Tabriz Journal of Electrical Engineering (TJEE), vol. 56, no. 1, 2026 Serial no. 115
DOI: 10.22034/tjee.2026.67364.5023

FPGA Implementation of a Multiplier-Free Ternary RNS Reverse Converter for a new 4-Moduli Set $\{3^n, 3^n-2, 3^n+2, 3^n-1\}$ Using Binary-Encoded Ternary Logic

Javad Ahsan, Ebrahim Farshidi*, Gholamreza Akbarizadeh

Department of Electrical Engineering/ Faculty of Engineering, Shahid Chamran University of Ahvaz, Ahvaz, Iran
E-mail addresses: j-ahsan@stu.scu.ac.ir (Javad Ahsan), farshidi@scu.ac.ir (Ebrahim Farshidi), g.akbari@scu.ac.ir (Gholamreza Akbarizadeh)
*Corresponding author

Received: 19/05/2025, Revised 02/12/2025, Accepted: 18/02/2026.

Abstract
In this paper, we present a novel FPGA-based implementation of a multiplier-free reverse converter for the residue number system (RNS) using a new 4-moduli set $\{3^n, 3^n-2, 3^n+2, 3^n-1\}$. The proposed method leverages the unbalanced ternary logic system to achieve efficient residue-to-binary conversion with reduced computational complexity and hardware resource usage. The core architecture eliminates the need for hardware multipliers by exploiting mathematical simplifications and residue properties inherent to the chosen moduli set. Ternary digits are encoded in binary to enable implementation on conventional FPGA platforms. The design is suitable for high-speed applications. Since no comparable ternary RNS reverse converter exists in the current literature, a baseline method was implemented for evaluation. Simulation and synthesis results confirm the efficiency of the proposed design in terms of area, speed, and power consumption, making it a promising solution for low-power, high-performance signal processing and cryptographic systems.

Keywords
Residue number system (RNS), reverse converter, ternary logic systems, modular arithmetic.

1. Introduction

The residue number system (RNS) is a non-weighted number system that enables parallel, carry-free arithmetic by representing numbers using a set of pairwise co-prime moduli [1]. This unique property makes RNS particularly attractive for high-speed and low-power applications such as digital signal processing (DSP) [2], cryptography [3], and neural network accelerators [4]. A critical component of any RNS-based system is the reverse converter [5], which reconstructs the original number from its residues. The efficiency of this converter significantly impacts the overall performance and practicality of the RNS system.

In recent years, the exploration of ternary logic systems has gained traction due to their potential to reduce interconnects, increase data density, and lower power consumption [6]. Combining ternary logic with RNS offers an opportunity to develop highly efficient computational architectures.

Furthermore, the adoption of ternary arithmetic offers several structural and computational advantages for RNS architectures. Each ternary digit in a given base b carries $\log_2 b$ bits of information. Thus, each trit carries almost 1.6 bits of information [7], resulting in shorter numerical representations and fewer arithmetic steps during reverse conversion. This reduces the number of required adders, shifters, and modular operations in the MRC process. Moreover, ternary-based moduli exhibit simpler reduction properties, enabling multiplier-free reconstruction with lower logic complexity. The ternary (base-3) number system has recently attracted considerable attention due to its suitability for emerging nano electronic technologies such as CNTFETs [8] and GNRFETs [9], which naturally support multi-level logic behaviour. These technologies motivate the exploration of ternary-based arithmetic structures in RNS architectures. Recent research has demonstrated considerable progress in leveraging these technologies to realize efficient RNS-based architectures [10].

For instance, the authors in [11] presented a CNTFET-based ternary multiplexer demonstrating significant improvements in delay, power consumption, and PDP, highlighting the suitability of CNTFET devices for multi-valued logic implementations in modern VLSI systems. Similarly, the work in [12] introduced a hybrid forward/reverse RNS converter architecture, aiming to reduce hardware overhead by reusing computational blocks, which underlines the importance of efficient reverse converter design in practical RNS-based processors.

This paper explores the optimization of reverse conversion techniques in RNS, emphasizing their

significance in high-performance cryptographic systems while considering the potential of ternary logic and next-generation transistor technologies for practical implementation. This work proposes an efficient multiplier-free reverse converter for a new 4-moduli RNS set defined as $\{3^n, 3^n-2, 3^n+2, 3^n-1\}$. The proposed moduli set offers a favourable dynamic range and is well-suited to ternary logic representation. A key advantage of this design is the elimination of hardware multipliers, which are typically resource-intensive and power-hungry in FPGA implementations [13]. Instead, we employ modular arithmetic simplifications to optimize the design. Although the proposed converter is designed using ternary logical principles, current FPGA architectures do not support native ternary hardware. Therefore, all ternary operations were implemented using binary-encoded ternary digits (two-bit encoding per trit). To the best of our knowledge, no prior work has addressed the implementation of a reverse converter for a 4-moduli set using ternary logic. Therefore, we have developed a baseline binary-compatible converter to compare performance metrics. The simulation results validate the proposed architecture in terms of speed, area, and power consumption, highlighting its applicability for high-speed and energy-efficient VLSI systems.

