Regular Paper

Performance Evaluation of Signed-Digit Architecture for Weighted-to-Residue and Residue-to-Weighted Number Converters with Moduli Set $(2^n - 1, 2^n, 2^n + 1)$

Shuangching Chen† and Shugang Wei†

High-speed signed-digit (SD) architectures for weighted-to-residue (WTOR) and residue-to-weighted (RTOW) number conversions with the moduli set $(2^n, 2^n - 1, 2^n + 1)$ are proposed. The complexity of the conversion has been greatly reduced by using compact forms for the multiplicative inverse and the properties of modular arithmetic. The simple relationships of WTOR and RTOW number conversions result in simpler hardware requirements for the converters. The primary advantages of our method are that our conversions use the modulo m signed-digit adder (MSDA) only and that the constructions are simple. We also investigate the modular arithmetic between binary and SD number representation using circuit designs and a simulation, and the results show the importance of SD architectures for WTOR and RTOW number conversions. Compared to other converters, our methods are fast and the execution times are independent of the word length. We also propose a high-speed method for converting an SD number to a binary number representation.

1. Introduction

In a binary number system, the carry propagation during addition limits the speed of arithmetic operation. A well-known property of the residue number system (RNS) is that the ith residue digit of the sum, difference, and product of the operands is exclusively dependent on the ith digits [1),2)]. Digital signal processing hardware based on RNS is currently considered important for high-speed and low-cost hardware realization [3)]. However, residue number arithmetic cannot be used for some applications, because RNS does not have weights in the residue digits. Moreover, fast conversions between the residue numbers and weighted numbers are required. Residue number arithmetic with the moduli set $(2^n - 1, 2^n, 2^n + 1)$ has been widely used, because residue addition can be implemented using a binary adder [4),5)]. When the ordinary binary number system is used in residue arithmetic, the binary arithmetic speed which is bounded by the size of the numbers is the prncipal cause of timing problems.

The redundant binary number representation was first introduced by Avizienis in 1961 [6)]. A radix-two signed-digit (SD) number system [6),7)], which has a set of {-1,0,1} and no need for a separate sign digit, is well known to offer advantages in an arithmetic circuit implementation without the carry propagation. Such number representation systems possess sufficient redundancy to allow for the annihilation of carry or borrow chains, and hence result in fast propagation-free addition and subtraction. A novel residue arithmetic hardware algorithm using a radix-two SD number representation has been proposed to implement the modulo m multiplication for the symmetric RNS [8),9)]. The key to increasing the computation speed of such multiplication is to perform fast modular additions with a large modulus. The advantages of fast multiplication and addition have been clearly demonstrated, although output decoding is difficult when comparing the magnitudes of two residue numbers.

The weighted-to-residue (WTOR) and residue-to-weighted (RTOW) number conversions are the crucial steps for any successful RNS application. For general moduli sets, residue-to-binary number conversions are based on the Chinese remainder theorem (CRT) or mixed radix conversion (MRC). However, a direct implementation of CRT is unprofitable because it is based on the modulo M operation where M is large. A number of residue-to-binary number converters have been proposed [10)~12)] for the moduli set $(2^n - 1, 2^n, 2^n + 1)$. The algorithm of Andraos et al. [10)] uses multiplicative inverses to simplify the CRT and that of Piestrak [11)] is an improved algorithm. Arithmetic modulo $2^{2n} - 1$ is required, and the implementation units of carry save adders (CSAs) and

† Department of Computer Science, Gunma University

multiplexers improve its speed. The approach of Conway et al. [12] uses several coefficients to establish the dynamic range but the CRT equation becomes more complex.

The mixed radix conversion technique is another method of RNS-to-binary number conversion. The classical Szabo-Tanaka [1] procedure consists of a MRC and an additional adjustment that is slow and requires n computational steps to get the result, where n is the number of residue system moduli. Several MRC approaches have been developed [13)~16)]. The method of Huang [14] uses look-up tables only at the first stage of the implementation, followed by regular standard binary hardware. To get mixed radix coefficients, the MRC method requires n modulo additions and n single input table look-up stages, where the modulo addition can be replaced using 2's complement addition followed by a correction factor on the adder output. The moduli set $(2^n-1, 2^n, 2^n+1)$ is a recent variation of the well-known MRC technique [16]. The reason for taking the moduli set $(2^n-1, 2^n, 2^n+1)$ is that the residue adder can be efficiently performed in an end-around-carry adder. In addition, the multiplicative inverses of the set result in simplified expressions for the conversion throughout the mathematical relationships. However, the carry propagation will arise inside the mixed radix digits during addition/subtraction and the speed of the arithmetic is limited. In the residue-to-weighted number conversion, the crucial work is using an efficient multiplier and treating the subtraction. In this paper, the SD number system is newly introduced to avoid the carry propagation and simplify the arithmetic for number conversions. We also introduce the idea of multiplicative inverses of the most popular special moduli set $(2^n-1, 2^n, 2^n+1)$. Therefore, a multiplier is not needed for our conversion because the embedded multiplications can be replaced by simple shift-left operations. Moreover, we also find that the timing depends on the order of the moduli $2^n-1, 2^n$ and 2^n+1. The design results show that the proposed residue number arithmetic circuits are faster than those based on binary arithmetic.

In the following section, we give the definition and several properties of a redundant modular representation for RNS, by which an efficient arithmetic algorithm with radix-two SD numbers can be constructed. In Section 3, novel weighted-to-residue and residue-to-weighted number conversion algorithms are proposed. The conversion process depends on simple mathematical relationships without multiplications. Section 4 shows the hardware design and the simulation results, including the performance of our proposed conversions. Finally, Section 5 concludes this work.

