How to Teach Residue Number System to Computer Scientists and Engineers

Keivan Navi, Amir Sabbagh Molahosseini, and Mohammad Esmaeildoust

***Abstract*—The residue number system (RNS) has been an important research field in computer arithmetic for many decades, mainly because of its carry-free nature, which can provide high-performance computing architectures with superior delay specifications. Recently, research on RNS has found new directions that have resulted in the introduction of efficient algorithms and hardware implementations for RNS with much better performance than previous ones. Furthermore, the applicability of RNS in many computer and digital signal processing (DSP) applications has simultaneously greatly increased. Hence, the need is evident for the development of a new and well-organized RNS teaching method with emphasis on recent achievements. In this paper, a step-by-step teaching process for RNS that describes RNS design in separate parts is presented. Each part is investigated in detail, taking into account the recent research performed on RNS. The compatibility of the proposed method with the new RNS research trends makes this method suitable for researchers as well as students.**

***Index Terms*—Forward converter, moduli sets, residue arithmetic, residue number system (RNS), reverse converter.**

I. Introduction

THE residue number system (RNS) is principally taught to graduate students; an introduction to RNS can also be taught as a part of a computer arithmetic course for graduate or undergraduate students of computer science or engineering. Recently, new algorithms and architectures have been introduced that make RNS less difficult to use, which has led to improvement in the performance of RNS. The resources used for teaching are mainly based on chapters of the books [1] and [2]. These describe RNS in a general way, but while giving a good introduction to RNS, these textbooks do not cover new achievements in the field.

The major issues in efficient design of RNS are the moduli set selection, forward conversion, residue arithmetic unit, and reverse conversion. Each RNS system is based on a moduli set, which involves a set of pair-wise relatively prime integers. The dynamic range of an RNS system is defined in terms of the product of the moduli, and it denotes the interval of integers exclusively represented in RNS. The forward converter decomposes a weighted binary number into a residue represented number with regard to the moduli set. The residue arithmetic unit includes a modular adder, subtractor, and multiplier for each modulo channel. The reverse converter transforms a residue represented number into its equivalent weighted binary number. To understand RNS, each issue, with its related problems and design methods, must be carefully investigated.

The fundamental concepts that should be covered in RNS teaching can be classified as follows:

1) the definition of RNS with its properties and some important applications;
2) moduli set selection considerations;
3) forward converter design;
4) residue arithmetic unit parts;
5) reverse converter design.

In this paper, a five-step teaching method is presented that takes into account new research achievements in realizing RNS. First, the principle of RNS with a numerical example is explained in Section II. Next, considerations in the selection of the moduli set, with the significant impact of this on different parts of RNS, are described in Section III. Forward converter design is surveyed in Section IV. Section V provides an overview of the residue arithmetic unit with concentration on modulo adder and multiplier design. The reverse converter, which is the most complex part of RNS, is investigated in Section VI. Finally, assessment results and achievements are presented in Section VII.

II. Residue Number System

The RNS is an unconventional number system that is defined in terms of relatively prime moduli set $\{m_1, m_2, \ldots, m_n\}$ that is $\gcd(m_i, m_j) = 1$ for $i \neq j$. A weighted number X can be represented as $X = (x_1, x_2, \ldots, x_n)$, where

$$x_i = X \bmod m_i = |X|_{m_i}, \quad 0 \leq x_i < m_i. \tag{1}$$

Such a representation is unique for any integer X in the range $[0, M-1]$, where M is the dynamic range of the moduli set $\{m_1, m_2, \ldots, m_n\}$, which is equal to the product of m_i terms $(M = m_1, m_2, \ldots, m_n)$ [3].

In RNS, the weighted operands are converted into residue representations by a forward converter. Then, arithmetic operations such as addition, subtraction, and multiplication are performed on RNS numbers in parallel without carry propagation between residue digits. Hence, RNS results in parallel and high-speed addition, subtraction, and multiplication. Each modulus of the moduli set has its own arithmetic processor that consists of a modulus adder, a modulus subtractor, and a modulus multiplier. Next, to use the result of arithmetic operations outside of RNS, the resulted RNS number should be converted into its corresponding weighted binary number by using a reverse converter. Fig. 1 shows the block diagram of a typical RNS system.

Manuscript received July 11, 2009; revised October 12, 2009; accepted March 31, 2010. Date of publication May 03, 2010; date of current version February 02, 2011.

K. Navi and M. Esmaeildoust are with the Faculty of Electrical and Computer Engineering, Shahid Beheshti University, Tehran 1983963113, Iran (e-mail: navi@sbu.ac.ir; m_doust@sbu.ac.ir).

A. S. Molahosseini (amir.sabbagh@sr.iau.ac.ir) is with the Microelectronic LAB, Shahid Beheshti University, Tehran 1983963113, Iran.

Digital Object Identifier 10.1109/TE.2010.2048329

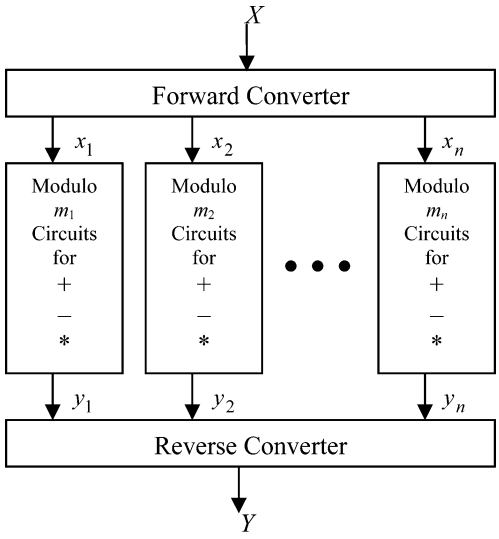

Fig. 1. Block diagram of a typical RNS system.

Example 1: Consider an RNS system based on the moduli set $\{5, 7, 9\}$. The RNS representations of the weighted numbers 53 and 72 can be computed as follows:

$$|53|_5 = 3, \quad |53|_7 = 4, \quad |53|_9 = 8$$
$$|72|_5 = 2, \quad |72|_7 = 2, \quad |72|_9 = 0.$$

Therefore, based on the moduli set $\{5, 7, 9\}$, the weighted numbers 53 and 72 have RNS representation as (3, 4, 8) and (2, 2, 0), respectively. Now, the addition on these RNS numbers can be performed as follows:

$$|3 + 2|_5 = 0, \quad |4 + 2|_7 = 6, \quad |8 + 0|_9 = 8.$$

The weighted number 125 in RNS number is equivalent to (0, 6, 8), and also verification can be done as

$$|125|_5 = 0, \quad |125|_7 = 6, \quad |125|_9 = 8.$$

Thus, it can be seen that the addition in the RNS can be performed in parallel without carry propagation between residue digits. Subtraction and multiplication can be executed in a similar way.