Recent studies have shown that ternary RNS architectures can significantly reduce hardware complexity and simplify modular arithmetic, especially through multiplier-free reconstruction techniques and high-radix operations. For example, [14] demonstrated that the ternary 3-moduli set $\{3^n, 3^n-1, 3^n-2\}$ enables efficient implementation with lower logic cost and improved dynamic-range utilization. Similarly, [7] reported notable improvements in delay and power for the same 3-moduli set using optimized MRC-based FPGA realizations.

However, all existing ternary RNS reverse converters are limited to 3-moduli sets, and no research has explored a 4-moduli ternary RNS architecture. Moreover, there is no reported multiplier-free or FPGA-oriented reverse converter for any 4-moduli ternary set.

Therefore, a clear research gap exists in developing a low-complexity, multiplier-free, reverse converter for a 4-moduli ternary RNS set an issue that this work directly addresses.

The main contributions of this paper include the proposal of a novel four-moduli set $\{3^n, 3^n-2, 3^n+2, 3^n-1\}$, specifically designed for ternary-based computation. The proposed moduli set achieves an extended dynamic range while maintaining minimal hardware complexity. Additionally, a multiplier-free reverse converter architecture is introduced. This converter eliminates the need for hardware multipliers by utilizing mathematical transformations and the modular properties of the proposed moduli set, resulting in a highly efficient hardware implementation.

The rest of the paper is organized as follows**:** Section 2 provides the necessary mathematical background on the residue number system (RNS), the mixed radix conversion (MRC) method, and ternary logic fundamentals. Section 3 details the proposed reverse converter architecture, including the mathematical formulation and hardware mapping. Section 4 presents the FPGA implementation results, including simulation and synthesis metrics. Finally, Section 5 concludes the paper.

2. Mathematical Background

2.1. Residue number system

The residue number system (RNS) [15] is a non-weighted number system representing a large integer as a set of smaller integers called residues. These residues are calculated with respect to a set of pairwise co-prime moduli. Given a set of k moduli $\{m_1, m_2, \ldots, m_k\}$, where each m_i is a positive integer and $GCD(m_i, m_j) = 1$ for all $i \neq j$, any integer X in the range $[0, M-1]$, where $M = m_1 m_2 \ldots . m_k$, can be uniquely represented as:

$$X \equiv (x_1, x_2, \ldots, x_k)\left(mod\,(m_1, m_2, \ldots, m_k)\right) \tag{1}$$

where each residue $x_i = X \bmod m_i$.

RNS offers several advantages, particularly for high speed and parallel computation. Since each residue is computed independently, arithmetic operations such as addition, subtraction, and multiplication can be performed on the residues in parallel without carry propagation between digits. This carry-free nature leads to faster computation and better scalability, especially in applications like digital signal processing (DSP), cryptography, and fault-tolerant systems.

2.2. Mixed radix conversion

Mixed radix conversion (MRC) [16] is a method used to reconstruct an integer from its residues in the residue number system (RNS). The MRC avoids modular multiplications, making it suitable for hardware implementations where reducing the complexity of arithmetic operations is crucial.

In MRC, a number X is represented by its residues $(x_1, x_2, \ldots, x_k)$ with respect to moduli set $\{m_1, m_2, \ldots, m_k\}$. The reconstruction of X is done sequentially by calculating mixed radix digits through the following steps:

$$v_1 = x_1 \tag{2}$$

$$v_2 = |(x_2 - v_1) \mid m_1 |^{-1}_{m_2}|_{m_2} \tag{3}$$

$$v_k = \left| \begin{pmatrix} \left((x_k - v_1) \left| m_1^{-1} \right|_{m_k} - v_2 \right) \left| m_2^{-1} \right|_{m_k} \\ -\ldots - v_{k-1} \end{pmatrix} \left| m_{k-1}^{-1} \right|_{m_k} \right|_{m_k} \tag{4}$$

Thus, the weighted number is calculated as follows:

$$X = v_k \prod_{i=1}^{k-1} m_i + \ldots + v_3 m_2 m_1 + v_2 m_1 + v_1. \tag{5}$$

In this work, the MRC method forms the basis for developing a multiplier-free reverse converter for the proposed ternary RNS moduli set.

2.3. Ternary logic system

Ternary logic is a form of multi-valued logic that operates with three distinct logic levels instead of the conventional two levels used in binary systems [7]. In this work, the unbalanced ternary system is adopted, where the logic values are represented by the digits $\{0,1,2\}$. This form is more suitable for practical hardware implementation due to its simplicity in logic gate design and signal representation [14].

The basic arithmetic components in ternary logic circuits for any computational operation are ternary half adders and ternary full adders. The typical behaviour of a ternary half adder with inputs *M*, *N* is defined in Table I.

Table I: Ternary half adder

M	N	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

Another important gate in ternary logic design is the STI (standard ternary inverter). The STI is a fundamental unary operator in ternary logic, and its function is to invert a ternary value based on predefined logic [17]. The typical behaviour of an STI is defined in Table II.

Table II: Standard ternary inverter

Input(x)	STI(x)
0	2
1	1
2	0

While ternary logic offers theoretical advantages, it cannot be directly implemented with standard CMOS-based hardware due to the lack of multi-level support in conventional FPGAs and ASICs. Therefore, in this work, ternary logic operations are mapped to binary logic using encoded representations of ternary digits, allowing ternary RNS arithmetic to be realized efficiently on existing binary hardware while maintaining its computational benefits. Table III demonstrates the mapped binary representation.

