of three types: posibits {0, +1}, negabits {-1, 0}, and unibits {-1, +1}. A posibit has a lower value equal to 0 whereas a negabit and a unibit have a lower value equal to -1. Furthermore, a posibit and a negabit use a gap size equal to 1 whereas a unibit uses a gap size equal to 2 [11]. All three twits require one bit for their representation and use bias encoding, that is, their lower value is encoded in binary with logical 0 whereas their upper value is encoded in binary with logical 1. The dot notations, symbolic notations and binary encodings of the twits are given in Table I.

978-0-7695-4076-4/10 $26.00 © 2010 IEEE
DOI 10.1109/ISVLSI.2010.23

TABLE I. DOT NOTATION, SYMBOLIC NOTATION AND BINARY ENCODING 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)

SUT-RNS has been proposed as a redundant, high-radix, encoding for modulo 2^n±1 numbers [17] [18]. Every SUT-RNS-encoded number consists of k radix-2^h SUT digits (n=k×h), where each SUT digit consists of (h+1) twits, that is, (h-1) posibits, 1 negabit and 1 unibit. The negabit along with the posibits represent the radix-2^h main part [-2^{h-1}, $+2^{h-1}$-1] whereas the unibit represents the transfer part of the SUT digit. The symbolic and dot notation of a k-digit ($D_k \ldots D_1$) SUT-RNS number X are shown in Fig. 1. The maximum representable number is equal to $X_{MAX} = +2^{h-1}(2^{kh}-1)/(2^h-1)$ whereas the minimum representable number is equal to $X_{MIN} = (-2^{h-1}-1)(2^{kh}-1)/(2^h-1)$. For example, when n=6 (k=2 and h=3), representable numbers are between -45 and +36. A modulo 2^n+1 (2^n-1) number X, $X \in [0, 2^n]$ ($X \in [0, 2^n\text{-}1]$ assuming a double representation of zero), can be encoded in SUT-RNS by utilizing the positive range of the SUT-RNS encoding for some values of X and the negative range for the remaining values. Assuming the values of n, k and h of the previous example, number 7 can be encoded as +7 in both moduli cases whereas number 37 can be encoded as -28 in modulo 2^6+1 since $|37|_{65} = |-28|_{65}$ and can be encoded as -26 in modulo 2^6-1 since $|37|_{63} = |-26|_{63}$.

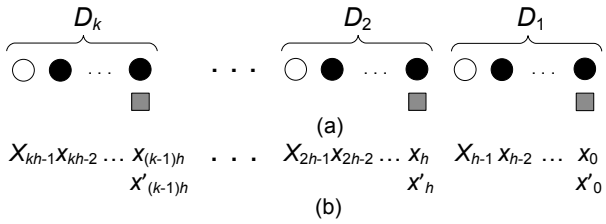

Figure 1. (a) Dot and (b) symbolic notation of a k-digit radix-2^h SUT-RNS number X

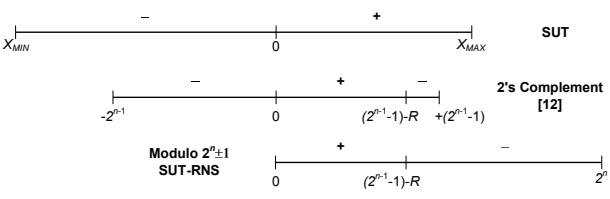

Figure 2. Positive and negative range of the SUT-RNS encoding

III. FORWARD CONVERTERS

In order to utilize the adder, subtractor and multiplier circuits that were proposed in [18] for SUT-RNS, one has to use forward converters to derive the SUT-RNS encodings of the input operands. In [18] no such circuits have been presented. [12] reported an algorithm for converting a signed two's complement number to an SUT representation. However this algorithm cannot be used as is in the SUT-RNS encoding since in this case the input is a modulo 2^n+1 or a modulo 2^n-1 unsigned number. We present in this section efficient forward converters of a modulo 2^n±1 number to the SUT-RNS encoding. We consider the two cases of moduli separately.

