2^n RNS Scalers for Extended 4-Moduli Sets

Leonel Sousa, *Senior Member, IEEE*

Abstract—Scaling is a key important arithmetic operation and is difficult to perform in Residue Number Systems (RNS). This paper proposes a comprehensive approach for designing efficient and accurate 2^n RNS scalers for important classes of moduli sets that have large dynamic ranges. These classes include the traditional 3-moduli set, but the exponent of the power of two modulo is augmented by a variable value x ($\{2^n - 1, 2^{n\underline{\pm x}}, 2^n + 1\}$), and any extended set with an additional modulo m_4 ($\{2^n - 1, 2^{n\underline{\pm x}}, 2^n + 1[, \underline{m_4}]\}$). The proposed approach embeds scaling into the formulation of the Chinese remainder theorem and the mixed radix system, and it exploits the properties of the target moduli sets to perform scaling explicitly in the RNS domain. This is accomplished by operating hierarchically on each channel without requiring reverse and forward conversions. Simple memoryless VLSI architectures are proposed based on the obtained formulations. The relative assessment indicates that not only are these architectures comprehensive and suitable for configurable systems, but they are also more efficient than the related state of the art in terms of both performance and energy. The experimental results obtained for a 90 nm CMOS ASIC technology show improvements in the area-delay product, normalized with respect to the dynamic range, of up to 57% and 146% with the proposed scalers for the augmented 3-moduli set (dynamic range of $4n - 1$ bits) and an extended 4-moduli set (dynamic range of $6n$ bits), respectively. These improvements increase to 64.9% and 263% when the energy required per scaling is measured. The proposed scalers are not only flexible and cost-effective, but they are also suitable for designing and implementing energy-constrained devices, particularly mobile systems.

Index Terms—Residue number system, scaling, Chinese remainder theorem, VLSI architecture, ASIC, FPGA.

1 Introduction

Residue number systems (RNS) are alternative representations to positional number systems. The primary benefit of RNS is that their carry-free arithmetic can be computed both independently and in parallel in each of the channels to perform integer addition, subtraction and multiplication [1]. The usefulness of RNS has been tested in several applications, including public-key cryptography [2] [3] and linear signal processing, such as digital communication systems [4], [5], [6]. For these applications, RNS provide improved performance with less energy consumption, which is a fundamental advantage in real-time processing and embedded systems, particularly for energy autonomous devices.

The conversion between representations, i.e., the reverse conversion between RNS and the weighted binary system, requires auxiliary operations that impose overhead and cannot be individually computed within each RNS channel. These operations do not have a significant impact on the complexity of the RNS-based arithmetic only in the cases of computationally intensive applications. This is the case of public-key cryptography, which requires multiplications of very large numbers, and linear signal processing, which requires the iterative computation of a large number of multiply-accumulate operations. However, unlike floating-point arithmetic, integer fixed-point arithmetic often requires scaling to ensure that the computed results do not exceed the dynamic range.

Scaling, which is a division by a constant, is an operation that cannot be directly implemented in RNS within each channel but, unlike the reverse conversion, must be completed several times during the processing. For example, the iterative nature of the digital signal processing, ranging from finite and infinite impulse response filters to fast Fourier and wavelet transforms, often imposes expanded dynamic ranges if no scaling is applied. Either the RNS is designed to support the dynamic range required in the worst case, or scaling must be applied to avoid overflow, similarly to traditional fixed point computation. Therefore, scaling is an essential operation to limit the required dynamic range, in order to prevent overflow while maintaining the cost of the RNS-based data-paths low and the performance high. A truly representative example of such problem is the case of recursive algorithms for adaptive filtering, for which a mixed RNS-Binary approach must be adopted to meet the constraints of the physical channel equalization [5]: a binary implementation of the least mean squares (LMS) algorithm is used to avoid complex and expensive scaling circuits in RNS, and the large variable FIR takes advantage of the RNS implementation.

There have been several recent contributions to the literature on RNS scaling. The early approaches exploited the Chinese Remainder Theorem (CRT) to design approximate scaling techniques for integers represented in RNS [7], [8], [9]. For example, two algorithms for approximating the scaling under certain assumptions are proposed in [7]. One of these approximate scaling algorithms was thoroughly analyzed

• *Leonel Sousa is with the Department of Electrical and Computer Engineering, Instituto Superior Técnico, Universidade de Lisboa, and at Instituto de Engenharia e de Sistemas de Computadores (INESC-ID), Portugal, e-mail: leonel.sousa@inesc-id.pt.*

in [8]. Two different bands of error were identified: only small errors occur in one band, but in the other band, errors can assume values in the order of the modulus.

Other approaches obtain exact solutions for RNS scaling by applying modular reduction to the integer function of division. To obtain exact solutions, computationally intensive base extension methods are applied [10], [11], [12], [13]. Most of the scaling algorithms in this class accelerate the evaluation of $\langle X \rangle_K$ by assuming scale factors that are the product of subset moduli of the original set.

Most of the aforementioned RNS scaling algorithms use Look-Up-Tables (LUTs); thus, their implementation in hardware requires memory components, read-only memories (ROMs) or read-write random access memories (RAM), to support different moduli sets. These memory-based implementations do not scale well for two main reasons: *i*) the hardware cost increases sharply with the dynamic range, and *ii*) they do not support pipelining, the processing throughput is therefore limited and decreases with the memory size. Few adder-based designs have been proposed for RNS scaling [14], [15], [16], [17]. Most of these proposals consider the particular scaling factor 2^n, which is usually adopted in the weighted binary system. On the one hand, [14] considered more general moduli sets and scaling factors, assuming that the moduli in the set and the scaling factor are relatively prime, which resulted in high hardware costs and low performance. On the other hand, a very efficient RNS 2^n scaler was proposed in [15] and extended in [16] for signed numbers and in [17] to consider programmable power-of-two scaling factors, though it was restricted to the traditional 3-moduli set $\{2^n-1, 2^n, 2^n+1\}$. The number theoretic properties of this 3-moduli set has led to the design of very efficient arithmetic units, such as converters to and from RNS [18], [19], [20], but these converters provide a limited dynamic range (DR) that is less than $(3n)$-bit.

The work presented here starts by observing that most of the proposed moduli sets for RNS can be considered extended and augmented moduli sets: *i*) augmented 3-sets that result from increasing the power of two modulo, even as a variable [21]; *ii*) extended 4-moduli sets, which introduce a fourth element to the traditional 3-moduli set [22], [23], [24] or the augmented 3-moduli set [25], [26], [27], [28]. Therefore, an algorithm that can design efficient 2^n scalers for all such extended or augmented moduli sets represents an important step toward the design of efficient RNS scalers that are neither overly generic to provide competitive performance figures nor excessively constrained to a single moduli set. This paper proposes algorithms with these exact features by expanding the results in [15] for augmented 3-moduli sets and by hierarchically applying the CRT and the mixed radix system (MRS) to design cost-effective VLSI scalers for augmented and extended 4-moduli sets. Moreover, the proposed unified scalers for the moduli sets can be employed in reconfigurable systems to support dynamic RNS-based systems when modifying the moduli set at run time [29]. It is experimentally shown that very efficient VLSI scalers are obtained using the proposed approaches, which can help in the design of systems faced with constraints in processing time, cost, and power consumption.

This paper is organized as follows. Section 2 presents the background material and introduces the notation. Section 3 proposes the formulation for scaling with the augmented 3-moduli sets $\{2^n-1, 2^{n+x}, 2^n+1\}$. Similarly, Section 4 proposes formulations for 4-moduli supersets of the type $\{2^n-1, 2^{n+x}, 2^n+1, m_4\}$. VLSI architectures for scaling based on the derived formulations are designed in Section 5. The relative performance of the proposed scalers is evaluated and experimentally assessed in Section 6 regarding the related state of the art. Finally, in Section 7, conclusions are drawn.

2 Background

By defining a basis $\{m_1, m_2, \cdots m_N\}$ of pairwise co-prime elements, where m_i is known to be a modulus, any integer X within a DR $M = \prod_{i=1}^{N} m_i$ can be represented in the RNS domain by the N-tuple of residues $(R_1, R_2, \cdots R_N)$. To simplify the presentation of the methods and the description of the architectures proposed herein, the following notation is adopted.

- For an n-bit array of a generic value α_i, bits are noted in a range from the most significant bit (MSB) to the least significant bit (LSB) as $\alpha_{i(n-1)}, \cdots, \alpha_{i(0)}$, and the sequence of bits from k to j is represented as $\alpha_{i(k:j)}$.
- R_i denotes the residue for m_i; R_i is the least positive remainder obtained after dividing X by m_i ($R_i = \langle X \rangle_{m_i}$), for which the n-bit array representation is $r_{i(n-1)}, \cdots, r_{i(0)}$.
- The residue for a composed moduli $\prod_{i=l}^{n} m_i$ is represented as $R_{n \leftrightarrow l}$ $(0 < l < n \leq N)$.
- S_i denotes the residue for m_i of the scaled integer.
- The designation "channel m_i" is adopted to refer to the set of modulo m_i operations.