2. Residue Number System and Residue Arithmetic with SD Number Representation

We consider a residue number system that has a set of relatively prime moduli, $\{2^n, 2^n-1, 2^n+1\}$. A residue digit with respect to a modulus m_i is represented by the number set:

$$l_{m_i} = \{0, \cdots, (m_i - 1)\}. \tag{1}$$

Let $M = \prod_{i=1}^{3} m_i = 2^n(2^n-1)(2^n+1)$. Szabo et al. [1] proved that if $0 \le A < M$, then the integer A has a one-to-one correspondence to its RNS representation. The integer A is uniquely represented by a 3-tuple (A_1, A_2, A_3), where

$$A_i = |A|_{m_i} = A - [A/m_i] \times m_i, \tag{2}$$

for $i = 1, 2, 3$. In the above equation, $[A/m_i]$ is the integer part, and each residue digit is defined as the remainder of least magnitude when A is divided by m_i.

A residue number x can be represented by an n-digit radix-two SD number representation as follows:

$$x = x_{n-1}2^{n-1} + x_{n-2}2^{n-2} + \cdots + x_0 \tag{3}$$

where $x_i \in \{-1, 0, 1\}$, which can be denoted as $x = (x_{n-1}, x_{n-2}, \cdots, x_0)_{SD}$. In the SD number representation, x has a value in the range of $[-(2^n-1), 2^n-1]$. However, it is difficult to know if x is in l_m, because of the redundancy of the SD number representation. (The subscript i is omitted.)

To simplify the manipulation of the modular operation in an SD number representation, we apply the definition that each residue digit has the following redundant residue number set:

$$L_m = \{-(2^n-1), \cdots, 0, \cdots, (m-1), \cdots, (2^n-1)\}, \tag{4}$$

Thus, x must be in L_m when it is expressed in an n-digit SD number representation. Obviously,

$$\begin{aligned} -x &= -(x_{n-1}, x_{n-2}, \cdots, x_0)_{SD} \\ &= (-x_{n-1}, -x_{n-2}, \cdots, -x_0)_{SD} \end{aligned}$$

is also in L_m.

[Definition 1] Let X be an integer and m be a modulus. Then $x = \langle X \rangle_m$ is defined as an integer in L_m. When $|X|_m \neq 0$, x has one of two possible values:

$$x = \langle X \rangle_m = |X|_m, \tag{5}$$

and

$$x = \langle X \rangle_m = |X|_m - \mathrm{sign}(|X|_m) \times m, \tag{6}$$

where

$$\mathrm{sign}(x) = \begin{cases} -1 & x < 0 \\ 1 & x \geq 0 \end{cases}.$$

When $|X|_m = 0$ and $m = 2^n - 1$, x has three possible values, that is, $-m$, 0 and m. □

The integer set l_m in (1) is a partial set of L_m. The intermediate results calculated in L_m are used for fast residue arithmetic. If necessary for a final result, they can be converted into l_m. Thus, the addition, subtraction and multiplication of 3-tuples $A = (A_1, A_2, A_3)$ and $B = (B_1, B_2, B_3)$ in an RNS can be represented as follows:

$$A \pm B = (\langle A_1 \pm B_1 \rangle_{m_1}, \langle A_2 \pm B_2 \rangle_{m_2}, \langle A_3 \pm B_3 \rangle_{m_3}), \tag{7}$$

$$A \times B = (\langle A_1 \times B_1 \rangle_{m_1}, \langle A_2 \times B_2 \rangle_{m_2}, \langle A_3 \times B_3 \rangle_{m_3}). \tag{8}$$

Obviously, the following properties exist in the redundant modular representation.

[Property 1]

Let a and b be integers. Then

(a) $\mathrm{abs}(\langle a \rangle_m) \leq 2^n - 1$,

(b) $\langle a + b \rangle_m \equiv \langle \langle a \rangle_m + \langle b \rangle_m \rangle_m$,

(c) $\langle a \times b \rangle_m \equiv \langle \langle a \rangle_m \times \langle b \rangle_m \rangle_m$,

and

(d) $\langle -a \rangle_m \equiv -\langle a \rangle_m$,

where $abs(x)$ is the absolute value of x and $\equiv$ indicates binary congruent modulo m. □

3. Mixed Radix Number System

The basic structures of our converters are shown in **Fig. 1**. A good way to approach this paper is to consider that the RNS with the SD number representation is integrated with the weighted number system by WTOR and RTOW number conversions.

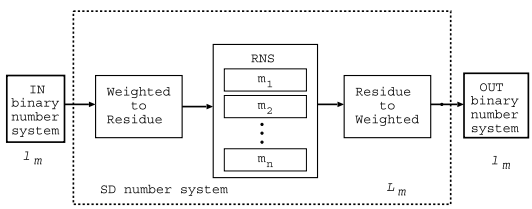

Fig. 1 Architecture of arithmetic operation.

Mixed radix conversion is fairly simple in principle. It would be usable in its given form in a residue computation system, because the residue computation system would be equipped to perform the arithmetic operation modulo m_i, not modulo M as required by the Chinese remainder theorem.

A number X as input or output in the architecture for moduli set $\{m_1, m_2, m_3\}$ can be expressed in mixed radix form:

$$X = a_3(w_3) + a_2(w_2) + a_1(w_1), \tag{9}$$

where $0 \leq a_i < m_i$ for i=1,2,3. The term a_i is mixed and $w_1 = 1, w_2 = m_1$ and $w_3 = m_2 m_1$ are radices. The mixed radix representation of X is denoted by (a_3, a_2, a_1), where the digits are listed in order of decreasing significance. The moduli $(2^n - 1, 2^n, 2^n + 1)$ residue number system has gained popularity because a modulo m addition in this system can be efficiently performed in an end-around-carry adder. For this reason, this type of RNS can be expected to play an increasing role in an RNS with weighted-to-residue and residue-to-weighted number conversions. Let $m_3 = 2^n, m_2 = 2^n - 1$ and $m_1 = 2^n + 1$ then $w_1 = 1, w_2 = 2^n + 1$ and $w_3 = (2^{2n} - 1)$. Notice that in Fig. 1, the input and output of the binary number system must be in the legitimate range $[0, m_3 m_2 m_1 - 1]$.