Division [4], sign detection [5], and magnitude comparison [6] are difficult RNS operations. Consequently, RNS is commonly used in situations where we only need addition, subtraction, and multiplication. Today, RNS is used for reducing the power dissipation in high-performance systems. RNS can tolerate a large reduction in the supply voltage, compared to the corresponding binary architecture, while achieving a particular delay specification [7]. RNS has many applications in digital signal processing (DSP) [8]. In particular, RNS is an interesting and useful tool for the implementation of high-speed FIR filters [9]–[11]. Also, RNS image coding can offer high-speed and low-power very large scale integration (VLSI) implementation for secure image processing [12], [13]. In addition to these merits, redundant RNS offers new approaches to the design of error detection and correction codes [14], [15].

III. Moduli Set Selection

The moduli selection has an important role in the design of RNS systems. The dynamic range, the speed, and the hardware implementation of RNS systems depend on the form and the number of the moduli chosen [16]. Various moduli sets have been suggested for RNS; these can be categorized as follows.

A. Binary Moduli Sets

It is well known that powers of two related moduli $(2^n, 2^n \pm 1)$ can simplify the required arithmetic operations and result in efficient hardware implementation of the RNS system by using usual binary hardware. In [17], a systematic approach for selecting binary moduli sets is presented, and it is shown that the numbers $2^n \pm 1$ and 2^n can be used as moduli choices for RNS. The dynamic range of moduli set can be determined by multiplying the moduli; however, the number of bits of the product is usually used for representing the dynamic range.

The $3n$-bit dynamic range moduli set $\{2^n - 1, 2^n, 2^n + 1\}$ [18] is the most famous RNS moduli set because of its simple and well-formed balanced moduli; however, arithmetic operations with respect to the modulus $2^n + 1$ are more complex than for the moduli 2^n and $2^n - 1$. Therefore, in $3n$-bit dynamic range RNS moduli sets $\{2^{n-1} - 1, 2^n - 1, 2^n\}$ [19] and $\{2^n - 1, 2^n, 2^{n+1} - 1\}$ [20], modulus $2^n + 1$ has been substituted by moduli $2^{n-1} - 1$ and $2^{n+1} - 1$, respectively, to accelerate the speed of the RNS arithmetic unit. Today, the dynamic range provided by these moduli sets is not sufficient for applications that require larger dynamic range with more parallelism. Hence, $4n$-bit dynamic range four-moduli sets $\{2^n - 3, 2^n + 1, 2^n - 1, 2^n + 3\}$ [21], $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$, and $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1\}$ [22] have been introduced. Moreover, several four-moduli sets with larger dynamic ranges than $4n$-bit such as $\{2^n - 1, 2^n, 2^n + 1, 2^{2n} + 1\}$ [23], $\{2^n - 1, 2^n + 1, 2^{2n} - 2, 2^{2n+1} - 3\}$ [24], $\{2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1\}$, and $\{2^n - 1, 2^n + 1, 2^{2n}, 2^{2n} + 1\}$ [25] have been also suggested. Among these, the moduli sets $\{2^n - 1, 2^n, 2^n + 1, 2^{2n} + 1\}$ and $\{2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1\}$ are both in the category of $5n$-bit dynamic range sets, and they are known as conversion-friendly and arithmetic-friendly moduli sets, respectively. As stated in [25], the set $\{2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1\}$ can result in a faster arithmetic unit than the moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{2n} + 1\}$, while the reverse converter for the set $\{2^n - 1, 2^n, 2^n + 1, 2^{2n} + 1\}$ can be designed easily and more efficiently than the reverse converter for the moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1\}$. In addition to these, some $5n$-bit dynamic range five-moduli sets such as $\{2^n, 2^n - 1, 2^n + 1, 2^n - 2^{(n+1)/2} + 1, 2^n + 2^{(n+1)/2} + 1\}$ [26], $\{2^n - 1, 2^n, 2^n + 1, 2^{n-1} - 1, 2^{n+1} - 1\}$ [27], and $\{2^n, 2^{n/2} - 1, 2^{n/2} + 1, 2^n + 1, 2^{2n-1} - 1\}$ [28] have been proposed.

It should be noted that as the number of moduli in the set increases, the RNS complexity will increase. Thus, the RNS systems based on four- and five-moduli sets are more complex than those based on three-moduli sets. Hence, three-moduli sets with larger dynamic range than the traditional three-moduli sets have been considered. These moduli sets are $\{2^n - 1, 2^n, 2^{2n+1} - 1\}$ [29], $\{2^n - 1, 2^n + 1, 2^{2n} + 1\}$ [30], $\{2^n, 2^{2n} - 1, 2^{2n} + 1\}$ [31], and $\{2^{2n}, 2^n - 1, 2^{n\pm1} - 1\}$ [32]. In particular, as reported in [31] and [32], the RNS systems based on moduli sets $\{2^{2n}, 2^n - 1, 2^{n\pm1} - 1\}$ and $\{2^n, 2^{2n} - 1, 2^{2n} + 1\}$ have better

TABLE I
SUMMARY OF DIFFERENT MODULI SETS

Three Moduli sets	Four Moduli sets	Five Moduli sets
$\{2^n-1, 2^n, 2^n+1\}$ [18]	$\{2^n-3, 2^n+1, 2^n-1, 2^n+3\}$ [21]	$\{2^n, 2^n-1, 2^n+1, 2^n-2^{(n+1)/2}+1,$ $2^n+2^{(n+1)/2}+1\}$ [26]
$\{2^{n-1}-1, 2^n-1, 2^n\}$ [19]	$\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ $\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$ [22]	
$\{2^n-1, 2^n, 2^{n+1}-1\}$ [20]		$\{2^n-1, 2^n, 2^n+1, 2^{n-1}-1, 2^{n+1}-1\}$ [27]
$\{2^n-1, 2^n, 2^{2n+1}-1\}$ [29]	$\{2^n-1, 2^n, 2^n+1, 2^{2n}+1\}$ [23]	
$\{2^n-1, 2^n+1, 2^{2n}+1\}$[30]	$\{2^n-1, 2^n+1, 2^{2n}-2, 2^{2n+1}-3\}$[24]	$\{2^n, 2^{n/2}-1, 2^{n/2}+1, 2^n+1, 2^{2n-1}-1\}$ [28]
$\{2^n, 2^{2n}-1, 2^{2n}+1\}$ [31]	$\{2^n-1, 2^n, 2^n+1, 2^{2n+1}-1\}$ $\{2^n-1, 2^n+1, 2^{2n}, 2^{2n}+1\}$ [25]	
$\{2^{2n}, 2^n-1, 2^{n+1}-1\}$ [32]		
$\{2^\alpha, 2^\beta-1, 2^\beta+1\}$ [33]		

performance than those based on four- and five-moduli sets, respectively, for same dynamic range requirement. Recently, the general moduli $\{2^\alpha, 2^\beta-1, 2^\beta+1\}$ [33], where $\alpha < \beta$, have been introduced. The values of α and β are set to regulate the required dynamic range and also to obtain a balanced moduli set. As shown in [33], the general moduli $\{2^\alpha, 2^\beta-1, 2^\beta+1\}$ where $\alpha < \beta$ can offer a high-performance RNS, especially for reverse conversion. Table I shows the summary of different moduli sets.

