PAPER

Efficient MRC-Based Residue to Binary Converters for the New Moduli Sets $\{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$ and $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$

Amir Sabbagh MOLAHOSSEINI^{†a)}, Chitra DADKHAH^{††b)}, Keivan NAVI^{†††c)}, Nonmembers,
and Mohammad ESHGHI^{†††d)}, Member

SUMMARY In this paper, the new residue number system (RNS) moduli sets $\{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$ and $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$ are introduced. These moduli sets have $4n$ -bit dynamic range and well-formed moduli which can result in high-performance residue to binary converters as well as efficient RNS arithmetic unit. Next, efficient residue to binary converters for the proposed moduli sets based on mixed-radix conversion (MRC) algorithm are presented. The converters are ROM-free and they are realized using carry-save adders and modulo adders. Comparison with the other residue to binary converters for $4n$ -bit dynamic range moduli sets shown that the presented designs based on new moduli sets $\{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$ and $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$ are improved the conversion delay and result in hardware savings. Also, the proposed moduli sets can lead to efficient binary to residue converters, and they can speed-up internal RNS arithmetic processing, compared with the other $4n$ -bit dynamic range moduli sets.

key words: residue to binary converter, mixed-radix conversion (MRC), residue number system (RNS), computer arithmetic

1. Introduction

The residue number system (RNS) is a carry-free number system, which has been used as an alternative to the weighted number system for achieving high-performance computing systems [1]. In RNS, the weighted operands are converted into residue representations. 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. [2], [3]. The RNS has many applications in digital signal processing (DSP) [4], [5]. In particular, RNS is an interesting and useful tool for the implementation of high-speed FIR filters [6], [7]. Also, implementation issues of some RNS-based DSP processors are reported in [8] and [9]. RNS has applications in image processing systems, especially RNS image coding can offers high-speed VLSI implementation of secure image processing algorithms [10]. In addition to these, redundant RNS offers new approaches to the design of the error detection and correction codes [11], [12].

Manuscript received December 1, 2008.

[†]The author is with Science and Research Branch, Islamic Azad University, Tehran, Iran.

^{††}The author is with the Department of Electrical Engineering, Khaje Nasir Toosi University of Technology, Tehran, Iran.

^{†††}The authors are with the Department of Electrical and Computer Engineering, Shahid Beheshti University, GC. Tehran, Iran.

a) E-mail: amir.sabbagh@sr.iau.ac.ir

b) E-mail: dadkha@eed.kntu.ac.ir

c) E-mail: navi@sbu.ac.ir

d) E-mail: m-eshghi@sbu.ac.ir

DOI: 10.1587/transinf.E92.D.1628

The dynamic range (DR) of an RNS system is defined in terms of the product of the moduli of the moduli set, and it denotes the interval of integers can be uniquely represented in RNS. The proper selection of a moduli set has an important role in the design of the RNS systems, because the speed of internal RNS arithmetic circuits as well as the complexity of residue to binary converter depend on the form and the number of the moduli [2]. The powers of two related numbers such as 2^k and $2^k \pm 1$, have been extensively used in RNS moduli sets. Because, RNS systems based on these moduli can be efficiently implemented by using usual binary hardwares. Specially, the modulo adders and multipliers for the moduli of the form 2^k are just the binary adders and multipliers with truncation [13]. Also, operations in modulo $2^k - 1$ are regular and can be easily performed by end-around carry scheme. But modulo $2^k + 1$ operations are much more complex and are usually the bottleneck for RNS arithmetic units [14].

The moduli sets which have been suggested for RNS, can be classified based on their DR. The $3n$ -bit DR moduli sets are $\{2^n - 1, 2^n, 2^n + 1\}$ [15], $\{2^{n-1} - 1, 2^n - 1, 2^n\}$ [16], and $\{2^n - 1, 2^n, 2^{n+1} - 1\}$ [17], [18]. The DR provided by these moduli sets is not sufficient for applications which require larger dynamic range. Hence, $4n$ -bit DR four-moduli sets such as $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1\}$ [19], [20], $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ [20]–[22] and $\{2^n - 3, 2^n + 1, 2^n - 1, 2^n + 3\}$ [23] have been introduced. These four-moduli sets include balanced moduli, but some of their multiplicative inverses have inelegant forms which resulted in increasing the cost and the delay of the residue to binary converter. Furthermore, with increasing the number of moduli in the moduli set, the complexity of RNS is also increase. So, the RNS systems based on four-moduli sets are more complex than those based on three-moduli sets. Hence, three-moduli sets with supporting larger DR than the traditional three-moduli sets such as $\{2^n - 1, 2^n, 2^{2n+1} - 1\}$ [24], $\{2^n - 1, 2^n + 1, 2^{2n} + 1\}$ [25] and $\{2^n, 2^{2n} - 1, 2^{2n} + 1\}$ [26] have been proposed. The first and second moduli sets have $4n$ -bit DR, while the third has $5n$ -bit DR. Due to the mathematical properties of the moduli sets $\{2^n - 1, 2^n + 1, 2^{2n} + 1\}$ and $\{2^n, 2^{2n} - 1, 2^{2n} + 1\}$, they have simple multiplicative inverses for residue to binary converter design, but the modulo $2^{2n} + 1$ leads to increasing the execution latency of the RNS arithmetic units. Recently, in [27] the general three-moduli set $\{2^\alpha, 2^\beta - 1, 2^\beta + 1\}$ where $\alpha < \beta$ introduced for offering large dynamic range with low complexity, and an adder-based residue to binary converter for

this moduli set is presented in [27]. The residue to binary converter architecture of [27] is very simple and efficient but the time performance of arithmetic unit of RNS systems based on its moduli set is restricted to the modulo $2^\beta + 1$. Also, with the general three-moduli set $\{2^\alpha, 2^\beta - 1, 2^\beta + 1\}$ where $\alpha < \beta$, we can not derive a $4n$ -bit DR moduli set.

In this paper, after describing necessary background in Sect. 2, we concentrate on $4n$ -bit DR moduli sets, and introduce two new $4n$ -bit DR RNS moduli sets $\{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$ and $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$ where $n \geq 4$. All of the moduli of the proposed moduli sets are in the forms 2^k or $2^k - 1$, and this leads to simple and efficient implementation of the binary to residue and residue to binary converters as well as internal RNS arithmetic circuits. Next, efficient designs of residue to binary converter for the moduli sets $\{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$ and $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$ based on mixed-radix conversion (MRC) algorithm are presented in Sect. 3. The converter architectures are adder-based and pipelinable, resulting in high-performance hardware designs. The proposed residue to binary converters have lower hardware costs and better conversion delays than the other residue to binary converters for $4n$ -bit DR moduli sets, and also the new introduced moduli sets are result in efficient binary to residue converters and they can speed-up internal RNS arithmetic processing than the other $4n$ -bit DR moduli sets as shown in Sect. 4.

2. Background

A residue number system is defined in terms of relatively-prime moduli set $\{P_1, P_2, \dots, P_n\}$ that is $\gcd(P_i, P_j) = 1$ for $i \neq j$. A weighted number X can be represented as $X = (x_1, x_2, \dots, x_n)$, where

$$x_i = X \bmod P_i = |X|_{P_i}, \quad 0 \leq x_i < P_i \quad (1)$$

Such a representation is unique for any integer X in the range $[0, M - 1]$, where $M = P_1 P_2 \dots P_n$ is the dynamic range of the moduli set $\{P_1, P_2, \dots, P_n\}$ [28]. Addition, subtraction and multiplication on residues can be performed in parallel, because there is no carry propagation between residue digits in RNS.