A. Modulo 2^n-1

Consider an n-bit modulo 2^n-1 number X=$x_{n-1} \ldots x_0$. The algorithm for forward conversion presented in [12] can correctly encode in SUT-RNS every value of X that lies in the positive range of the SUT-RNS encoding (see Fig. 2). However, for all values of X that are encoded in SUT-RNS in the negative range, a value decreased by one compared to the correct modulo 2^n-1 value is produced since modulo 2^n arithmetic is used instead and $|X-2^n|_{2^n-1} = |X-(2^n-1)-1|_{2^n-1} = |X-1|_{2^n-1}$. Hence, for all the SUT-RNS encoded values of X that lie in the negative range, we have to increase the corresponding value of X by one in order to get the correct modulo 2^n-1 value.

The following two-step algorithm is a modification of the forward conversion algorithm of [12] that deals with the above-mentioned problem and performs correct modulo 2^n-1 forward SUT-RNS conversion:

Step I: Compute $y=X+R+s=y'+s$, where y and $R=(2^{kh}-1)/(2^h-1)$ denote n-bit operands, $y'=X+R$, and s denotes a sign indication bit whose value is equal to 0 when the value of X lies in the positive range of the SUT-RNS encoding and is equal to 1 when the value of X lies in the negative range. According to Fig. 2, $s = \begin{cases} 0, & X+R < 2^{n-1} \\ 1, & X+R \geq 2^{n-1} \end{cases}$. Hence, sign s can be derived by the logical equation $s = y'_{n-1} \vee c_{n-1}$, where y'_{n-1} and c_{n-1} are the most significant bit and carry out of the (X+R) addition and $\vee$ denotes the logical OR operation. A straightforward solution for deriving the bits of y uses a binary adder for deriving y' and a controllable incrementer for incorporating s. However, those two operations can be efficiently merged in a parallel-prefix-based adder (see Fig. 3). The X and R operands can be driven to a parallel-prefix structure that in $\log_2 n$ levels can derive the n carries (c_{n-1}, …, c_0) of X+R. Then, s can be derived by an XOR and an OR gate, as $s = (hs_{n-1} \oplus c_{n-2}) \vee c_{n-1}$, where $hs_{n-1} = x_{n-1} \oplus R_{n-1}$ is the half-sum bit of the most significant bit position. An extra prefix level can then be used for adding the value of s and for producing a new set of carries (c'_{n-2}, …, c'_0). Finally, the n-bits of y can be derived by 2-input XOR gates. We have to note that since R is a constant that has a value equal to 1 in all bit positions with weights 2^{ih}, $0 \leq i < k$, and a value equal to 0 in all other bit positions, the parallel-prefix structure can be significantly simplified.

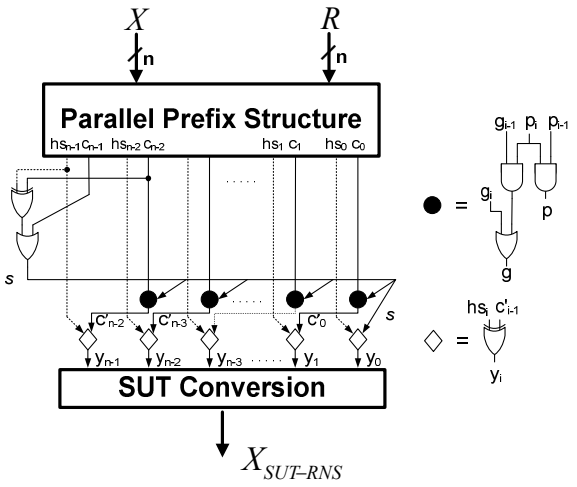

Figure 3. Proposed modulo 2^n-1 SUT-RNS forward converter

Step II: Use the following logic equations [12] to transform the bits of y to the corresponding SUT-RNS encoding of X, denoted as $X_{SUT\text{-}RNS}$, assuming that $y_{-1}=0$ and that $\wedge$ denotes the logical AND operation, while $\bar{z}$ denotes the logical NOT operation on bit z:

a. negabits : $X_{ih-1} = \bar{y}_{ih-1}$ for $1 \le i \le k$

b. posibit : $x_{ih} = y_{ih} \oplus y_{ih-1}$ for $0 \le i \le k$-1

c. posibits : $x_{ih-j} = y_{ih-j}$ for $1 \le i \le k$ and $2 \le j < h$

d. unibits : $x'_{ih} = y_{ih} \wedge y_{ih-1}$, for $0 \le i \le k$-1

Step II implies that the unibit x'_0 is always equal to 0. The complete circuit structure that realizes the above algorithm is given in Fig. 3 and is capable of dealing with both representations of zero in modulo 2^n-1 arithmetic, that is, 0 and 2^n-1.

Example 1: Suppose that $n=k\times h=2\times 4=8$ and $X=153$. Then $R=17$, $y'=170$, $s=1$ and $y=153+17+1=171=10101011_2$. According to Step II, $X_7 = \bar{y}_7 = 0$, $x_6 = y_6 = 0$, $x_5 = y_5 = 1$, $x_4 = y_4 \oplus y_3 = 1$, $x'_4 = y_4 \wedge y_3 = 0$, $X_3 = \bar{y}_3 = 0$, $x_2 = y_2 = 0$, $x_1 = y_1 = 1$, $x_0 = y_0 \oplus y_{-1} = 1$, and $x'_0 = y_0 \wedge y_{-1} = 0$, thus $X_{SUT-RNS} = \left\langle \begin{matrix} X_7 x_6 x_5 x_4 X_3 x_2 x_1 x_0 \\ x'_4 \quad x'_0 \end{matrix} \right\rangle = \left\langle \begin{matrix} 00110011 \\ 0 \quad 0 \end{matrix} \right\rangle =$ $(-6)\times 2^4+(-6)\times 2^0 = |-102|_{255} = 153$.

B. Modulo 2^n+1

Consider now a $(n+1)$-bit modulo 2^n+1 number $X=x_n x_{n-1}\ldots x_0 \in [0, 2^n]$. The forward converter of [12] could be used to encode X in SUT-RNS. However, for all values of X that lie in the negative range of the SUT-RNS encoding an increased by one value compared to the correct one would be produced since $\left|X-2^n\right|_{2^n+1} = \left|X-(2^n+1)+1\right|_{2^n+1} = \left|X+1\right|_{2^n+1}$. Hence, for all these values of X we have to decrease by one in order to get the correct modulo 2^n+1 SUT-RNS encoding.

The following algorithm is similar to the one presented previously for modulo 2^n-1 and performs modulo 2^n+1 SUT-RNS forward conversion.

Step I: Compute $y = X + R - s = X + (R-1) + \bar{s} = x_n 2^n + y' + \bar{s}$, where y and $R=(2^{kh}-1)/(2^h-1)$ denote n-bit operands, $y' = \left|X\right|_{2^n} + (R-1)$, and s denotes the sign indication bit. According to Fig. 2, $s = \begin{cases} 0, & X+R < 2^{n-1} \\ 1, & X+R \ge 2^{n-1} \end{cases} = \begin{cases} 0, & X+(R-1) < 2^{n-1}-1 \\ 1, & X+(R-1) \ge 2^{n-1}-1 \end{cases} = \begin{cases} 0, & X+(R-1) < 2^{n-1}-1 \\ 1, & X+(R-1) \ge 2^{n-1} \\ 1, & X+(R-1) = 2^{n-1}-1 \end{cases}$. The first two conditions can be identified by the logical equation $s=y'_{n-1} \vee c_{n-1} \vee x_n$, where y'_{n-1} and c_{n-1} are the most significant bit and carry out of the $\left|X\right|_{2^n} + (R-1)$ addition, respectively. Note that s also incorporates the most significant bit of X, x_n, in order to add the value $x_n 2^n$. Hence, the n least significant bits of X and $(R-1)$ are driven to an n-bit parallel-prefix structure. Then $\bar{s}$ is derived by an XOR and a NOR gate while an extra parallel prefix level is used to add the value of $\bar{s}$ and produce y.