IV. FORWARD CONVERTER

A forward converter [41] consists of independent converters for each modulus of the moduli set for transforming a weighted integer binary number into its equivalent RNS representation. Forward conversion is a simple process and can be efficiently realized using multi-operand modular adders. As an example, a forward converter for the RNS based on the general three-moduli set $\{2^\alpha, 2^\beta-1, 2^\beta+1\}$ [33], [72] can be designed as follows.

Consider an m-bits weighted number A, which can be represented in binary as follows:

$$A = (a_{m-1}a_{m-2}\ldots a_1a_0)_2. \tag{2}$$

The residues (x_1, x_2, x_3) corresponding to the moduli set $\{2^\alpha, 2^\beta-1, 2^\beta+1\}$ must be computed. First, the residue in moduli 2^α is obtained as

$$\begin{aligned} x_1 &= |A|_{2^\alpha} = |(a_{m-1}\ldots a_1a_0)_2|_{2^\alpha} \\ &= |(a_{m-1}\ldots a_{\alpha+1}a_\alpha)_2 \times 2^\alpha + (a_{\alpha-1}\ldots a_1a_0)_2|_{2^\alpha} \\ &= (a_{\alpha-1}\ldots a_1a_0)_2. \end{aligned} \tag{3}$$

Therefore, it is enough to consider the least significant α bits, and the rest of the bits of A will be ignored, as they are multiples of 2^α. Now, the modulus $(2^\beta-1)$ must be investigated. The number A (2) can be partitioned into j consecutive β-bit blocks as follows:

$$A = T_{j-1}2^{(j-1)\beta} + \cdots + T_1 2^\beta + T_0 = \sum_{i=0}^{j-1} T_i \cdot 2^{i\beta}. \tag{4}$$

Next, the residue of A in modulus $(2^\beta-1)$ can be computed as

$$\begin{aligned} x_2 = |A|_{2^\beta-1} &= \left|T_{j-1}2^{(j-1)\beta} + \cdots + T_1 2^\beta + T_0\right|_{2^\beta-1} \\ &= |T_{j-1} + \cdots + T_1 + T_0|_{2^\beta-1} = \left|\sum_{i=0}^{j-1} T_i\right|_{2^\beta-1}. \end{aligned} \tag{5}$$

The above equation can be implemented with a multi-operand modular adder [42] that consists of a modulus $(2^\beta-1)$ carry save adder (CSA) tree followed by a modulus $(2^\beta-1)$ carry propagate adder (CPA). Finally, the residue of A in modulus $(2^\beta+1)$ can be calculated as follows:

$$\begin{aligned} x_3 = |A|_{2^\beta+1} &= \left|T_{j-1}2^{(j-1)\beta} + \cdots + T_1 2^\beta + T_0\right|_{2^\beta+1} \\ &= |T_{j-1} + \cdots - T_1 + T_0|_{2^\beta+1} = \left|\sum_{i=0}^{j-1} (-1)^i T_i\right|_{2^\beta+1}. \end{aligned} \tag{6}$$

Implementation of (6) also relies on a modulus $(2^\beta+1)$ multi-operand adder [42]. Therefore, forward converter for the general moduli set $\{2^\alpha, 2^\beta-1, 2^\beta+1\}$ is composed of two multi-operand modular adders that work in parallel.

Example 2: Consider the moduli set $\{2^n, 2^{n+1}-1, 2^{n+1}+1\}$ where $n = 2$, i.e., $\{4, 7, 9\}$. This moduli set is a special case of the moduli set $\{2^\alpha, 2^\beta-1, 2^\beta+1\}$ for $\alpha = n$ and $\beta = n+1$. In this RNS, the residues of the weighted number $A = 86$ can be calculated by using (3), (5), and (6) as follows:

$$\begin{aligned} x_1 &= |A|_{2^\alpha} = |86|_4 = |(01010110)_2|_4 = (10)_2 = 2 \\ x_2 &= |A|_{2^\beta-1} = |86|_7 = |(01010110)_2|_7 \\ &= |(01)_2 + (010)_2 + (110)_2|_7 = |1+2+6|_7 = 2 \\ x_3 &= |A|_{2^\beta+1} = |86|_9 = |(01010110)_2|_9 \\ &= |(01)_2 - (010)_2 + (110)_2|_9 = |1-2+6|_9 = 5. \end{aligned}$$

Therefore, the residue representation of 86 is (2, 2, 5).

V. RESIDUE ARITHMETIC UNIT

RNS is suitable for computing applications that require only addition, subtraction, and multiplication. Other operations such as division, sign detection, and magnitude comparison are the

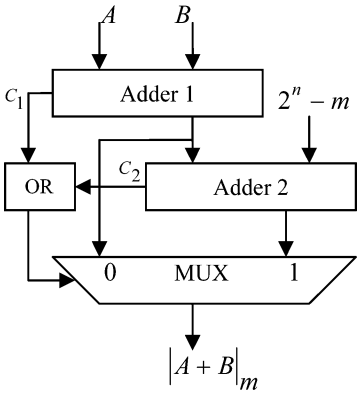

Fig. 2. A general modulus adder architecture.

difficult RNS operations. Hence, RNS is usually used in special-purpose processors (like DSP processors [43]) rather than in general purpose processors. The residue arithmetic unit of RNS consists of a modulus adder, subtractor, and multiplier for each modulus channel. In other words, each modulus of the moduli set has its own arithmetic processor composed of modulus arithmetic circuits for performing addition, subtraction, and multiplication.

A. Modulus Adder

Modulus addition is a fundamental operation in RNS, as modular adders are necessary building blocks for designing modulus multipliers [44], and also modulus subtraction can be done by using modular adders. Therefore, the modulus adder design must be investigated in detail.

The basic formula for performing the general modulus addition of the n-bit numbers A and B is

$$|A+B|_m = \begin{cases} A+B-m, & \text{if } A+B \geq m \\ A+B, & \text{otherwise.} \end{cases} \tag{7}$$

As shown in [45], the above equation can be rewritten as (8), shown at the bottom of the page. A simple hardware implementation of this equation is proposed in [46]. The realization relies on two binary adders and a multiplexer that is controlled by the carry outputs of the adders for selecting the correct sum, as shown in Fig. 2. This architecture can be parallelized to reduce the delay.

The general modular adder architectures can be specialized for the popular RNS moduli forms, i.e., $2^n, 2^n-1$ and 2^n+1. A modulo adder for 2^n is just a simple n-bit binary adder that ignores the carry output. Also, modulo 2^n-1 addition can be mechanized by an n-bit binary adder with end-around carry (EAC) [47]. However, modulus adders for 2^n+1 have more complex architectures due to the problem of $(n+1)$-bit circuits for modulo 2^n+1. Hence, for reducing complexity and improving efficiency, the diminished-one number system is used in the design of a modulo 2^n+1 adder.