Table III: Ternary to binary mapping method

Ternary value	binary representation
0	00
1	01
2	10

Accordingly, each ternary digit (trit) is encoded using two binary bits to enable implementation on standard binary hardware.

3. Proposed Method

To utilize the four-moduli set $\{3^n, 3^n-2, 3^n+2, 3^n-1\}$, after verifying the pairwise co-primeness of the moduli, the necessary computations for designing a residue number system (RNS) that can be efficiently implemented on suitable hardware are investigated.

3.1. Proof of Coprimality

The Euclidean algorithm is used to determine the greatest common divisor (GCD) of two numbers. The GCD of two numbers *a* and *b* is denoted as GCD (*a, b*). To prove that two numbers are co-prime, their GCD must be 1. Thus, the pairwise co-primeness of the proposed moduli set is verified as follows:

$$GCD(3^n-1, 3^n) = GCD(3^n-1, 1) = 1 \quad (6)$$

$$GCD(3^n-1, 3^n+2) = GCD(3^n-1, 3) = 1 \quad (7)$$

$$GCD(3^n-1, 3^n-2) = GCD(3^n-2, 1) = 1$$

$$(8)\ GCD(3^n, 3^n+2) = GCD(3^n, 2) = 1$$

$$(9)\ GCD(3^n, 3^n-2) = GCD(3^n-2, 2) = 1$$

$$(10)\ GCD(3^n-2, 3^n+2) = GCD(3^n-2, 4) = 1$$

$$(11)$$

Based on the above relations, it can be concluded that the proposed moduli set consists of pairwise co-prime moduli. Therefore, designing a reverse converter for this moduli set is feasible.

3.2. Proposed reverse converter

As previously mentioned, the design of a reverse converter is the most critical part of an RNS, as it poses significant challenges. To reduce system complexity, a two-level design approach is adopted to compute the weighted number. The first level employs the moduli set $\{3^n-2, 3^n\}$ and the mixed-radix conversion (MRC) algorithm, while the second level computes the final weighted number using the MRC algorithm again, this time with the moduli set $\{3^n(3^n-2), 3^n+2, 3^n-1\}$.

Figure 1 illustrates the overall design scheme.

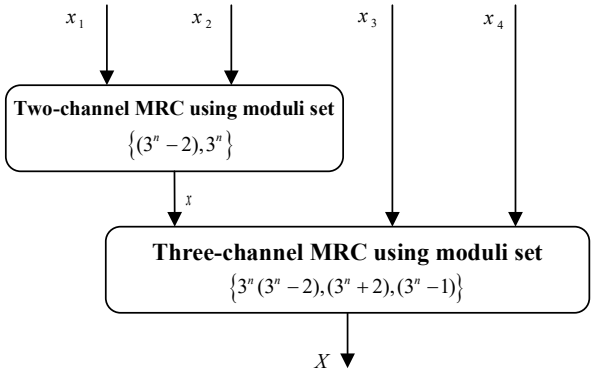

Fig 1. Two-level design of the four-modulus reverse converter

3.2.1. Level 1: Computation of x

Consider the moduli $P_1 = 3^n - 2, P_2 = 3^n$, along with their corresponding residue (x_1, x_2) . The MRC algorithm is used to compute *x* as follows:

$$x = v_2 P_1 + v_1 \tag{12}$$

where the coefficients *v* are computed as:

$$v_1 = x_1 \tag{13}$$

$$v_2 = |(x_2 - v_1)\,|\,P_1\,|_{P_2}^{-1}|_{P_2} \tag{14}$$

v_2 is computed as follows:

$$v_2 = |(x_2 - x_1)\,|\,3^n - 2\,|_{3^n}^{-1}|_{3^n} \tag{15}$$

The required multiplicative inverse is computed as:

$$|\,3^n - 2\,|_{3^n}^{-1} \rightarrow |\,k \times 3^n - 2\,|_{3^n}^{-1} = 1 \rightarrow k = \frac{3^n - 1}{2} \tag{16}$$

Here, *k* can be expressed as:

$$k = \frac{3^n - 1}{2} = \left\{3^0 + 3^1 + ... + 3^{n-1}\right\} \tag{17}$$

Thus, equation (15) becomes:

$$\begin{aligned} v_2 &= |(x_2 - x_1)(\frac{3^n - 1}{2})\,|_{3^n} = \\ &= |\,M \times (3^0 + 3^1 + ... + 3^{n-1})\,|_{3^n} \end{aligned} \tag{18}$$

where:

$$M = |x_2 - x_1|_{3^n} = |x_2 + STI\,(x_1) + 1|_{3^n} \tag{19}$$

$$\begin{aligned} v_2 &= \left|M^0 + M^1 + ... + M^{n-1}\right|_{3^n} = \\ &= \left|M + LS\,(M,1) + ... + LS\,(M,n-1)\right|_{3^n} \end{aligned} \tag{20}$$

where *LS*(*M*,*n*) represents an *n*-bit left circular shift of *M*. Since the final result is computed modulo 3^n, only the *n* least significant bits (LSBs) of *M* are needed, as higher bits are multiples of 3^n. Thus, additions and shifts can ignore carry propagation beyond the *n*-th bit.
Finally, *x* is computed using equation (12):

$$\begin{aligned} x &= x_1 + (3^n - 2) \times v_2 = x_1 + 3^n v_2 - 2v_2 = \\ &= x_1 + 3^n v_2 - (v_2 + v_2) = \\ &= v_2 \,\&\, x_1 + STI\,(v_2 + v_2) + 1 \end{aligned} \tag{21}$$

The block diagram for computing x is shown in figure 2.