Step II: Use the same logic equations as in the modulo 2^n-1 case to transform the bits of y to the corresponding SUT-RNS encoding $X_{SUT\text{-}RNS}$. The above algorithm produces the correct SUT-RNS encoding for all values of X except when $X+(R-1)=2^{n-1}-1$ (third condition). In this case we still have to subtract one. This can be easily achieved by deriving a signal m indicating the case where $X+(R-1)=2^{n-1}-1$ and correcting in this case the SUT-RNS encoding only of the least significant SUT digit. The logic equation for m is: $m = \overline{hs_{n-1}} \wedge hs_{n-2} \wedge \ldots \wedge hs_0$, where hs_{n-1}, …, hs_0 denote the half-sum bits of the parallel-prefix structure. m can be derived as fast as the carries of the parallel-prefix structure and hence it doesn't increase the delay of the forward converter. The twits of the least significant SUT digit are then derived by the following logic equations:

a. negabit : $X_{h-1} = m \oplus \bar{y}_{h-1}$

b. posibit : $x_1 = y_1$

c. posibits : $x_{h-j} = m \oplus y_{h-j}$, for $2 \le j \le h$, $j \ne h$-1

d. unibit : $x'_0 = m$

Hence, unibit x'_0 is equal to 1 only when $X+(R-1)=2^{n-1}-1$. The complete circuit structure that realizes the above algorithm is given in Fig. 4.

Example 2: Suppose that $n=k\times h=2\times 4=8$ and $X=153$. Then $R=17$, $\bar{s} = 0$, $m=0$ and $y=153+16+0=169=10101001_2$. According to Step II, $X_7 = \bar{y}_7 = 0$, $x_6 = y_6 = 0$, $x_5 = y_5 = 1$, $x_4 = y_4 \oplus y_3 = 1$, $x'_4 = y_4 \wedge y_3 = 0$, $X_3 = m \oplus \bar{y}_3 = 0$, $x_2 = m \oplus y_2 = 0$, $x_1 = y_1 = 0$, $x_0 = m \oplus y_0 = 1$, and $x'_0 = m = 0$, thus $X_{SUT-RNS} = \left\langle \begin{matrix} X_7 x_6 x_5 x_4 X_3 x_2 x_1 x_0 \\ x'_4 \quad x'_0 \end{matrix} \right\rangle = \left\langle \begin{matrix} 00110001 \\ 0 \quad 0 \end{matrix} \right\rangle =$ $(-6)\times 2^4+(-8)\times 2^0 = |-104|_{257} = 153$.

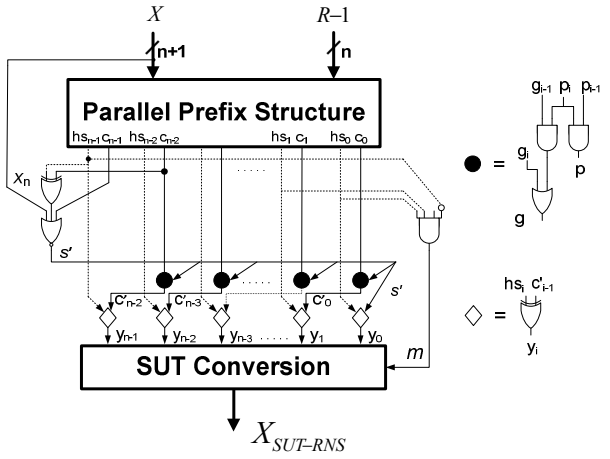

Figure 4. Proposed modulo 2^n+1 SUT-RNS forward converter

IV. Reverse Converters

We present in this section efficient reverse converters of an SUT-RNS encoded modulo 2^n±1 number to its corresponding binary encoding. We consider the two cases of moduli separately.