The modulus adder structure based on simple binary adders suffers from high latency caused by the carry propagation chain in the ripple carry adders. Hence, in [48] parallel prefix adders are used for realizing modular addition. Later, high-speed parallel prefix modular adders for moduli 2^n-1 and 2^n+1 are introduced in [49] and [50], respectively. Also, an efficient parallel prefix adder architecture that is designed for performing general modular addition is proposed in [51]. In addition to these, the carry save diminished-one number system [52] was recently introduced and used together with parallel prefix adder structures for deriving a low-power and high-performance modulo 2^n+1 adder design.

B. Modulus Multiplication

Modular multipliers can be constructed based on different methods. There are, broadly speaking, two general ways of performing modulus multiplication [53]. The first one is based on multiplying two residues with a conventional binary multiplier, and then reducing the result with regard to the modulus. A realization of this method for the moduli set $\{2^n-1, 2^n, 2^n+1\}$ has been presented in [54]. The modulus 2^n multiplication can be simply performed using a binary multiplier, and then the least significant n bits are considered as the result. Based on the method shown in [54], the moduli $(2^n \pm 1)$ multipliers can be also designed using regular binary multipliers followed by reduction units that consist of modulus adders. As indicated in [53], the multiplication-then-reduction method needs a lot of space to store the product (which usually has a large value) for future use in the reduction unit. The second way to design modular multipliers is interleaving multiplication and reduction. The Montgomery algorithm [55] computes the modular multiplication in this way, being based on one reduction at each multiplication iteration. An RNS Montgomery modular multiplier for large dynamic range operands based on the mixed-radix approach is presented in [56]. There are also many specialized designs for performing moduli $(2^n \pm 1)$ multiplications since 2^n-1 and 2^n+1 are the popular RNS moduli. Wang *et al.* used a Wallace tree for designing efficient multipliers for moduli 2^n+1 [57] and 2^n-1 [58]. Also, in [48], moduli $(2^n \pm 1)$ multipliers based on the parallel-prefix concept and end-around carry adders are presented. A new design method for a modulo 2^n-1 multiplier based on modified Booth is described in [59]. In [60], a universal modified Booth modulo 2^n+1 multiplier based on bit-pair recoding is introduced. Also, in [61], an improved modulo 2^n+1 multipliers without using Booth recoding is proposed. Furthermore, the stored-unibit-transfer (SUT) redundant representation has been introduced and used to design a modulus 2^n+1 multiplier [62].

$$\begin{aligned} |A+B|_m &= \begin{cases} A+B-m+2^n-2^n, & \text{if } A+B+2^n-m \geq 2^n \\ A+B, & \text{otherwise} \end{cases} \\ &= \begin{cases} |A+B+2^n-m|_{2^n}, & \text{if } A+B+2^n-m \geq 2^n \\ A+B, & \text{otherwise} \end{cases} \end{aligned} \tag{8}$$

In addition to these, index multipliers [11], [63], and [64] have been considered to replace the multiplication modulus with a prime number, with the additional benefit of offering significant savings. As described in [11], the isomorphism between the multiplicative group $Q = \{1, 2, \ldots, m_l - 1\}$, with the multiplication modulus m_j, and the additive group $I = \{0, 1, \ldots, m_l - 2\}$, with the addition modulus $m_l - 1$, can be used to achieve the mapping $q = \Phi_l^{-1}(i) = |g_l^i|_{m_l}$, where $I = 1, 2, \ldots, L, q \in Q$ and $i \in I$. Then, the multiplication can be performed by

$$|q_1 q_2|_{m_l} = g^{|i_1+i_2|_{m_l-1}}. \tag{9}$$

Therefore, three steps are needed to perform multiplication [63]: first, finding the required indexes i's for each number; second, adding the modulus of the index $m_l - 1$; and third, doing the inverse index transformation. Note that lookup tables (LUT) can be used for implementation of the first and third steps. It should be mentioned that the modulus is restricted to being a prime number, which is why prime modules are used in the Quadratic Residue Number Systems (QRNS) for complex processing. The moduli sets used in QRNS are prime numbers in the form of $4\mathrm{k} + 1$ [70]. The only problem of QRNS is that choosing prime numbers of $4\mathrm{k} + 1$ form is considered as a restriction. Therefore, in [71], Generalized QRNS (GQRNS) was presented, in which there is no need for prime numbers in the form of $4\mathrm{k} + 1$, and any moduli set with pair-wise relativity prime can be used. In this case, the index multipliers can be used to perform the required modular multiplications. For example, an index arithmetic QRNS-based design of complex FIR filters is presented in [11].

VI. Reverse Converter

The reverse converter translates a residue represented number into its equivalent weighted number. It is an important part of the RNS system because the conversion delay should not counteract the speed gain of the RNS arithmetic unit. Furthermore, most algorithms for performing difficult RNS operations, such as division and magnitude comparison, are based on reverse conversion. Thus, an efficient design of the reverse converter could improve the overall performance of the RNS system and increase its applicability. Due to these considerations, most research on RNS is focused on designing efficient reverse converters. The moduli set and conversion algorithm both determine the complexity of the reverse converter. In other words, a judicious selection of the moduli set, with a dynamic range suitable for the application, and of a conversion algorithm compatible with the properties of the selected moduli set, could result in a high-performance hardware design for a reverse converter.

The algorithms of reverse conversion are principally based on the Chinese remainder theorem (CRT), mixed-radix conversion (MRC), and new Chinese remainder theorems (New CRTs). In addition to these, novel conversion algorithms have been proposed, which are not as general as CRT, MRC, or New CRTs. Some of the novel conversion algorithms that are dedicated for the moduli sets $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ and $\{r^n - 2, r^n - 1, r^n\}$ are presented in [65] and [66], respectively.

To design a reverse converter, the values of moduli of the moduli set must first be substituted in conversion algorithm formulas. Then, the resulting equations should be simplified by using arithmetic properties. Finally, simplified equations would be realized using hardware components such as full adders, half-adders, logic gates, or LUT. Now, consider the RNS number $(x_1, x_2, \ldots, x_n)$ with moduli set $(P_1, P_2, \ldots, P_n)$. In the following, the reverse conversion algorithms are investigated.

A. Chinese Remainder Theorem

By CRT [2], an RNS number can be converted into the weighted number X as follows:

$$X = \left| \sum_{i=1}^{n} |x_i N_i|_{P_i} M_i \right|_M \tag{10}$$

where $M = P_1 P_2 \ldots P_n, M_i = M/P_i$, and $N_i = |M_i^{-1}|_{P_i}$ is the multiplicative inverse of M_i modulus P_i. The CRT can be implemented in parallel channels followed by a modulus M adder. This modulus adder is very large and can result in inefficient hardware implementation of the reverse converter. Recently, in [20] and [31], the CRT is used for deriving reverse converters for the three-moduli sets $\{2^n - 1, 2^n, 2^{n+1} - 1\}$ and $\{2^n, 2^{2n} - 1, 2^{2n} + 1\}$, respectively.