In the RNS forward conversion (to convert the integer number X from binary to RNS), residue R_i is computed for each channel $1 \leq i \leq N$. The main advantage of the RNS is the possibility of performing the same operations in parallel that one would perform using integers in each channel: for three integers A, B and C, all of which are smaller than M, and o, an addition/subtraction or a multiplication, $C = A \ o \ B$ is computed independently in each channel ($c_i = \langle a_i \ o \ b_i \rangle_{m_i}$). The RNS reverse conversion, which is a more complex operation, returns the representation of an integer from RNS to the binary domain.

2.1 Modular arithmetic properties

The characteristics and properties of modular arithmetic are exploited in this paper. Property 1 expresses well-known, simple-to-prove, and useful equalities for power of two moduli.

Property 1:

$$\begin{aligned}
\langle 2^n \times R_i \rangle_{2^n+1} &= \langle -R_i \rangle_{2^n+1} \\
\langle 2^x \times R_i \rangle_{2^n-1} &= \langle R_i \rangle_{2^n-1} \text{ for } x = n \ ; \\
(\text{for } 0 < x < n) &= r_{i(n-x-1)}, \cdots, r_{i(0)}, r_{i(n-1)}, \cdots, r_{i(n-x)}
\end{aligned}$$

Scaling the integer X by given K can be expressed, by definition, as

$$\lfloor \frac{X}{K} \rfloor = \frac{X - \langle X \rangle_K}{K} \tag{1}$$

Lemmas 1 and 2 can be used to simplify the modular reduction for moduli that possess certain characteristics.

Lemma 1:

$$\langle \lfloor k \times A \rfloor \rangle_{k \times p} = \lfloor k \times \langle A \rangle_p \rfloor; \ A, p, k \times p \in \mathbb{N} \ ; k \in \mathbb{Q} \tag{2}$$

Proof: The remainder operation is computed as

$$\langle \lfloor k \times A \rfloor \rangle_{k \times p} = \lfloor k \times A \rfloor - \lfloor \frac{\lfloor k \times A \rfloor}{k \times p} \rfloor \times k \times p$$

Although $k \in \mathbb{Q}$, because $k \times p \in \mathbb{N}$:

$$\lfloor \frac{\lfloor k \times A \rfloor}{k \times p} \rfloor = \lfloor \frac{\cancel{k} \times A}{\cancel{k} \times p} \rfloor \ ,$$

therefore,

$$\begin{aligned}
\langle \lfloor k \times A \rfloor \rangle_{k \times p} &= \lfloor k \times (A - \lfloor \frac{A}{p} \rfloor \times p) \rfloor = \\
&= \lfloor k \times \langle A \rangle_p \rfloor \ .
\end{aligned}$$

□

Lemma 2:

$$\left\langle \langle A \rangle_q \right\rangle_p = \langle A \rangle_p \ \text{; for any } q = k \times p; \ A, k, p \in \mathbb{N} \tag{3}$$

Proof: The remainder operation can be represented as $\langle A \rangle_q = A - k_1 \times q$, $0 \leq A - k_1 \times q < q$. Thus,

$$\begin{aligned}
\left\langle \langle A \rangle_q \right\rangle_p &= \langle A - k_1 \times q \rangle_p = \\
&= \left\langle \langle A \rangle_p - \langle k_1 \times q \rangle_p \right\rangle_p
\end{aligned}$$

Because $q = k \times p$,

$$\left\langle \langle A \rangle_q \right\rangle_p = \left\langle \langle A \rangle_p - \langle k_1 \times q \rangle_p \right\rangle_p = \langle A \rangle_p \ .$$

□

2.2 RNS reverse conversion

In the RNS reverse conversion, the integer X can be obtained from its residues $(R_1, R_2, \cdots R_N)$ by using the CRT (4) or adopting the MRS (5). The CRT allows one to compute in parallel the terms of the sum that are required to perform the reverse conversion by applying modulo M arithmetic (4):

$$X = \left\langle \sum_{i=1}^{N} M_i \times \langle M_i^{-1} \rangle_{m_i} \times R_i \right\rangle_M \ , \tag{4}$$

where $M_i = M/m_i$, and $\langle M_i^{-1} \rangle_{m_i}$ is the multiplicative inverse of M_i with respect to m_i ($\langle M_i \times M_i^{-1} \rangle_{m_i} = 1$).

Though most of the arithmetic required for the MRS has the width of the RNS channels, iterative computation is required (5):

$$\begin{aligned}
X &= y_1 + m_1 \times (y_2 + m2 \times (y_3 + m_3 \times (\cdots))) \qquad (5) \\
y_1 &= R_1 \\
y_2 &= \left\langle (R_2 - y_1) \times \langle m_1^{-1} \rangle_{m_2} \right\rangle_{m_2} \\
y_3 &= \left\langle \left((R_3 - y_1) \times \langle m_1^{-1} \rangle_{m_3} - y_2 \right) \times \langle m_2^{-1} \rangle_{m_3} \right\rangle_{m_3} \\
&\cdots
\end{aligned}$$

To support the class of augmented and extended 4-moduli sets whose general form is $\{m_1 = 2^n - 1, m_2 = 2^{n+x}, m_3 = 2^n + 1, m_4\}$ (where $0 \leq x \leq n$ and m_4 is any integer value that is pairwise co-prime with all other elements), hierarchical reverse conversion is performed on two consecutive levels: *i*) the the CRT (4) is computed for the 3-moduli set $\{2^n - 1, 2^{n+x}, 2^n + 1\}$, and *ii*) the MRS (5) is applied to calculate the final value X.

Multiplicative inverses (M_i^{-1}) in Lemma 3 are applied in the CRT to obtain the integer $R_{3\leftrightarrow1}$ and $\lfloor \frac{R_{3\leftrightarrow1}}{2^n} \rfloor$ from the residues in the moduli set $\{2^n - 1, 2^{n+x}, 2^n + 1\}$ $(0 \leq x \leq n)$.

Lemma 3: The following multiplicative inverses (M_i^{-1}) exist for $0 \leq x \leq n$.

$$\begin{aligned}
M_1^{-1} &= \left\langle [2^{n+x} \times (2^n + 1)]^{-1} \right\rangle_{2^n-1} = \\
&= \left\langle 2^{2n-x-1} \right\rangle_{2^n-1} \qquad (6) \\
M_2^{-1} &= \left\langle [(2^n - 1) \times (2^n + 1)]^{-1} \right\rangle_{2^{n+x}} = \\
&= \left\langle 2^{n+x} - 1 \right\rangle_{2^{n+x}} \qquad (7) \\
M_3^{-1} &= \left\langle [2^{n+x} \times (2^n - 1)]^{-1} \right\rangle_{2^n+1} = \\
&= \left\langle 2^{2n-x-1} \right\rangle_{2^n+1} \qquad (8)
\end{aligned}$$

Proof: By applying the definition of multiplicative inverse $\langle M_i \times M_i^{-1} \rangle_{m_i} = 1$ to (6) (7) and (8),

$$\begin{aligned}
\left\langle M_1 \times 2^{2n-x-1} \right\rangle_{2^n-1} &= \left\langle 2^{3n-1} \times 2 \right\rangle_{2^n-1} = 1 \\
\left\langle M_2 \times (2^{n+x} - 1) \right\rangle_{2^{n+x}} &= \left\langle -2^{2n} + 1) \right\rangle_{2^{n+x}} = 1 \\
\left\langle M_3 \times 2^{2n-x-1} \right\rangle_{2^n+1} &= \left\langle 2^{3n-1} \times (-2) \right\rangle_{2^n+1} = 1 \ .
\end{aligned}$$

$\square$

The MRS (5) is then applied to compute the final value $\lfloor \frac{X}{2^n} \rfloor$ by considering the 2-moduli set $\{\prod_{i=1}^{3} m_i = (2^{2n}-1)\times 2^{n+x}, m_4\}$ and the residues $R_{3\leftrightarrow 1}$ and R_4. Naturally, the final value is produced directly in the first level of the scaler when only the 3-moduli set is considered because $\lfloor \frac{X}{2^n} \rfloor = \lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \rfloor$. The multiplicative inverses for the 4-moduli sets depend on the values that m_4 assumes.