Each RNS-based processing system consists of a binary to residue converter, an arithmetic unit and a residue to binary converter. The binary to residue converter encode a weighted binary number into a residue represented number, with regard to the moduli set, and can be implemented by using multi-operand modular adders. The RNS arithmetic unit includes modular adder, subtractor and multiplier for each modulo channel. The residue to binary converter decode a residue represented number into its equivalent weighted binary number. The residue to binary converter is very important part of the RNS systems. Because, conversion delay should not counteract the speed gain of RNS arithmetic unit. Besides, the residue to binary conversion can be used for performing difficult RNS operations such as division, sign detection and magnitude comparison [2]. The algorithms of residue to binary conversion

are principally based on Chinese remainder theorem (CRT) and mixed-radix conversion. Now, consider the RNS number $(x_1, x_2, \dots, x_n)$ with moduli set $(P_1, P_2, \dots, P_n)$. In the following, the residue to binary conversion algorithms are described.

By CRT [2], the number X can be calculated from its residue representation by

$$X = \left| \sum_{i=1}^m |x_i N_i|_{P_i} M_i \right|_M \quad (2)$$

where

$$M = P_1 P_2 \dots P_M \quad (3)$$

$$M_i = M / P_i \quad (4)$$

$$N_i = |M_i^{-1}|_{P_i} \quad (5)$$

The N_i is the multiplicative inverse of M_i modulo P_i . The CRT can be implemented in parallel channels followed by a modulo M adder. The problem is that, this modulo adder is very big, and it can results in inefficient hardware implementation for residue to binary converter.

By MRC [2], [29], an RNS number can be converted into its equivalent weighted binary number X as follows

$$X = v_n \prod_{i=1}^{n-1} P_i + \dots + v_3 P_2 P_1 + v_2 P_1 + v_1 \quad (6)$$

The coefficients v_i s can be obtained from residues by

$$v_1 = x_1 \quad (7)$$

$$v_2 = \left| (x_2 - v_1) |P_1^{-1}|_{P_2} \right|_{P_2} \quad (8)$$

$$v_3 = \left| \left((x_3 - v_1) |P_1^{-1}|_{P_3} - v_2 \right) |P_2^{-1}|_{P_3} \right|_{P_3} \quad (9)$$

In the general case, we have

$$v_n = \left| \left(\left((x_n - v_1) |P_1^{-1}|_{P_n} - v_2 \right) |P_2^{-1}|_{P_n} - \dots - v_{n-1} \right) |P_{n-1}^{-1}|_{P_n} \right|_{P_n} \quad (10)$$

where $|P_i^{-1}|_{P_j}$ denotes the multiplicative inverse of P_i modulo P_j . MRC is a sequential algorithm which is not suitable for moduli sets with more than four moduli. But, for two and three-moduli sets with well-formed moduli, MRC can leads to simple residue to binary conversion equations which can be efficiently realized in hardware.

3. Residue to Binary Converters

In this section, MRC is used to derive efficient residue to binary conversion algorithms for the moduli sets $\{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$ and $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$. Also, adder-based hardware implementations of the conversion algorithms are presented.

3.1 Converter for the Moduli Set $\{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$

For design of a residue to binary converter, firstly the values of moduli of the moduli set must substitute in conversion algorithm formulas. Then, the resulted equations should be simplified by using arithmetic properties. Finally, simplified equations would realize using hardware components such as full adders, half adders and logic gates. The following Theorems, Lemmas and Properties are needed for derivation of residue to binary conversion algorithm.

Theorem 1: The moduli set $\{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$ includes pairwise relatively prime moduli.

Proof: Since $2^n - 1$ and $2^{n+1} - 1$ are two odd numbers, and 2^{2n} is an even number, it is clear that 2^{2n} is relatively prime to the numbers $2^n - 1$ and $2^{n+1} - 1$. Also, the moduli $2^n - 1$ and $2^{n+1} - 1$ are relatively prime, because they are previously used in the RNS moduli set $\{2^n - 1, 2^n, 2^{n+1} - 1\}$ [17].

Lemma 1: The multiplicative inverse of 2^{2n} modulo $(2^n - 1)$ is $k_1 = 1$.

Proof: It is easy to find that,

$$|2^n|_{2^n-1} = 1 \quad (11)$$

Therefore,

$$|k_1 \times 2^{2n}|_{2^n-1} = |1 \times 2^n \times 2^n|_{2^n-1} = |1 \times 1 \times 1|_{2^n-1} = 1 \quad (12)$$

Lemma 2: The multiplicative inverse of 2^{2n} modulo $(2^{n+1} - 1)$ is $k_2 = 2^2$.

Proof: By substituting the value of k_2 , we have

$$\begin{aligned} |k_2 \times 2^{2n}|_{2^{n+1}-1} &= |2^2 \times 2^{2n}|_{2^{n+1}-1} = |2^{2n+2}|_{2^{n+1}-1} \\ &= |2^{n+1} \times 2^{n+1}|_{2^{n+1}-1} = |1 \times 1|_{2^{n+1}-1} = 1 \end{aligned} \quad (13)$$

Lemma 3: The multiplicative inverse of $(2^n - 1)$ modulo $(2^{n+1} - 1)$ is $k_3 = -2$.

Proof: It is clear that,

$$\begin{aligned} |k_3 \times (2^n - 1)|_{2^{n+1}-1} &= |-2 \times (2^n - 1)|_{2^{n+1}-1} \\ &= |-2^{n+1} + 2|_{2^{n+1}-1} = |-1 + 2|_{2^{n+1}-1} = 1 \end{aligned} \quad (14)$$

Theorem 2: In the RNS based on the moduli set $\{P_1, P_2, P_3\} = \{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$, the number X can be calculated from its corresponding residues (x_1, x_2, x_3) by

$$X = x_1 + 2^{2n}Z + 2^{2n}(2^n - 1)T \quad (15)$$

where

$$Z = |x_2 - x_1|_{2^n-1} \quad (16)$$

$$T = |((x_3 - x_1)2^2 - Z)(-2)|_{2^{n+1}-1} \quad (17)$$

Proof: By letting $P_1 = 2^{2n}$, $P_2 = 2^n - 1$, $P_3 = 2^{n+1} - 1$ and the values of multiplicative inverses from Lemmas 1, 2 and 3 into (6)–(9), Eqs. (15)–(17) are obtained.

Theorem 2 is used for designing an efficient residue to binary converter. But before implementing it, according to the following properties, (15)–(17) can be simplified for deriving an efficient hardware design.

Property 1: The residue of a negative residue number $(-v)$ in modulo $(2^n - 1)$ is the one's complement of v , where $0 \leq v < 2^n - 1$ [26].

Property 2: The multiplication of a residue number v by 2^P in modulo $(2^n - 1)$ is carried out by P bit circular left shift, where P is a natural number [26].