3.2.2. Level 2: Computation of the final weighted number X

The second-level computation is performed by considering the moduli $P_1 = 3^n(3^n - 2)$, $P_2 = 3^n + 2$, $P_3 = 3^n - 1$ with corresponding residues (x, x_3, x_4) . The MRC algorithm is again applied to compute the final weighted number *X*:

$$X = v_3 P_2 P_1 + v_2 P_1 + v_1 \tag{22}$$

where the coefficients *v* are computed as:

$$v_1 = x \tag{23}$$

$$v_2 = |(x_3 - v_1)\,|\,P_1\,|_{P_2}^{-1}|_{P_2} \tag{24}$$

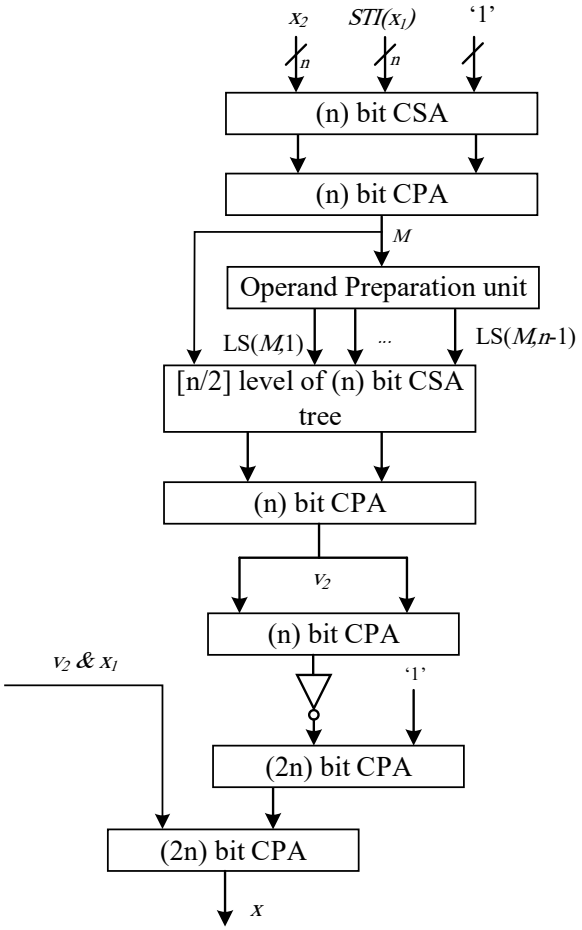

Fig 2. Block diagram for computing *x*

$$v_3 = |((x_4 - v_1)\,|\,P_1\,|_{P_3}^{-1} - v_2)\,|\,P_2\,|_{P_3}^{-1}|_{P_3} \tag{25}$$

v_2 is computed as follows:

$$\begin{aligned} v_2 &= |(x_3 - x)\,|\,3^n(3^n - 2)\,|_{3^n+2}^{-1}|_{3^n+2} = \\ &|T \times |\,3^n(3^n - 2)\,|_{3^n+2}^{-1}|_{3^n+2} \end{aligned} \tag{26}$$

where:

$$\begin{aligned} T &= |\,x_3 - x\,|_{3^n+2} = |x_3 + STI\,(x) + 1 + 2|_{3^n+2} = \\ &= |x_3 + STI\,(x) + 3|_{3^n+2} \end{aligned} \tag{27}$$

The required multiplicative inverse is computed as:

$$|\,3^n(3^n - 2)\,|_{3^n+2}^{-1} =$$

$$\begin{cases} if \rightarrow n = 2k \quad \rightarrow k = 1,2,3,... \rightarrow 3^0 + 3^n - \sum_{j=n-1}^{1(odd)} 3^j \\ if \rightarrow n = 2k+1 \rightarrow k = 1,2,3,... \rightarrow 3^1 - 3^0 + \sum_{j=3}^{n(odd)} 3^{j-1} \end{cases} \tag{28}$$

Thus, equation (26) is computed in two cases:

Case 1: n=2k (even)

$$\begin{aligned} v_2 &= |T \times |\, 3^n(3^n-2)\,|_{3^n+2}^{-1}|_{3^n+2} = \\ &= |T \times (3^0+3^n-3^{n-1}-3^{n-3}-3^{n-5}-\ldots-3^1)\,|_{(3^n+2)} = \\ &= \left| \begin{pmatrix} T + LS(T,n) - LS(T,n-1) - LS(T,n-3) \\ -LS(T,n-5) - \ldots - LS(T,1) \end{pmatrix} \right|_{3^n+2} \end{aligned} \quad (29)$$

Case 2: n=2k+1 (odd)

$$\begin{aligned} v_2 &= \left| T \times \left| 3^n\left(3^n-2\right) \right|_{3^n+2}^{-1} \right|_{3^n+2} = \\ &= \left| T \times \left(3^1-3^0+3^2+3^4+3^6+\ldots+3^n\right) \right|_{3^n+2} = \\ &= \left| \begin{pmatrix} LS(T,1) - T + LS(T,2) + LS(T,4) \\ +LS(T,6) - \ldots - LS(T,n) \end{pmatrix} \right|_{3^n+2} \end{aligned} \quad (30)$$

