Fast and energy-efficient FPGA realization of RNS reverse converter for the ternary 3-moduli set $\{3^n-2, 3^n-1, 3^n\}$

Pooriya NavaeiLavasani^{1,2} · Shokoufeh Adeli^{2,3} · MohammadReza Taheri^{2,4} · Mohammad Hossein Moaiyeri^{1,2} · Keivan Navi^{2,4}

Received: 15 August 2019 / Accepted: 14 January 2020 / Published online: 25 January 2020
© Springer Nature Switzerland AG 2020

Abstract

Ternary number representation has been known as one of the efficient methods for the implementation of digital systems. Employing residue number system (RNS) in ternary digital signal processing (DSP) applications could offer the advantages of ternary number representation along with further privileges such as improving parallelism and fault tolerance. Efficacious design of reverse converter as a crucial block of residue number system has always been of great interest to researchers. In this paper, an efficient reverse converter for ternary RNS with moduli set $\{3^n-2, 3^n-1, 3^n\}$, based on the mixed-radix conversion (MRC) algorithm is presented. The proposed algorithm reduces the number of adders with large bit-width and the dependencies between consecutively process blocks. The employed technique in the proposed algorithm and the reverse converter architecture improves the delay and area on the FPGA Virtex Ultrascale + family platform in 14 nm/16 nm FinFET technology as compared to the related works.

Keywords Computer arithmetic · Residue number system · Ternary logic · Reverse converter · FPGA

1 Introduction

Over the past few decades, the binary numeral system has gained more popularity over the other radices due to the binary essence of electronic switching devices. However, the binary system is not always the best choice. Recent studies have revealed that ternary can be the most efficient number representation for the implementation of digital systems, since it is the closest integer number to the optimum radix e [1]. The rapid density growth of chips will inevitably result in a complex wire routing and packaging, which pose serious challenges in modern chips [2]. These issues subsequently restrict the performance of the contemporary binary integrated circuits, as the speed and power consumption of integrated circuits are dominantly

affected by the dimensions and complexity of their interconnects [3].

A well-promising approach to address these challenges is to exploit ternary logic in designing digital integrated circuits [4, 5] which enable transferring more data per wire with considerably high-density memories compared to their binary counterparts. Consequently, the number of required interconnects and routing complexity of integrated circuits can be reduced [6]. Furthermore, using ternary systems creates an opportunity to implement real-life applications, such as process control and robotics, with more efficiency [7].

Over the last few years, several researches have been conducted for efficient realization of the arithmetic unit (AUs) based on ternary structures. Considering the conducted studies, the design presented in [8] used binary

✉ Keivan Navi, navi@sbu.ac.ir | ¹Department of Electrical Engineering, Shahid Beheshti University, Tehran, Iran. ²Nanotechnology and Quantum Computing Lab, Shahid Beheshti University, Tehran, Iran. ³Department of Engineering, Khatam University, Tehran, Iran. ⁴Department of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran.

encoding for balanced ternary numbers and performed ternary arithmetic function directly on ternary numbers. The past history of ternary systems alongside the available and promising nanotechnologies such as carbon nanotube field-effect transistor (CNFET) and graphene nanoribbon field-effect transistors (GNRFET) and interconnects for implementing the future ternary integrated circuits [2–6], inspired us to continue research in the field of ternary arithmetic.

In this paper, we employ the unbalanced representation for the realization of the ternary arithmetic. A significant problem in the arithmetic structures with weighted number representation is the long carry propagation chain, which has been intensified with the growth of processing demands. Therefore, unconventional numeral systems have attracted the researchers' attention to addressing this problematic issue. One of the sounds promising alternative numeral systems is the residue number system (RNS) [9]. RNS has become a remarkable number system in computer arithmetic, especially during the recent decades, regarding its limited carry-propagation nature between residue channels. In RNS, each modulo performs arithmetic operations independent of the other channels. Thus, a large number can be divided into smaller parts (residues), and operations can be executed independently in parallel. Therefore, long and slow operations, such as addition and multiplication, can be performed on smaller operands in a parallel manner [10–13]. In addition, RNS has outstanding potential for fast and energy-efficient computing in DSP systems and remarkable properties in digital systems, including error detection, error correction, and fault tolerance. Therefore, its possible applications include digital filters, convolutions, correlations, discrete Fourier transform (DFT), fast Fourier transform (FFT), computer networks, DNA arithmetic, cloud storage, cryptographic systems, and fault-tolerant hybrid memory design [9, 12, 14–16].