We represent numbers redundantly based on the redundant binary representation. This mainly has the following advantages over a binary number system.

(1) Parallel operation
The carry propagation that arises in the binary number system can be avoided using the redundant number representation.

(2) Fast subtraction
Subtraction can be done using addition with invers addend. In the redundant number representation, the NOT logical operation can be implemented by replacing Y with $-Y$, and we do not need any gates for the NOT logical operation.

(3) Word length
According to the SD number representation, all the residues in the moduli set $(2^n, 2^n - 1, 2^n + 1)$ are within n-digits, so the SD number representation is more efficient than the binary, which needs $(n + 1)$-bit binary numbers for the moduli $(2^n + 1)$.

Table 1 Rules for adding binary SD numbers.

	$abs(x_i) = abs(y_i)$	$abs(x_i) \neq abs(y_i)$	
		$(x_i + y_i) \times (x_{i-1} + y_{i-1}) \leq 0$	$(x_i + y_i) \times (x_{i-1} + y_{i-1}) > 0$
w_i	0	$x_i + y_i$	$-(x_i + y_i)$
c_i	$(x_i + y_i)/2$	0	$x_i + y_i$

3.1 Weighted-to-Residue Number Conversion

The proposed method for conversion from the weighted number system to the residue number system is for a specific moduli set $m_3 = 2^n, m_2 = 2^n - 1, m_1 = 2^n + 1$. Let B be a $3n$-digit number, represented as (9), and given by

$$B = a_3 \times (m_2 m_1) + a_2 \times (m_1) + a_1 \quad (10)$$

where $0 \leq B < 2^n(2^n - 1)(2^n + 1)$. Let $b_1 = B \bmod 2^n + 1$, $b_2 = B \bmod 2^n - 1$ and $b_3 = B \bmod 2^n$, where b_1, b_2 and b_3 are residue digits for the moduli $2^n + 1, 2^n - 1$ and 2^n. Evidently, $b_1 = a_1$. Then, b_2 and b_3 are converted as follows:

$$\begin{aligned} b_2 &= \left| |a_2(2^n - 1) + 2a_2|_{2^n-1} + |a_1|_{2^n-1} \right|_{2^n-1} \\ &= \left| |2a_2|_{2^n-1} + |a_1|_{2^n-1} \right|_{2^n-1} \quad (11) \\ b_3 &= \left| |(2^{2n} - 1)a_3|_{2^n} + |a_2|_{2^n} + |a_1|_{2^n} \right|_{2^n} \\ &= \left| -a_3 + |a_2 + a_1|_{2^n} \right|_{2^n} \quad (12) \end{aligned}$$

We can implement the above number conversion by using binary arithmetic method. In the definition of the SD number for RNS, the legitimate range of b_i is $[-(m_i - 1), (m_i - 1)]$. We use the modulo operation $\langle \cdot \rangle_m$ to deal with $|\cdot|_m$ using the SD number representation so that we do not consider whether the converted result is ≥ 0. Therefore, the conversion can achieve high speed. Moreover, the b_is are directly applicable to next-generation residue SD architecture.

Algorithm A

Let a_1, a_2 and a_3 be n-digit mixed radix numbers for modulo $2^n + 1, 2^n - 1$ and 2^n, and let b_1, b_2 and b_3 be n-digit residue numbers. The a_i and b_i are represented as SD numbers for $i = 1, 2, 3$.

(1) procedure for b_1
$b_1 = a_1$;

(2) procedure for b_2
(2A) $Z1 = \langle 2a_2 \rangle_{2^n-1}$;
(2B) $b_2 = \langle Z1 + a_1 \rangle_{2^n-1}$;

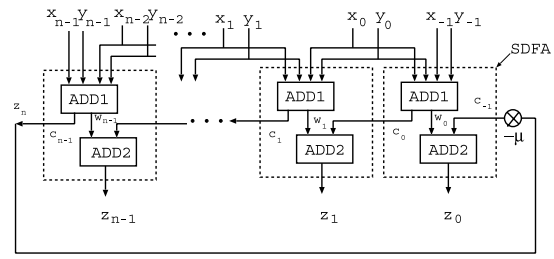

Fig. 2 Modulo m signed-digit adder (MSDA).

(3) procedure for b_3
(3A) $R1 = \langle a_2 + a_1 \rangle_{2^n}$;
(3B) $R2 = \langle -a_3 \rangle_{2^n}$;
(3C) $b_3 = \langle R2 + R1 \rangle_{2^n}$;

In (2A), we perform a modular doubling. We first shift a_2 to the left and re-insert it to the least significant digit (LSD). In (3B) and (3C), we use one modulo m signed-digit Addition (MSDA) by replacing the addend with $-a_3$. Therefore, we need three MSDAs [8] for (2B), (3A) and (3C). In the above algorithm, b_1, b_2 and b_3 can be performed in parallel.