B. Mixed-Radix Conversion

By MRC [2], [67], the number X can be calculated from residues by

$$X = v_n \prod_{i=1}^{n-1} P_i + \cdots + v_3 P_2 P_1 + v_2 P_1 + v_1. \tag{11}$$

The coefficients $\{v_1, v_2, \ldots, v_n\}$ can be obtained from residues by

$$v_1 = x_1 \tag{12}$$

$$v_2 = \left| (x_2 - v_1) \left| P_1^{-1} \right|_{P_2} \right|_{P_2} \tag{13}$$

$$v_3 = \left| ((x_3 - v_1) \left| P_1^{-1} \right|_{P_3} - v_2) \left| P_2^{-1} \right|_{P_3} \right|_{P_3}. \tag{14}$$

In the general case

$$v_n = \left| \left(\left((x_n - v_1) \left| P_1^{-1} \right|_{P_n} - v_2 \right) \times \left| P_2^{-1} \right|_{P_n} - \cdots - v_{n-1} \right) \left| P_{n-1}^{-1} \right|_{P_n} \right|_{P_n} \tag{15}$$

$|P_i^{-1}|_{P_j}$ denotes the multiplicative inverse of P_i modulus P_j. The MRC is a sequential algorithm that is not suitable for moduli sets with more than four moduli. However, for two-moduli sets, MRC leads to a simple conversion equation that can be efficiently realized in hardware. Hence, MRC usually is used in combinatorial conversion architectures. In these architectures, the moduli set is partitioned into two parts. Then, each part is separately converted by using other conversion algorithms. Finally, the two results are combined by using MRC. Such an implementation is employed in [22] and [27] for designing efficient reverse converters for the moduli sets $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1\}$, $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$, and $\{2^n - 1, 2^n, 2^n + 1, 2^{n-1} - 1, 2^{n+1} - 1\}$. Furthermore, to reduce the complexity of MRC, a two-level implementation of MRC is proposed in [29] to achieve a reverse converter for the moduli set $\{2^n - 1, 2^n, 2^{2n+1} - 1\}$.

C. New Chinese Remainder Theorem 1

The weighted number X can be computed by New CRT-I [23], [68] as follows:

$$X = x_1 + P_1 \times \left| k_1(x_2 - x_1) + k_2 P_2(x_3 - x_2) + \cdots + k_{n-1} P_2 P_3 \cdots P_{n-1}(x_n - x_{n-1}) \right|_{P_2 P_3 \cdots P_n} \quad (16)$$

where

$$|k_1 \times P_1|_{P_2 P_3 \cdots P_n} = 1 \quad (17)$$

$$|k_2 \times P_1 \times P_2|_{P_3 \cdots P_n} = 1 \quad (18)$$

$$|k_{n-1} \times P_1 \times P_2 \times \cdots \times P_{n-1}|_{P_n} = 1. \quad (19)$$

By New CRT-I, the size of the final modulo adder is reduced in comparison to the traditional CRT. In particular, if the first modulus of the moduli set is selected in the form 2^k, and the multiplication of other moduli is in the form $2^k - 1$, the New CRT-I can be implemented by only a multi-operand modulus adder. Such architectures are reported in [18], [23], [25], and [33].

D. New Chinese Remainder Theorem 2

The following algorithm finds the weighted number X based on New CRT-II where $P_1 < P_2 < \ldots < P_n$ [68], [69]:

Algorithm: translate $((x_1, x_2, \ldots, x_n), (P_1, \ldots, P_n), X)$

1) if $n > 2$, let $t = \lfloor n/2 \rfloor$, then
   translate $((x_1, \ldots, x_t), (P_1, \ldots, P_t), N_1)$,
   $M_1 = P_1 \ldots P_t$,
   translate $((x_{t+1}, \ldots, x_n), (P_{t+1}, \ldots, P_n), N_2)$,
   $M_2 = P_{t+1} \ldots P_n$,
   findno (N_1, N_2, M_1, M_2, X).
2) if $n = 2$, then *findno* (x_1, x_2, P_1, P_2, X).
3) if $n = 1$, then $X = |x_1|_{P_1}$.

Procedure: findno (x_1, x_2, P_1, P_2, X)

1) find a k_0 such that $|k_0 P_2|_{P_1} = 1$,
2) $X = x_2 + P_2 |k_0(x_1 - x_2)|_{P_1}$.

It can be seen that the general algorithm of New CRT-II has a tree-like architecture, and it reduces the final modulus adder size more than New CRT-I. Based on this algorithm, for the four-moduli set $\{P_1, P_2, P_3, P_4\}$, the number X can be calculated from its corresponding residues (x_1, x_2, x_3, x_4) by using the following equations [24]:

$$X = Z + P_1 P_2 |k_1(Y - Z)|_{P_3 P_4} \quad (20)$$

$$Z = x_1 + P_1 |k_2(x_2 - x_1)|_{P_2} \quad (21)$$

$$Y = x_3 + P_3 |k_3(x_4 - x_3)|_{P_4} \quad (22)$$

where

$$|k_1 P_1 P_2|_{P_3 P_4} = 1 \quad (23)$$

$$|k_2 P_1|_{P_2} = 1 \quad (24)$$

$$|k_3 P_3|_{P_4} = 1. \quad (25)$$

k_1, k_2, and k_3 are the multiplicative inverses. Recently, two adder-based implementations of New CRT-II have been proposed for the reverse converter of the four-moduli sets $\{2^n - 1, 2^n + 1, 2^{2n} - 2, 2^{2n+1} - 3\}$ [24] and $\{2^n - 1, 2^n, 2^n + 1, 2^{2n+1} - 1\}$ [25].

Example 3: In the RNS based on the moduli set $\{P_1, P_2, P_3\} = \{2, 3, 5\}$, by using MRC, CRT, New CRT-I, and New CRT-II, the residue represented number $(x_1, x_2, x_3) = (1, 2, 3)$ can be converted into its equivalent weighted decimal number as follows.