Arithmetic ternary blocks with weighted number representation also deal with the long carry chain problem. During the past decade, the substantial benefits of ternary number representation along with the great privileges of RNS, such as parallelism, have become of great interest for researchers. In [17], for further reduction of the area, power, and latency of the multiplication operation, ternary valued logic has been adopted in a logarithmic version of RNS called Residue Logarithmic Number System (RLNS). RNS has also mapped to ternary for efficient ternary-RNS implementations with a variety of moduli sets in [18]. In the design of each RNS platform, the choice of moduli set plays a significant role in the overall system performance. The reason is that the forms and the number of the moduli in the set directly influence the system's latency, hardware complexity, and dynamic range [18]. Among the suggested moduli set, $\{3^n - 2, 3^n - 1, 3^n\}$ shows a good

enough dynamic range and a more straightforward and efficient implementation as compared to the other moduli sets [19]. One of the challenging parts of the RNS systems is the residue to the weighted conversion part according to its significant delay and area overhead. In [19] and [20], two different reverse converters for the moduli set $\{3^n - 2, 3^n - 1, 3^n\}$ have been developed based on the Chinese Remainder Theorem (CRT) and Mixed Radix Conversion (MRC). The former design suffers from the extreme complexity of the large bit-length and complex modular adder, which imposes a considerable delay and area occupation to the reverse converter part. To address this serious issue, in [20], an efficient residue to weighted converter was presented for the moduli set mentioned above, based on a two-level MRC algorithm. Nevertheless, computing in two levels of series MRC blocks leads to using adders with large bit-width, especially for the second level, which results in $3n$ -digit ternary adders for calculating the weighted number. These large adders increase the computation delay of the circuit.

Accordingly, these observations have motivated us to seek for an alternative design to overcome these issues. The proposed algorithm eliminates the issues described above by employing one level of MRC block with moduli set $\{3^n - 2, 3^n - 1, 3^n\}$, and dividing the calculation into subparts, which reduces the critical path and intermediates carry propagate adders. Therefore, the computation delay of our proposed design is reduced compared to its previous counterparts. The FPGA implementation results authenticate the superiority of the proposed approach in terms of performance and energy efficiency as compared to its previous counterpart. Furthermore, we implement the box blur kernel filter as a digital image processing application based on the proposed and previous approaches, and the results shows the excel of the proposed approach.

The rest of the paper is arranged as follows: Sect. 2 presents a conceptual study of the ternary and RNS systems. In Sect. 3, the state-of-the-art ternary RNS blocks and the ternary RNS reverse converter architecture are introduced in detail. In Sect. 4, the description of the proposed algorithm and its hardware implementation are presented. Section 5 contains the simulation results and the measurements by using FPGA. Finally, Sect. 6 concludes the paper.

2 Background

In this section, we briefly describe some concepts of the unbalanced ternary arithmetic system as well as RNS, which will be employed in our proposed design.

2.1 Ternary in brief

Ternary number representation is a place-value notation for numbers using the powers of 3. Any number $X \in R$ from the decimal system can be represented in the ternary arithmetic system by using the following equation [21]:

$$X = \dots + c_2 \cdot 3^2 + c_1 \cdot 3^1 + c_0 \cdot 3^0 + c_{-1} \cdot 3^{-1} + \dots = \sum_{-m}^n c_k \cdot 3^k \quad (1)$$

where k is the power index, and c_k denotes the appropriate coefficients to represent a given number X .

In the ternary representation, each digit is called a *trit* (ternary digit). Each digit that forms a number in a given base b carries $\log_2 b$ bits of information. Thus, each *trit* carries almost 1.6 bits [22]. Moreover, the maximum number U that can be represented with $w + 1$ symbols (size) in an arithmetic system of radix $r > 1$ is upper-bounded by r^{w+1} [21]. As stated before, the balanced and unbalanced modes are two ways to encode numbers by employing ternary logic, as shown in Table 1 [23].

The unbalanced ternary system represents any number by using digits that belong to the digit set of $\{0, 1, 2\}$. For coding the unbalanced ternary numbers with binary hardware, each *trit* is indicated by two bits. From now on, we consider ternary as an unbalanced ternary system. The ternary arithmetic operations such as addition, subtraction, multiplication, and division are highly analogous to binary arithmetic operations by means of regulation [24]. For instance, the rules for ternary addition are given in Table 2.

The fundamental ternary arithmetic components for any computational operation are the half adder and full adder blocks. The ternary half adder with inputs A and B operates based on the following equations [25]:

$$Sum = (A + B) \bmod 3 \quad (2)$$

$$Carry = \left\lfloor \frac{(A + B)}{3} \right\rfloor \quad (3)$$

where $\lfloor \cdot \rfloor$ denotes the floor function. The truth table of the ternary half adder is shown in Table 3. The ternary full adder with the inputs A, B , and C_{in} operates based on the following equations [25]:

Table 1 Ternary encoding of base-3 digits

Truth value	Unbalanced	Balanced
False	0	-1
Unknown	1	0
True	2	+1

Table 2 Ternary addition rules

Augend	Addend	Sum	Carry
0	0	0	0
0	1	1	0
0	2	2	0
1	1	2	0
1	2	0	1
2	2	1	1

$$Sum = (A + B + C_{in}) \bmod 3 \quad (4)$$

$$Carry = \left\lfloor \frac{(A + B + C_{in})}{3} \right\rfloor \quad (5)$$

The 4-input ternary full adder with the inputs A, B, C , and C_{in} operates based on the following equations [25]:

$$Sum = (A + B + C + C_{in}) \bmod 3 \quad (6)$$

$$Carry = \left\lfloor \frac{(A + B + C + C_{in})}{3} \right\rfloor \quad (7)$$

The standard ternary inverter (STI) operates based on the following equation [26]:

$$STI(A) = \begin{cases} A, & A = 1 \\ 2 - A, & A \neq 1 \end{cases} \quad (8)$$

There are different kinds of arithmetic adder implementations for ternary signals. In this paper, we employ ripple carry adder (RCA). The RCAs with two input vectors are realized based on the ternary half adder (Eqs. 2, 3) and the 3-input ternary full adder (Eqs. 4, 5). Furthermore, the RCAs with three input vectors are implemented based on the 3-input ternary full adder (Eqs. 4, 5) and the 4-input ternary full adder (Eqs. 6, 5).