A. Modulo 2^n-1

In order to get the binary encoding X of an SUT-RNS encoded modulo 2^n-1 number $X_{SUT-RNS}$, we need to add in modulo 2^n-1 the following 4 n-bit vectors, as shown in dot notation in Fig. 5:

(a) the P=$0x_{kh-2}\ldots x_{(k-1)h}\ 0x_{(k-1)h-2}\ldots x_{(k-2)h}\ \ldots\ 0x_{h-2}\ldots x_0$ posibits vector.

(b) the negabits vector denoted as N. Vector N in modulo 2^n-1 arithmetic is equal to N=$X_{kh-1}1\ldots1\ X_{(k-1)h-1}1\ldots1\ \ldots\ X_{h-1}1\ldots1$. This is justified as follows: Due to the bias encoding, a negabit n_i with a weight equal to 2^i represents a value equal to $-2^i\overline{n}_i$.

Hence, $N=\left|\sum_{i=1}^{k}(-2^{ih-1}\overline{X}_{ih-1})\right|_{2^n-1}=\left|(2^n-1)-\sum_{i=1}^{k}(2^{ih-1}\overline{X}_{ih-1})\right|_{2^n-1}$.

Since, for every bit z it holds that $1-\overline{z}=z$, we conclude that N = $X_{kh-1}1\ldots1\ X_{(k-1)h-1}1\ldots1\ \ldots\ X_{h-1}1\ldots1$, that is, it consists of k h-bit patterns $X_{ih-1}1\ldots1$, $1\le i\le k$.

(c) the unibits vector denoted as U. Unibits can be treated as doublebits or equivalently as posibits in the next higher bit position, as long as we also consider a correction equal to -R. Hence, U =$0\ldots0x'_{(k-1)h}0\ 0\ldots0x'_{(k-2)h}\ 0\ \ldots\ 0\ldots0x'_0 0$.

(d) the constant correction vector $C^-=\left|-R\right|_{2^n-1}=\left|-(2^{kh}-1)/(2^h-1)\right|_{2^n-1}$ = $1\ldots10\ \ldots\ 1\ldots10$, which consists of k h-bit patterns $1\ldots10$.

Instead of using a 4-operand modulo 2^n-1 adder, we can merge the 4 vectors in two and use only a 2-operand modulo 2^n-1 adder. The posibits of P along with the negabits of N form an n-bit vector denoted as PN. PN is actually the main part of the SUT-RNS encoded number. The remaining constant bits of P and N along with the U vector and the constant C^- vector can be replaced by an n-bit vector $PNUC^-$ defined as $PNUC^-$ = $b_{n-1}\ldots b_0$, where $b_{(i+1)h-1}\ldots b_{ih}=x'_{ih}\overline{x'_{ih}}\ldots\overline{x'_{ih}}0$, $0\le i\le k$-1. Since x'_0=0, the h least significant bits are equal to $01\ldots10$, whereas the remaining $(k-1)h$ bits are repeating h-bit patterns of $x'_{ih}\overline{x'_{ih}}\ldots\overline{x'_{ih}}0$, with $1\le i\le k$-1. Hence, PN and $PNUC^-$ vectors are driven in a modulo 2^n-1 adder which derives the binary encoding of $X_{SUT-RNS}$, as shown in Fig. 6.