Consider the three-moduli set $\{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$ and let the corresponding residues of the integer number X be (x_1, x_2, x_3) . The residues have bit-level representation as

$$x_1 = (\underbrace{x_{1,2n-1}x_{1,2n-2} \cdots x_{1,1}x_{1,0}}_{2n \text{ bits}})_2 \quad (18)$$

$$x_2 = (\underbrace{x_{2,n-1}x_{2,n-2} \cdots x_{2,1}x_{2,0}}_{n \text{ bits}})_2 \quad (19)$$

$$x_3 = (\underbrace{x_{3,n}x_{3,n-1} \cdots x_{3,1}x_{3,0}}_{n+1 \text{ bits}})_2 \quad (20)$$

Equation (15) can rewrite as

$$X = x_1 + 2^{2n}H \quad (21)$$

where

$$H = Z + (2^n - 1)T \quad (22)$$

$$Z = |x_2 - x_1|_{2^n-1} \quad (23)$$

$$T = |((x_3 - x_1)2^2 - Z)(-2)|_{2^{n+1}-1} \quad (24)$$

Firstly, (23) is simplified as follow

$$Z = |w_1 + w_2|_{2^n-1} \quad (25)$$

where

$$w_1 = |x_2|_{2^n-1} = \left| \underbrace{x_{2,n-1} \cdots x_{2,1}x_{2,0}}_n \right|_{2^n-1} = \underbrace{x_{2,n-1} \cdots x_{2,1}x_{2,0}}_n \quad (26)$$

$$\begin{aligned} w_2 &= |-x_1|_{2^n-1} = \left| -(\underbrace{x_{1,2n-1} \cdots x_{1,1}x_{1,0}}_{2n}) \right|_{2^n-1} \\ &= \left| -(\underbrace{x_{1,2n-1} \cdots x_{1,n+1}x_{1,n}}_n \times 2^n + \underbrace{x_{1,n-1} \cdots x_{1,1}x_{1,0}}_n) \right|_{2^n-1} \end{aligned} \quad (27)$$

Now, (27) can be divided in two parts as below

$$w_{21} = \left| -(\underbrace{x_{1,n-1} \cdots x_{1,1}x_{1,0}}_n) \right|_{2^n-1} = \underbrace{\bar{x}_{1,n-1} \cdots \bar{x}_{1,1}\bar{x}_{1,0}}_n \quad (28)$$

$$\begin{aligned} w_{22} &= \left| -2^n(\underbrace{x_{1,2n-1} \cdots x_{1,n+1}x_{1,n}}_n) \right|_{2^n-1} \\ &= \underbrace{\bar{x}_{1,2n-1} \cdots \bar{x}_{1,n+1}\bar{x}_{1,n}}_n \end{aligned} \quad (29)$$

Therefore, (25) become as follows

$$Z = |w_1 + w_{21} + w_{22}|_{2^n-1} \quad (30)$$

Next, we simplify (24) by rewriting it as

$$T = |2Z - 2^3(x_3 - x_1)|_{2^{n+1}-1} \quad (31)$$

Equation (31) can be separated into four parts as

$$\begin{aligned} w_3 &= |2Z|_{2^{n+1}-1} = \left| 2(0 \underbrace{Z_{n-1} \cdots Z_1 Z_0}_n) \right|_{2^{n+1}-1} \\ &= \underbrace{Z_{n-1} \cdots Z_1 Z_0}_n 0 \end{aligned} \quad (32)$$

$$\begin{aligned} w_4 &= |-2^3 x_3|_{2^{n+1}-1} = \left| -2^3 (\underbrace{x_{3,n} \cdots x_{3,1} x_{3,0}}_{n+1}) \right|_{2^{n+1}-1} \\ &= \underbrace{\bar{x}_{3,n-3} \cdots \bar{x}_{3,1} \bar{x}_{3,0}}_{n-2} \bar{x}_{3,n} \bar{x}_{3,n-1} \bar{x}_{3,n-2} \end{aligned} \quad (33)$$

$$\begin{aligned} w_5 &= |2^3 x_1|_{2^{n+1}-1} = \left| 2^3 (\underbrace{x_{1,2n-1} \cdots x_{1,1} x_{1,0}}_{2n}) \right|_{2^{n+1}-1} \\ &= \left| 2^3 (\underbrace{x_{1,2n-1} \cdots x_{1,n+2} x_{1,n+1}}_{n-1} \times 2^{n+1} + \underbrace{x_{1,n} \cdots x_{1,1} x_{1,0}}_{n+1}) \right|_{2^{n+1}-1} \end{aligned} \quad (34)$$

The above equation can be parsing as follow

$$\begin{aligned} w_{51} &= \left| 2^3 (\underbrace{x_{1,n} \cdots x_{1,1} x_{1,0}}_{n+1}) \right|_{2^{n+1}-1} \\ &= \underbrace{x_{1,n-3} \cdots x_{1,1} x_{1,0}}_{n-2} x_{1,n} x_{1,n-1} x_{1,n-2} \end{aligned} \quad (35)$$

$$\begin{aligned} w_{52} &= \left| 2^3 \times 2^{n+1} (\underbrace{00 x_{1,2n-1} \cdots x_{1,n+2} x_{1,n+1}}_{n-1}) \right|_{2^{n+1}-1} \\ &= \underbrace{x_{1,2n-2} \cdots x_{1,n+2} x_{1,n+1}}_{n-2} 00 x_{1,2n-1} \end{aligned} \quad (36)$$

Therefore, (31) can be calculated by

$$T = |w_3 + w_4 + w_{51} + w_{52}|_{2^{n+1}-1} \quad (37)$$

Finally, H in (22) is obtained as

$$H = P - Q \quad (38)$$

where

$$P = Z + 2^n T = \underbrace{T_n \cdots T_1 T_0}_{n+1} \underbrace{Z_{n-1} \cdots Z_1 Z_0}_n \quad (39)$$

$$Q = T = \underbrace{0 \cdots 00}_n \underbrace{T_n \cdots T_1 T_0}_{n+1} \quad (40)$$

Example 1: Consider the moduli set $\{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$ where $n = 4$. The weighted number X can be calculated from its RNS representation (97, 12, 29) as follow
For $n = 4$ the moduli set is $\{256, 15, 31\}$ and also residues have binary representation as

$$x_1 = 97 = (01100001)_2$$

$$x_2 = 10 = (1100)_2$$

$$x_3 = 29 = (11101)_2$$

By letting values of residues and $n = 4$ in (26), (28), (29) and (30), we have

$$w_1 = (1100)_2 = 12$$

$$w_{21} = (1001)_2 = 9$$

$$w_{22} = (1110)_2 = 14$$

$$Z = |12 + 9 + 14|_{15} = 5 = (0101)_2$$

Next, From (32), (33), (35), (36) and (37), we have

$$w_3 = (01010)_2 = 10$$

$$w_4 = (10000)_2 = 16$$

$$w_{51} = (11000)_2 = 24$$

$$w_{52} = (01000)_2 = 8$$

$$T = |10 + 16 + 24 + 8|_{31} = 27 = (11011)_2$$

Finally, by substituting the required values in (39), (40), (38) and (21), we have

$$P = (\underbrace{11011}_5 \underbrace{0101}_4)_2 = 437$$

$$Q = (\underbrace{0000}_4 \underbrace{11011}_5)_2 = 27$$

$$H = 437 - 27 = 410 = (110011010)_2$$

$$X = 97 + 2^8 \times 410 = (\underbrace{110011010}_9 \underbrace{01100001}_8)_2 = 105057$$

To verify the result, we calculate the residues as follow

$$x_1 = |105057|_{256} = 97$$

$$x_2 = |105057|_{15} = 12$$

$$x_3 = |105057|_{31} = 29$$

Therefore, the weighted integer number 105057 in the RNS based on the moduli set $\{256, 15, 31\}$ has representation as (97, 12, 29).