MRC conversion algorithm:

$|k_1 \times 2|_3 = 1 \Rightarrow k_1 = |2^{-1}|_3 = 2$
$|k_2 \times 2|_5 = 1 \Rightarrow k_2 = |2^{-1}|_5 = 3$
$|k_3 \times 3|_5 = 1 \Rightarrow k_3 = |3^{-1}|_5 = 2$
$v_1 = 1$
$v_2 = |(2 - 1) \times 2|_3 = 2$
$v_3 = |((3 - 1) \times 3 - 2) \times 2|_5 = 3$
$X = 3 \times 3 \times 2 + 2 \times 2 + 1 = 23$

CRT conversion algorithm:

$M = 2 \times 3 \times 5 = 30$
$M_1 = 30/2 = 15 \Rightarrow N_1 = |15^{-1}|_2 = 1$
$M_2 = 30/3 = 10 \Rightarrow N_2 = |10^{-1}|_3 = 1$
$M_3 = 30/5 = 6 \Rightarrow N_3 = |6^{-1}|_5 = 1$
$X = |1 \times 1 \times 15 + 2 \times 1 \times 10 + 3 \times 1 \times 6|_{30} = 23$

New CRT-I conversion algorithm:

$|k_1 \times 2|_{3 \times 5} = 1 \Rightarrow k_1 = 8$
$|k_2 \times 2 \times 3|_5 = 1 \Rightarrow k_2 = 1$
$X = 1 + 2 \times |8 \times (2 - 1) + 1 \times 3 \times (3 - 2)|_{15} = 23$

New CRT-II conversion algorithm:

$|k_1 \times 2 \times 3|_5 = 1 \Rightarrow k_1 = 1$
$|k_2 \times 2|_3 = 1 \Rightarrow k_2 = 2$
$Z = 1 + 2 \times |2 \times (2 - 1)|_3 = 5$
$Y = 3$
$X = 5 + 6 \times |1 \times (3 - 5)|_5 = 23$

VII. Assessment Results and Achievements

The proposed method has been taught to 15 Ph.D. students and 34 M.S. students as a part of *"Computer Arithmetic"* and *"Computational Aspects of VLSI Design"* courses from the second semester of 2007 to the first semester of 2009 at Shahid Beheshti University, Tehran, Iran, and the Science and Research branch of the Islamic Azad University (IAU), Tehran, Iran. The reaction of students to this method was very positive, and the achievements of those courses were significant, resulting in many research journal and conference papers. Also, one of the graduated Ph.D. students (an academic member of IAU) who participated in these courses has proposed a new course called *"Residue Number System"* as an advanced topic in computer architecture.

The students were asked to answer some questions about the effectiveness of this method to assess its value. The questionnaire consists of seven questions shown in Table II. The students were asked to grade the questions from 1 (disappointed) to 5 (satisfied). The average score of each question is shown in Table II. Question 1 elicits the amount of knowledge students had about RNS before taking these courses and illustrates that they were not very familiar with RNS. Question 2 shows the role of the instructor and his/her contribution to increase students' knowledge about RNS. Question 3 investigates if the method helped the students to come up with new ideas in the RNS field. Question 4 queried how applicable the students found RNS for practical applications. Question 5 asks if they were encouraged to participate in a team work activity. Question 6 asks if students are motivated to continue their research in this field. Finally, question 7 assesses the extent to which students would use this method if required to teach RNS in their future careers. The

TABLE II
QUESTIONNAIRE ANSWERED BY THE STUDENTS

Questions	Score
1- Have you had any previous knowledge of RNS?	1.72
2- How much did the instructor contribute to increasing your knowledge about RNS?	4.36
3- Has the method used in this class helped you to come up with new ideas in RNS?	4.4
4- How much do you find the method used applicable?	3.97
5- Does the method encourage you to participate in a team work activity?	3.7
6- Do you find RNS to be a suitable topic for an M.S. thesis or Ph.D. dissertation?	4.05
7- If you are supposed to teach RNS, would you use this method?	4.2
Total	3.77

average score for all questions is 3.77, which shows the degree of satisfaction of students taught RNS by this method.

VIII. CONCLUSION

One of the best ways to enhance the overall performance of the arithmetic units of applications based on addition, subtraction, and multiplication is to use RNS. In these kinds of applications, therefore, RNS considerations are inevitable. While some computer scientists and engineers are interested in learning the Residue Number System (RNS), existing textbooks do not completely cover the actual needs of RNS teaching. This paper presents a step-by-step RNS teaching method based on the state-of-the-art of RNS research. The learning process suggested in this paper is designed for a computer scientist or engineer in that their undergraduate knowledge is taken into account and a wide area of state-of-the-art work is investigated.

ACKNOWLEDGMENT

The authors would like to thank Dr. B. Yoberd for his literature contribution.