3 Scaler for augmented 3-moduli sets

By using the CRT to scale an integer represented in the RNS domain by a constant k, the following holds:

$$\left\lfloor \frac{X}{k} \right\rfloor = \left\lfloor \frac{1}{k} \times \left\langle \sum_{i=1}^{N} M_i \times \left\langle M_i^{-1} \right\rangle_{m_i} \times R_i \right\rangle_M \right\rfloor \tag{9}$$

By applying Lemma 1 to (9), we have

$$\left\lfloor \frac{X}{k} \right\rfloor = \left\langle \left\lfloor \frac{1}{k} \times \sum_{i=1}^{N} \times M_i \times \left\langle M_i^{-1} \right\rangle_{m_i} \times R_i \right\rfloor \right\rangle_{\frac{M}{k}} . \tag{10}$$

Instantiating (10) for $k = 2^n$ and the augmented 3-moduli sets and using the multiplicative inverses in Lemma 3, $\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \rfloor$ can computed using (11).

$$\begin{aligned}\left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor &= \big\langle \lfloor (2^x \times (2^n+1)) \times 2^{2n-x-1} \times R_1 + \\ &+ \frac{(2^{2n}-1)}{2^n} \times (2^{n+x}-1) \times R_2 + \\ &+ (2^n-1) \times 2^{2n-1} \times R_3 \rfloor \big\rangle_{(2^{2n}-1)\times 2^x}\end{aligned} \tag{11}$$

Because the first and last operands in (11) are always integers, the floor function must only be applied to the operand associated with R_2:

$$\begin{aligned}&\left\lfloor \tfrac{(2^{2n}-1)\times(2^{n+x}-1)\times R_2}{2^n} \right\rfloor = \\ &(2^{2n+x}-2^n-2^x) \times R_2 + \left\lfloor \tfrac{R_2}{2^n} \right\rfloor = \\ &(2^{2n+x}-2^n-2^x) \times R_2 + r_{2(n+x-1:n)} .\end{aligned} \tag{12}$$

By applying (12) on (11), we have

$$\begin{aligned}\left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor &= \big\langle (2^n+1) \times 2^{2n-1} \times R_1 + \\ &+ (2^{2n+x}-2^n-2^x) \times R_2 + r_{2(n+x-1:n)} + \\ &+ (2^n-1) \times 2^{2n-1} \times R_3 \big\rangle_{(2^{2n}-1)\times 2^x} .\end{aligned} \tag{13}$$

Analysing (13) indicates that the major challenge in obtaining a representation of $\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \rfloor$ directly in each channel is formulating the scaling in the channel (2^{n+x}) because $(2^{2n}-1)\times 2^x$ is not a multiple of 2^{n+x}. However, by applying Lemma 1,

$$\begin{aligned}\left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor \right\rangle_{2^{n+x}} &= \Big\langle \big\langle (2^{3n-1}+2^{2n-1}) \times R_1 \\ &- 2^n \times R_2 + (2^{3n-1}-2^{2n-1}) \times R_3 \\ &+ r_{2(n+x-1:n)} \big\rangle_{(2^{2n}-1)\times 2^x} \Big\rangle_{2^{n+x}} \\ &= \Big\langle \big\langle 2^x \times 2^{n-x} \times \big[(2^{2n-1}+2^{n-1}) \times R_1 \\ &- R_2 + (2^{2n-1}-2^{n-1}) \times R_3 \big] \\ &+ r_{2(n+x-1:n)} \big\rangle_{(2^{2n}-1)\times 2^x} \Big\rangle_{2^{n+x}} \\ &= \Big\langle \Big\langle 2^x \underbrace{\left\langle 2^{n-x} \left[(2^{2n-1}+2^{n-1}) \times R_1 - R_2 + (2^{2n-1}-2^{n-1}) \times R_3 \right] \right\rangle_{(2^{2n}-1)}}_{\phi} \\ &+ r_{2(n+x-1:n)} \Big\rangle_{(2^{2n}-1)\times 2^x} \Big\rangle_{2^{n+x}} .\end{aligned} \tag{14}$$

Given that, by definition, $\phi \leq 2^{2n}-2$ and $r_{2(n+x-1:n)} \leq 2^x - 1$,

$$2^x \times \phi + r_{2(n+x-1:n)} \leq 2^{2n+x} - 2^{x+1} + 2^x - 1 < (2^{2n}-1) \times 2^x , \tag{15}$$

(16) is used to compute $S_2 = \langle \lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \rfloor \rangle_{2^{n+x}}$ by applying (15) on (14).

$$\begin{aligned}S_2 &= \big\langle 2^x \big\langle 2^{n-x} \left[(2^{2n-1}+2^{n-1}) \times R_1 - R_2 \right. \\ &+ \left. (2^{2n-1}-2^{n-1}) \times R_3 \right] \big\rangle_{(2^{2n}-1)} + r_{2(n+x-1:n)} \big\rangle_{2^{n+x}} .\end{aligned} \tag{16}$$

To compute $S_1 = \langle \lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \rfloor \rangle_{2^n-1}$ and $S_3 = \langle \lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \rfloor \rangle_{2^n+1}$, one can apply both Lemma 2 (because $(2^{2n}-1)\times 2^x = (2^n-1)\times(2^n+1)\times 2^x$) and Property 1 to (13), leading to (17) and (18), respectively.

$$\begin{aligned}S_1 &= \langle R_1 - R_2 + r_{2(n+x-1:n)} \rangle_{2^n-1} \\ &= \langle R_1 - \langle R_2 \rangle_{2^n} \rangle_{2^n-1} \qquad (17) \\ S_3 &= \langle R_2 + r_{2(n+x-1:n)} - R_3 \rangle_{2^n+1} \\ &= \langle \langle R_2 \rangle_{2^n} - R_3 \rangle_{2^n+1} . \qquad (18)\end{aligned}$$

Using this comprehensive formulation, we can obtain results for any values of x in the range $0 \leq x \leq n$. For example, when $x = 0$ ($r_{2(n+x-1:n)} = 0$), the results should be the same as those presented in [15]. Further, for moduli 2^n-1 and 2^n+1, exactly the same expressions were obtained, though slightly different formulations for modulo 2^n are provided in that study. The formulation in [15] is

$$\begin{aligned}\left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor \right\rangle_{2^n} &= \langle \langle (2^{2n-1}+2^{n-1}) \times R_1 - 2^n \times R_2 \\ &+ (2^{2n-1}+2^{n-1}-1) \times R_3 \rangle_{(2^{2n}-1)} \rangle_{2^n} .\end{aligned} \tag{19}$$

However, because $\left\langle 2^{3n-1}\right\rangle_{(2^{2n}-1)} = 2^{n-1}$, for $x = 0$, we can obtain (20) from (16), which is equivalent to (19).

$$\left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor \right\rangle_{2^n} = \left\langle \langle (2^{n-1} + 2^{2n-1}) \times R_1 - 2^n \times R_2 + (2^{n-1} + 2^{2n} - 1 - 2^{2n-1}) \times R_3 \rangle_{(2^{2n}-1)} \right\rangle_{2^n} \tag{20}$$

$R_{3\leftrightarrow 1}$ might be considered either the final integer value of the conversion for the 3-moduli sets or an intermediate result obtained while proceeding to the second level of the conversion to 4-moduli sets. It is worth noting that based on (16), the value of $\left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor$ is obtained before the final reduction (modulo 2^{n+x}). Moreover, it is simple to obtain a representation of $R_{3\leftrightarrow 1}$ by using (21) because the addition is performed by simply concatenating the n LSB of R_2.

$$R_{3\leftrightarrow 1} = \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor \times 2^n + \langle R_{3\leftrightarrow 1} \rangle_{2^n} = \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor \times 2^n + r_{2(n-1:0)} \ . \tag{21}$$

4 Extended 4-moduli supersets

The MRS (5) is applied in the second level of the proposed hierarchical scaling method to obtain the final integer scaled value. From the residues $R_{3\leftrightarrow 1}$ and R_4 for the 4-moduli sets $\{(2^{2n}-1) \times 2^{n+x}, m_4\}$ ($M_4 = \frac{M}{m_4} = 2^{n+x} \times (2^{2n}-1)$), we have

$$X = R_{3\leftrightarrow 1} + \left\langle (R_4 - R_{3\leftrightarrow 1}) \times \langle M_4^{-1} \rangle_{m_4} \right\rangle_{m_4} \times M_4 \tag{22}$$

(23) is obtained by scaling (22) by 2^n:

$$\left\lfloor \frac{X}{2^n} \right\rfloor = \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor + \left\langle (R_4 - R_{3\leftrightarrow 1}) \times \langle M_4^{-1} \rangle_{m_4} \right\rangle_{m_4} \times \frac{\cancel{2^n}}{\cancel{2^n}} \times 2^x \times (2^n - 1) \times (2^n + 1) \ . \tag{23}$$

Computing (23) modulo 2^{n+x} to obtain S_2 leads to (24). The first term of (24) can be computed using (16), and $R_{3\leftrightarrow 1}$ can be obtained from (21).

$$S_2 = \left\langle \left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor \right\rangle_{2^{n+x}} + \left\langle (R_4 - R_{3\leftrightarrow 1}) \times \langle M_4^{-1} \rangle_{m_4} \right\rangle_{m_4} \times (-2^x) \right\rangle_{2^{n+x}} \tag{24}$$

Similarly, S_4 is given by (25):

$$S_4 = \left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor + (R_4 - R_{3\leftrightarrow 1}) \times \langle M_4^{-1} \rangle_{m_4} \times (2^{2n+x} - 2^x) \right\rangle_{m_4} \tag{25}$$

The complexity of (24) and (25) depends primarily on the values chosen for m_4 and x, which define not only the required modular arithmetic but also the complexity of the multiplicative inverse. This is not the case for the moduli $2^n - 1$ and $2^n + 1$. It is worth noting that the scaled values on these channels do not change upon the introduction of m_4 into the moduli set; therefore, (17) and (18) can be used on (26) and (27).

$$S_1 = \left\langle \left\lfloor \frac{X}{2^n} \right\rfloor \right\rangle_{2^n-1} = \left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor \right\rangle_{2^n-1} \tag{26}$$

$$S_3 = \left\langle \left\lfloor \frac{X}{2^n} \right\rfloor \right\rangle_{2^n+1} = \left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor \right\rangle_{2^n+1} \tag{27}$$

The next sub-sections are devoted to case studies of RNS scaling on 4-moduli sets with different characteristics. Because the scaled values S_1 and S_3 are independent of the values of m_4 and x, we focus our attention on the other two channels, 2^{n+x} and m_4.