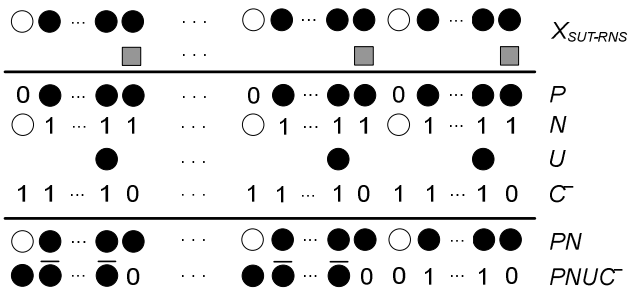

Figure 5. Vector formation for modulo 2^n-1 SUT-RNS reverse conversion

Example 3: Suppose that n=k×h=2×4=8 and X=104=7×2^4-8×2^0. The SUT-RNS encoding of X is equal to $X_{SUT-RNS}=\left\langle X_7x_6x_5x_4X_3x_2x_1x_0\right\rangle=\left\langle\begin{matrix}1110\ 0001\\ x'_4\quad x'_0\end{matrix}\right\rangle=\left\langle\begin{matrix}1110\ 0001\\ 1\quad\ 0\end{matrix}\right\rangle$. According to the previous discussion PN=11100001 and $PNUC^-$=10000110. A modulo 255 adder (which is equivalent to an end-around-carry binary adder) with PN and $PNUC^-$ as inputs produces the value 01101000 at the output which is equal to 104.

B. Modulo 2^n+1

A similar approach can be used in the modulo 2^n+1 case as well. In order to get the binary encoding X of an SUT-RNS encoded modulo 2^n+1 number $X_{SUT-RNS}$, we need to add in modulo 2^n+1 arithmetic 4 n-bit vectors (see Fig. 7): vectors P, N and U for the posibits, negabits and unibits, respectively, which are equal to those in the modulo 2^n-1 case and a constant correction vector C^+ which in the case of modulo 2^n+1 is equal to $C^+=\left|2-R\right|_{2^n+1}$. The constant term 2 is justified by the

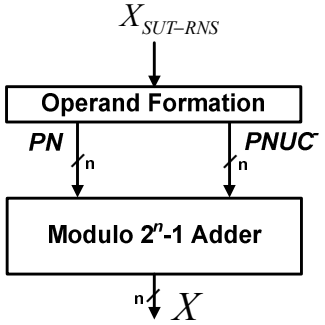

Figure 6. Proposed modulo 2^n-1 SUT-RNS reverse converter

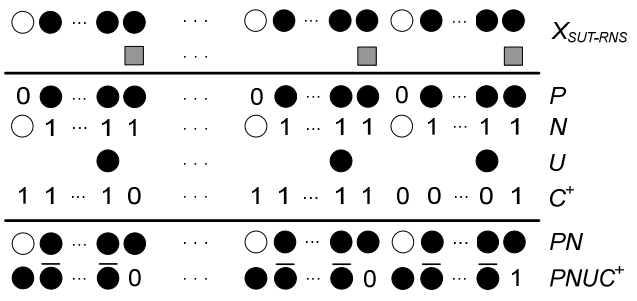

Figure 7. Vector formation for modulo 2^n+1 SUT-RNS reverse conversion

fact that the negabits vector in modulo 2^n-1 and the corresponding negabits vector in modulo 2^n+1 always differ by 2.

The 4 vectors can be merged in two: the *PN* vector which is the main part of the SUT-RNS encoded number *X* and the $PNUC^+$ vector which depends on the transfer part of $X_{SUT\text{-}RNS}$ and is equal to $PNUC^+ = b_{n-1} \ldots b_0$, where $b_{h-1} \ldots b_0 = x'_0 \overline{x'_0} \ldots \overline{x'_0} 1$ and $b_{(i+1)h-1} \ldots b_{ih} = x'_{ih} \overline{x'_{ih}} \ldots \overline{x'_{ih}} 0$, $1 \le i \le k-1$. The two *n*-bit vectors *PN* and $PNUC^+$ are then driven to an enhanced diminished-one modulo 2^n+1 adder [19] that produces the (*n*+1)-bit binary encoding of $X_{SUT\text{-}RNS}$, as shown in Fig. 8. We have to note that in $PNUC^+$ a constant correction term equal to -1 is also taken into account since a diminished-one adder always increases the sum of its two input operands by one.