Hardware architecture of the residue to binary converter for the moduli set $\{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$ is shown in Fig. 1. Implementation is based on Eqs. (30), (37), (38) and (21). Firstly the operand preparation unit (OPU) 1 prepares the needed operands (26), (28) and (29) by using simply manipulating the routing of the bits of the residues, and some NOT gates. $2n$ NOT gates are used for performing the inversions of (28) and (29). Implementation of (30) requires a 3-operand modulo $(2^n - 1)$ adder [30] which consists of an n -bit carry-save adder (CSA) with end-around carry (EAC) followed by a modulo $(2^n - 1)$ adder. The modulo $(2^n - 1)$ adder can be implemented with different methods [30]–[33]. In this paper, we considered the carry propagate adder (CPA) with EAC for the implementation of the modulo adders of the residue to binary converters. Therefore, modulo $(2^n - 1)$ adder relies on an n -bit CPA with EAC [30]. The delay of an CSA with EAC is the same as that of a full adder (FA), and the delay of an CPA with EAC is twice of the delay of a regular CPA, while has the same hardware complexity.

Realization of (37) relies on a 4-operand modulo

Fig. 1 The converter for moduli set $\{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$.

$(2^{n+1} - 1)$ adder [31], and it is implemented by two $(n+1)$ -bit CSA's with EAC followed by a $(n+1)$ -bit CPA with EAC for performing the modulo $(2^{n+1} - 1)$ addition. The OPU 2 requires $(n+1)$ NOT gates for performing the inversions of (33). Also, since (32) and (36) have one and two bits of zeros, three FA's in CSA2 and CSA3 are reduced to one and two pairs of XOR/AND gates, respectively. Finally, implementation of (38) needs a $(2n+1)$ -bit subtractor. This

Table 1 Characterization of each part of the proposed residue to binary converter for moduli set $\{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$.

Parts	FA	NOT	XOR /AND pairs	XNOR /OR pairs	Delay
OPU1	—	$2n$	—	—	t_{NOT}
CSA1	n	—	—	—	t_{FA}
CPA1	n	—	—	—	$2nt_{\text{FA}}$
OPU2	—	$n+1$	—	—	t_{NOT}
CSA2	n	—	1	—	t_{FA}
CSA3	$n-1$	—	2	—	t_{FA}
CPA2	$n+1$	—	—	—	$(2n+2)t_{\text{FA}}$
OPU3	—	$n+1$	—	—	t_{NOT}
CPA3	$n+1$	—	—	n	$(2n+1)t_{\text{FA}}$
Total	$6n+1$	$4n+2$	3	n	$(6n+6)t_{\text{FA}} + 3t_{\text{NOT}}$

subtractor can be implemented by a $(2n+1)$ -bit regular CPA with '1' carry-in, and $(n+1)$ NOT gates. Also, since the inversion of Q (40) has n bits of 1's, n FA's of the CPA3 are reduced to n pairs of XNOR/OR gates. It should be noted that the carry-out of CPA3 will be ignored. Also, since x_1 is a $2n$ -bit number, no computational hardware is needed to compute $x_1 + 2^{2n}H$ in (21). The desired result is obtained by concatenating x_1 with H . Details of area and delay, of each part of the converter are shown in Table 1.

3.2 Converter for the Moduli Set $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$

By using a similar derivation like before, the residue to binary converter for the moduli set $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$ is obtained as follow.

Theorem 3: The moduli set $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$ consists of pairwise relatively prime moduli.

Proof: Similar to the proof of theorem 1, it can be seen that 2^{2n} is relatively prime to the numbers $2^n - 1$ and $2^{n-1} - 1$. Also, since the moduli $2^n - 1$ and $2^{n-1} - 1$ previously used together in the RNS moduli set $\{2^{n-1} - 1, 2^n - 1, 2^n\}$ [16], they are relatively prime.

Lemma 4: The multiplicative inverse of 2^{2n} modulo $(2^{n-1} - 1)$ is $k_4 = 2^{n-3}$.

Proof: It is obvious that,

$$\begin{aligned} |k_4 \times 2^{2n}|_{2^{n-1}-1} &= |2^{n-3} \times 2^{2n}|_{2^{n-1}-1} = |2^{3n-3}|_{2^{n-1}-1} \\ &= |2^{n-1} \times 2^{n-1} \times 2^{n-1}|_{2^{n-1}-1} = |1 \times 1 \times 1|_{2^{n-1}-1} = 1 \end{aligned} \quad (41)$$

Lemma 5: The multiplicative inverse of $(2^n - 1)$ modulo $(2^{n-1} - 1)$ is $k_5 = 1$.

Proof: First, it can be seen that,

$$|2^n|_{2^{n-1}-1} = |2 \times 2^{n-1}|_{2^{n-1}-1} = 2 \quad (42)$$

Now, we have

$$\begin{aligned} |k_5 \times (2^n - 1)|_{2^{n-1}-1} &= |1 \times (2^n - 1)|_{2^{n-1}-1} \\ &= |1 \times (2 - 1)|_{2^{n-1}-1} = 1 \end{aligned} \quad (43)$$

Theorem 4: In the RNS based on the moduli set $\{P_1, P_2, P_3\} = \{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$, the number X can be

calculated from its corresponding residues (x_1, x_2, x_3) by

$$X = x_1 + 2^{2n}Z + 2^{2n}(2^n - 1)T \quad (44)$$

where

$$Z = |x_2 - x_1|_{2^{n-1}} \quad (45)$$

$$T = |(x_3 - x_1)2^{n-3} - Z|_{2^{n-1-1}} \quad (46)$$

Proof: By substituting $P_1 = 2^{2n}$, $P_2 = 2^n - 1$, $P_3 = 2^{n-1} - 1$ and the values of multiplicative inverses from Lemmas 1, 4 and 5 into (6)–(9), Eqs. (44)–(46) are obtained.

Theorem 4 can be simplified by employing Properties 1 and 2, for reducing hardware complexity. The steps for simplifying (45) are same with those of Theorem 2, and hence we only note mathematical equations without any description.

$$X = x_1 + 2^{2n}H \quad (47)$$

$$H = Z + (2^n - 1)T \quad (48)$$

$$Z = |w_1 + w_{21} + w_{22}|_{2^{n-1}} \quad (49)$$

where

$$w_1 = |x_2|_{2^{n-1}} = \left| \underbrace{x_{2,n-1} \cdots x_{2,1} x_{2,0}}_n \right|_{2^{n-1}} = \underbrace{x_{2,n-1} \cdots x_{2,1} x_{2,0}}_n \quad (50)$$

$$w_{21} = \left| -\underbrace{(x_{1,n-1} \cdots x_{1,1} x_{1,0})}_n \right|_{2^{n-1}} = \underbrace{\bar{x}_{1,n-1} \cdots \bar{x}_{1,1} \bar{x}_{1,0}}_n \quad (51)$$

$$w_{22} = \left| -2^n \underbrace{(x_{1,2n-1} \cdots x_{1,n+1} x_{1,n})}_n \right|_{2^{n-1}} = \underbrace{\bar{x}_{1,2n-1} \cdots \bar{x}_{1,n+1} \bar{x}_{1,n}}_n \quad (52)$$