REFERENCES

[1] I. Koren, *Computer Arithmetic Algorithms*. Englewood Cliffs, NJ: Prentice-Hall, 1993.
[2] B. Parhami, *Computer Arithmetic: Algorithms and Hardware Design*. Oxford, U.K.: Oxford Univ. Press, 2000.
[3] F. J. Taylor, "Residue arithmetic: A tutorial with examples," *Computer*, vol. 17, pp. 50–62, May 1984.
[4] J. H. Ysng, C. C. Chang, and C. Y. Chen, "A high-speed division algorithm in residue number system using parity-checking technique," *Int. J. Comput. Math.*, vol. 81, no. 6, pp. 775–780, 2004.
[5] T. Tomczak, "Fast sign detection for RNS $(2^n-1, 2^n, 2^n+1)$," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 55, no. 6, pp. 1502–1511, Jun. 2008.
[6] L. Sousa, "Efficient method for magnitude comparison in RNS based on two pairs of conjugate moduli," in *Proc. 18th IEEE Int. Symp. Comput. Arithmetic*, 2007, pp. 240–250.
[7] T. Stouratitis and V. Paliouras, "Considering the alternatives in low-power design," *IEEE Circuits Devices Mag.*, vol. 7, no. 4, pp. 23–29, Jul. 2001.
[8] G. C. Cardarilli, A. Nannarelli, and M. Re, "Residue number system for low-power DSP applications," in *Proc. 41st IEEE Asilomar Conf. Signals, Syst., Comput.*, 2007, pp. 1412–1416.
[9] W. K. Jenkins and B. J. Leon, "The use of residue number systems in the design of finite impulse response digital filters," *IEEE Trans. Circuits Syst.*, vol. CAS-24, no. 4, pp. 191–201, Apr. 1977.
[10] R. Conway and J. Nelson, "Improved RNS FIR filter architectures," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 51, no. 1, pp. 26–28, Jan. 2004.
[11] J. Ramirez, U. Meyer-Base, and A. Garcia, "Efficient RNS-based design of programmable FIR filters targeting FPL technology," *J. Circuits, Syst. Comput.*, vol. 14, no. 1, pp. 165–177, 2005.
[12] W. Wei *et al.*, "RNS application for digital image processing," in *Proc. 4th IEEE Int. Workshop Syst.-on-Chip Real-Time Appl.*, 2004, pp. 77–80.
[13] A. Ammar, A. Al Kabbany, M. Youssef, and A. Emam, "A secure image coding using residue number systems," in *Proc. 18th Nat. Radio Sci. Conf.*, 2001, pp. 399–405.
[14] L. L. Yang and L. Hanzo, "Redundant residue number system based error correction codes," in *Proc. IEEE Veh. Technol. Conf.*, 2001, vol. 3, pp. 1472–1476.
[15] S. Timarchi and K. Navi, "Efficient class of redundant residue number system," in *Proc. IEEE Int. Symp. Intell. Signal Process.*, 2007, pp. 1–6.
[16] W. Wang, M. N. S. Swamy, and M. O. Ahmad, "Moduli selection in RNS for efficient VLSI implementation," in *Proc. IEEE Int. Symp. Circuits Syst.*, 2003, vol. 4, pp. IV-512–IV-515.
[17] M. Abdallah and A. Skavantzos, "A systematic approach for selecting practical moduli sets for residue number systems," in *Proc. 27th IEEE Int. Symp. Syst. Theory*, 1995, pp. 445–449.
[18] Y. Wang, X. Song, M. Aboulhamid, and H. Shen, "Adder based residue to binary numbers converters for $(2^n-1, 2^n, 2^n+1)$," *IEEE Trans. Signal Process.*, vol. 50, no. 7, pp. 1772–1779, Jul. 2002.
[19] W. Wang, M. N. S. Swamy, M. O. Ahmad, and Y. Wang, "A high-speed residue-to-binary converter and a scheme of its VLSI implementation," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 47, no. 12, pp. 1576–1581, Dec. 2000.
[20] P. V. A. Mohan, "RNS-to-binary converter for a new three-moduli set $\{2^{n+1}-1, 2^n, 2^n-1\}$," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 54, no. 9, pp. 775–779, Sep. 2007.
[21] P. V. A. Mohan, "New reverse converters for the moduli set $\{2^n-3, 2^n-1, 2^n+1, 2^n+3\}$," *J. Electron. Commun.*, vol. 62, no. 9, pp. 643–658, 2008.
[22] P. V. A. Mohan and A. B. Premkumar, "RNS-to-binary converters for two four-moduli set $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ and $\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 6, pp. 1245–1254, Jun. 2007.
[23] B. Cao, C. H. Chang, and T. Srikanthan, "An efficient reverse converter for the 4-moduli set $\{2^n-1, 2^n, 2^n+1, 2^{2n}+1\}$ based on the new chinese remainder theorem," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 50, no. 10, pp. 1296–1303, Oct. 2003.
[24] W. Zhang and P. Siy, "An efficient design of residue to binary converter for four moduli set $(2^n-1, 2^n+1, 2^{2n}-2, 2^{2n+1}-3)$ based on new CRT II," *J. Inf. Sci.*, vol. 178, no. 1, pp. 264–279, 2008.
[25] A. S. Molahosseini, K. Navi, C. Dadkhah, O. Kavehei, 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 Syst. I, Reg. Papers*, vol. 57, no. 4, pp. 823–835, Apr. 2010.
[26] A. A. Hiasat, "VLSI implementation of new arithmetic residue to binary decoders," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 13, no. 1, pp. 153–158, Jan. 2005.
[27] B. Cao, C. H. Chang, and T. Srikanthan, "A residue-to-binary converter for a new five-moduli set," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 5, pp. 1041–1049, May 2007.
[28] A. S. Molahosseini, C. Dadkhah, and K. Navi, "A new five-moduli set for efficient hardware implementation of the reverse converter," *IEICE Electron. Exp.*, vol. 6, no. 14, pp. 1006–1012, 2009.
[29] A. S. Molahosseini, K. Navi, and M. K. Rafsanjani, "A new residue to binary converter based on mixed-radix conversion," in *Proc. 3rd IEEE Int. Conf. Inf. Commun. Technol., From Theory Appl.*, 2008, pp. 1–6.
[30] W. Wang, M. N. S. Swamy, M. O. Ahmad, and Y. Wang, "A study of the residue-to-binary converters for the three-moduli sets," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 50, no. 2, pp. 235–243, Feb. 2003.
[31] A. Hariri, K. Navi, and R. Rastegar, "A new high dynamic range moduli set with efficient reverse converter," *J. Comput. Math. Appl.*, vol. 55, no. 4, pp. 660–668, 2008.
[32] A. S. Molahosseini, C. Dadkhah, K. Navi, and M. Eshghi, "Efficient MRC-based residue to binary converters for the new moduli sets $\{2^{2n}, 2^n-1, 2^{n+1}-1\}$ and $\{2^{2n}, 2^n-1, 2^{n-1}-1\}$," *IEICE Trans. Inf. Syst.*, vol. E92-D, no. 9, pp. 1628–1638, 2009.