Table 3 The truth table of the ternary half adder

Inputs		Outputs	
A	B	Sum	Carry
0	0	0	0
0	1	1	0
0	2	2	0
1	0	1	0
1	1	2	0
1	2	0	1
2	0	2	0
2	1	0	1
2	2	1	1

2.2 RNS in brief

RNS is defined by a set of mutually prime integers, called the moduli set. In a given moduli sets like $\{m_1, m_2, \dots, m_n\}$, for each pair of moduli $\text{GCD}(m_i, m_j) = 1$ where $i \neq j$. The dynamic range of the system is shown by $M = m_1.m_2 \dots m_n$. Any integer X member of $[0, M - 1]$, can be represented in RNS uniquely ($X \Rightarrow (|X|_{m_1}, |X|_{m_2}, \dots, |X|_{m_n})$) [12], where, $|X|_{m_i} = x_i = X \bmod m_i$ and $x_i \in [0, m_i)$.

Arithmetic operations such as addition, subtraction, and multiplication can be performed in parallel channels for each RNS modulo (m_i), since there is no carry propagation through the RNS moduli. Considering X and Y as operands, and Z as the result of an arithmetic operation, any operation in RNS can be performed as

$$Z \xrightarrow{RNS} (z_1, z_2, \dots, z_n), Z = X \otimes Y \xrightarrow{RNS} z_i = x_i \otimes_{m_i} y_i \quad (9)$$

where $\otimes$ is one of the addition, subtraction, and multiplication operands, and $\otimes_{m_i}$ denotes the aforementioned operands in modulo m_i [9].

The addition modulo ($3^n - k$) of two residue values, is formulated as [12]

$$|a + b|_{(3^n - k)} = \begin{cases} a + b, & a + b < (3^n - k) \\ a + b - (3^n - k), & a + b \geq (3^n - k) \end{cases} \quad (10)$$

The subtraction of two residue values can be performed as [12]

$$|a - b|_{(3^n - k)} = |a + \text{STI}(b) + 1 - K|_{(3^n - k)} \quad (11)$$

Each RNS commonly contains three vital parts: first, the weighted to residue converter (forward converter), arithmetic operators, and the residue to weighted number converter (reverse converter) [27]. The forward converter accomplishes the transformation of a weighted number into a residue number, based on an specific moduli set. The arithmetic unit performs the basic arithmetic operations such as addition, subtraction, and multiplication on the residue numbers. The reverse converter transforms the residue numbers to the weighted numbers.

One of the methods, which is widely used for converting residue numbers to weighted numbers, is the Mixed Radix Conversion (MRC) algorithm [28]. In MRC, the weighted number X can be calculated as

$$X = a_n \prod_{i=1}^n m_i + \dots + a_3 m_2 m_1 + a_2 m_1 + a_1 \quad (12)$$

where a_i 's are called the mixed-radix coefficients and are calculated as

$$a_n = \left| \left(\left((x_n - a_1) |m_1^{-1}|_{m_n} - a_2 \right) |m_2^{-1}|_{m_n} - \dots - a_{n-1} \right) |m_{n-1}^{-1}|_{m_n} \right|_{m_n} \quad (13)$$

where $n > 1$ and $a_1 = x_1$. Thus, for a ternary moduli set, which includes three moduli $\{m_1, m_2, m_3\}$, Eq. 12 can be simplified as

$$X = a_3 m_2 m_1 + a_2 m_1 + a_1 \quad (14)$$

where

$$a_1 = x_1, \quad (15)$$

$$a_2 = \left| (x_2 - a_1) |m_1^{-1}|_{m_2} \right|_{m_2}, \quad (16)$$

$$a_3 = \left| \left((x_3 - a_1) |m_1^{-1}|_{m_3} - a_2 \right) |m_2^{-1}|_{m_3} \right|_{m_3} \quad (17)$$

In the above equations, $|m_i^{-1}|_{m_j}$ is the multiplicative inverse of m_i modulo m_j . For any integer a , the modular multiplicative inverse is an integer x , which satisfies the equation $ax \equiv 1(\bmod m)$.

3 Related works

For our targeted moduli set, several contributions have been conducted in the literature regarding the efficient design of the abovementioned RNS blocks. Efficient forward converters for the moduli set $\{3^n - 2, 3^n - 1, 3^n\}$ were suggested in [19]. In [29] and [30], different designs for ternary to RNS converter based on the redundant numbers representation method for the moduli set $\{3^n - 2, 3^n - 1, 3^n\}$ were presented. In [19], an RNS modular adder for the moduli set $\{3^n - 2, 3^n - 1, 3^n\}$ was presented. Based on the aforementioned redundant method, efficient modulo adders were also proposed in [29] and [30]. Furthermore, in [31], a novel architecture for multiply-accumulate (MAC) units for the moduli set $\{3^n - 2, 3^n - 1, 3^n\}$ was proposed. This MAC unit was employed in the realization of DSP algorithms, such as FIR in the ternary domain. The results given in [31] indicated the superiority of the ternary RNS compared to its weighted counterparts in the implementations of digital filters. Besides the above-mentioned studies, some researches have been carried out for designing reverse converter for the moduli set $\{3^n - 2, 3^n - 1, 3^n\}$. As the main aim of this study is to investigate the way to improve the performance of the reverse converter for $\{3^n - 2, 3^n - 1, 3^n\}$. In this section, we review the state-of-the-art implementation of the reverse converter with this moduli set based on the algorithm