Now, (46) can be simplified as follows

$$T = |2^{n-3}x_3 - 2^{n-3}x_1 - Z|_{2^{n-1-1}} \quad (53)$$

By dividing (53) in separate parts, we have

$$w_3 = |2^{n-3}x_3|_{2^{n-1-1}} = \left| 2^{n-3} \underbrace{(x_{3,n-2} \cdots x_{3,1} x_{3,0})}_{n-1} \right|_{2^{n-1-1}} = x_{3,1} x_{3,0} \underbrace{x_{3,n-2} \cdots x_{3,3} x_{3,2}}_{n-3} \quad (54)$$

$$w_4 = \left| -2^{n-3}x_1 \right|_{2^{n-1-1}} = \left| -2^{n-3} \underbrace{(x_{1,2n-1} \cdots x_{1,1} x_{1,0})}_{2n} \right|_{2^{n-1-1}} = \left| \begin{array}{l} -2^{n-3}(x_{1,2n-1} x_{1,2n-2} \times 2^{2n-2} \\ + x_{1,2n-3} \cdots x_{1,n} x_{1,n-1} \times 2^{n-1} \\ + x_{1,n-2} \cdots x_{1,1} x_{1,0}) \end{array} \right|_{2^{n-1-1}} \quad (55)$$

The above equation can be parsing as follow

$$w_{41} = \left| -2^{n-3} \times 2^{2n-2} \underbrace{(0 \cdots 00)}_{n-3} x_{1,2n-1} x_{1,2n-2} \right|_{2^{n-1-1}}$$

$$= \bar{x}_{1,2n-1} \bar{x}_{1,2n-2} \underbrace{1 \cdots 11}_{n-3} \quad (56)$$

$$w_{42} = \left| -2^{n-3} \times 2^{n-1} \underbrace{(x_{1,2n-3} \cdots x_{1,n} x_{1,n-1})}_{n-1} \right|_{2^{n-1-1}} = \left| -2^{n-3} \times 2^{n-1} \underbrace{(x_{1,2n-3} \cdots x_{1,n+2} x_{1,n+1} x_{1,n} x_{1,n-1})}_{n-3} \right|_{2^{n-1-1}} = \bar{x}_{1,n} \bar{x}_{1,n-1} \underbrace{\bar{x}_{1,2n-3} \cdots \bar{x}_{1,n+2} \bar{x}_{1,n+1}}_{n-3} \quad (57)$$

$$w_{43} = \left| -2^{n-3} \underbrace{(x_{1,n-2} \cdots x_{1,1} x_{1,0})}_{n-1} \right|_{2^{n-1-1}} = \left| -2^{n-3} \underbrace{(x_{1,n-2} \cdots x_{1,3} x_{1,2} x_{1,1} x_{1,0})}_{n-3} \right|_{2^{n-1-1}} = \bar{x}_{1,1} \bar{x}_{1,0} \underbrace{\bar{x}_{1,n-2} \cdots \bar{x}_{1,3} \bar{x}_{1,2}}_{n-3} \quad (58)$$

For the last term of (53), we have

$$w_5 = |Z|_{2^{n-1-1}} = \left| -\underbrace{(Z_{n-1} \cdots Z_1 Z_0)}_n \right|_{2^{n-1-1}} = \left| -(Z_{n-1} \times 2^{n-1} + \underbrace{Z_{n-2} \cdots Z_1 Z_0}_{n-1}) \right|_{2^{n-1-1}} \quad (59)$$

By dividing (59), w_{51} and w_{52} are obtained as follow

$$w_{51} = \left| -2^{n-1} \underbrace{(0 \cdots 00 Z_{n-1})}_{n-2} \right|_{2^{n-1-1}} = \underbrace{1 \cdots 11}_{n-2} \bar{Z}_{n-1} \quad (60)$$

$$w_{52} = \left| -\underbrace{(Z_{n-2} \cdots Z_1 Z_0)}_{n-1} \right|_{2^{n-1-1}} = \underbrace{\bar{Z}_{n-2} \cdots \bar{Z}_1 \bar{Z}_0}_{n-1} \quad (61)$$

Since both least significant $(n-3)$ bits of w_{41} in (56) and most significant $(n-2)$ bits of w_{51} in (60) are 1's, we can use the following vectors instead of w_{41} and w_{51} .

$$w'_{41} = \bar{x}_{1,2n-1} \bar{x}_{1,2n-2} \underbrace{1 \cdots 11}_{n-4} \bar{Z}_{n-1} \quad (62)$$

$$w'_{51} = \underbrace{1 \cdots 11}_n \quad (63)$$

We know that,

$$w'_{51} = \left| \underbrace{1 \cdots 11}_{n-1} \right|_{2^{n-1-1}} = |2^{n-1} - 1|_{2^{n-1-1}} = 0 \quad (64)$$

Therefore, (53) can be calculated by

$$T = |w_3 + w_{42} + w_{43} + w_{52} + w'_{41}|_{2^{n-1-1}} \quad (65)$$

Finally, (48) can be computed by

$$H = P - Q \quad (66)$$

where

$$P = Z + 2^n T = \underbrace{T_{n-2} \cdots T_1 T_0}_{n-1} \underbrace{Z_{n-1} \cdots Z_1 Z_0}_n \quad (67)$$

$$Q = T = \underbrace{0 \cdots 00}_n \underbrace{T_{n-2} \cdots T_1 T_0}_{n-1} \quad (68)$$

Example 2: Given the moduli set $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$ where $n = 4$. The RNS number (73, 14, 5) can be converted into its equivalent weighted number X as follows

For $n = 4$ the moduli set is $\{256, 15, 7\}$ and residues have binary representation as

$$x_1 = 73 = (01001001)_2$$

$$x_2 = 14 = (1110)_2$$

$$x_3 = 5 = (101)_2$$

Substituting values in (49)–(52), we have

$$w_1 = (1110)_2 = 14$$

$$w_{21} = (1011)_2 = 11$$

$$w_{22} = (0110)_2 = 6$$

$$Z = \lfloor 14 + 11 + 6 \rfloor_{15} = 1 = (0001)_2$$

Then, From (54), (57), (58), (61), (62) and (65), we have

$$w_3 = (011)_2 = 3$$

$$w_{42} = (101)_2 = 5$$

$$w_{43} = (101)_2 = 5$$

$$w_{52} = (110)_2 = 6$$

$$w'_{41} = (101)_2 = 5$$

$$T = \lfloor 3 + 5 + 5 + 6 + 5 \rfloor_7 = 3 = (011)_2$$

Finally, letting values in (66)–(68) and (47),

$$P = (\underbrace{011}_3 \underbrace{0001}_4)_2 = 49$$

$$Q = (\underbrace{0000}_4 \underbrace{011}_3)_2 = 3$$

$$H = 49 - 3 = 46 = (0101110)_2$$

$$X = 73 + 2^8 \times 46 = (\underbrace{010111001001001}_7 \underbrace{001}_8)_2 = 11849$$

To verify the result, we have

$$x_1 = \lfloor 11849 \rfloor_{256} = 73$$

$$x_2 = \lfloor 11849 \rfloor_{15} = 14$$

$$x_3 = \lfloor 11849 \rfloor_7 = 5$$

So, the weighted number 11849 in the RNS based on the moduli set $\{256, 15, 7\}$ has representation as (73, 14, 5).

Also, hardware realization of the residue to binary converter for the moduli set $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$ is based on (49), (65), (66) and (47), and it is depicted in Fig. 2. It can be seen that, the residue to binary converter composed of a 3-operand modulo $(2^n - 1)$ adder, a 5-operand modulo $(2^{n-1} - 1)$ adder and a $(2n - 1)$ -bit CPA. Since the proposed residue to binary converter for the moduli set $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$ has a similar architecture as the residue to binary converter for the moduli set $\{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$, details about hardware implementation have been omitted. The characterization of different parts of the residue to binary converter for the moduli set $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$ is presented in Table 2.