The proposed converters are constructed by using MSDAs. A circuit diagram of an n-digit MSDA with n SD full adders (SDFAs) is shown in **Fig.2**. One SDFA consists of one ADD1 and one ADD2. The ADD1 generates the intermediate sum and the intermediate carry, and the ADD2 sums the low intermediate carry and the intermediate sum. Let c_i and w_i be the carry and the intermediate sum of the ith digit position, respectively. Their values are set as shown in **Table 1** with respect to the values of x_i, y_i, x_{i-1} and y_{i-1}. Thus, the modulo m addition can be performed in parallel without the carry propagation [8]. We use $\otimes$ to mean a 1-by-1 multiplier.

[Example 1] We now illustrate the conversion procedure for $n = 3$ and $B = 411 = a_3(m_2 m_1) + a_2(m_1) + a_1$ where $m_3 = 2^3, m_2 = 2^3 - 1$ and $m_1 = 2^3 + 1$. Thus, the inputs, the mixed radix digits, are $a_3 = 6 = (110)$, $a_2 = 3 = (011)$ and $a_1 = 6 = (110)$. Encoding the binary representation to the SD number representation is not difficult if a sign bit is inserted for all binary bits. In the following procedure, the addition is replaced with an MSDA.

Hence $b_1 = (110)$, and

$$\begin{aligned} b_2 &= \langle\langle 2(011)\rangle_{2^3-1} + (110)\rangle_{2^3-1} \\ &= \langle(110) + (110)\rangle_{2^3-1} \\ &= (101) \\ b_3 &= \langle(\bar{1}\bar{1}0) + \langle(011) + (110)\rangle_{2^3} \\ &= \langle(\bar{1}\bar{1}0) + (001)\rangle_{2^3} \\ &= (1\bar{1}1) \end{aligned}$$

where $\bar{1} = -1$. Thus, the output is $((1\bar{1}1), (101), (110))$. □

3.2 Arithmetic of Residue Number System

We now consider the arithmetic of RNS. A number in RNS is represented by the residue of each modulus, and arithmetic operations are accomplished based on each modulus. Suppose that two numbers, B and D, are represented as $B = (b_3, b_2, b_1)$ and $D = (d_3, d_2, d_1)$ in RNS for moduli $m_3 = 2^n, m_2 = 2^n - 1$ and $m_1 = 2^n + 1$. The arithmetic of B and D in RNS satisfies Eqs. (7) and (8).

Because the moduli are independent of each other, there is no carry propagation among them. All residue digits in the binary system are positive by the definition of the mod operation. If $x_i = b_i - d_i < 0$, then $x_i = m_i + (b_i - d_i)$. However, in the SD number system, by Definition 1, we do not consider whether x_i is ≥ 0 or < 0, and this enables high speed. Moreover, no carry propagation arises during the addition inside the residue number digits.

[Example 2] We provide an example of the arithmetic of residue numbers with SD number representation. Let $B = ((1011), (1010), (10\bar{1}1))$ and $D = ((0110), (01\bar{1}0), (0101))$ for the moduli $2^4, 2^4 - 1$ and $2^4 + 1$. Consider $B + D$ in RNS. We use three MSDAs to calculate it, and this procedure is shown in **Fig. 3**. Note that c_{-1} is evaluated using the end-around-carry. The result is $((0001), (\bar{1}101), (\bar{1}1\bar{1}\bar{1}))$.

These characteristics allow RNS computations with SD number representations to be complemented more quickly.

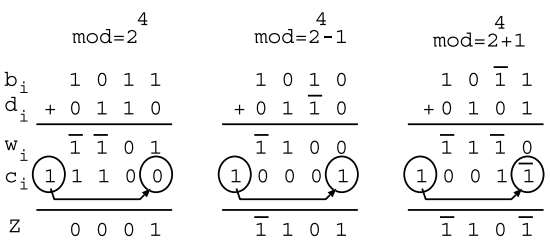

Fig. 3 MSDA in RNS.

3.3 Residue-to-Weighted Number Conversion

Next, we consider the conversion of the residue number to the weighted format with respect to the moduli set $(2^n, 2^n - 1, 2^n + 1)$. To recover the residue number representation of X, the basic formula of MRC is applied.

Let X correspond to the residue (x_3, x_2, x_1) for moduli $(m_3, m_2, m_1) = (2^n, 2^n - 1, 2^n + 1)$. The converter computes the number X from the 3-tuple (x_3, x_2, x_1), and the mixed radix notation is as follows:

$$a_1 = x_1 \tag{13}$$

$$a_2 = \left| |\frac{1}{m_1}|_{m_2} \times |x_2 - a_1|_{m_2} \right|_{m_2} \tag{14}$$

$$a_3 = \left| |\frac{1}{m_2}|_{m_3} \times \left| (|\frac{1}{m_1}|_{m_3} \times |x_3 - a_1|_{m_3} - a_2) \right|_{m_3} \right|_{m_3}. \tag{15}$$

Modulo m multiplications and modulo m subtractions are needed to compute a_2 and a_3. The above number conversion is usually implemented using binary number arithmetic [16]. As pointed out earlier, in the SD number system, we do not need any gates for the NOT logical operation, so the modulo m subtraction is easily implemented by replacing Y with $-Y$ in the modulo m addition. In the following, we introduce compact forms of the multiplicative inverse for the moduli set $(2^n - 1, 2^n, 2^n + 1)$. The modulo m multiplication can be implemented by shifting/inverting the multiplicand. The proof is shown in Property 2.

[Property 2] Let $m_3 = 2^n, m_2 = 2^n - 1$ and $m_1 = 2^n + 1$. The multiplicative inverses are as follows:

$$\left\langle \frac{1}{2^n - 1} \right\rangle_{2^n} = -1 \tag{16}$$

$$\left\langle \frac{1}{2^n + 1} \right\rangle_{2^n} = 1 \tag{17}$$

$$\left\langle \frac{1}{2^n + 1} \right\rangle_{2^n - 1} = 2^{n-1}. \tag{18}$$

The proofs of (16), (17), (18) are based on the fact that $|x|\frac{1}{x}||_{m_j} = 1$.