suggested in [20]. The algorithm is based on two levels of series MRC blocks for the residue to weighted conversion. The first and the second level of the MRC blocks operate with the moduli set $\{3^n - 2, 3^n\}$, and $\{3^n(3^n - 2), 3^n - 1\}$, respectively. The hardware implementation of [20] is shown in Fig. 1.

4 Proposed design

In this section, the proposed reverse converter for ternary RNS $\{3^n - 2, 3^n - 1, 3^n\}$ is described.

4.1 Proposed method

Based on the MRC algorithm (see Eq. 14), a novel RNS reverse converter for the moduli set of $\{3^n - 2, 3^n - 1, 3^n\}$ is proposed. In contrast to the design presented in [20], our design employs only one level of MRC algorithm for calculating the weighted number X . In the proposed design, firstly, the mixed-radix coefficients of a_2 and a_3 are calculated considering Eqs. 16 and 17. Then, a combination of those coefficients with x_1 yields X as follows:

$$\begin{aligned} X &= a_3 m_2 m_1 + a_2 m_1 + a_1 = m_1(a_2 + a_3 m_2) + x_1 \\ &= 3^n(a_2 + a_3(3^n - 1)) + x_1 = 3^n(a_2 + 3^n a_3 - a_3) + x_1 \\ &= ((a_3 \& a_2) - a_3) \& x_1 \end{aligned} \quad (18)$$

Fig. 1 The design of [20]

where $\&$ denotes the concatenation operator of two operands. The necessities multiplicative inverse to calculate the aforementioned mixed-radix coefficients are as follows:

$$K_1 = |m_1^{-1}|_{m_2} \rightarrow |K_1 \times m_1|_{m_2} = 1 \rightarrow |K_1 \times 3^n|_{(3^n-1)} = 1 \rightarrow K_1 = 1 \quad (19)$$

$$\begin{aligned} K_2 &= |m_1^{-1}|_{m_3} \rightarrow |K_2 \times m_1|_{m_3} = 1 \rightarrow |K_2 \times 3^n|_{(3^n-2)} \\ &= 1 \xrightarrow{[20]} K_2 = \frac{3^n - 1}{2} = \underbrace{111 \dots 1}_n \end{aligned} \quad (20)$$

$$\begin{aligned} K_3 &= |m_2^{-1}|_{m_3} \rightarrow |K_3 \times m_2|_{m_3} \\ &= 1 \rightarrow |K_3 \times (3^n - 1)|_{(3^n-2)} = 1 \rightarrow K_3 = 1 \end{aligned} \quad (21)$$

According to Eqs. 19, 20 and 21, the mixed-radix coefficients can be simplified as follows:

$$a_2 = |(x_2 - x_1)|_{(3^n-1)} \quad (22)$$

$$a_3 = |(x_3 - x_1) \times (111 \dots 1)_n - a_2|_{(3^n-2)} \quad (23)$$

Considering $T = (x_3 - x_1)$, and $(111 \dots 1)_n = 3^0 + 3^1 + \dots + 3^{(n-1)}$, Eq. 23 can be rewritten as

$$a_3 = \left| \underbrace{T^0 + T^1 + \dots + T^{(n-1)}}_{\alpha} - a_2 \right|_{(3^n-2)} \quad (24)$$

where T^i indicates the i -digit left shifting of T . To calculate a_3 , all of the terms must be reduced in modulo $(3^n - 2)$ based on Eq. 25.

$$\begin{aligned} &|(y_{2n-1} \dots y_{n+1} y_n y_{n-1} \dots y_1 y_0)|_{(3^n-2)} \\ &= 2(y_{2n-1} \dots y_{n+1} y_n) + (y_{n-1} \dots y_1 y_0) \end{aligned} \quad (25)$$

According to Eq. 25, two times of the n most significant digits of T^i 's should be added to the n least significant digits of T^i 's. In the process of α reduction to two n -digit numbers, if the length of any vector exceeds from n , the excess part should be treated in a similar way to Eq. 25. This process continues until the addition of these two vectors using an n -digit adder produces no carry.