4.1 Case studies $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} \pm 1\}$

The 4-moduli set $\{2^n - 1, 2^n, 2^n + 1, 2^{n+1} - 1\}$, as proposed in [22] and designated here by 4-Mod A, is composed of relatively prime numbers for even values of n and provides a DR of $(4n)$-bit. In the first level of the scaler, $x = 0$, and on the second level, $m_4 = 2^{n+1} - 1$. $\left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor \right\rangle_{2^n}$ is computed by using (20); for the second level, the value of the multiplicative inverse ψ is provided in [22] for $n \geq 4$:

$$\psi = \left\langle \left[2^n \times (2^{2n} - 1)\right]^{-1} \right\rangle_{2^{n+1}-1} = 2^n + 2^{n-2} + \cdots + 2^4 + 2 \ . \tag{28}$$

All of the operands required to compute S_2 and S_4 using (29) and (30), respectively, are thus available.

$$S_2 = \left\langle \left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor \right\rangle_{2^n} + \langle (R_4 - R_{3\leftrightarrow 1})\psi \rangle_{2^{n+1}-1} (-1) \right\rangle_{2^n} \tag{29}$$

$$S_4 = \left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor + (R_4 - R_{3\leftrightarrow 1})\psi \times (2^{n-1} - 1) \right\rangle_{2^{n+1}-1} \tag{30}$$

Table 1 illustrates the application of the proposed hierarchical scaling method. As an example, for $n = 8$, the method corresponds to the moduli set $\{m_1 = 255, m_2 = 256, m_3 = 257, m_4 = 511\}$ and provides a 32-bit DR. This table also considers the straightforward RNS scaling method, which applies reverse conversion followed by both scaling on the binary domain and forward conversion, to verify the correctness of the proposed approach.

The other extended 4-moduli set $\{2^n - 1, 2^n + 1, 2^n, 2^{n+1} + 1\}$ [22], herein designated as 4-Mod B, for $m_4 = 2^{n+1} + 1$ and odd n provides a DR of $(4n+1)$-bit. For this 4-moduli set (28), (29) and (30) become (31) (the terms in n exist until $n - 2k = 5$ for $n \geq 5$), (32) and (33), respectively.

$$\zeta = \left\langle \left[2^n \times (2^{2n} - 1)\right]^{-1} \right\rangle_{2^{n+1}+1} = 2^n + 2^{n-2} + \cdots + 2^{n-2k} + 2^4 - 2 \tag{31}$$

TABLE 1: Example of scaling in RNS the integer $X = 2^{30} - 1$ by 2^n for the 4-moduli A and $n = 8$ ($2^8 = 256$)

Moduli	$2^n - 1 = 255$	$2^n = 256$	$2^n + 1 = 257$	$2^{n+1} - 1 = 511$
Original residues	$R_1 = 63$	$R_2 = 255$	$R_3 = 192$	$R_4 = 7$
Reverse conv. to obtain X		$2^{30} - 1$	$\lfloor (2^{30} - 1)/2^8 \rfloor = 2^{22} - 1$	
(Forward conv. $\lfloor X/2^8 \rfloor = 2^{22} - 1$))				
Residues of scaled result S_i	S_1=63	S_2=255	S_3=63	S_4=15
Applying the proposed formulation for 2^8 RNS scaling				
$S_1 = \left\langle \lfloor \frac{2^{30}-1}{256} \rfloor \right\rangle_{255}$ (26)	$\langle R_1 - R_2 \rangle_{255} = \langle 63 - 255 \rangle_{255} = 63$			
$S_3 = \left\langle \lfloor \frac{2^{30}-1}{256} \rfloor \right\rangle_{257}$ (27)	$\langle R_2 - R_3 \rangle_{257} = \langle 255 - 192 \rangle_{257} = 63$			
$\left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{256} \right\rfloor \right\rangle_{256}$ (20)	$\langle \langle (128 + 32768) \times 63 - 256 \times 255 + (128 + 65535 - 32768) \times 192 \rangle_{65535} \rangle_{256} = \langle 63 \rangle_{256} = 63$			
$R_{3\leftrightarrow 1}$ (21)	$\left\lfloor \frac{R_{3\leftrightarrow 1}}{256} \right\rfloor \times 256 + \langle R_{3\leftrightarrow 1} \rangle_{256} = 63 \times 256 + 255 = 16383$			
$\psi = \left\langle \left[2^8 \times (2^{16} - 1) \right]^{-1} \right\rangle_{2^9 - 1}$ (28)	$2^8 + 2^6 + 2^4 + 2 = 338$			
$S_2 = \left\langle \left\lfloor \frac{X}{2^n} \right\rfloor \right\rangle_{256}$ (29)	$\left\langle 63 + \langle (7 - 16383) \times 338 \rangle_{511} \times (-1) \right\rangle_{256} = 255$			
$S_4 = \left\langle \left\lfloor \frac{X}{2^n} \right\rfloor \right\rangle_{511}$ (30)	$\left\langle \langle 63 \rangle_{511} + \langle (7 - 16383) \times 338 \rangle_{511} \times 127 \right\rangle_{511} = 15$			

$$
\begin{aligned}
S_2 &= \left\langle \left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor \right\rangle_{2^n} + \langle (R_4 - R_{3\leftrightarrow 1})\zeta \rangle_{2^{n+1}+1} (-1) \right\rangle_{2^n} & (32)\\
S_4 &= \left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor + (R_4 - R_{3\leftrightarrow 1})\zeta \times (-2^{n-1} - 1) \right\rangle_{2^{n+1}+1} & (33)
\end{aligned}
$$

4.2 Case study $\{2^n - 1, 2^{2n}, 2^n + 1, 2^{2n+1} - 1\}$

The 4-moduli set $\{2^n - 1, 2^{2n}, 2^n + 1, 2^{2n+1} - 1\}$ [25] [30], herein designated as 4-Mod C, enlarges the DR to ($6n$)-bit by adopting the augmented 3-moduli set $\{2^n - 1, 2^{2n}, 2^n + 1\}$. By using (16) with $x = n$ in the first level of the scaler, (34) is obtained.

$$
\begin{aligned}
\left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor \right\rangle_{2^{2n}} &= \left\langle 2^n \times \left\langle (2^{2n-1} + 2^{n-1}) \times R_1 - R_2 \right.\right. \\
&+ \left.(2^{2n-1} - 2^{n-1}) \times R_3 \right\rangle_{(2^{2n}-1)} \\
&+ \left. r_{2(2n-1:n)} \right\rangle_{2^{2n}} . \qquad (34)
\end{aligned}
$$

The multiplicative inverse χ [25] is provided in (35):

$$\chi = \left\langle \left[2^{2n} \times (2^{2n} - 1) \right]^{-1} \right\rangle_{2^{2n+1}-1} = \left\langle -2^{2n+3} \right\rangle_{2^{2n+1}-1} = -4 \qquad (35)$$

(36) and (37) are used to compute S_2 and S_4, respectively.

$$
\begin{aligned}
S_2 &= \left\langle \left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor \right\rangle_{2^{2n}} + \langle 4(R_{3\leftrightarrow 1} - R_4) \rangle_{2^{2n+1}-1} (-2^n) \right\rangle_{2^{2n}} & (36)\\
S_4 &= \left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor + (R_4 - R_{3\leftrightarrow 1})(2^{n+1}) \right\rangle_{2^{2n+1}-1} . & (37)
\end{aligned}
$$

5 VLSI ARCHITECTURES

The hybrid architecture for scaling depicted in Fig. 1 performs reverse conversion between RNS and the binary representations, and in the middle, it applies scaling by a power of two, which is a straightforward operation in the binary domain. However, as stated above, this is a computationally expensive and inefficient solution for RNS scaling.