Proof of (16):

$$\begin{aligned} |(2^n - 1) \cdot (-1)|_{2^n} &= |-2^n + 1|_{2^n} \\ &= 1 \end{aligned}$$

Proof of (17):

$$|(2^n+1)\cdot(1)|_{2^n} = |2^n+1|_{2^n} = 1$$

Proof of (18):

$$\begin{aligned}|(2^n+1)\ \cdot\ (2^{n-1})|_{2^n-1} &= |(2^n-1)(2^{n-1})+2^n|_{2^n-1}\\ &= |2^n|_{2^n-1}\\ &= |(2^n-1)+1|_{2^n-1}\\ &= 1\end{aligned}$$

□

The most efficient order of the moduli is $m_3 = 2^n$, $m_2 = 2^n-1$ and $m_1 = 2^n+1$. The reason is as follows. Because the longest delay path is for calculating a_3 in the order $m_3 = 2^n$,$m_2 = 2^n-1$,$m_1 = 2^n+1$, the multiplication needed for a_3 is $|\frac{1}{m_1}|_{m_3} = 1$, whose multiplication is not required, and $|\frac{1}{m_3}|_{m_3} = -1$, whose multiplicative inverses are all sign digits. In SD number representation, the inversion procedure is simple.

In the following procedure, we use operation $\langle\cdot\rangle_m$ instead of $|\cdot|_m$. Substituting Eqs. (16), (17) and (18) into Eqs. (14) and (15) yields

$$a_2 = \langle 2^{n-1}\times\langle x_2-a_1\rangle_{m_2}\rangle_{m_2} \qquad (19)$$

$$a_3 = \langle(-1)\times\langle\langle x_3-a_1\rangle_{m_3}-a_2)\rangle_{m_3}\rangle_{m_3} = \left\langle -\langle(x_3-a_1)\rangle_{m_3}+(a_2)\right\rangle_{m_3}. \qquad (20)$$

The coefficients imply simple shift-left operations. Note that $-(m_1-1)\le a_1\le m_1-1$, $-(m_2-1)\le a_2\le m_2-1$, and $-(m_3-1)\le a_3\le m_3-1$.

[Example 3] Suppose that $X = 43$ and that its residue representation is $((011),(01\bar{1}),(111))$ for moduli $2^3, 2^3-1$, and 2^3+1. The proposed residue-to-weighted number conversion, shown in **Fig. 4**, derives the mixed radix digit $a_1 = (111) = 7, a_2 = (0\bar{1}\bar{1}) = -3$ and $a_3 = (01\bar{1}) = 1$. According to (9), $1(63)-3(9)+7 = 43 = X$. □

Algorithm B

Let x_1, x_2 and x_3 be the residue numbers for modulo $2^n+1, 2^n-1$ and 2^n, and let a_1, a_2 and a_3 be mixed radix digits.

(1) procedure for a_1
$a_1 = x_1$;

(2) procedure for a_2
(2A) $Z1 = \langle x_2-a_1\rangle_{m_2}$;
(2B) $a_2 = \langle 2^{n-1}Z1\rangle_{m_2}$;

(3) procedure for a_3
(3A) $R1 = \langle x_3-a_1\rangle_{m_3}$;
(3B) $R2 = \langle -R1\rangle_{m_3}$;

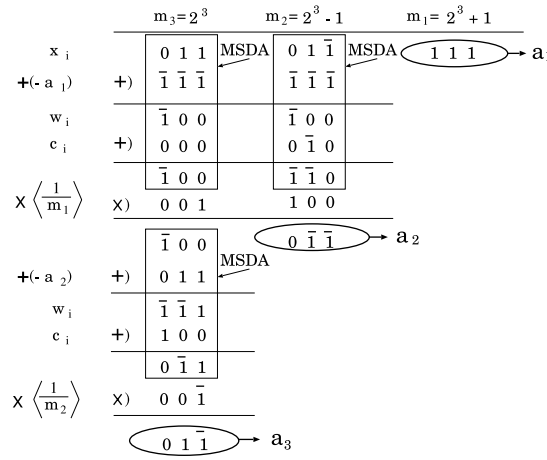

Fig. 4 Example of residue-to-weighted number conversion.

(3C) $R3 = \langle R2+a_2\rangle_{m_3}$;

In (2A), (3A) and (3C), we use MSDA by changing the addends to $-a_1$ and $-R1$. In (2B), the most significant $(n-1)$ positions will shift to the left and are re-inserted into the LSD.

3.4 Conversion to Binary Numbers

Because one-to-one correspondence between input and output is needed, we convert the $T = a_3(2^{2n}-1)+a_2(2^n+1)+a_1$ into l_m. First, we convert a_1, a_2 and a_3 to the 2's complement $(n+1)$-bit binary numbers $a_1^+, a_1^-, a_2^+, a_2^-, a_3^+$ and a_3^-, where the a_i^+ are the positive digits and the a_i^- are the negative digits for a_i $(i = 1,2,3)$. Then, we use three n-bit prefix adders to calculate $SA = a_1^+ + a_1^-$, $SB = a_2^+ + a_2^-$ and $SC = a_3^+ + a_3^-$ in parallel, where SA, SB and SC are 2's $(n+1)$-bit complement numbers. The prefix adder is based on the Kogge-Stone tree structure [18] which uses the associative operator $'o'$ defined in Ref. [19] to implement the carry computation. Note that $S = SC(2^{2n}-1)+SB(2^n+1)+SA$, and that S is an ordinary binary number. If $S<0$, then we add M to S and S will be returned to l_m, where $M = 2^n(2^n-1)(2^n+1)$. Therefore, $S+M = (2^n+SC)(2^{2n}-1)+SB(2^n+1)+SA$. We can consider 2^n+SC to mean "add '1' to the $(n+1)$th position of SC". Thus, the output is within the legitimate range $[0, M-1]$ and it is also a binary number representation. Notice that SC is an $(n+1)$-bit ordinary binary representation, and SB and SA are 2's $(n+1)$-bit complement ordinary binary representations.