For modulo (3^n+2) reduction, the following algorithm is used to handle overflow beyond n bits [18]:

$$\begin{aligned} &\left| a_{2n-1}a_{2n-2}\ldots a_{n+1}a_n a_{n-1}\ldots a_1 a_0 \right|_{3^n+2} = \\ &= \left| \left(a_{2n-1}a_{2n-2}\ldots a_{n+1}a_n\right) \times 3^n + \left(a_{n-1}\ldots a_1 a_0\right) \right|_{3^n+2} = \\ &= \left| \left(a_{2n-1}a_{2n-2}\ldots a_{n+1}a_n\right) \times (3^n+2-2) + \left(a_{n-1}\ldots a_1 a_0\right) \right|_{3^n+2} = \\ &= \left(a_{n-1}\ldots a_1 a_0\right) - 2 \times \left(a_{2n-1}a_{2n-2}\ldots a_{n+1}a_n\right) \end{aligned} \quad (31)$$

The block diagram for computing v_2 is shown in figure 3.

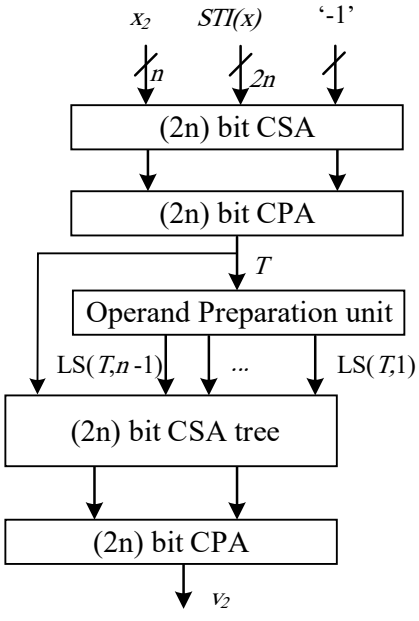

Fig 3. Block Diagram for Computing v_2

v_3 is computed by considering equation (25) as follows:

$$v_3 = \left| \begin{matrix} ((x_4 - x)\,|\,3^n(3^n-2)\,|_{3^n-1}^{-1} - v_2) \\ \times\,|\,3^n+2\,|_{3^n-1}^{-1} \end{matrix} \right|_{3^n-1} \quad (32)$$

The required multiplicative inverses are computed as:

$$\begin{aligned} &|\,3^n(3^n-2)\,|_{3^n-1}^{-1} \rightarrow |\,k_2 \times 3^n(3^n-2)\,|_{3^n-1} = 1 \\ &\rightarrow k_2 = -1 \end{aligned} \quad (33)$$

$$\begin{aligned} &\left| 3^n+2 \right|_{3^n-1}^{-1} \rightarrow \left| k_3 \times (3^n+2) \right|_{3^n-1} = 1 \\ &\rightarrow k_3 = 3^{n-1} \end{aligned} \quad (34)$$

Thus, equation (32) becomes:

$$v_3 = \left| (x - x_4 - v_2)3^{n-1} \right|_{3^n-1} = \left| E \times 3^{n-1} \right|_{3^n-1} \quad (35)$$

where:

$$\begin{aligned} E &= \left| x - x_4 - v_2 \right|_{3^n-1} = \left| x - (x_4 + v_2) \right|_{3^n-1} = \\ &= \left| x + STI\left(\overset{n}{00\ldots0} \,\&\, \left(x_4 + v_2\right) \right) \right|_{3^n-1} \end{aligned} \quad (36)$$

For modulo (3^n-1) reduction, the following algorithm is used [18]:

$$\begin{aligned} &\left| a_{2n-1}a_{2n-2}\ldots a_{n+1}a_n a_{n-1}\ldots a_1 a_0 \right|_{3^n-1} = \\ &= \left| \left(a_{2n-1}a_{2n-2}\ldots a_{n+1}a_n\right) \times 3^n + \left(a_{n-1}\ldots a_1 a_0\right) \right|_{3^n-1} = \\ &= \left| \left(a_{2n-1}a_{2n-2}\ldots a_{n+1}a_n\right) \times (3^n-1+1) + \left(a_{n-1}\ldots a_1 a_0\right) \right|_{3^n-1} = \\ &= \left(a_{2n-1}a_{2n-2}\ldots a_{n+1}a_n\right) + \left(a_{n-1}\ldots a_1 a_0\right) \end{aligned} \quad (37)$$

Thus, v_3 is computed as:

$$v_3 = \left| E \times 3^{n-1} \right|_{3^n-1} = \left| CLS\,(E, n-1) \right|_{3^n-1} \quad (38)$$

where $CLS\,(\cdot)$ denotes a circular left shift. The block diagram for computing v_3 is shown in figure 4.