Fig. 2 The converter for moduli set $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$.

4. Performance Evaluation

The residue to binary converters which presented in Sect. 3 are designed for the new $4n$ -bit DR moduli sets $\{2^{2n}, 2^n - 1,$

Table 2 Characterization of each part of the proposed residue to binary converter for moduli set $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$.

Parts	FA	NOT	XOR /AND pairs	XNOR /OR pairs	Delay
OPU4	—	$2n$	—	—	t_{NOT}
CSA4	n	—	—	—	t_{FA}
CPA4	n	—	—	—	$2nt_{\text{FA}}$
OPU5	—	$3n$	—	—	t_{NOT}
CSA5	$n-1$	—	—	—	t_{FA}
CSA6	$n-1$	—	—	—	t_{FA}
CSA7	3	—	—	$n-4$	t_{FA}
CPA5	$n-1$	—	—	—	$(2n-2)t_{\text{FA}}$
OPU6	—	$n-1$	—	—	t_{NOT}
CPA6	$n-1$	—	—	n	$(2n-1)t_{\text{FA}}$
Total	$6n-1$	$6n-1$	—	$2n-4$	$(6n+1)t_{\text{FA}} + 3t_{\text{NOT}}$

$2^{n+1} - 1\}$ and $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$. Hence, to verify the performance of these residue to binary converters, they have to be compared with the other residue to binary converters for $4n$ -bit DR moduli sets. Such moduli sets are $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1\}$, $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$, $\{2^n - 3, 2^n + 1, 2^n - 1, 2^n + 3\}$, $\{2^n - 1, 2^n, 2^{2n+1} - 1\}$ and $\{2^n - 1, 2^n + 1, 2^{2n} + 1\}$. These moduli sets belong to the category of $4n$ -bit DR moduli sets, and each one has its own properties. The moduli sets $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1\}$ and $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ were introduced in [19] and [21], respectively. Recently, new designs of the residue to binary converters for these moduli sets based on MRC have been proposed by Mohan et al. [20]. As shown in [20], the residue to binary converters of [20] have better performance than those of [19] and [21]. Therefore, we only compare our residue to binary converters with the residue to binary converters of [20]. Also, a high-speed converter for the moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$ is presented in [22]. Sheu et al. [23] proposed the balanced moduli set $\{2^n - 3, 2^n + 1, 2^n - 1, 2^n + 3\}$, but because of using moduli of the forms $2^k \pm 3$, their residue to binary converter needs three ROMs, and this resulted in increasing the cost and the delay of the converter. Recently, Molahosseini et al. [24] proposed the modulus $(2^k + 1)$ -free moduli set $\{2^n - 1, 2^n, 2^{2n+1} - 1\}$ for having fast residue arithmetic unit. The converter presented in [24] derived by using a two-level implementation of MRC algorithm, and this leads to increasing the latency.

Wang et al. [25] designed a residue to binary converter architecture for the moduli set $\{2^n - 1, 2^n + 1, 2^{2n} + 1\}$. They used a modified CRT formulation to derive an adder-based residue to binary converter for this moduli set. The general structure of the residue to binary converter of [25] consists of two $2n$ -bit CSAs with EAC, a $2n$ -bit modulo $(2^{2n} - 1)$ adder and a $4n$ -bit binary adder. They used the method of [32] for performing the modulo $(2^{2n} - 1)$ addition, and by using different methods for implementing the $4n$ -bit binary adder, two versions of the residue to binary converter are presented in [25]. The residue to binary converters of [20]–[24] as well as the proposed designs, are using CPAs with

EAC and regular CPAs for implementing modulo of the forms $(2^k - 1)$ adders and vector binary adders, respectively. Therefore, for a fair comparison, we consider a $2n$ -bit CPA with EAC, and a $4n$ -bit regular CPA for implementation of the modulo $(2^{2n} - 1)$ adder and $4n$ -bit binary adder of [25], respectively. In this case, the delay of the residue to binary converter of [25] is $(1 + 1 + 4n + 4n)t_{\text{FA}} = (8n + 2)t_{\text{FA}}$, where t_{FA} denotes the delay of an FA. Furthermore, the presented residue to binary converters can be compared with the converter of the $5n$ -bit DR moduli set $\{2^n, 2^{2n} - 1, 2^{2n} + 1\}$ [26], because similar to the proposed moduli sets, the moduli set $\{2^n, 2^{2n} - 1, 2^{2n} + 1\}$ is an unconventional three-moduli set which offers larger DR than $3n$ -bit. Similar to the other converters, we consider a $4n$ -bit CPA with EAC for the implementation of the modulo $(2^{4n} - 1)$ adder of the residue to binary converter of [26]. Table 3 compares the hardware requirements and conversion delays of the residue to binary converters. It can be seen that, the proposed residue to binary converters have less conversion delay than all other converters. Furthermore, the hardware requirements of the proposed converters are less than other converters except the residue to binary converter of the moduli set $\{2^n, 2^{2n} - 1, 2^{2n} + 1\}$. The residue to binary converter of the moduli set $\{2^n, 2^{2n} - 1, 2^{2n} + 1\}$ [26] has the lowest hardware cost. Also, comparison of the proposed residue to binary converters with together showed that, the converter for the moduli set $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$ is faster, while the residue to binary converter for the moduli set $\{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$ has lower hardware cost.

Until now, it is shown that the proposed residue to binary converters have better performance than the other residue to binary converters for $4n$ -bit DR moduli sets. But other important issues in the design of RNS systems are the speed of the internal RNS arithmetic processing and the binary to residue conversion. Here, we used the method of [26] for comparing the time-performance of the different RNS moduli sets. It is the magnitude of the largest modulo that dictates the speed of arithmetic operations; however, speed and cost also depend on the moduli chosen [2]. Thus, the modulo 2^{2n} determines the overall speed of the RNS systems based on the proposed moduli sets $\{2^{2n}, 2^n - 1, 2^{n+1} - 1\}$ and $\{2^{2n}, 2^n - 1, 2^{n-1} - 1\}$. Also, the overall speed of arithmetic unit of RNS systems based on the moduli sets $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1\}$, $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$, $\{2^n - 3, 2^n - 1, 2^n + 1, 2^n + 3\}$ and $\{2^n - 1, 2^n, 2^{2n+1} - 1\}$ are restricted to the time performance of the moduli $2^{n+1} + 1$, $2^n + 1$, $2^n + 3$ and $2^{2n+1} - 1$, respectively. In addition to these, in the moduli sets $\{2^n - 1, 2^n + 1, 2^{2n} + 1\}$ and $\{2^n, 2^{2n} - 1, 2^{2n} + 1\}$, the critical modulo is $2^{2n} + 1$. Next, we consider the unit gate delays of parallel-prefix modular adders of [33], [34] and [35] for estimating the delay of addition in moduli of the forms $2^k - 1$, $2^k + 1$ and $2^k + 3$, respectively. It should be noted that, there is no modular adder which specially designated for modulo of the forms $2^k + 3$. Hence, we must use the generic modular adders for performing modulo $2^n + 3$ addition. Also, modulo 2^k adder can perform addition with approximately the same speed as the modulo $2^k - 1$ adder

Table 3 Performance comparison of the different residue to binary converters.