4. Hardware Design and Performance Evaluation

In binary logic, a single memory space is all that is needed for a digit (bit), while in ternary logic the sign of the digit requires an extra space. In hardware implementation, an SD digit x is encoded as a 2-bit binary code defined as $x = [x^s, x^a]$, where x^s is the sign and x^a is the absolute value. This demands more hardware resources, but it also reallocates space, which affects the overall speed. We use a hardware description language, VHDL, to design the residue arithmetic circuits for the implementation of the proposed converters. Then, we performed a simulation under the conditions of 1-μm CMOS gate array technology. To compare the performance of the proposed circuits and conventional ones, we designed a binary architecture using the same technology by using a synthesis tool called Design Complier from Synopsys. The implementations are relative between binary number arithmetic and SD number arithmetic. Therefore, the performance comparison results may have the same relative rates under advanced design technology such as 0.3-μm CMOS gate array technology.

4.1 Proposed Architecture

The proposed weighted-to-residue number converter consists of three MSDAs, which are used for (2B), (3A) and (3C) in Algorithm A, and one shift block. The WTOR converter is shown in **Fig. 5**. The MSDA consists of n identical sub-blocks SDFAs, as shown in Fig.2 and the shift block is for shifting the most significant digit to the left and re-inserting it into the LSD corresponding to (2A) in Algorithm A. In Fig. 5, $\ominus$, which corresponds to (3B) in Algorithm A, means to replace y_i with $-y_i$. Note that b_1, b_2 and b_3 can be implemented in parallel by the proposed architecture.

The proposed residue-to-weighted number converter, which consists of three MSDAs and one shift block, is shown in **Fig. 6**. The three MSDAs are for steps (2A), (3A) and (3C) in Algorithm B. The shift block which corresponds to (2B) in Algorithm B, is for shifting the most significant $(n-1)$ positions to the left and reinserting them into the LSD.

The proposed WTOR and RTOW number converters are very simple, and only three MSDAs are required for both.

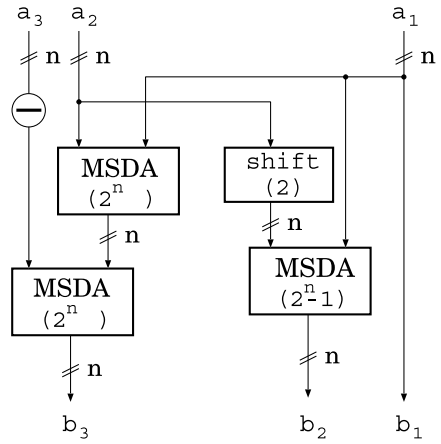

Fig. 5 Weighted-to-residue number converter.

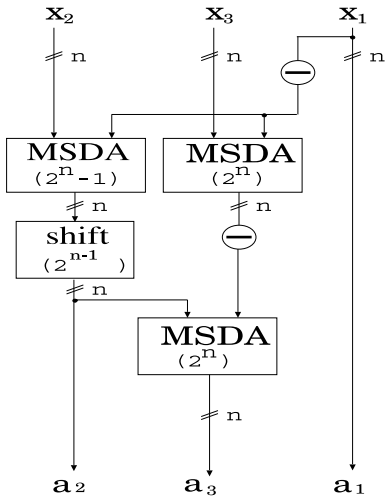

Fig. 6 Residue-to-weighted number converter.

4.2 Performance Evaluation and Comparison

Our aim is to enable high-speed conversion, so drawbacks in terms of area are not problems. Because a residue number computing system includes a number of residue number arithmetic circuits such as adders and multipliers (see Fig. 1), we list the evaluations of residue adders and multipliers not only in their binary number representations but also in SD number representations. The performance of the MSDA with the gate-level design is shown in **Table 2**. The data in the first column is the dynamic range for different choices of n. The simulation results show that the delay time of MSDA is independent of n. Kalampoukas, et al. [20] gives a good way to deal with modulo $(2^n - 1)$ adders using parallel-prefix structure [19]. Their model

assumes that each gate, excluding exclusive-OR, counts as one elementary gate or both area and delay. Their delay time [20] is $2\log_2 n + 3$ and their chip area is $3n\log_2 n + 4n$. However, this technique cannot be applied to the modulo $(2^n + 1)$ adder because the end-around-carry is negative. On the other hand, MSDA can deal with the modulo (2^n+1) adder in a manner similar to the modulo $(2^n - 1)$ adder, because the SD number representation is redundant. When the modulus $2^n - 1$ is large, MSDA has more advantages than Kalampoukas, et al. [20]. The relationship is shown in **Fig.7**.

Next, we compare the modular SD multiplier with the binary multiplier. Efstathiou, et al. [17] gives an efficient binary modulo $2^n - 1$ multiplier. A Booth-code method was used to reduce the numbers of modulo $2^n - 1$ partial products and thus the chip area is smaller. In the modulo $2^n + 1$ multiplier, we calculate $(n \times n)$-bit multiplication, and the result is mod $(2^n + 1)$ [5]. The modular SD multiplier is designed as a binary tree structure [8] to increase speed. The multiplier in RNS is shown in **Table 3**. The SD multiplier is faster than the binary multiplier.

