

# Contributions to residue-number-system/signed-digit arithmetic

A. Lindström, M. Nordseth, L. Bengtsson and A. Omondi

*Abstract:* Improvements to existing signed-digit modulo adders and multipliers are proposed, and new converters for the residue signed-digit number system are described for the moduli  $\{2^n - 1, 2^n, 2^n + 1\}$ . An existing highly efficient signed-digit cell is used to develop new modulo adders, which combined with a new scheme to half the number of partial products results in improved signed-digit modulo multipliers. The novel converters facilitate integration with normal binary systems.

*Introduction:* The use of the signed-digit (SD) system has recently been suggested as a way of eliminating the remaining carry propagation in residue number system (RNS) arithmetic [1]. Utilizing the carry-free properties of SD with RNS arithmetic also helps simplify the implementation of crucial RNS arithmetic operations. This letter makes two main contributions: the improvement of circuits that have previously been proposed for residue signed-digit (RNS+SD) arithmetic and the design of novel converters for the RNS+SD representation.

*SD modulo addition:* The most common rules for adding two SD integers,  $X$  and  $Y$ , are shown in Table 1, in which  $c_i$  denotes the

**Table 1** Rules for adding SD numbers

$x_i y_i$	00	01	0 $\bar{1}$	0 $\bar{1}$	11	$\bar{1}\bar{1}$	$\bar{1}\bar{1}$
$x_{i+1} y_{i-1}$	-	at neither is $\bar{1}$	at least one is $\bar{1}$	at neither is $\bar{1}$	at least one is $\bar{1}$	-	-
$c_{i+1}$	0	1	0	0	$\bar{1}$	1	$\bar{1}$
$u_i$	0	$\bar{1}$	1	$\bar{1}$	1	0	0

carry and  $u_i$  the interim sum. These rules eliminate any carry propagation when the final sum is calculated,  $s_i = c_i + u_i, s_i \in \{\bar{1}, 0, 1\}$ .

The binary coding we shall use is  $y = [y^- y^+] \in \{\{00\}, \{01\}, \{10\}\}$  to represent 0, 1 and  $\bar{1}$  respectively. This section shows that speed and area gains can be made by taking an existing highly efficient SD addition cell and adapting it for residue operation. The SD adder cell described in [2] is used as foundation for our modulo adder. The logic for this SD adder cell is

$$\begin{aligned}
 p_i &= \overline{x_{i-1}^- + y_{i-1}^-} \\
 xid_i &= \overline{x_i^- + x_i^+}, \quad yid_i = \overline{y_i^- + y_i^+} \\
 z_{i+1} &= \overline{xid_i \cdot yid_i \cdot p_i + xid_i \cdot yid_i \cdot p_i + x_i^+ \cdot yid_i + y_i^+ \cdot xid_i} \\
 t_i &= \overline{xid_i \cdot yid_i \cdot p_i + xid_i \cdot yid_i \cdot p_i +} \\
 &\quad + \overline{xid_i \cdot yid_i \cdot p_i + xid_i \cdot yid_i \cdot p_i} \\
 s_i^- &= t_i \cdot z_i, \quad s_i^+ = \overline{t_i \cdot z_i}.
 \end{aligned} \tag{1}$$

Modulo  $2^n + 1$  addition (where  $n$  is the word length) can then be performed by including an end-around-carry in the computations in (1):

$$\begin{aligned}
 p_0 &= \overline{x_{n-1}^+ + y_{n-1}^+} \\
 z_0 &= \overline{xid_{n-1} \cdot yid_{n-1} \cdot p_{n-1} + xid_{n-1} \cdot} \\
 &\quad \cdot \overline{yid_{n-1} \cdot p_{n-1} + x_{n-1}^- \cdot yid_{n-1} + y_{n-1}^- \cdot xid_{n-1}}
 \end{aligned} \tag{2}$$

For modulo  $2^n - 1$  addition, the end-around operation instead becomes  $p_0 = \overline{x_{n-1}^+ + y_{n-1}^+}$  and  $z_0 = z_n$ . This results in modulo

adders that are just a few gates larger than the normal SD adder but which are equally fast. Note that compared to a binary  $2^n + 1$  modulo adder, the SD counterpart has the advantage that it only needs  $n$  digits to represent the result. If the result is greater than  $2^n - 1$ , then the output will be given in the negative range instead, i.e. if the operation  $x \bmod m, \langle x \rangle_m$ , has the positive solution  $a$ , then

$a - m$  is the valid result in the negative range. In fact, the negative range will be used whenever it is suitable.