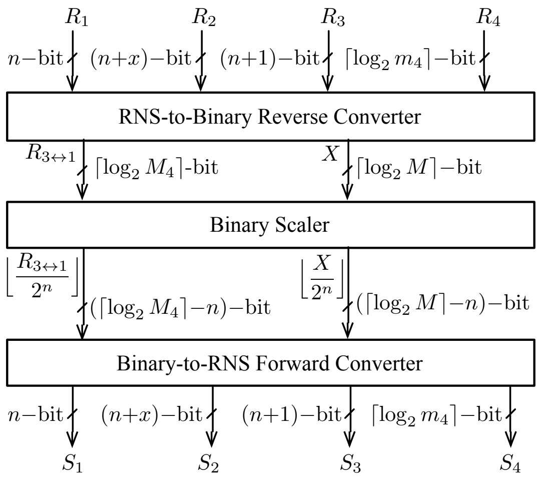

R_1
R_2
R_3
R_4
n−bit
$(n{+}x)$−bit
$(n{+}1)$−bit
$\lceil \log_2 m_4 \rceil$−bit
RNS-to-Binary Reverse Converter
$R_{3\leftrightarrow 1}$
$\lceil \log_2 M_4 \rceil$-bit
X
$\lceil \log_2 M \rceil$−bit
Binary Scaler
$\left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor$
$(\lceil \log_2 M_4 \rceil - n)$−bit
$\left\lfloor \frac{X}{2^n} \right\rfloor$
$(\lceil \log_2 M \rceil - n)$−bit
Binary-to-RNS Forward Converter
n−bit
$(n{+}x)$−bit
$(n{+}1)$−bit
$\lceil \log_2 m_4 \rceil$−bit
S_1
S_2
S_3
S_4

Fig. 1: Hybrid scaler performs reverse conversion to binary, binary scaling, and direct conversion to RNS.

A generic architecture proposed to design RNS scalers is presented in Fig. 2. The RNS scaler on the left-hand side of Fig. 2 operates directly in the channels of any 3-moduli set of the class $\{2^n - 1, 2^{n+x}, 2^n +$

$1\}$ $(0 \leq x \leq n)$. The scaler on the right-hand side of Fig. 2 also operates in the RNS domain, but for 2-moduli sets of the class $\{2^{2n}-1 \times 2^{n+x}, m_4\}$, where both moduli are co-prime. The internal architectures of those scalers are directly derived from the formulations derived in the previous sections.

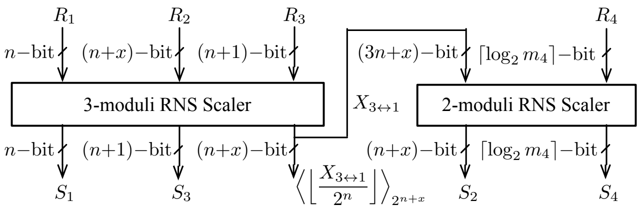
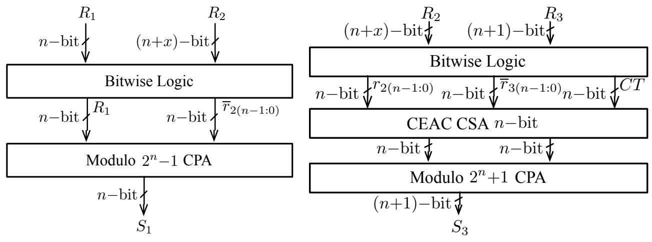


Fig. 2: General architecture of the proposed RNS scalers.

The architecture for the 3-moduli set, as depicted in Fig. 2, provides the scaled values in the three channels by applying the architectures in Fig. 3a), Fig. 3b) and Fig. 4) to compute (26), (27) and (16), respectively, in parallel.



(a) Channel 2^n-1 (b) Channel 2^n+1

Fig. 3: Architecture of the proposed RNS scaler for 3-moduli sets: S_1, S_3.

The architectures shown in Fig. 3a) and Fig. 3b) were proposed in [15] for the moduli set $\{2^n-1, 2^n, 2^n+1\}$. In Fig. 3a), two n-bit operands are added by using a modulo 2^n-1 carry propagate adder (CPA) [31]; $\langle -R_2 \rangle_{2^n-1} = \overline{R}_2$, where the overline represents the one's complement operation. In Fig. 3b), a Carry Save Adder with Complemented End-Around Carry (CSA-CEAC) and a modulo 2^n+1 CPA [31] are used to compute $\left\langle \left\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \right\rfloor \right\rangle_{2n+1}$. Complementing the carry-out bit of the CSA and performing end around-carry, a modulo 2^n+1 addition is implemented, but with the result incremented by one. Therefore, although only two $(n+1)$-bit operands have to be added, the CSA-CEAC is required to add a Correction Term (CT), which arises not only because the result obtained using the CEAC technique is incremented by one but also because $\langle -R_3 \rangle_{2^n+1} = \langle \overline{R}_3 + 2 \rangle_{2^n+1}$. In Fig. 3b, the n-th bit of $\overline{r_3}$ is also accounted for by the CT.

The architecture in Fig. 4 was designed to compute $R_{3\leftrightarrow 1}$ scaled by 2^n directly in the channel 2^{n+x}. This

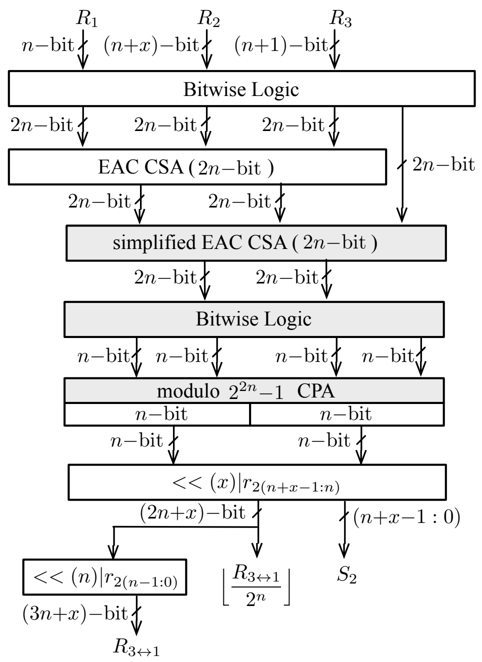


Fig. 4: Architecture of the proposed RNS scaler for 3-moduli sets; it may be used singly (S_2) or as as the first level for 4-moduli sets ($R_{3\leftrightarrow 1}$).

channel requires more hardware resources and imposes more delay because it applies modulo $2^{2n}-1$ arithmetic. However, this burden can be alleviated by applying Property 1 to the terms in (16) modulo $2^{2n}-1$:

$$(2^{2n-1}+2^{n-1})R_1 \overset{\text{mod } 2^{2n}-1}{\equiv}$$
$$r_{1(0)}, r_{1(n-1)}, \cdots r_{1(0)}, r_{1(n-1)}, \cdots, r_{1(1)} \tag{38}$$
$$-R_2 \overset{\text{mod } 2^{2n}-1}{\equiv} \overbrace{1, \ \cdots, 1}^{n-x}, \overline{r}_{2(n+x-1)}, \cdots, \overline{r}_{2(0)} \tag{39}$$
$$2^{2n-1} \times R_3 \overset{\text{mod } 2^{2n}-1}{\equiv} r_{3(0)}, \overbrace{0, \ \cdots, 0,}^{n-1} r_{3(n)}, \cdots, r_{3(1)} \tag{40}$$
$$-2^{n-1} \times R_3 \overset{\text{mod } 2^{2n}-1}{\equiv} \overline{r}_{3(n)}, \overline{r}_{3(n-1)}, \cdots \overline{r}_{3(0)}, \overbrace{1, \cdots, 1}^{n-1} \tag{41}$$

and by rewriting (40) and (41) to obtain (42) and (43).

$$(2^{2n-1}-2^{n-1})R_3 \overset{\text{mod } 2^{2n}-1}{\equiv}$$
$$r_{3(0)}, \overbrace{0, \ \cdots 0,}^{n-1}, r_{3(n)}, \overbrace{1, \ \cdots, 1}^{n-1} \tag{42}$$
$$+\overline{r}_{3(n)}, \overline{r}_{3(n-1)}, \cdots, \overline{r}_{3(0)}, r_{3(n-1)}, \cdots, r_{3(1)} \ . \tag{43}$$

Therefore, to compute (16), a Carry Save Adder with End-Around Carry (CSA-EAC) and a second simplified CSA-EAC, as highlighted in Fig. 4, are used. One of the operands of this last adder has the values of only two bits as variables from a total of $2n$ (operand in (42)). The multiplication by 2^{n-x} modulo $2^{2n}-1$ in (16) is performed by rotating the obtained carry and sum bit vectors, $(n-x)$ positions to the left (operation performed in the 'Bitwise Logic' component highlighted in Fig. 4). It can also be observed in (16) that only the n LSB of the modulo $2^{2n}-1$ sum are required, which are left shifted by x positions ('$<<(x)$' in Fig. 4) and concatenated with $r_{2(n+x-1:n)}$ ('|' symbol in Fig. 4) to compute S_2. Therefore, the n-bit CPA in Fig. 4 performs modulo $2^{2n}-1$ addition but only computes the n LSB bits of the sum. The $R_{3\leftrightarrow 1}$ can also be obtained by taking the $(2n)$-bit sum from the modulo $2^{2n}-1$ CPA in Fig. 4 ($\lfloor \frac{R_{3\leftrightarrow 1}}{2^n} \rfloor$), shifting left the result n positions and concatenating it with the n least significant bits of R_2 ($r_{2(n-1:0)}$), as stated in (21).