Converter	Moduli set	FA	NOT	XOR/AND pairs	XNOR/OR pairs	Others	Delay
[20]-C1	$\{2^n-1, 2^n, 2^{n+1}, 2^{n+1}-1\}$	$9n+5+k^*$	$6n+1$	–	$2n$	–	$((23n+12)/2)t_{FA}$
[20]-C1ROM	$\{2^n-1, 2^n, 2^{n+1}, 2^{n+1}-1\}$	$8n+4$	$6n+1$	–	$2n$	ROM	$(9n+6)t_{FA}$
[22]	$\{2^n-1, 2^n, 2^{n+1}, 2^{n+1}-1\}$	$10n+6+k^*$	$7n+2$	–	$6n+2$	MUX	$((15n+22)/2)t_{FA}$
[20]-C2	$\{2^n-1, 2^n, 2^{n+1}, 2^{n+1}+1\}$	$n^2+12n+12$	$6n+7$	–	$2n$	MUX	$(16n+22)t_{FA}$
[20]-C2ROM	$\{2^n-1, 2^n, 2^{n+1}, 2^{n+1}+1\}$	$9n+10$	$5n+6$	–	$2n$	MUX ROM	$(11n+14)t_{FA}$
[23]	$\{2^n-3, 2^n+1, 2^{n-1}, 2^{n+3}\}$	$26n+8$	–	–	–	ROM	$(7n+8)t_{FA}+2t_{ROM}$
[24]	$\{2^n-1, 2^n, 2^{2n+1}-1\}$	$9n+2$	$7n+3$	2	$2n$	–	$(10n+5)t_{FA}$
[25]-C4	$\{2^n-1, 2^{2n}+1, 2^{2n}+1\}$	$8n+2$	–	$2n-2$	–	–	$(8n+2)t_{FA}$
[26]	$\{2^n, 2^{2n}-1, 2^{2n}+1\}$	$5n+2$	$3n+1$	$2n-1$	$n-1$	–	$(8n+1)t_{FA}+t_{NOT}$
Proposed	$\{2^{2n}, 2^n-1, 2^{n+1}-1\}$	$6n+1$	$4n+2$	3	n	–	$(6n+6)t_{FA}+3t_{NOT}$
Proposed	$\{2^{2n}, 2^n-1, 2^{n-1}-1\}$	$6n-1$	$6n-1$	–	$2n-4$	–	$(6n+1)t_{FA}+3t_{NOT}$

* $k=(n-4)(n+1)/2$

Table 4 Comparison the speed of the different moduli sets.

Moduli set	Critical modulo	Delay
$\{2^n-1, 2^n, 2^{n+1}, 2^{n+1}-1\}$	2^{n+1}	$2\log_2(n)+6$
$\{2^n-1, 2^n, 2^{n+1}, 2^{n+1}+1\}$	$2^{n+1}+1$	$2\log_2(n+1)+6$
$\{2^n-3, 2^n+1, 2^{n-1}, 2^{n+3}\}$	2^{n+3}	$2\log_2(n-1)+7$
$\{2^n-1, 2^n, 2^{2n+1}-1\}$	$2^{2n+1}-1$	$2\log_2(n+0.5)+5$
$\{2^n-1, 2^{2n}+1, 2^{2n}+1\}$	2^{2n+1}	$2\log_2(n)+8$
$\{2^n, 2^{2n}-1, 2^{2n}+1\}$	2^{2n}	$2\log_2(n)+5$
$\{2^{2n}, 2^n-1, 2^{n+1}-1\}$		
$\{2^{2n}, 2^n-1, 2^{n-1}-1\}$		

of [33]. The unit gate delays of the adders of [33], [34] and [35] are $2\log_2(k_1)+3$, $2\log_2(k_2)+6$ and $2\log_2(k_3-1)+7$, respectively. Therefore, by substituting $k_1 = 2n$, $2n+1$, $k_2 = n+1$, n , $2n$ and $k_3 = n$, the unit gate delays are obtained, and the results are noted in Table 4. According to this table, it is clear that the addition in modulo 2^{2n} is faster than the other moduli. Hence, the arithmetic unit of RNS systems based on moduli sets $\{2^{2n}, 2^n-1, 2^{n+1}-1\}$ and $\{2^{2n}, 2^n-1, 2^{n-1}-1\}$ are the fastest than those based on moduli sets $\{2^n-1, 2^n, 2^{n+1}, 2^{n+1}-1\}$, $\{2^n-1, 2^n, 2^{n+1}, 2^{n+1}+1\}$, $\{2^n-3, 2^n-1, 2^n+1, 2^n+3\}$, $\{2^n-1, 2^n+1, 2^{2n}+1\}$, $\{2^n-1, 2^n, 2^{2n+1}-1\}$ and $\{2^n, 2^{2n}-1, 2^{2n}+1\}$. Especially, it can be seen that, while the proposed moduli sets have three moduli, they are faster than the four-moduli sets.

Now, we investigate the binary to residue conversion for the different moduli sets. The binary to residue converter consists of independent converters for each modulo of the moduli set for transforming a weighted integer binary number into its equivalent RNS representation. The binary to residue converter for the proposed moduli sets $\{2^{2n}, 2^n-1, 2^{n+1}-1\}$ and $\{2^{2n}, 2^n-1, 2^{n-1}-1\}$ can be simply implemented. The modulo 2^{2n} reduction needs no computational hardware, the desired result can be obtained by only considering the least significant $2n$ bits of the weighted binary number. Also, reduction circuits for the moduli 2^n-1 , $2^{n+1}-1$ and $2^{n-1}-1$ can be implemented with multi-operand modular adders (MOMAs) [31].

Hence, each of the proposed moduli sets as well as the moduli sets $\{2^n-1, 2^n, 2^{2n+1}-1\}$ and $\{2^n, 2^{2n}-1, 2^{2n}+1\}$ requires two MOMAs. But, the MOMAs for the moduli $2^{2n+1}-1$ and $2^{2n}+1$ have longer delay and higher cost than the MOMAs for the moduli $2^{n+1}-1$ and $2^{n-1}-1$ which are need for our moduli sets. Furthermore, for the moduli sets $\{2^n-1, 2^n+1, 2^{2n}+1\}$, $\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$ and $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$, we need three MOMAs. Particularly, binary to residue converter for the moduli set $\{2^n-3, 2^n-1, 2^n+1, 2^n+3\}$ requires four MOMAs. Therefore, it is clear that the proposed moduli sets can result in more efficient binary to residue converters than the other moduli sets.

5. Conclusion

In this paper, efficient residue to binary converters for the new RNS moduli sets $\{2^{2n}, 2^n-1, 2^{n+1}-1\}$ and $\{2^{2n}, 2^n-1, 2^{n-1}-1\}$ are presented. The converters for these moduli sets are achieved by an adder-based implementation of the MRC algorithm, with better performance compared to the other residue to binary converters for $4n$ -bit DR moduli sets. Also, it is showed that, with the new proposed moduli sets, the binary to residue converter as well as the internal RNS arithmetic circuits can be implemented efficiently, resulting in high-performance RNS systems.