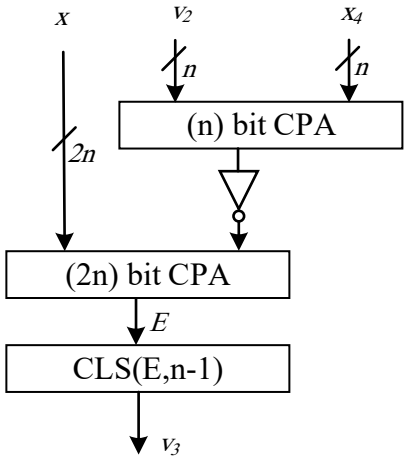

Fig 4. Block diagram for computing v_3

Finally, the weighted number X is computed using equation (22):

$$\begin{aligned} X &= v_3 P_2 P_1 + v_2 P_1 + v_1 = \\ &= 3^n\left(3^n-2\right)\left(3^n+2\right)v_3 + 3^n\left(3^n-2\right)v_2 + x = \\ &= \left(3^{3n} - 4\times 3^n\right)v_3 + 3^{2n}v_2 - 2\times 3^n v_2 + x = \\ &= v_3 \,\&\, v_2 \,\&\, x - 3^n\left(4\times v_3 + 2\times v_2\right) \end{aligned} \quad (39)$$

This concludes the design of the reverse converter for the proposed four-modulus set. The two-level approach ensures computational efficiency and hardware optimization.

3.3. Numerical example

If n=3, the moduli set will be {25,27,29,26}. Given the weighted number 75981, the residues will be as follows:

$$x_1 = |75981|_{25} = (6)_{10} = (020)_3$$

$$x_2 = |75981|_{27} = (3)_{10} = (010)_3$$

$$x_3 = |75981|_{29} = (1)_{10} = (001)_3$$

$$x_4 = |75981|_{26} = (9)_{10} = (100)_3$$

Note that $|A|_B$ donate A mod B and $(C)_D$ donate the number C in base D.

Now we use the proposed method to calculate the weighted number from the corresponding residues.

Calculation of the first level is as follows:

$$v_1 = x_1 = (020)_3$$

$$M = |x_2 - x_1|_{27} = |(010) - (020)|_{27} = (220)_3$$

$$v_2 = |M + LS(M,1) + LS(M,2)|_{27} =$$
$$|(220)_3 + (200)_3 + (000)_3|_{27} = (120)_3$$

$$x = v_2 \,\&\, x_1 + STI(v_2 + v_2) + 1 = (112010)_3$$

Now we can calculate the final weighted number via second level calculation as fallows:

$$v_1 = x = (112010)_3$$

$$T = |x_3 - x|_{29} = (222)_3$$

$$v_2 = \left|T \times \left(3^1 - 3^0 + 3^2\right)\right|_{29} = |LS(T,1) - T + LS(T,2)|_{29}$$

$$= \left|(2220)_3 - (222)_3 + (22200)_3\right|_{29} = (221)_3$$

$$E = |x - x_4 - v_2|_{26} =$$

$$\left|(112010)_3 - (100 + 221)_3\right|_{26} = (100)_3$$

$$v_3 = |CLS(E,2)|_{26} = (010)_3$$

$$X = v_3 \,\&\, v_2 \,\&\, x - 3^n \left(4 \times v_3 + 2 \times v_2\right) =$$

$$= (010221112010)_3 - (2022000) =$$

$$= (10212020010)_3 = (75981)_{10}$$

Table IV: Performance comparison of the proposed and baseline reverse converters for different values of *n*.

Design	n	Delay (ns)	Area Utilization (CLB LUTs)	Dynamic Power (mW)	EDP ($\times 10^{-18}$ Js)	FOM (1 /(ns×LUT×mW)) $\times 10^{-6}$
Proposed method	4	12.32	177	20	3.04	22.93
Baseline method	4	31.70	1108	21	21.10	1.36
Proposed method	5	15.78	296	27	6.72	7.93
Baseline method	5	35.08	1492	34	41.84	0.56
Proposed method	6	16.12	357	33	8.58	5.27
Baseline method	6	37.52	1626	40	56.31	0.41

4. Simulation results

To evaluate the performance of the proposed multiplier-free ternary RNS reverse converter, a custom baseline implementation was developed using conventional reverse conversion techniques.

The baseline design relies on mixed radix conversion (MRC) and modular multiplication implemented in binary logic. Since, to the best of our knowledge, no existing design in the literature addresses a reverse converter for a 4-moduli set, particularly using ternary logic, this baseline was created to enable a meaningful comparison and highlight the advantages of the proposed architecture.

The proposed and baseline designs were described in VHDL and synthesized using Xilinx Vivado 2018, targeting a Virtex UltraScale+ xcvu3p-ffvc1517-3-e. Functional simulation was carried out using Vivado simulator, verifying correctness across the entire dynamic range. Due to the lack of native ternary hardware support, all ternary operations were mapped into binary using a 2-bit encoding per trit. The design leveraged binary logic to emulate ternary structures such as adders and the standard ternary inverter (STI), ensuring compatibility with commercial FPGAs.

Although the proposed design was not deployed on a physical FPGA board, the complete hardware implementation flow was performed, including synthesis, placement and routing, static timing analysis, and power estimation using Xilinx Vivado. Because the architecture is fully combinational and does not involve memory elements or I/O interaction, on-board testing does not provide additional insight beyond the implementation reports generated by the FPGA design tools.

Therefore, all reported delay, area, and power values correspond to the post-implementation results, which accurately reflect the real hardware performance of the proposed architecture on the target FPGA device.