4.1.1 A numerical example

Assume the moduli set $\{3^n - 2, 3^n - 1, 3^n\}$, with $n = 2$, and the weighted number is $271 = (101001)_3$. The symbol $( )_3$

indicates the value of the number in ternary. Accordingly, the residue number is given by:

$$x_1 = |327|_9 = 1 = (01)_3$$

$$x_2 = |327|_8 = 7 = (21)_3$$

$$x_3 = |327|_7 = 5 = (12)_3$$

By using Eqs. 15, 22 and 23, the mixed-radix coefficients can be calculated as

$$a_1 = 1 = (01)_3$$

$$a_2 = |(7-1)|_8 = 6 \xrightarrow{\text{Ternary}} a_2 = |(21-01)_3|_8 = (20)_3$$

$$a_3 = |(5-1) \times 4 - 6|_7 = 3 \xrightarrow{\text{Ternary}} a_3 = |((12-01) \times 11 - 12)_3|_7 = (10)_3$$

By substituting values in Eqs. 14 and 18, we have

$$X = 3 \times 9 \times 8 + 6 \times 9 + 1 = 271$$

$$\begin{aligned} X &= (((10)_3 \& (20)_3) - (10)_3) \& (01)_3 \\ &= ((1020)_3 - (10)_3) \& (01)_3 \\ &= (1010)_3 \& (01)_3 = (101001)_3 \end{aligned}$$

Therefore, the above results verify the correctness of the proposed reverse converter algorithm.

4.2 Hardware architecture of the proposed method

For hardware implementation, the proposed architecture can be divided into three parts. According to Eq. 26.

$$X = \underbrace{\left(\left(\overbrace{a_3}^{\text{Part II}} \& \overbrace{a_2}^{\text{Part I}} \right) - a_3 \right) \& x_1}_{\text{Part III}} \quad (26)$$

Part I calculates a_2 base on Eq. 22, which can be executed by an n -digit modulo $(3^n - 1)$, ternary adder, as shown in Fig. 2a. Based on Eq. 10, for the modulo $(3^n - 1)$, if the result is greater than or equal to the $(3^n - 1)$, the result will be added to the 3's complement of the modulo, which is $3^n - (3^n - 1) = 1$. To increase the parallelism of the modulo $(3^n - 1)$ ternary adder, the result and the same result plus one can be generated simultaneously, and the correct value can be selected by a multiplexer as shown in Fig. 2b. According to Eq. 11 the subtraction of two residues in $(3^n - 1)$ modulo can be rewritten as

$$|(x_2 - x_1)|_{(3^n - 1)} = |(x_2 + STI(x_1) + 1 - 1)|_{(3^n - 1)} = |x_2 + STI(x_1)|_{(3^n - 1)} \quad (27)$$

Thus, $-x_1$ can be realized by an STI, as shown in Fig. 2b.

Part II calculates a_3 base on Eq. 24. As shown in Fig. 3, first, $T = (x_3 - x_1)$ is computed by a regular n -digit ternary adder. Then, T^i , a_2 and the Const value are fed to a $2n$ -digit modulo $(3^n - 2)$ ternary carry save adder (CSA) tree, which reduces the Sum and Carry of the output to n digits. From Eq. 11 the subtraction of two residues in $(3^n - 2)$ modulo can be calculated as

Fig. 2 Hardware implementation of Part I; hardware schematic (a), hardware detailed design (b)

Fig. 3 Hardware implementation of Part II

$$\begin{aligned} |(x_3 - x_1)|_{(3^n-2)} &= |(x_3 + STI(x_1) + 1 - 2)|_{(3^n-2)} \\ &= |x_3 + STI(x_1) - 1|_{(3^n-2)} \end{aligned} \quad (28)$$

By applying Eq. 28 in Eq. 23, the subtraction can be transformed into addition as follow

$$\begin{aligned} a_3 &= |(x_3 - x_1) \times (111 \dots 1)_n - a_2|_{(3^n-2)} \\ &= |(x_3 + STI(x_1) - 1) \times (111 \dots 1)_n + STI(a_2) - 1|_{(3^n-2)} \\ &= |(x_3 + STI(x_1)) \times (111 \dots 1)_n + STI(a_2) - (111 \dots 1)_n - 1|_{(3^n-2)} \\ &= |(x_3 + STI(x_1)) \times (111 \dots 1)_n + STI(a_2) + ((3^n - 2) \\ &\quad - (111 \dots 1)_n - 1)|_{(3^n-2)} \\ &= \left| \underbrace{(x_3 + STI(x_1)) \times (111 \dots 1)_n}_T + STI(a_2) + \underbrace{(111 \dots 102)_n}_{Const} \right|_{(3^n-2)} \end{aligned} \quad (29)$$

The circuit corresponding to Eq. 29 is presented in Fig. 3.

The $2n$ -digit modulo $(3^n - 2)$ ternary CSA tree is a modified CSA based on Eq. 29. That is, the CSA minimum levels required for reducing $2n$ ternary digits to n ternary digits in modulo $(3^n - 2)$ depends on n , since the addition should be continued as long as no carry digits with the trit position higher than n remains. An example of the employed algorithm in modulo $(3^n - 2)$ ternary CSA tree is shown

Fig. 4 $2n$ -digit modulo $(3^n - 2)$ ternary algorithm

in Fig. 4. In each level of this CSA tree, first, the digits with trit positions more than n should be doubled and moved to the n lower position than its current position. Then, in every column, digits should be grouped and reduced to a *sum* and a *carry*. For the first level, the three inputs T^0 , $STI(a_2)$, and $Const$ have no digits with the bit position more than n . However, the inputs from T^0 to T^{n-1} have digits that exceed from position n , and each digit that exceeds from position n should be doubled and moved to the n lower position than its current position. After moving all digit to the right position of line n , the reduction can be performed on each column. The digits in columns could be grouped into packages with 2, 3, or 4 members for reduction, since we have ternary half-adder, 3-input full-adder, and 4-input full-adder for addition. The method is the same for every level, and will continue to the point that only one line of *Sum* and one line of *Carry* remain, like level 5. Then, we apply the method one more time, to make sure that no *Carry* with position higher than n remains (Fig. 4). Finally, the *Sum* and *Carry* are accumulated with a regular n -digit ternary adder (Fig. 3).