The hardware logic needed for an SD adder can be dramatically simplified if one or both of the inputs to the adder are in binary representation. The rules in Table 1 can be simplified in both cases by limiting it to just the possible input cases, thus eliminating the influence of  $x_{i-1}$  and  $y_{i-1}$ . The translation of these reduced rules into logic is straightforward and can be used directly for modulo  $2^n - 1$  addition by using a simple end-around-carry,  $c_0 = c_n$ . However,

this is not feasible in the case of modulo  $2^n + 1$  addition where instead the equations in (1) and (2) can be simplified for binary or mixed input.

*SD modulo multiplication:* The partial products in modulo multiplication can be summed in a tree of SD modulo adders as described in [1], which uses  $n$  partial products. The number of partial products can however be reduced to  $n/2$ , where  $n$  is the operand length, by considering two digits of  $y$  simultaneously as follows

$$\begin{aligned}
 \langle x \cdot y \rangle_m &= \left\langle \sum_{i=0}^{\frac{n}{2}-1} \langle 2^{2i} (y_{2i} \cdot x + y_{2i+1} \cdot 2x) \rangle_m \right\rangle_m \\
 &= \left\langle \sum_{i=0}^{\frac{n}{2}-1} \langle 2^{2i} \cdot rp_i \rangle_m \right\rangle_m = \left\langle \sum_{i=0}^{\frac{n}{2}-1} \langle ppi_i \rangle_m \right\rangle_m
 \end{aligned}$$

where  $rp_i \in \{-3x, -2x, -x, 0, x, 2x, 3x\}$ . These values can be generated using only one SD modulo adder (to generate  $3x = 2x + x$ ). Shift, rotation (see 3) and negation do not require any additional gates. The partial products,  $ppi_i = \langle 2^{2i} \cdot rp_i \rangle_m$ , can

be generated by rotation if the modulo is  $2^n - 1$ ,  $2^n$  or  $2^n + 1$  by using the rules in (3). These rotations can be accomplished by wiring connections appropriately.

$$\begin{aligned}
 \langle 2^a \cdot y \rangle_{2^p-1} &= [y_{p-1-a} \dots y_0 y_{p-1} \dots y_{p-a}] \\
 \langle 2^a \cdot y \rangle_{2^p} &= [y_{p-1-a} \dots y_0 0_{p-1} \dots 0_{p-a}] \\
 \langle 2^a \cdot y \rangle_{2^p+1} &= [y_{p-1-a} \dots y_0 (-y_{p-1}) \dots (-y_{p-a})] \\
 p \geq a, y &= [y_{p-1} \dots y_0]
 \end{aligned} \tag{3}$$

Compared to an addition tree that sums all  $n$  partial products without recoding, the top level consisting of  $n/2$  adders is replaced by one adder and some logic. The suggested reduction of partial products results in reduced hardware complexity with a negligible slow-down of the circuit.

*RNS+SD forward converter:* A new forward converter for the  $\{2^n - 1, 2^n, 2^n + 1\}$  moduli set has been developed. The converter structure is based on the binary-to-RNS converter described in [3] but adapted for RNS+SD. The use of SD arithmetic is very convenient because, in contrast with a conventional binary implementation, the use of simplified SD residue adders reduces the complexity of the converter.

For a  $3n$ -bit binary integer,  $Y$ , in the range  $[0, M)$ , where  $M = (2^n - 1)2^n(2^n + 1)$ , the vectors  $k_0 = y_{n-1} \dots y_0$ ,  $k_1 = y_{2n-1} \dots y_n$ ,  $k_2 = y_{3n-1} \dots y_{2n}$  are created. With these vectors the following operations

$$\begin{aligned}
r_1 &= \langle \langle k_2 + k_0 \rangle_{2^{n-1}} + k_1 \rangle_{2^{n-1}} \\
r_2 &= k_0 \\
r_3 &= \langle \langle k_2 + k_0 \rangle_{2^{n+1}} - k_1 \rangle_{2^{n+1}}
\end{aligned} \tag{4}$$

are then performed to obtain the residues  $\{r_1, r_2, r_3\}$ , which is the RNS representation of  $Y$ .

The modulo operations in (4) become trivial with the SD system and are no more complex than normal SD additions. Also, since the input,  $Y$ , is a binary integer, the modulo SD adders can be greatly simplified, as they will only require a few gates per bit. This results in a very small and fast (constant time) converter.