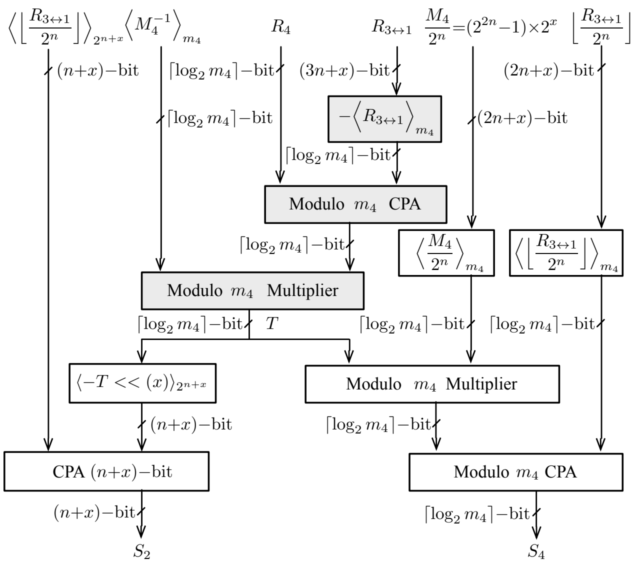

$\langle\lfloor\frac{R_{3\leftrightarrow1}}{2^n}\rfloor\rangle_{2^{n+x}}$
$\langle M_4^{-1}\rangle_{m_4}$
R_4
$R_{3\leftrightarrow1}$
$\frac{M_4}{2^n}=(2^{2n}-1)\times 2^x$
$\lfloor\frac{R_{3\leftrightarrow1}}{2^n}\rfloor$
$(n+x)$−bit
$\lceil\log_2 m_4\rceil$−bit
$(3n+x)$−bit
$(2n+x)$−bit
$-\langle R_{3\leftrightarrow1}\rangle_{m_4}$
Modulo m_4 CPA
$\langle\frac{M_4}{2^n}\rangle_{m_4}$
Modulo m_4 Multiplier
T
$\langle -T<<(x)\rangle_{2^{n+x}}$
CPA $(n+x)$−bit
S_2
S_4

Fig. 5: Second level of the proposed scalers for generic 4-moduli sets, (24), (25); the highlighted shared units are assumed to support generic modulo m_4 co-prime with M_4.

The architecture of the second level of the hierarchical scalers that support 4-moduli sets of the form $\{M_4=(2^{2n}-1)\times 2^{n+x}, m_4\}$ is depicted in Fig. 5. (24) and (25) both require the computation of $T=\left\langle (R_4-R_{3\leftrightarrow 1})\times\langle M_4^{-1}\rangle_{m_4}\right\rangle_{m_4}$, which is performed using the highlighted components in Fig. 5. Moreover, on the left-hand side of Fig. 5, a shifter and a $(n+x)$-bit binary CPA are used to produce the final result in the channel 2^{n+x} (S_2), but on the right side, a multiplier and a CPA, both modulo m_4, are used to produce S_4.

The complexity of the modulo m_4 multipliers in Fig. 5 is variable. The highlighted modulo m_4 multiplier may range from a simple shift unit (case study in subsection 4.2) to a multiplier implemented with an adder tree (case study in subsection 4.1), depending on the particular value of m_4 and, consequently, the operand $\langle M_4^{-1}\rangle_{m_4}$. The same consideration applies to the other modulo m_4 multiplier in the data path that is used to compute $\langle\lfloor\frac{X}{2^n}\rfloor\rangle_{m_4}$, which can be also reduced to a very simple arithmetic unit depending on the value of $\langle M_4\rangle_{m_4}$.

Although these general architectures can be tuned to perform optimizations of specific parametrizations and moduli-sets, i.e., for particular values of x and m_4, this work is primarily focused on general architectures that can be used as a framework to develop comprehensive scalers. For example, for the case study of the moduli set $\{2^n-1, 2^{2n}, 2^n+1, 2^{2n+1}-1\}$ in subsection 4.2, the architecture of the second level of the proposed scalers is presented in Fig. 6. In comparison with Fig. 5, the data path in Fig. 6 is quite simplified. Both of the modulo m_4 multipliers of Fig. 5 are mapped to simple bitwise operations in Fig. 6 according to (36) and (37). Therefore, both multipliers have a negligible impact on the total delay and the circuit area of the scaler. Moreover, the reduction modulo m_4 of $R_{3\leftrightarrow 1}$ is performed using the $2n+1$-bit CSA-EAC introduced in Fig. 6.

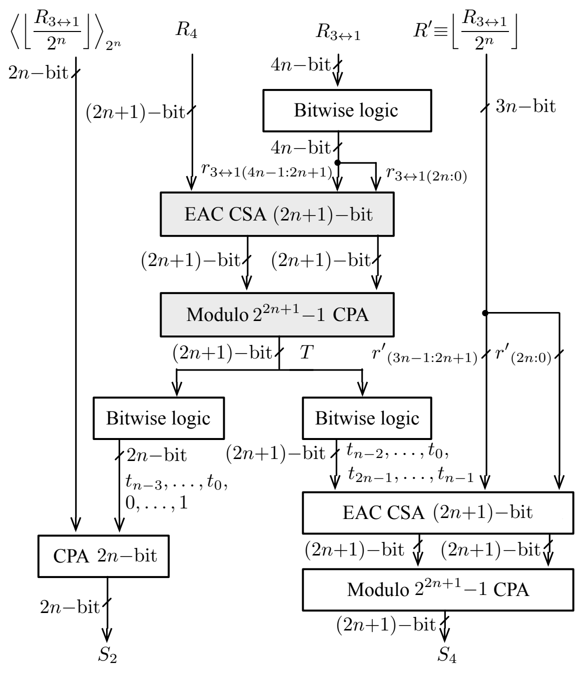

$\langle\lfloor\frac{R_{3\leftrightarrow1}}{2^n}\rfloor\rangle_{2^n}$
R_4
$R_{3\leftrightarrow1}$
$R'\equiv\lfloor\frac{R_{3\leftrightarrow1}}{2^n}\rfloor$
$2n$−bit
$4n$−bit
$(2n+1)$−bit
$3n$−bit
Bitwise logic
$r_{3\leftrightarrow1(4n-1:2n+1)}$
$r_{3\leftrightarrow1(2n:0)}$
EAC CSA $(2n+1)$−bit
Modulo $2^{2n+1}-1$ CPA
T
$r'_{(3n-1:2n+1)}$
$r'_{(2n:0)}$
$t_{n-3},\ldots,t_0, 0,\ldots,1$
$t_{n-2},\ldots,t_0, t_{2n-1},\ldots,t_{n-1}$
CPA $2n$−bit
S_2
S_4

Fig. 6: Second level of the proposed scaler, (36), (37), for the the 4-mod C ($\{2^n-1, 2^{2n}, 2^n+1, 2^{2n+1}-1\}$).

6 Assessment and Experimental Evaluation

To assess the proposed RNS scalers and perform a technology-agnostic comparison with the related state of the art, the simple unit-gate (U-G) model used in [15] is also adopted here. Because the conclusions are similar and due to space limitations, we present a detailed evaluation for only two of the addressed scalers: the 3-moduli set in Fig. 3 and Fig. 4 (for $x=n$) and the 4-Mod C $\{2^n-1, 2^{2n}, 2^n+1, 2^{2n+1}-1\}$ in

Fig. 6. Experimental results, obtained by designing an Application-Specific Integrated Circuit (ASIC) and configuring Field Programming Gate Arrays (FPGAs), are provided for all of the scalers considered in this paper.

6.1 Evaluation based on the U-G model

In the U-G model, a two-input monotonic gate, such as a NAND gate, is considered to have one unit of area and one unit of delay. An XOR gate is deemed to require two units of area and to impose a delay of two units. The area and delay of an inverter is a negligible fraction of a unit, and it is thus assumed to require zero units of area and delay. Based on the U-G model, a Full Adder (FA) has seven units of area and four units of delay.

Binary adders and 2^k-1 and 2^k+1 modulo adders are the main requirements for implementing the scalers' architectures described in section 5. The circuit area and delay for each RNS channel can be independently evaluated using the U-G model. In this assessment, the adopted adders are similar to those presented in [15]. The adopted Ling modulo 2^n-1 adders [32] require $3n\lceil\log_2 n-1\rceil+12n$ units of area and impose a delay of $2\lceil\log_2 n-1\rceil+3$ units. Modulo 2^n+1 adders for the diminished-1 representation [33] require $4.5n\lceil\log_2 n\rceil+0.5n+6$ units of area and impose a delay of $2\lceil\log_2 n\rceil+3$ units. For binary adders, $1.5n\lceil\log_2 n\rceil+5n$ units of area and $2\lceil\log_2 n\rceil+3$ units of time are assumed [33].