The final part (Part III) performs Eq. 26 by concatenating a_3 and a_2 , following by adding $-a_3$ using a regular $2n$ -digit ternary adder, and finally concatenating the obtained result with x_1 . Figure 5 shows the hardware implementation of this part. A regular $2n$ -digit ternary adder performs the addition. Since the a_3 bit length is n , and as $-a_3 = STI(a_3) + 1$, to perform the addition, we should concatenate $(22 \dots 2)_n$ with $STI(a_3)$ and implement the regular $2n$ -digit ternary adder with carry-in 1.

This implementation has less delay and remarkable performance compared to the reverse converter proposed in [20], since it uses only a single level of design, and employs fewer carry propagate adder in its critical path.

Fig. 5 Hardware implementation of Part III

Table 4 Trits coding by bits

Ternary value	Binary representation
0	00
1	01
2	10

Implementation of ternary architectures is currently feasible by mapping ternary structures into binary hardware. Therefore, each ternary digit is mapped into two bits, and the mapping method is shown in Table 4.

By employing this coding, any ternary arithmetic block can be mapped to binary coding. The fundamental blocks that are used in the implementation of the proposed design and the presented design in [20] include ternary half adder, ternary full adder, and the STI. The truth table of these blocks with binary coding are shown in Tables 5, 6 and 7, respectively.

To provide better-operating conditions for synthesis tools to perform the optimum synthesis, VHDL with/select signal assignment is used to describe the ternary half adder, and full adder. That is, the combination of inputs is used as the selector and the values of outputs are used as different cases. Furthermore, Table 7 indicates that the STI can be implemented on binary with a single binary XNOR on the bit with a higher bit position.

5 Simulation results

In this section, the performance of the proposed design, as well as its related counterpart, are explored and evaluated. Moreover, the impact of the proposed reverse converter is evaluated in box blur filter as a real-world implementation for a DSP system. The proposed ternary RNS reverse converter architecture is examined extensively and compared with the structure presented in [20], using VHDL code and

Table 5 Ternary half adder mapped to binary

Inputs		Outputs	
A	B	Sum	Carry
00	00	00	00
00	01	01	00
00	10	10	00
01	00	01	00
01	01	10	00
01	10	00	01
10	00	10	00
10	01	00	01
10	10	01	01

Table 6 Ternary full adder mapped to binary

Inputs			Outputs		Inputs			Outputs	
A	B	C _{in}	Sum	Carry	A	B	C _{in}	Sum	Carry
00	00	00	00	00	01	01	10	01	01
00	00	01	01	00	01	10	00	00	01
00	00	10	10	00	01	10	01	01	01
00	01	00	01	00	01	10	10	10	01
00	01	01	10	00	10	00	00	10	00
00	01	10	00	01	10	00	01	00	01
00	10	00	10	00	10	00	10	01	01
00	10	01	00	01	10	01	00	00	01
00	10	10	01	01	10	01	01	01	01
01	00	00	01	00	10	01	10	10	01
01	00	01	10	00	10	10	00	01	01
01	00	10	00	01	10	10	01	10	01
01	01	00	10	00	10	10	10	00	10
01	01	01	00	01					

Table 7 Standard ternary inverter (STI) mapped to binary

Input (a)	STI (a)
00	10
01	01
10	00

Xilinx Vivado software. First, the VHDL code functionality is verified by simulation. Then, it is synthesized and implemented on Xilinx Virtex Ultrascale+ family in 14 nm/16 nm FinFET technology. All these examinations have taken place on xcvu3p-ffvc1517-3-e device with the default strategy of Vivado for synthesis and implementation. The ternary circuit presented in [20] considered as only related work in our evaluation pool, as it outperformed the other related designs.

5.1 Performance evaluation

The performance of the two designs is examined for different values of n . The delay, dynamic power and area utilization of the proposed design and [20] after the implementation process are given in Table 8. Since the dynamic

power of a Xilinx FPGA is known as design power and static power is the power required for FPGA to operate normally, only dynamic power is considered for comparison [32].

As the bit-width n impacts the implementation of the design, the performance and energy-efficiency of the designs are evaluated with different bit-width, ranging from 5 up to 7. The delay, the dynamic power, area utilization, the PDP, and the EDP of the designs are plotted in Figs. 6a–c and 7a, b, respectively.

According to the results, the proposed design excels its counterpart [20], in all performance metrics except area utilization. Although the design suggested in [20] has a better area utilization, the power delay product (PDP) and energy-delay product (EDP), which reflect the energy efficiency of a given circuit, are reduced tremendously in the proposed design compared to the previous design. Moreover, by increasing the bit-width (n), the delay and EDP of the proposed design continue to decrease even more. The delay and EDP start decreasing from 11 and 33% in $n=5$ –19% and 41% in $n=7$. Also, PDP is decreased from 26% in $n=5$ –28% in $n=7$.