[33] A. S. Molahosseini, K. Navi, O. Hashemipour, and A. Jalali, "An efficient architecture for designing reverse converters based on a general three-moduli set," *J. Syst. Archit.*, vol. 54, no. 10, pp. 929–934, 2008.
[34] K. W. Current, V. G. Oklobdzija, and D. Maksimovic, "Low-energy logic circuit techniques for multiple valued logic," in *Proc. IEEE Int. Symp. Multiple-Valued Logic*, 1996, pp. 86–90.
[35] E. Dubrova, "Multiple-valued logic in VLSI: Challenges and opportunities," in *Proc. NORCHIP Conf.*, 1999, pp. 340–350.
[36] M. A. Soderstrand and R. A. Escott, "VLSI implementation in multiple-valued logic of an FIR digital filter using residue number system arithmetic," *IEEE Trans. Circuits Syst.*, vol. CAS-33, no. 1, pp. 5–25, Jan. 1986.
[37] M. Abdallah and A. Skavantzos, "On multimoduli residue number systems with moduli of forms $r^a, r^b - 1, r^c + 1$," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 7, pp. 1253–1266, Jul. 2005.
[38] M. Hosseinzadeh, K. Navi, and S. Gorgin, "A new moduli set for residue number system: $\{r^n - 2, r^n - 1, r^n\}$," in *Proc. IEEE Int. Conf. Elect. Eng.*, 2007, pp. 1–6.
[39] M. Hosseinzadeh and K. Navi, "A new moduli set for residue number system in ternary valued logic," *J. Appl. Sci.*, vol. 7, pp. 3729–3735, 2007.
[40] A. S. Molahosseini, K. Navi, and M. K. Rafsanjani, "Efficient forward and reverse converters for a new high-radix moduli set," in *Proc. 3rd IEEE Int. Symp. Inf. Technol.*, 2008, pp. 1–4.
[41] B. Guan and E. V. Jones, "Fast conversion between binary and residue numbers," *Electron. Lett.*, vol. 24, no. 19, pp. 1195–1197, 1988.
[42] S. J. Piestrak, "Design of residue generators and multioperand modular adders using carry-save adders," *IEEE Trans. Comput.*, vol. 43, no. 1, pp. 68–77, Jan. 1994.
[43] R. Chaves and L. Sousa, "RDSP: A RISC DSP based on residue number system," in *Proc. Euromicro Symp. Digital Syst. Design: Archit., Methods, Tools*, 2003, pp. 128–135.
[44] A. Hiasat, "New efficient structure for a modular multiplier for RNS," *IEEE Trans. Comput.*, vol. 49, no. 2, pp. 170–174, Feb. 2000.
[45] H. T. Vergos and C. Efstathiou, "On the design of efficient modular adders," *J. Circuits, Syst., Comput.*, vol. 41, no. 5, pp. 965–972, 2005.
[46] M. A. Bayoumi, G. A. Jullien, and W. C. Miller, "A VLSI implementation of residue adder," *IEEE Trans. Circuits Syst.*, vol. CAS-34, no. 3, pp. 284–288, Mar. 1987.
[47] S. J. Piestrak, "A high speed realization of a residue to binary converter," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 42, no. 10, pp. 661–663, Oct. 1995.
[48] R. Zimmermann, "Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication," in *Proc. 14th IEEE Symp. Comput. Arithmetic*, 1999, pp. 158–167.
[49] L. Kalampoukas *et al.*, "High-speed parallel-prefix modulo $2^n - 1$ adders," *IEEE Trans. Comput.*, vol. 49, no. 7, pp. 673–679, Jul. 2000.
[50] C. Efstathiou, H. T. Vergos, and D. Nikolos, "Fast parallel-prefix modulo $2^n + 1$ adder," *IEEE Trans. Comput.*, vol. 53, no. 9, pp. 1211–1216, Sep. 2004.
[51] R. A. Patel, M. Benaissa, N. Powell, and S. Boussakta, "Novel power-delay-area-efficient approach to generic modular addition," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 54, no. 6, pp. 1279–1292, Jun. 2007.
[52] S. Timarchi, O. Kavehei, and K. Navi, "Low powe modulo $2^n + 1$ adder based on carry save diminshed-one number system," *Amer. J. Appl. Sci.*, vol. 5, no. 4, pp. 312–319, 2007.
[53] N. Nedjah and L. M. Mourelle, "A review of modular multiplication methods and respective hardware implementations," *Informatica*, vol. 30, pp. 111–129, 2006.
[54] A. A. Hiasat and H. S. Zohdy, "Design and implementation of a fast and compact residue-based semi-custom VLSI arithmetic chip," in *Proc. 37th Midwest Symp. Circuits Syst.*, 1994, pp. 428–431.
[55] P. L. Montgomery, "Modular multiplication without trial division," *Math. Comput.*, vol. 44, pp. 519–521, 1985.
[56] J. C. Bajard, L. S. Didier, and P. Kornerup, "An RNS montgomery modular multiplication algorithm," *IEEE Trans. Comput.*, vol. 47, no. 7, pp. 766–776, Jul. 1998.
[57] Z. Wang, G. A. Jullien, and W. C. Miller, "An efficient tree architecture for modulo $2^n + 1$ multiplication," *J. VLSI Signal Process.*, vol. 14, no. 3, pp. 241–248, 1996.
[58] Z. Wang, G. A. Jullien, and W. C. Miller, "An algorithm for multiplication modulo $(2^n - 1)$," in *Proc. IEEE Midwest Symp. Circuits Syst.*, 1996, pp. 1301–1304.
[59] C. Efstathiou, H. T. Vergos, and D. Nikolos, "Modified booth modulo $2^n + 1$ multipliers," *IEEE Trans. Comput.*, vol. 53, no. 3, pp. 370–374, Mar. 2004.
[60] L. Sousa and R. Chaves, "A universal architecture for designing efficient modulo $2^n + 1$ multipliers," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 6, pp. 1166–1178, Jun. 2005.
[61] H. T. Verogs and C. Efstathiou, "Design of efficient modulo $2^n + 1$ multipliers," *IET Comput. Digit. Tech.*, vol. 1, no. 1, pp. 49–57, 2007.
[62] S. Timarchi and K. Navi, "Arithmetic circuits of redundant SUT-RNS," *IEEE Trans. Instrum. Meas.*, vol. 58, no. 9, pp. 2959–2968, Sep. 2009.
[63] G. A. Jullien, "Implementation of multiplication, modulo a prime number, with applications to number theoretic transform," *IEEE Trans. Comput.*, vol. C-29, no. 10, pp. 899–905, Oct. 1980.
[64] D. Radhakrishnan and Y. Yuan, "Novel approaches to the design of the VLSI RNS multipliers," *IEEE Trans. Circuits Syst.*, vol. 39, no. 1, pp. 52–54, Jan. 1992.
[65] M. Hosseinzadeh, A. S. Molahosseini, and K. Navi, "An improved reverse converter for the moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$," *IEICE Electron. Exp.*, vol. 5, no. 17, pp. 672–677, 2008.
[66] M. Hosseinzadeh, A. S. Molahosseini, and K. Navi, "A fully parallel reverse converter," *Int. J. Elect., Comput., Syst. Eng.*, vol. 1, no. 3, pp. 183–187, 2007.
[67] B. Cao, C. H. Chang, and T. Srikanthan, "Adder based residue to binary converters for a new balanced 4-moduli set," in *Proc. 3rd IEEE Symp. Image, Signal Process. Anal.*, 2003, vol. 2, pp. 820–825.
[68] Y. Wang, "Residue-to-binary converters based on new Chinese remainder theorems," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 47, no. 2, pp. 197–205, Feb. 2000.
[69] Y. Wang, X. Song, M. Aboulhamid, and H. Shen, "A new algorithm for RNS magnitude comparison based on new chinese remainder theorem II," in *Proc. 9th Great Lakes Symp. VLSI*, 1999, pp. 362–365.
[70] S. H. Leung, "Application of residue number systems to complex digital filters," in *Proc. IEEE Asilmar Conf. Circuits, Syst., Comput.*, Pacific Grove, CA, Nov. 1981, pp. 70–74.
[71] M. A. Soderstrand, T. G. Johnson, and G. A. Clark, "Use of generalized quadratic RNS arithmetic in adaptive filters," in *Proc. IEEE Asilomar Conf. Circuits, Syst., Comput.*, Pacific Grove, CA, Nov. 1985, pp. 102–106.
[72] J. H. McClellan and C. M. Rader, *Number Theory in Digital Signal Processing*. Englewood Cliffs, NJ: Prentice-Hall, 1979.

Keivan Navi received the M.Sc. degree in electrical engineering (computer hardware) from Sharif University of Technology, Tehran, Iran, in 1990, and the Ph.D. degree in computer architecture from Paris XI University, Paris, France, in 1995.

He is currently an Associate Professor with the Faculty of Electrical and Computer Engineering, Shahid Beheshti University, Tehran, Iran. His research interests include the Residue Number System, carbon nanotubes, single-electron transistors, reversible logic design, interconnection networks, and quantum computing.

Amir Sabbagh Molahosseini was born in Kerman, Iran, in 1983. He received the B.Sc. degree from Shahid Bahonar University of Kerman, Kerman, Iran, in 2005, and the M.Sc. degree (with highest honors) from Islamic Azad University (IAU), Science and Research Branch, Tehran, Iran, in 2007, both in computer engineering. He is currently working toward the Ph.D. degree in computer engineering at the Science and Research Branch of IAU. His research interests include VLSI design and computer arithmetic with emphasis on residue number system.

Mohammad Esmaeildoust received the B.Sc. degree in hardware engineering from Shahed University, Tehran, Iran, in 2006, and the M.Sc. degree in computer architecture from Shahid Beheshti University of Technology, Tehran, Iran, in 2008. He is currently a Ph.D. candidate in computer architecture at Shahid Beheshti University of Technology. He is working on reconfigurable computing and computer arithmetic, especially in the area of the residue number system.