References

- [1] T. Stouratits and V. Paliouras, "Considering the alternatives in low-power design," IEEE Circuits and Devices, vol.7, pp.23–29, 2001.
- [2] B. Parhami, Computer Arithmetic: Algorithms and Hardware Design, Oxford University Press, 2000.
- [3] M. Lu, Arithmetic and Logic in Computer Systems, John Wiley & Sons, 2004.
- [4] M.A. Soderstrand, W.K. Jenkins, Q.A. Jullien, and F.J. Taylor, Residue number system arithmetic: Modern applications in digital signal processing, IEEE Press, 1986.
- [5] G.C. Cardarilli, A. Nannarelli, and M. Re, "Residue number system for low-power DSP applications," Proc. 41st Asilomar Conference on Signals, Systems, and Computers, 2007.
- [6] 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, 1977.
- [7] R. Conway and J. Nelson, "Improved RNS FIR filter architectures," *IEEE Trans. Circuits Syst. II*, vol.51, no.1, pp.26–28, 2004.
- [8] R. Chaves and L. Sousa, "RDSP: A RISC DSP based on residue number system," *Proc. Euromicro Symposium on Digital System Design: Architectures, Methods and Tools*, 2003.
- [9] E. Diclaudio, F. Piazza, and G. Orlandi, "Fast combinatorial RNS processors for DSP applications," *IEEE Trans. Comput.*, vol.44, no.5, pp.624–331, 1995.
- [10] W. Wang, M.N.S. Swamy, and M.O. Ahmad, "RNS application for digital image processing," *Proc. 4th IEEE International Workshop on System-on-Chip for Real Time Applications*, 2004.
- [11] L.L. Yang and L. Hanzo, "Redundant residue number system based error correction codes," *Proc. IEEE Vehicular Technology Conference*, 2001.
- [12] S. Timarchi and K. Navi, "Efficient class of redundant residue number system," *Proc. IEEE International Symposium on Intelligent Signal Processing*, 2007.
- [13] W. Wang, M.N.S. Swamy, and M.O. Ahmad, "Moduli selection in RNS for efficient VLSI implementation," *Proc. IEEE International Symposium on Circuits and Systems*, 2003.
- [14] C.H. Chang, S. Menon, B. Cao, and T.A. Srikanthan, "A configurable dual moduli multi-operand modulo adder," *Proc. IEEE International Symposium on Circuits and Systems*, 2005.
- [15] 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, 2002.
- [16] 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*, vol.47, no.12, pp.1576–1581, 2000.
- [17] 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*, vol.54, no.9, pp.775–779, 2007.
- [18] S.H. Lin, M.H. Sheu, and C.H. Wang, "Efficient VLSI design of a residue-to-binary converter for the moduli set $(2^n, 2^{n+1} - 1, 2^n - 1)$," *IEICE Trans. Inf. & Syst.*, vol.E91-D, no.7, pp.2058–2060, July 2008.
- [19] M. Bhardwaj, T. Srikanthan, and C.T. Clarke, "A reverse converter for the 4-moduli superset $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} + 1\}$," *Proc. IEEE Symposium on Computer Arithmetic*, 1999.
- [20] 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*, vol.54, no.6, pp.1245–1254, 2007.
- [21] A.P. Vinod and A.B. Premkumar, "A residue to binary converter for the 4-moduli superset $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$," *J. Circuits Syst. Comput.*, vol.10, pp.85–99, 2000.
- [22] 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 Electronics Express*, vol.5, no.17, pp.672–677, 2008.
- [23] M.H. Sheu, S.H. Lin, C. Chen, and S.W. Yang, "An efficient VLSI design for a residue to binary converter for general balance moduli $(2^n - 3, 2^n + 1, 2^n - 1, 2^n + 3)$," *IEEE Trans. Circuits Syst. II*, vol.51, no.2, pp.152–155, 2004.
- [24] A.S. Molahosseini, K. Navi, and M.K. Rafsanjani, "A new residue to binary converter based on mixed-radix conversion," *Proc. 3rd International Conference on Information and Communication Technologies: From Theory to Application*, 2008.
- [25] 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*, vol.50, no.2, pp.235–243, 2003.
- [26] A. Hariri, K. Navi, and R. Rastegar, "A new high dynamic range moduli set with efficient reverse converter," *Elsevier Journal of Computers and Mathematics with Applications*, vol.55, no.4, pp.660–668, 2008.
- [27] A.S. Molahosseini, K. Navi, O. Hashemipour, and A. Jalali, "An efficient architecture for designing reverse converters based on a general three-moduli set," *Elsevier Journal of Systems Architecture*, vol.54, no.10, pp.929–934, 2008.
- [28] F.J. Taylor, "Residue arithmetic: A tutorial with examples," *Computer*, vol.17, pp.50–62, 1984.
- [29] C.K. Koc, "A fast algorithm for mixed-radix conversion in residue arithmetic," *Proc. IEEE International Conference on Computer Design: VLSI in Computers and Processors*, 1989.
- [30] S.J. Piestrak, "A high speed realization of a residue to binary converter," *IEEE Trans. Circuits Syst. II*, vol.42, no.10, pp.661–663, 1995.
- [31] S.J. Piestrak, "Design of residue generators and multioperand modular adders using carry-save adders," *IEEE Trans. Comput.*, vol.423, no.1, pp.68–77, 1994.
- [32] C. Efstathiou, D. Nikolos, and J. Kalamatianos, "Area-time efficient modulo $2^n - 1$ adder design," *IEEE Trans. Circuits Syst. II*, vol.41, no.7, pp.463–467, 1994.
- [33] L. Kalampoukas, D. Nikolos, C. Efstathiou, H.T. Vergos, and J. Kalamatianos, "High-speed parallel-prefix modulo $2^n - 1$ adders," *IEEE Trans. Comput.*, vol.49, no.7, pp.673–679, 2000.
- [34] 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, 2004.
- [35] 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*, vol.54, no.6, pp.1279–1292, 2007.

Amir Sabbagh Molahosseini received his B.Sc. degree in Computer Engineering from Shahid Bahonar University of Kerman, Iran in 2005, and the M.Sc. degree in Computer Architecture from Islamic Azad University (IAU), Science and Research Branch, Tehran, Iran in 2007. He is currently working toward his Ph.D degree in Computer Engineering at Science & Research Branch of IAU, Tehran, Iran. His research interests are Residue Number System and VLSI.

Chitra Dadkhah is Assistant Professor in Electrical Engineering Faculty of K.N.Toosi University of Technology. She received the B.S. degree in Software Engineering from Shahid Beheshti University, Tehran, Iran in 1990. She received the M.S. in Computer Engineering (Artificial Intelligence), IASI Department, University of Paris 11 (Orsay), France, 1993. She also received PhD in Computer Engineering (AI), Department of Computer Engineering & IT, Amir Khabir University (Polytechnique), Tehran, Iran, 2005. Her research interests include Soft Computing, Residue Number System, Expert System, Verification & Validation of the Knowledge.

Keivan Navi received his B.Sc and M.Sc in Computer Hardware engineering from Shahid Beheshti University in 1987 and Sharif University of Technology in 1990, respectively. He also received his Ph.D. of Computer Architecture from Paris XI University in 1995. He is currently associate professor in faculty of electrical and computer engineering of Shahid Beheshti University. His research interests include VLSI design, Single Electron Transistors (SET), Carbon Nano Tube, Computer Arithmetic, Intercon-

nection Network and Quantum Computing. He has published over 30 ISI and research journal papers and over 70 IEEE, international and national conference paper.

Mohammad Eshghi (BS'78, Ms'89, Phd'94) is born in Shahroud, Iran in 1954 and got his BS in electrical engineering from Sharif University in 1978. He got his MS degree in EE from Ohio University, Athens, Ohio, and his Ph.D in EE from Ohio State University, Columbus, Ohio, USA, in 1989 and 1994, respectively. He is now with the Electrical and Computer Engineering Faculty at Shahid Beheshti University, Tehran Iran. His research interests including Digital Signal processing,

Digital circuit design, implementation and test on Field Programmable Gate Array (FPGA), and quantum computing.