Table 8 Simulation results

Design	n	Delay (ns)	Dynamic power (mW)	Area utilization (CLB LUTs)	PDP (pJ)	EDP ($\times 10^{-18}$ J S)
Proposed design	5	7.29	50	376	365	2.66
Design of [20]	5	8.16	60	343	490	3.99
Proposed design	6	7.37	60	419	442	3.25
Design of [20]	6	8.38	71	438	595	4.99
Proposed design	7	8.05	72	655	580	4.67
Design of [20]	7	9.90	76	524	753	7.46

Fig. 6 Delay (a), dynamic power (b) and area utilization (c) of proposed design and the design of [20] for different n

Fig. 7 PDP (a) and EDP (b) of proposed design and the design of [20] for different n

5.2 Application evaluation

The proposed design is evaluated in a real-world implementation for a DSP system. The RNS systems are widely used in DSP systems. Digital image processing as a subcategory of the DSP systems has various filters that their performance could be improved by RNS [33]. Accordingly, to evaluate the performance of the proposed design and the design of [20], the box blur filter is implemented on both RNS systems with the same forward converter and arithmetic units. The box blur filter is a linear low-pass filter in image processing [34], which is frequently used to approximate a Gaussian blur for blurring and smoothing the image. A low-pass filter can be used to remove noises from the image. Filtering an image can be visualized by a convolving filter kernel on the image. A kernel is a small

matrix that filters a pixel of an image based on its neighbors. The box blur kernel filter can be written as [34]

$$\frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \quad (30)$$

According to Eq. 30, the kernel is well suited for ternary implementation, since the fraction part is power of 3 and can be performed by shifting the total number 2 times to the left, which is cost-free for the circuit implementation. The arithmetic unit of RNS consists of a 5-digit adder, a 2-trit counter, and a 5-digit register. In the first clock, the element (1, 1) of the matrix is added to the empty register and in each clock the next elements are added to the register. After nine clocks, the addition is complete and the result is delivered to the RNS reverse converter.

Table 9 Results and comparisons

Design	Pipelined	Delay (ns)	Dynamic power (mW)	Area utilization (CLB LUTs)	PDP (nJ)	EDP ($\times 10^{-18}$ J S)
Proposed design	No	28.06	68	676/114	1.9	53.3
Design of [20]	No	28.83	76	656/114	2.19	63.2
Proposed design	Yes	54.72	47	632/130	2.57	141
Design of [20]	Yes	58.38	55	555/124	3.21	187

Therefore, it takes nine clocks for the circuit to generate the right value of the filter. Thus, the proposed design and [20] in two methods can be analyzed. In the first method, we count the clock for the arithmetic unit and multiply it by nine and then, add the delay of the forward converter and proper reverse converter. In the second method, we could attach the forward converter, arithmetic unit, and proper reverse converter and pipeline them, in order to have one clock. The final value of the clock multiplied by 9 is the total delay of circuits. The delay, dynamic power, and area utilization of the box blur filter using the proposed design and [20] after the implementation process are given in Table 9.

The simulation results of this experiment indicate that the performance metrics except the area utilization of the proposed design outperforms the design presented in [20] in all the methods and implementations.

6 Conclusion

An innovative fast and energy-efficient RNS reverse converter for the moduli set $\{3^n - 2, 3^n - 1, 3^n\}$ have been proposed. The proposed design not only performs the processes with neither memory nor multiplier but also reduces the usage of adders with large bit-widths. The circuit evaluations validate that the new design surpasses the previous related design in terms of performance and energy efficiency, which are the main requirements of any digital systems, especially DSP systems.

Compliance with ethical standards

Conflict of interest The authors declare that they have no competing interests.