Example 4: Let *n*, *h*, *k* and *X* have the same values as in the previous example. The SUT-RNS encoding of *X* is equal to $X_{SUT-RNS} = \left\langle \begin{matrix} X_7 x_6 x_5 x_4 X_3 x_2 x_1 x_0 \\ x'_4 \quad\quad x'_0 \end{matrix} \right\rangle = \left\langle \begin{matrix} 1110\,0001 \\ 1 \quad\quad 0 \end{matrix} \right\rangle$. Then *PN* = 11100001 and $PNUC^+$ = 10000111. An enhanced diminished-one modulo 257 adder sums *PN* and $PNUC^+$ and produces the value 001101000 at its output which is equal to 104.

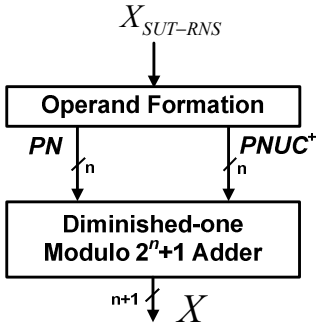

Figure 8. Proposed modulo 2^n+1 SUT-RNS reverse converter

V. Evaluation and Experimental Results

In this section we at first evaluate the forward and reverse converters that were proposed in Sections III and IV, respectively, and then, we present some experimental results based on CMOS VLSI circuit implementations.

The SUT-RNS forward converters for both modulo 2^n-1 and 2^n+1 are based on an *n*-bit parallel-prefix structure. A few gates are used to derive the sign bit *s* which is then added with an extra prefix level and a level of 2-input XOR gates. Finally, Step II of forward conversion requires some extra 2-input XOR gates in parallel. Since the parallel-prefix structure has a logarithmic delay and all remaining subcircuits have small constant delays, we conclude that the forward converters are very efficient in delay. The SUT-RNS reverse converters are also very efficient since they are based on modulo 2^n-1 or diminished-one modulo 2^n+1 adders whose input operands are formed at a minimum delay of an inverter. Furthermore, both the parallel-prefix structure in the forward converters and the modulo adders in the reverse converters can be designed using any desirable architecture.

We described in HDL forward and reverse converters for both moduli cases and for several values of *n*, *k* and *h*. In the forward converters case we considered a Kogge-Stone [20] parallel prefix structure. In the reverse converters case we considered modulo 2^n-1 and diminished-one modulo 2^n+1 adders that follow the architectures of [21] and [22], respectively. After validating the correct operation of the HDL descriptions via simulation, we synthesized them in a power-characterized 90nm CMOS technology, using a standard delay optimization script, and derived estimates for area, delay and average power dissipation. The attained results, given in Table II, indicate that the proposed converters are very fast and require small area and power dissipation. Since we are not aware of any other work on forward and reverse modulo $2^n \pm 1$ SUT-RNS converters, no comparison with other proposals is possible.

TABLE II. Experimental Results

			Forward Converters			Reverse Converters		
n	***k***	***h***	***Area (um^2)***	***Delay (ns)***	***Power (mW)***	***Area (um^2)***	***Delay (ns)***	***Power (mW)***
Modulo 2^n-1								
8	2	4	864	0.235	0.25	974	0.172	0.37
12	4	3	1462	0.270	0.50	1932	0.212	0.69
12	3	4	1457	0.269	0.43	1813	0.215	0.67
16	4	4	2218	0.278	0.62	2592	0.211	0.97
20	5	4	2554	0.313	0.70	3757	0.259	1.32
20	4	5	2485	0.307	0.61	3809	0.252	1.34
Modulo 2^n+1								
8	2	4	1191	0.250	0.29	1533	0.170	0.62
12	4	3	1559	0.285	0.47	2601	0.209	0.99
12	3	4	1467	0.287	0.37	2842	0.209	1.14
16	4	4	2345	0.292	0.59	3426	0.217	1.36
20	5	4	2777	0.331	0.73	5129	0.252	2.04
20	4	5	2706	0.320	0.61	5386	0.251	2.17

VI. CONCLUSIONS

Redundant number systems can be used to reduce the carry propagation inside each channel in an RNS. SUT has been proposed as a redundant high-radix encoding for RNS that can improve the efficiency of BSD-based RNS since it can utilize conventional arithmetic components such as full/half adders. We have presented in this paper, for the first time in the open literature, efficient forward and reverse converters for the SUT-RNS encoding for the two most commonly used moduli cases, that is, modulo $2^n \pm 1$. The forward converters are based on parallel-prefix binary adders and simple logic gates whereas the reverse converters are based on parallel-prefix modulo $2^n \pm 1$ adders and simple logic gates.