4.1. Performance Evaluation

To evaluate the efficiency of the proposed reverse converter architecture, we synthesized and simulated both the proposed method and a baseline binary-compatible design for different values of *n* (4, 5 and 6). The results, summarized in Table IV, include delay, area utilization, dynamic power consumption, and energy-delay product (EDP), which is a key figure of merit for energy efficiency. To enable a unified comparison between

theproposed architecture and the baseline design, a Figure of Merit (FOM) [19] was defined that jointly considers delay, area, and dynamic power consumption:

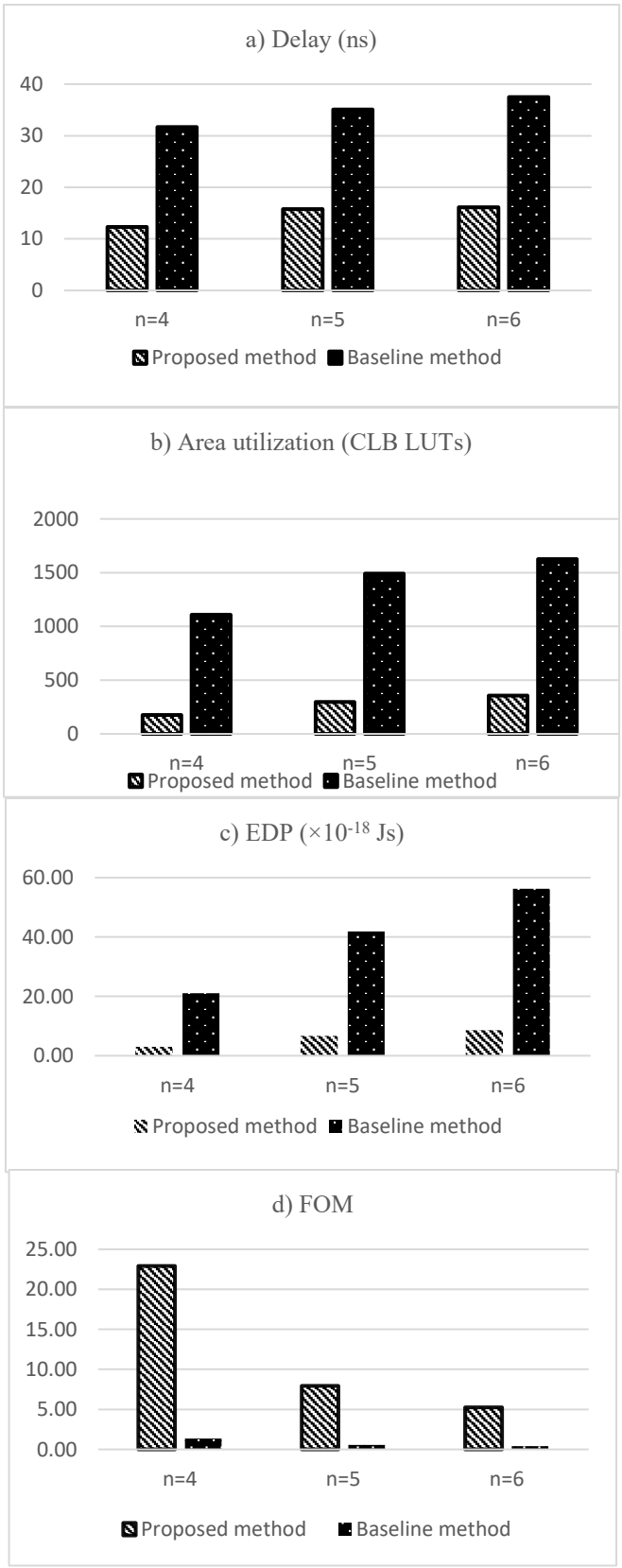

Fig. 5. Performance comparison of proposed and baseline designs. (a) Delay (ns), (b) Area utilization (CLB LUTs), (c) EDP ($\times 10^{-18}$ Js), (d) FOM.

$$FOM = \frac{1}{Delay \times Area \times Power} \tag{37}$$

A higher FOM indicates a more efficient design. This metric provides an objective means of comparing architectures with different resource and performance trade-offs.

All performance results have been updated to include this FOM for each value of

The results clearly demonstrate the superiority of the proposed method in all evaluated metrics. For instance, at $n = 6$, the proposed method achieves a 57% reduction in delay, 78% reduction in area, and 65% reduction in PDP compared to the baseline. The performance comparison between the proposed and baseline designs is illustrated in figure 5. Such performance gains highlight the potential of the proposed architecture for low-power, high-speed applications in VLSI systems, especially in domains like signal processing and cryptography where efficiency is paramount.

5. Conclusion

In this paper, we presented a novel FPGA implementation of a multiplier-free reverse converter for a new 4-moduli RNS set $\{3^n, 3^n-2, 3^n+2, 3^n-1\}$ using unbalanced ternary logic. The proposed architecture leverages the mathematical structure of the moduli set to eliminate the need for hardware multipliers, enabling efficient and low-complexity modular computation. Ternary digits were encoded in binary form to ensure compatibility with existing FPGA platforms, and an efficient design was adopted to support high-throughput applications. Due to the lack of existing comparable designs, a baseline binary-compatible version was implemented for performance evaluation. Simulation and synthesis results confirm the effectiveness of the proposed approach, demonstrating reductions in area, latency, and power consumption compared to conventional RNS designs. This work opens new possibilities for integrating ternary logic in practical VLSI systems, particularly in domains where parallelism, low power, and high speed are critical. Future work may explore extending the architecture to larger moduli sets, integrating full arithmetic units in ternary RNS, and developing custom ternary logic cells optimized for ASIC fabrication.