The required area and imposed delay of the proposed scalers are presented in Tables 2 and 3 for each channel. Analyzing Fig. 3a, only a modulo 2^n-1 CPA is used for the channel 2^n-1. The channel 2^n+1 requires a n-bit CSA and a final modulo 2^n+1 CPA, as depicted in Fig. 3b. It is worth to noting that both the CPA and each CSA-CEAC increment the result, which is offset by the correction term applied in Fig. 3b.

Table 2 and Table 3 consider for the 2^{n+x} channel of the the 3-moduli set (Fig. 4) to be the worst case with respect to the area and delay ($x=n$). For this channel, two $2n$-bit CSA-EAC and a modulo $2^{2n}-1$ CPA are used to add the four operands in (38) (39) and (42). It is worth noting that all but two bits of one of the operands in (42) assume a constant value; thus, the highlighted CSA-EAC in Fig. 4 only requires half adders for $(2n-2)$ bits from a total of $2n$. A half adder requires less than half of the area of an FA. Thus, the CSA-EAC in Fig. 4 only requires $7n$ units of area but imposes a delay of 4 units. Moreover, the modified binary adder of Fig. 4 requires an area corresponding to that of an n-bit binary adder and the delay of a modulo $2^{2n}-1$ modulo adder.

For the 4-mod C set and considering the architecture shown in Fig. 6, to simplify the presentation of the results, we use $\lceil\log_2(2n+1)\rceil \cong \lceil\log_2 2n\rceil$. The part of the computation common to both the 2^{2n} and $2^{2n+1}-1$ channels is performed using the highlighted components in Fig. 6: a $2n+1$-bit CSA-EAC, with $[14n+7]$ units of area and a delay of 4 units, and a modulo $2^{2n+1}-1$ CPA, with $[(6n+3)\lceil\log_2 n\rceil+24n+12]$ units of area and a delay of $[2\lceil\log_2 n\rceil+3]$ units. In the results presented in Tables 2 and 3, these units of area are accounted for in the 2^{2n} modulus, for which a final $2n$-bit binary CPA is also used and requires an additional $[3n\lceil\log_2 n\rceil+13n]$ units of area and $[2\lceil\log_2 n\rceil+5]$ units of delay. As observed in Fig. 6, an additional $2n+1$-bit EAC CSA and a modulo $2^{2n+1}-1$ CPA are used for the $2^{2n+1}-1$ channel, for which we must consider $[14n+7]$ and $[(6n+3)\lceil\log_2 n\rceil+24n+12]$ units of area, and 4 and $[2\lceil\log_2 n\rceil+3]$ units of delay, respectively.

TABLE 2: Circuit Area (A) required (U-G model) for the 3-moduli set ($x=n$) and the 4-mod C

Channel	A_{CPA}	A_{Total}
3-moduli set		
2^{n+x}	$3n\lceil\log_2 n\rceil+12n$	$3n\lceil\log_2 n\rceil+33n$
Common to 3 and 4 moduli sets		
2^n-1	$3n\lceil\log_2 n-1\rceil+12n$	$3n\lceil\log_2 n-1\rceil+12n$
2^n+1	$4.5n\lceil\log_2 n\rceil+0.5n+6$	$4.5n\lceil\log_2 n\rceil+7.5n+6$
Total 3-moduli	$10.5n\lceil\log_2 n\rceil+21.5n+6$	$10.5n\lceil\log_2 n\rceil+49.5n+6$
4-mod C		
2^{2n}	$(9n+3)\lceil\log_2 n\rceil+37n+12$	$(9n+3)\lceil\log_2 n\rceil+51n+19$
$2^{2n+1}-1$	$(6n+3)\lceil\log_2 n\rceil+24n+12$	$(6n+3)\lceil\log_2 n\rceil+38n+19$
Total 4-mod C	$(28.5n+6)\lceil\log_2 n\rceil+94.5n+30$	$(28.5n+6)\lceil\log_2 n\rceil+150.5n+44$

The total area required by the the proposed scaler for the 4-mod C set is obtained from Table 2 by adding the area of the components in each channel, leading to $(28.5n+6)\lceil\log_2 n\rceil+150.5n+44$ units. The critical path presented in Table 3, $6\lceil\log_2 n\rceil+25$ units, goes through the shared components of channels 2^{n+x} and $m_4=2^{n+1}-1$.

Table 4, which was also obtained using the U-G model, extends the relative assessment of the proposed scalers by considering other RNS scalers for $n=8$ and $n=16$. Again, the worst case scenario for an area and delay is obtained for the 3-moduli sets ($x=n$). The results in [15] are used as a benchmark because, to the best of the author's knowledge, that study proposed the most efficient scaler for the 3-moduli set $\{2^n-1, 2^n, 2^n+1\}$.

Because there is no specific scaler for the considered 4-moduli sets, the figures of merit obtained using the proposed scalers are compared with those registered by performing reverse conversion followed by binary

TABLE 3: Delay (D) imposed (U-G model) for the 3-moduli set ($x = n$) and the 4-mod C

Channel	D_{CPA}	D_{Total}
3-moduli set		
2^{n+x}	$2\lceil\log_2 n\rceil + 3$	$2\lceil\log_2 n\rceil + 11$
Common to 3 and 4 moduli sets		
$2^n - 1$	$2\lceil\log_2 n - 1\rceil + 3$	$2\lceil\log_2 n - 1\rceil + 3$
$2^n + 1$	$2\lceil\log_2 n\rceil + 3$	$2\lceil\log_2 n\rceil + 7$
Total 3-moduli	$2\lceil\log_2 n\rceil + 3$	$2\lceil\log_2 n\rceil + 11$
4-mod C		
2^{2n}	$4\lceil\log_2 n\rceil + 8$	$4\lceil\log_2 n\rceil + 12$
$2^{2n+1} - 1$	$4\lceil\log_2 n\rceil + 6$	$4\lceil\log_2 n\rceil + 14$
Total 4-mod C	$6\lceil\log_2 n\rceil + 11$	$6\lceil\log_2 n\rceil + 25$

TABLE 4: Comparative assessment regarding the related state of the art (U-G model): A-Area, D-Delay, $\frac{A\times D}{DR}$-Product Area Delay values normalized with respect to the DR expressed in bits

n	8			16		
Method	A	D	$\frac{A\times D}{DR}$	A	D	$\frac{A\times D}{DR}$
3 moduli set ($m_2 = 2^{2n}$)						
Proposed	654	17	**347**	1470	19	**437**
[15]	804	15	**503**	1812	17	**642**
4-mod C ($m_4 = 2^{2n+1} - 1$)						
Proposed	1950	43	**1747**	4548	45	**2195**
RC≻Sc≻FC [25]	2547	59	**3131**	5513	67	**3848**

scaling and forward conversion (RC≻Sc≻FC), which maps the representation back to the RNS domain (Fig. 1). This reverse converter was built in a two-level hierarchy: the first level was based on the CRT (4), and the second level was based on the MRS [25]. Scaling by a power of two in the binary domain is easily performed by a hardwired shift operation, so neither additional costs nor extra delay are incurred. The most efficient topologies are adopted to implement the forward converters [34]. Apart from the binary channel, which requires only truncation, these topologies apply to all of the moduli sets considered in this study, i.e., Property 1, CSA-EAC, CSA-CEAC, and modulo 2^k-1 and 2^k+1 CPAs, to implement the forward converters.

In addition to the area and delay, the values for the area-delay product, normalized with respect to the DR, are presented in Table 4. For the 3-moduli set, the proposed scaler supports a DR of approximately $4n$-bit, whereas the scaler in [15] only supports an approximately $3n$-bit DR. The normalized area-delay product figure in (44) is used for comparison purposes; an improvement greater than 1 indicates that the proposed scaler is more efficient than the related state of the art. An average improvement of approximately 45% is achieved by using the proposed scalers for the 3-moduli sets. This improvement increases significantly when the scaling is performed on the 4-mod C set: with significant reductions in both area and delay, we estimate an area-delay product average improvement of 78%.

$$\text{Improvement} = \frac{(\text{Area} \times \text{Delay})_{\text{reference}} \times \text{DR}_{\text{proposed}}}{(\text{Area} \times \text{Delay})_{\text{proposed}} \times \text{DR}_{\text{reference}}} \quad (44)$$

6.2 Experimental evaluation

The circuits of the proposed and related state-of-the-art scalers were described in synthesizable VHDL, and their functionality was thoroughly tested. A well-known library of arithmetic units [31], also written with synthesizable VHDL, was used. This library contains a structural specification of components, namely optimized prefix adders, that were employed to describe and implement the scalers. Using these HDL specifications, implementations targeting FPGA and ASIC were accomplished.