Tables 2 and 3 show that SD number architectures are faster than binary architectures in RNS arithmetic. Therefore, the high-speed converters architectures for WTOR and RTOW with SD number representation, as shown in Fig. 1, are important. The tables show that a residue number arithmetic system with the SD number arithmetic can have higher performance than that with the ordinary binary arithmetic.

Table 2 Performance of modulo $2^n - 1$ adder.

n	area (gates)		delay (ns)	
	PA [20]	SD number	PA [20]	SD number
4	60	120	4.95	5.46
8	136	240	5.7	5.46
16	378	480	8.82	5.46

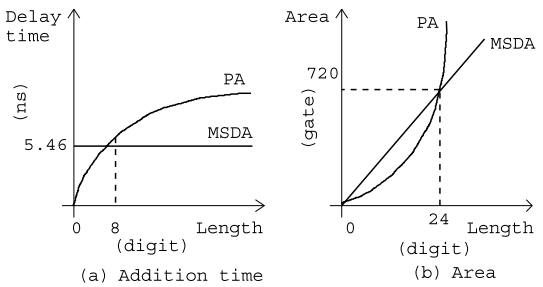

Fig. 7 Comparison of adders.

Table 4 shows a comparison of the performance of WTOR number converters with the architecture shown in Fig. 5. Because the longest delay path is dependent on the delay time of two MSDAs when n is large, the proposed WTOR number converter is faster than binary number converters.

The methods used for the RNS to binary conversion are based on CRT or MRC. The Andraos-Ahmad algorithm, which introduces compact forms of multiplicative inverses to simplify CRT, is a well-known technique. The algorithm uses four adders, two of which operate in parallel, to convert the moduli $(2^n + 1, 2^n, 2^n - 1)$ residue number into their binary equivalent. Recently, Piestrak [11] suggested a simplification of the Andraos-Ahmad technique: the value of the $-r_1$ modulo 2^{2n-1} of Andraos, et al. [10] can be easily obtained by manipulating r_1. Piestrak [11] proposed two methods. The first method, referred to as the cost-effective (CE) version, uses two $2n$-bit CSAs and one $2n$ CPA with an end-around-carry to calculate the $A + B + C - r_1$ of Andraos, et al. [10]. The other method, which is referred to as the high-speed (HS) version, uses two $2n$-bit CSAs and two parallel $2n$-bit CPAs followed by a multiplexer. Our technique is comparable to Piestrak's technique.

Now, we evaluate the proposed RTOW converter design.

Table 4 Performance of weighted-to-residue number converter.

n	area (gates)		delay (ns)	
	binary	SD number	binary	SD number
4	116	368	12.17	10.41
8	265	736	21.31	10.41
16	627	1,472	40.67	10.41

Table 3 Performance of parallel modulo $2^n \pm 1$ multipliers.

n	area (gates)			delay (ns)		
	binary [5] $(2^n + 1)$	binary [17] $(2^n - 1)$	SD number $(2^n \pm 1)$	binary [5] $(2^n + 1)$	binary [17] $(2^n - 1)$	SD number $(2^n \pm 1)$
4	177	136	474	19.72	19.72	16.94
8	748	456	2,106	42.03	37.03	22.92
16	3,226	2,103	8,826	91.85	73.75	29.94

Table 5 Performance of the residue-to-weighted number converter, with a comparison of cost-effective and high-speed methods.

n	area (gates)				delay (ns)			
	CRT		MRC		CRT		MRC	
	CE [11]	HS [11]	binary [16]	SD number	CE [11]	HS [11]	binary [16]	SD number
4	240	345	147	376	20.06	15.25	18.73	13.75
8	480	689	329	752	32.38	22.56	38.81	13.75
16	960	1,377	727	1,504	57.02	37.19	74.62	13.75

Table 6 Performance of prefix adder.

n	area (gates)	delay (ns)
4	48	5.79
8	119	7.75
16	284	9.83

$$Area_{RTOW} = 3A_{MSDA} + A_{shift} \quad (21)$$
$$Delay_{RTOW} = 2\triangle_{MSDA} \quad (22)$$

where A_{MSDA} and A_{shift} are the areas of the $MSDA$ and the shifter in Fig. 6, and the delay of the MSDA is $\triangle_{MSDA}$. The execution time of the MSDA is O(1), which is independent of n. We find the best order is $m_3 = 2^n - 1, m_2 = 2^n$ and $2^n + 1$ [16]. Their longest delay path is in calculating a_3, and the inversion can be easily replaced using 1's complement representation. A comparison between the proposed converter and CRT [11] is provided in **Table 5**. The results show that our converter is very fast and that the delay time is independent of n.

To convert SD number representation to binary number representation, we use prefix adders, as mentioned in Section 3.4. The performance of one prefix adder is illustrated in **Table 6**, which shows that the conversion is fast. The binary number representation is a 2's complement representation, but it is ≥ 0. The 2's complement representation is suitable for our converter.

5. Conclusion

We presented simple weighted-to-residue and residue-to-weighted number converters for moduli $2^n, 2^n - 1$ and $2^n + 1$. Our method requires only addition for WTOR and RTOW number conversion. The proposed converters have many advantages, which have been demonstrated throughout this paper using examples and analysis. Our simulation shows that the performance of our converters is comparable to that of binary architectures and that the proposed schemes are high-speed architectures. The RNS arithmetic operation based on the SD number system has been shown to be more efficient than the binary number system. In addition, the experimental results show that the proposed WTOR and RTOW number converters, which can be easily implemented on hardware and make the calculation less complicated and more efficient, are high-speed architectures. Thus, our method is more efficient and less complicated than the binary number system in the arithmetic operation. We also provided a method for converting an SD number to a binary number representation.