6. References

[1] K. Navi, A. S. Molahosseini, and M. Esmaeildoust, “How to teach residue number system to computer scientists and engineers”, *IEEE Transactions on Education*, vol. 54, no. 1, pp. 156–163, 2011.

[2] S. K. Singhal, S. Kumar, S. K. Patel, K. A. Rao, and G. Saxena, “An efficient method of modulo adder design for Digital Signal Processing applications,” *MethodsX*, vol. 14, p. 103–216, 2025.

[3] M. T. Gençoğlu, “Importance of Cryptography in Information Security,” *IOSR Journal of Computer Engineering (IOSR-JCE)*, vol. 21, no. 1, pp. 65–68, 2019.

[4] H. Nakahara and T. Sasao, “A High-speed Low-power Deep Neural Network on an FPGA based on the Nested RNS: Applied to an Object Detector”, 2018 IEEE

International Symposium on Circuits and Systems (ISCAS), May 2018, IEEE, pp. 1–5.
[5] M. Obeidi Daghlavi, M. R. Noorimehr, and M. Esmaeildoust, “Efficient two-level reverse converters for the four-moduli set {2n−1, 2n–1, 2n−1–1, 2n+1–1}”, *Analog Integrated Circuits and Signal Processing*, vol. 108, no. 1, pp. 75–87, 2021.
[6] B. Parhami and M. Mckeown, “Arithmetic with Binary-Encoded Balanced Ternary Numbers”, In Asilomar Conference on Signals, Systems and Computers, Nov 2013, IEEE, pp. 1130-1133.
[7] P. NavaeiLavasani, S. Adeli, M. R. Taheri, M. H. Moaiyeri, and K. Navi, “Fast and energy-efficient FPGA realization of RNS reverse converter for the ternary 3-moduli set {3n–2, 3n–1, 3n}”, *SN Appl Sci*, vol. 2, no. 2, pp. 269, 2020.
[8] S. Lin, Y. Bin Kim, and F. Lombardi, “Design of a ternary memory cell using CNTFETs”, *IEEE transactions on nanotechnology*, vol. 11, no. 5, pp. 1019–1025, 2012.
[9] S. Lone, A. Bhardwaj, A. K. Pandit, S. Gupta, and S. Mahajan, “A Review of Graphene Nanoribbon Field-Effect Transistor Structures”, *Journal of Electronic Materials*, vol. 50, no. 6, pp. 3169–3186, 2021.
[10] A. Malik, M. S. Hussain, and M. Hasan, “Energy-Efficient Exact and Approximate CNTFET-Based Ternary Full Adders”, *Circuits, Systems, and Signal Processing*, vol. 43, no. 5, pp. 2982–3003, 2024.
[11] الهام نیک بخت بیدگلی، داریوش دیدبان، «بررسی عملکرد مالتی پلکسر سه ارزشی مبتنی بر ترانزیستورهای اثر میدان نانولوله کربنی»، مجله مهندسی برق دانشگاه تبریز، جلد ۵۰، شماره ۲، صفحات ۹۴۳-۹۵۳، ۱۳۹۹.
[12] آزاده السادات عمرانی زرندی، امیر صباغ ملاحسینی، «طراحی ترکیبی مبدل های مستقیم و معکوس: شیوه ای نو برای کاهش پیچیدگی سخت افزاری سیستم اعداد مانده ای»، مجله مهندسی برق دانشگاه تبریز، جلد ۵۰، شماره ۳، صفحات ۱۳۱۵-۱۳۲۸، ۱۳۹۹.
[13] P. V. A. Mohan, “Reverse Converters for the Moduli Set {2 n+ 1− 3, 2 n− 1, 2 n− 1− 1}”, *IETE Journal of Education*, vol. 57, no. 1, pp. 31–43, 2016.
[14] A. S. Molahosseini, M. Hosseinzadeh, and K. Navi, “A Multiplier-Free Residue to Weighted Converter for the Moduli Set {3 n–2, 3 n–1, 3 n}”, *Con-temp Eng Sci*, vol. 1, pp. 71-80, 2008.
[15] J. Ahsan, M. Esmaeildoust, A. Kaabi, and V. Zarei, “Efficient FPGA implementation of RNS Montgomery multiplication using balanced RNS bases”, *Integration*, vol. 84, pp. 72–83, 2022.
[16] M. Esmaeildoust, D. Schinianakis, H. Javashi, T. Stouraitis, and K. Navi, “Efficient RNS implementation of elliptic curve point multiplication over GF(p)”, *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 8, pp. 1545–1549, 2013.
[17] M. Hosseinzadeh, S. J. Jassbi, and K. Navi, “A New Moduli Set {3 n-1, 3 n+ 1, 3 n+ 2, 3 n-2} in Residue Number System”, 10th International Conference on Advanced Communication Technology, Feb 2008, IEEE, pp. 1601–1603.
[18] M. Hosseinzadeh, K. Navi, and S. Gorgin, “A New Moduli Set for Residue Number System {rn 2, rn 1, rn}”, International Conference on Electrical Engineering, Apr 2007, IEEE, pp. 1–6.
[19] J. M. Rabaey, A. Chandrakasan, and B. Nikolic, “Digital integrated circuits”, vol. 2, Prentice hall Englewood Cliffs, 2002.