References

- Hayes B (2001) Third base. *Am Sci* 89(6):490–494
- Moaiyeri MH, Taheri ZM, Khezeli MR, Jalali A (2019) Efficient passive shielding of MWCNT interconnects to reduce crosstalk effects in multiple-valued logic circuits. *IEEE Trans Electromagn Compat* 61(5):1593–1601
- Khezeli MR, Moaiyeri MH, Jalali A (2017) Analysis of crosstalk effects for multiwalled carbon nanotube bundle interconnects in ternary logic and comparison with Cu interconnects. *IEEE Trans Nanotechnol* 16(1):107–117
- Srinivasuand B, Sridharan K (2017) A synthesis methodology for ternary logic circuits in emerging device technologies. *IEEE Trans Circuits Syst I Regul Pap* 64(8):2146–2159
- Tabrizchi S, Taheri M, Navi K, Bagherzadeh N (2019) Novel CNFET ternary circuit techniques for high-performance and energy-efficient design. *IET Circuits Devices Syst* 13(2):193–202
- Hamedani SG, Moaiyeri MH (2019) Impacts of process and temperature variations on the crosstalk effects in sub-10 nm multilayer graphene nanoribbon interconnects. *IEEE Trans Device Mater Reliab* 19(4):630–641
- Hurst SL (1984) Multiple-valued logic; its status and its future. *IEEE Trans Comput C-33*(12):1160–1179
- Parhami B, McKeown M (2013) Arithmetic with binary-encoded balanced ternary numbers. In: 2013 Asilomar conference on signals, systems and computers, Asilomar
- Chang C-H, Molahosseini AS, Zarandi AAE, Tay TF (2015) Residue number systems: a new paradigm to datapath optimization for low-power and high-performance digital signal processing applications. *IEEE Circuits Syst Mag* 15(4):26–44
- Stouraitis T, Paliouras V (2001) Considering the alternatives in low-power design. *IEEE Circuits Devices Mag* 7(4):23–29
- Omondi AR, Premkumar B (2007) Residue number systems: theory and implementations. Imperial College Press, London
- Molahosseini AS, de Sousa LS, Chang C-H (2017) Embedded systems design with special arithmetic and number systems. Springer International Publishing AG, Cham
- Szabo N, Tanaka R (1967) Residue arithmetic and its applications to computer technology. McGraw-Hill, New York
- Jenkins W, Leon B (1977) The use of residue number systems in the design of finite impulse response digital filters. *IEEE Trans Circuits Syst* 24(4):191–201
- Soderstrand M (1977) A high-speed low-cost recursive digital filter using residue number arithmetic. *Proc IEEE* 65(7):1065–1067
- Taylor FJ, Papadourakis G, Skavantzios A, Stouraitis A (1985) A radix-4 FFT using complex RNS arithmetic. *IEEE Trans Comput C-34*(6):573–576
- Valliammal SR, Palaniswami S (2016) Multiplier design utilizing tri valued logic for RLNS based DSP applications. *Circuits Syst* 7(04):417–433
- Abdallah M, Skavantzios A (2005) On multimoduli residue number systems with moduli of forms $r/\text{sup } a/$, $r/\text{sup } b/-1$, $r/\text{sup } c/+1$. *IEEE Trans Circuits Syst I Regul Pap* 52(7):1253–1266
- Hosseinzadeh M, Navi K (2007) A new moduli set for residue number system in ternary valued logic. *J Appl Sci* 7(23):3729–3735

20. Sabbagh A, Hosseinzadeh M, Navi K (2008) A multiplier-free residue to weighted converter for the moduli set $\{3n - 2, 3n - 1, 3n\}$. *Contemp Eng Sci* 1(2):71–80
21. Georgiou HV (2016) On the optimality of ternary arithmetic for compactness. arXiv preprint arXiv, vol. 1611.03715
22. Jones DW (2012) Fast ternary addition. The University of Iowa Department of Computer Science, 1 April 2012. [Online]. <http://homepage.divms.uiowa.edu/~jones/ternary/arith.shtml>
23. Toulabinejad M, Taheri M, Navi K, Bagherzadeh N (2019) Toward efficient implementation of basic balanced ternary arithmetic operations in CNFET technology. *Microelectron J* 90:267–277
24. Dhande AP, Ingole VT, Ghiye VR (2014) Ternary digital system: concepts and applications. SM Online Publishers LLC
25. Navi K, Doostaregan A, Moaiyeri MH, Hashemipour O (2011) A hardware-friendly arithmetic method and efficient implementations for designing digital fuzzy adders. *Fuzzy Sets Syst* 185(1):111–124
26. Moaiyeri MH, Doostaregan A, Navi K (2011) Design of energy-efficient and robust ternary circuits for nanotechnology. *IET Circuits Devices Syst* 5(4):285–296
27. Taheri M, Khani E, Esmaeildoust M, Navi K (2012) Efficient reverse converter design for five moduli set $\{2n, 22n + 1 - 1, 2n/2 - 1, 2n/2 + 1, 2n + 1\}$. *J Comput Model* 2(1):93–108
28. Taheri M, Shafiee N, Esmaeildoust M, Amirjamshidi Z, Sabbaghinadooshan R, Navi K (2015) A high speed residue-to-binary converter for balanced 4-moduli set. *J Comput Secur* 2(1):43–54
29. Khademolhosseini H, Roohi A (2011) A new redundant method on representing numbers with moduli set $\{3n, 3n - 1, 3n - 2\}$. In: 2011 International conference on computer, communication and electrical technology (ICCCET), Tamilnadu
30. Khademolhosseini H, Hosseinzadeh M (2011) A robust redundant residue representation in residue number system with moduli set $(rn-2, rn-1, rn)$. *World Acad Sci Eng Technol Int J Math Comput Phys Electr Comput Eng* 5(8):1170–1175
31. Ghosh A, Sinha A (2018) FPGA implementation of RNS adder based MAC unit in ternary value logic domain for signal processing algorithm and its performance analysis. In: IEEE electron devices Kolkata conference (EDKCON), Kolkata, 2018
32. Corporation X (2018) Vivado design suite user guide: power analysis and optimization (UG907). 20 December 2018. [Online]. https://www.xilinx.com/support/documentation/sw_manuals/xilinx2018_3/ug907-vivado-power-analysis-optimization.pdf
33. Chervyakov NI, Pavel AL, Nikolai NN, Dmitrii IK, Alexander SV, Danil VB (2019) Implementation of smoothing image filtering in the residue number system. In: 2019 8th Mediterranean conference on embedded computing (MECO). IEEE, pp 1–4
34. Shih FY (2010) Image processing and pattern recognition: fundamentals and techniques. Wiley-IEEE Press, Hoboken

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.