References

1) Szabo, N.S. and Tanaka, R.I.: *Residue Arithmetic and Its Applications to Computer Technology*, New York, McGraw-Hill (1967).
2) Paliouras V. and Stouraitis T.: Novel high-radix residue number system architectures, *IEEE Trans. circuits and systems II.*, Vol.47, No.10, pp.1059–1073 (Oct. 2000).
3) Sonderstrand, M.A., Jendins, W.K., Junllien, G.A. and Taylor, F.J.: *Residue Number System Arithmetic: Modern Applications in Digital Signal Processing*, IEEE Press, New York (1986).
4) Skavantzos, A. and Rao, P.B.: New multipliers modulo $2^n - 1$, *IEEE Trans. Comput.,* Vol.41, No.8, pp.957–961 (Aug. 1992).
5) Hiasat, A.: New memoryless mod $(2^n \pm 1)$ residue multiplier, *Electronics Letters,* Vol.28, No.3, pp.314–315 (Jan. 1992).
6) Avizienis, A.: Signed-digit number representations for fast parallel arithmetic, *IRE Trans. Elect. Comput.*, EC-10, pp.389–400 (Sep. 1961).
7) Parhami, B.: Carry-free addition of recoding binary signed-digit numbers, *IEEE Trans. comput.,* Vol.37, No.11, pp.1470–1476 (Nov. 1988).
8) Wei, S. and Shimizu, K.: A novel residue arithmetic hardware algorithm using a signed-digit number representation, *IEICE Trans. Inf. & Syst.*, Vol.E83-D, No.12, pp.2056–2064 (Dec. 2000).
9) Wei, S. and Shimizu, K.: Compact residue arithmetic multiplier based on the radix-4 signed-digit multiple-valued arithmetic circuits, *IEICE Trans. Electron.*, Vol.E82-C, No.9, pp.1647–1645 (Sep. 1999).

10) Andraos, S. and Ahmad, H.: A new efficient memoryless residue to binary converter, *IEEE Trans. Circuits Syst.*, Vol.CAS-35, pp.1441–1444 (Nov. 1988).
11) Piestrak, S.J.: A high-speed realization of a residue to binary number system converter, *IEEE Trans. Circuits Syst.II*, Vol.42, No.10, pp.661–663 (Oct. 1995).
12) Conway, R. and Neison, J.: Fast converter for 3 moduli RNS using new property of CRT, *IEEE Trans. Comput.*, Vol.48, No.8, pp.852–860 (Aug. 1999).
13) Baraniecka, A. and Jullien, G.: On decoding techniques for residue number system realizations of digital signal processing hardware, *IEEE Trans. Circuits Syst.,* Vol.CAS-25, pp.935–936 (Nov. 1978).
14) Huang, C.: A fully parallel mixed-radix conversion algorithm for residue number applications, *IEEE Trans. Comput.,* Vol.C-32, pp.398–402 (Apr. 1983).
15) Chakraborti, N., Soundararajan, J. and Reddy, A.: An implementation of mixed-radix conversion for residue number applications, *IEEE Trans. Comput.,* Vol.C-35, pp.762–764 (Aug. 1986).
16) Ananda Mohan, P.V.: Evaluation of fast conversion techniques for binary-residue number system, *IEEE Trans. Circuits Syst.-I,* Vol.45, No.10, pp.1107–1109 (Oct. 1998).
17) Efstathiou, C., Vergos, H.T. and Nikolos, D.: Modified booth modulo $2^n - 1$ multipliers, *IEEE Trans. Comput.*, Vol.53, No.3, pp.370–374 (Mar. 2004).
18) Kogge, P.M. and Stone, H.S.: A parallel algorithm for the efficient solution fo a general class of recurrence equations, *IEEE Trans. Comput.,* Vol.22, No.8, pp.783–791 (Aug. 1973).
19) Brent, R.P. and Kung, H.T.: A regular layout for parallel adders, *IEEE Trans. Comput.,* Vol.31, No.3, pp.260–264 (Mar. 1982).
20) Kalampoukas, L., Nikolos, D., Efstathiou, C., Vergos, H.T. and Kalamatianos, J.: High-speed parallel-prefix modulo $2^n - 1$ adders, *IEEE Trans. Comput.* Vol.49, No.7, pp.673–680 (July 2000).

(Received September 15, 2005)
(Accepted March 2, 2006)
(Online version of this article can be found in the IPSJ Digital Courier, Vol.2, pp.328–337.)

Shuangching Chen was born in Kaohsiung, Taiwan on July 18, 1972. He received the B.E. degree in Applied Mathematic from Feng Chia University, Taiwan, Republic of China in 1994, and the M.E. degree in Computer Science from Gunma University, Kiryu, Japan in 2002. He is currently a doctoral student at the Department of Computer Science, Gunma University. His research interests include parallel computer architecture, residue architecture, VLSI design and digital signal processing. He is a student member of IPSJ and IEICE.

Shugang Wei was born in Harbin, China on September 19, 1957. He received the B.E. degree in Radio Engineering from the Harbin Institute of Technology, Harbin, China, the M.E. degree in Computer Science from Gunma University, Kiryu, Japan and the Dr. Eng. degree in Electronic Engineering from Tohoku University, Sendai, Japan, in 1982, 1987, and 1990, respectively. He was an Assistant Professor with the Department of Radio Engineering, Harbin Institute of Technology from 1982 to 1984. In 1990 he joined Matsushita Communication Industrial Co., Ltd., Yokohama, Japan. At present he is an Associate Professor in the Department of Computer Science, Faculty of Engineering, Gunma University. His research interests include logic design, high-speed arithmetic circuits, VLSI systems and digital audio signal processing. Dr. Wei is a member of the Acoustical Society of Japan, IEICE and IEEE.