*RNS+SD backward converter:* A backward converter for the same moduli set has also been developed. The backward converter is based on a modified version of the New Chinese Remainder Theorems (CRT-1 as stated in [4]). CRT-1 states that

$$X = x_2 + 2^n \cdot \langle x_2 - x_3 + 2^{n-1}(2^n + 1)(x_1 - 2x_2 + x_3) \rangle_{2^{2n-1}} \tag{5}$$

where  $\{x_1, x_2, x_3\}$  is the RNS representation of  $X$ . Equation (5) can be written as

$$X = x_2 + 2^n \cdot \langle x_2 - 2^{2n}x_2 - x_3 - 2^n x_2 + 2^{n-1}(2^n + 1)(x_1 + x_3) \rangle_{2^{2n-1}} \tag{6}$$

and by using the first rotation rule in (3) with  $y = [0..0x_2]$  and  $a = p = 2n$ , Equation (6) simplifies to

$$X = x_2 + 2^n \cdot \langle -x_3 - 2^n x_2 + 2^{n-1}(2^n + 1)(x_1 + x_3) \rangle_{2^{2n-1}}$$

Now, by using the fact that  $x_3$  has  $n$ -digits in the SD system the above expression can be written as

$$X = x_2 + 2^n \cdot \langle a + b + c \rangle_{2^{2n-1}}$$

where  $a$ ,  $b$  and  $c$  have the digit length  $2n$  and are formed by concatenation and rotation:

$$\begin{aligned}
a &= [(-x_2)(-x_3)] \\
b &= [x_{1_0}x_{1_{n-1}} \dots x_{1_0}x_{1_{n-1}}x_{1_1}] \\
c &= [x_{3_0}x_{3_{n-1}} \dots x_{3_0}x_{3_{n-1}}x_{3_1}]
\end{aligned}$$

Forming  $a$ ,  $b$  and  $c$  can be done without using any gates but adding  $a$ ,  $b$  and  $c$  will however require two  $2n$ -digit SD modulo adders. The last step, i.e. the computation of  $x_2 + 2^n \cdot \langle \dots \rangle_{2^{2n-1}}$ , may also be done with concatenation and will thus not require any additional gates.

The upper part of  $X$ ,  $\langle a + b + c \rangle_{2^{2n-1}}$ , can be converted to binary by a  $2n$ -bit carry-look-ahead adder (CLA) and the lower part,  $x_2$ , can be converted by a  $n$ -bit CLA. Any carry-outs from the CLAs can safely be ignored as the two parts are bound by their moduli.

$X$  will be in the range  $(-M, M)$ ,  $M = (2^n - 1)2^n(2^n + 1)$ , because the SD residue adders also use the negative range. If it is necessary for the output from the converter to be in the range  $[0, M)$ , then the positive value of  $\langle X \rangle_M$  is the correct output and can be calculated by adding  $M$  to  $X$  if negative. This is done by adding 1's to the upper  $2n$ -bits of  $X$ . Addition with only 1's is very simple in the SD system as the rules in Table 1 can in that case be trivially simplified.

When converting to binary, it is simplest to just first add 1's to the upper  $2n$ -bits of  $X$ , before converting, and then convert the result and the original  $2n$ -bits to binary using two CLAs operating in parallel. The correct binary value can then be selected by examining the sign bit of the converted original  $2n$ -bits.

*Conclusions:* Improved RNS+SD arithmetic circuits have been presented and new RNS+SD converters for the moduli set  $\{2^n - 1, 2^n, 2^n + 1\}$  have been described. The developed RNS+SD converters make it possible to integrate RNS+SD circuits with conventional binary systems.

## References

- 1 WEI, S., SHIMIZU, K.: 'A novel residue arithmetic hardware algorithm using a signed-digit number representation', IEICE Trans. Inf. and Syst., Dec. 2000, Vol. E83-D, No. 12, pp. 2056-2064.
- 2 TAKAGI, N., YASUURA, H., and YAJIMA, S.: 'High-speed VLSI Multiplication Algorithm with a Redundant Binary Addition Tree', IEEE Trans. Computers, Sept. 1985, Vol. C-34, No. 9, pp. 789-796.
- 3 VINNAKOTA, B., and BAPESWARA RAO, V.V.: 'Fast Conversion Techniques for Binary-Residue Number systems', IEEE Trans. Circuits and Systems I, Dec. 1994, Vol. 41, No. 12, pp. 927-929.
- 4 WANG, Y.: 'Residue-to-Binary Converters Based on New Chinese Remainder Theorems', IEEE Trans. Circuits and Systems II, Mar. 2000, Vol. 47, No. 3, pp. 197-206.

## Authors' affiliations:

Anders Lindström [e8call@etek.chalmers.se], Michael Nordseth [e8mn@etek.chalmers.se], and Lars Bengtsson [labe@ce.chalmers.se] (Department of Computer Engineering, Chalmers University of Technology, SE-412 96 Gothenburg, Sweden)

Amos Omondi [amos.omondi@flinders.edu.au] (School of Informatics and Engineering, Flinders University, Bedford Park, SA 5042, Australia)