The incorporation of the proposed converters in the various already proposed forward and reverse converters from/to binary to/from RNS is currently under investigation. This will enable to convert binary representations to SUT-RNS and vice versa without using a residue representation as an intermediate step.

REFERENCES

[1] P. V. Ananda Mohan, Residue Number Systems: Algorithms and Architectures, Netherlands: Kluwer Academic Publishers, 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," 6th Euromicro Symp. on Digital System Design , pp. 128–135, 2003.

[4] P. G. Fernandez and A. Lloris, "RNS-based implementation of 8x8 point 2D-DCT over field-programmable devices," Electronics Letters, vol. 39, no. 1, pp. 21-23, 2003.

[5] Y. Liu and E. Lai, "Moduli set selection and cost estimation for RNS-based FIR filter and filter bank design," Design Automation for Embedded Systems, vol. 9, no. 2, pp. 123-139, 2004.

[6] G. Cardarilli, A. Nannarelli and M. Re, "Residue number system for low-power DSP applications," Asilomar Conference on Signals, Systems and Computers, pp. 1412-1416, 2007.

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

[8] U. Meyer-Baese, A. Garcia and F. Taylor, "Implementation of a communications channelizer using FPGAs and RNS arithmetic," VLSI Signal Processing, vol. 28, no. 1-2, pp. 115-128, 2001.

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

[10] A. Avizienis, "Signed-digit representation for fast parallel arithmetic," IRE Trans. on Electronic Computers, vol. EC-10, pp. 389-400, 1961.

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

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

[13] G. Jaberipur and B. Parhami, "Unified approach to the design of modulo-$(2^n \pm 1)$ adders based on signed-LSB representation of residues," IEEE Int. Symposium on Computer Arithmetic, pp. 57-64, 2009.

[14] A. Lindstrom, M. Nordseth, L. Bengtsson, and A. Omondi, "Arithmetic circuits combining residue and signed-digit representations," 8th Asia-Pacific Computer Systems Architecture Conf., vol. 2823, pp. 246-257, 2003.

[15] S. Wei, "A new residue adder with redundant binary number representation," 6th Int. IEEE North-East Workshop on Circuits and Systems, pp. 157-160, 2008.

[16] A. Persson and L. Bengtsson, "Forward and reverse converters and moduli set selection in signed-digit residue number systems," J. Signal Processing Systems, vol. 56, no.1, pp. 1-15, 2009.

[17] S. Timarchi and K. Navi, "Efficient class of redundant Residue Number System," IEEE Int. Symposium on Intelligent Signal Processing, pp. 475-780, 2007.

[18] S. Timarchi and K. Navi, "Arithmetic circuits of redundant SUT-RNS," IEEE Trans. on Instrumentation and Measurement, vol. 58, no. 9, pp. 2959-2968, 2009.

[19] H. T. Vergos, D. Bakalis and C. Efstathiou, "Fast modulo 2^n+1 multi-operand adders and residue generators," Integration, the VLSI Journal, vol. 43, no. 1, pp. 42-48, 2010.

[20] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," IEEE Trans. on Computers, vol. 22, no. 8, pp. 786-792, 1973.

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

[22] H. T. Vergos, C. Efstathiou and D. Nikolos, "Diminished-one modulo 2^n+1 adder design," IEEE Trans. on Computers, vol. 51, no. 12, pp. 1389-1399, 2002.