A Xilinx Virtex 4 (part xc4vlx200ff1513-11) FPGA, based on 90 nm CMOS technology, was targeted to obtain the programming bitstream using the Synopsys Synplicity Premier tools (version E-2010.09-SP2) for the synthesis procedure and the Xilinx ISE tools (version 12.4) for placing and routing. For the ASIC technology, we supported our implementation on a Faraday 90-nm standard cell library that was tailored for the UMC 90 nm logic SP/RVT Low-K, using the Synopsys Design Vision tools (version E-2010.12) to synthesize the design and the Cadence Encounter and NanoRoute tools (versions v09.12-s159 and v09.12-s013, respectively) for placing and routing[1]. For both the FPGA and ASIC technologies, no manual optimizations were introduced. The synthesis tools were set to target minimum delay, thus allowing the tool to use unconstrained resources and power consumption. The presented energy-per-scaling estimated values were obtained from the placed-and-routed circuit specifications for 20% of the switching activity. The Cadence Encounter built-in power reporting tool was employed to measure the dynamic and leakage power on the target ASIC technology.

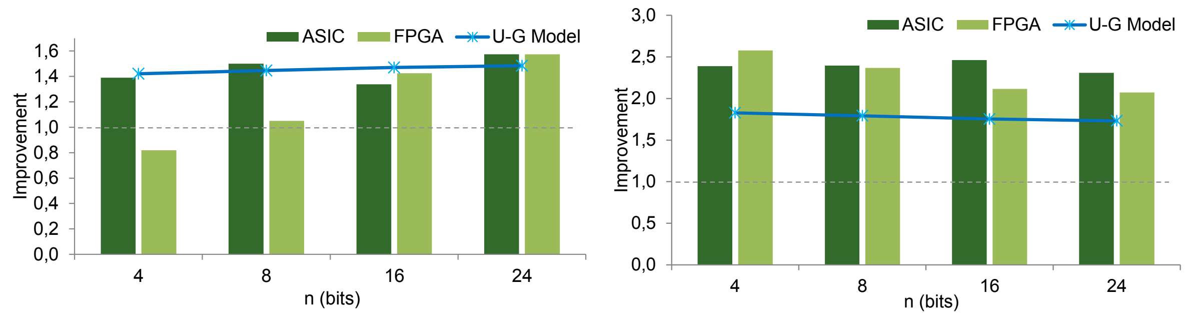
The experimental results obtained for the ASIC and the FPGA are presented in Table 5 for the 3-moduli set (with $x = n$) and the extended 4-moduli sets addressed in Section 4. The measures for $n = 8, 16$ are presented and correspond to dynamic ranges from 31-bit to 96-bit.

Let us first analyze the results obtained for the 3-moduli sets. The area and delay results that were experimentally obtained for the proposed and reference scalers are in line with the estimations derived

1. The HDL specification of the proposed and related art scalers are publicly available at *http://sips.inesc-id.pt/las/prototypes/rnsscalers/*.

TABLE 5: Performance figures of the proposed scalers for $n = 8$ and $n = 16$; the values normalized with respect to the DR are presented in parentheses.

	90nm ASIC						Virtex 4 FPGA			
	D (ns)		A $(\times 10^3 \mu m^2)$		P (mW)		D (ns)		A (slices)	
n	8	16	8	16	8	16	8	16	8	16
3 moduli set: for the Proposed $m_2 = 2^{2n}$ $(DR = 4n - 1\text{-bit})$; for [15] $m_2 = 2^n$ $(DR = 3n - 1\text{-bit})$										
Proposed	0.74 (0.024)	0.92 (0.015)	5.6 (0.18)	12.2 (0.19)	9.8 (0.32)	19.4 (0.31)	6.1 (0.20)	7.0 (0.11)	210 (6.77)	324 (5.14)
[15]	0.73 (0.031)	0.83 (0.018)	6.4 (0.28)	13.7 (0.29)	11.7 (0.51)	23.7 (0.50)	5.29 (0.23)	6.34 (0.13)	191 (8.30)	380 (8.09)
4-mod A $(m_2 = 2^n,\ m_4 = 2^{n+1} - 1,\ DR = 4n\text{-bit})$										
Proposed	2.25 (0.070)	2.78 (0.043)	17.5 (0.55)	40.6 (0.63)	85.8 (2.68)	213.5 (3.34)	17.75 (0.55)	21.10 (0.33)	503 (15.72)	1084 (16.94)
RC [22]≻Sc≻FC	2.86 (0.089)	3.5 (0.055)	21.6 (0.68)	51.2 (0.8)	117.8 (3.68)	294.5 (4.6)	21.88 (0.68)	26.50 (0.41)	569 (17.78)	1120 (17.50)
4-mod B $(m_2 = 2^n,\ m_4 = 2^{n+1} + 1,\ DR = 4n + 1\text{-bit})$										
Proposed	4.11 (0.125)	4.71 (0.072)	32.6 (0.99)	73.0 (1.12)	149.1 (4.52)	318.6 (4.9)	26.16 (0.79)	34.02 (0.52)	688 (20.85)	1312 (20.18)
RC [22]≻Sc≻FC	4.70 (0.142)	5.70 (0.088)	38.7 (1.17)	88.2 (1.36)	206.6 (6.26)	476.9 (7.34)	31.62 (0.96)	37.79 (0.58)	706 (21.39)	1362 (20.95)
4-mod C $(m_2 = 2^{2n},\ m_4 = 2^{2n+1} - 1,\ DR = 6n\text{-bit})$										
Proposed	1.65 (0.034)	1.86 (0.019)	18.7 (0.39)	38.4 (0.4)	66.7 (1.39)	127.8 (1.33)	12.02 (0.25)	14.47 (0.15)	455 (9.48)	887 (9.24)
RC [25]≻Sc≻FC	2.47 (0.051)	2.83 (0.029)	29.9 (0.62)	62.2 (0.65)	151.3 (3.15)	290.7 (3.03)	18.55 (0.39)	22.64 (0.24)	698 (14.54)	1199 (12.49)



(a) Area×Delay 3-moduli

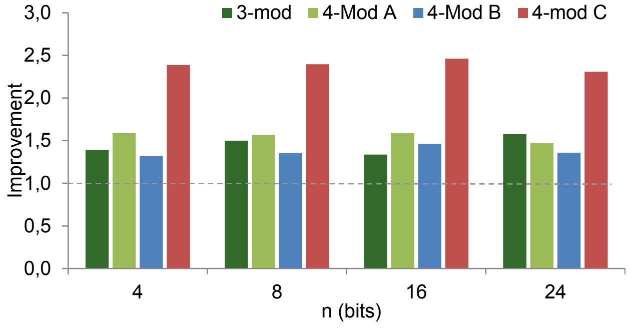
(b) Area×Delay 4-moduli

Fig. 7: Comparison with the state of the art of the delay-area product; values normalized with respect to the DR for the ASIC and FPGA (a line designed using the U-G model is superimposed): *a*) for the 3-moduli set, in relation to [15]; *b*) for the 4-moduli set, in relation to the RC≻Sc≻FC approach; an improvement metric greater than 1 indicates that the proposed scaler is more efficient than the related state of the art.

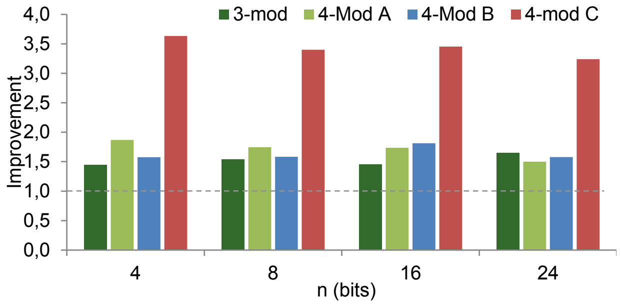
with the U-G model. Analyzing the results obtained for the ASIC, normalized with respect to the DR, the proposed scaler in comparison with [15] is 22.5% and 16.7% faster for $n = 8$ and $n = 16$, respectively, and the area is reduced by more than 30% on average. The proposed scalers also reduce the power consumption of the ASICs by approximately 18%, on average. The results in Table 5 also show that improvements of the same order are obtained for the area and delay when the proposed scalers for the 3-moduli set are implemented on FPGAs, i.e., for larger values of n.

Fig. 7 depicts the area-delay product improvements achieved using the proposed scalers, comparing the experimental results obtained for ASIC and FPGA with the estimates presented in Tables 2 and 3. As observed in these tables, the scalers proposed are compared with [15] for the 3-moduli set and with RC≻Sc≻FC method [25] for the 4-mod C set . The experimentally obtained results show that the improvements for the 3-moduli set follow the same trend and are in line with the prediction from the U-G model, except for scalers with a low n implemented on FPGAs. A maximum area-delay product improvement of approximately 57% is achieved for both the ASIC and the FPGA ($n = 24$), where we used the U-G model to estimate that this improvement would be approximately 48%. The minimum improvement for the ASIC (approximately 33%) occurs for $n = 16$, the case for which the deviation between the estimated and experimental results is the greatest (approximately 14%). The improvements obtained by implementing the proposed scalers for $n = 4, 8$ in FPGAs are negligible results that were not accurately estimated using the U-G model. This inaccuracy primarily arises because the logic is implemented on look-up-tables in memory and also because the interconnections, which are not considered in the U-G model, dominate the cost and performance in these cases.

The estimated advantage of using the proposed RNS scaler for the 4-mod C versus RC≻Sc≻FC [25]



(a) Area×Delay improvement



(b) Energy per scaling improvement

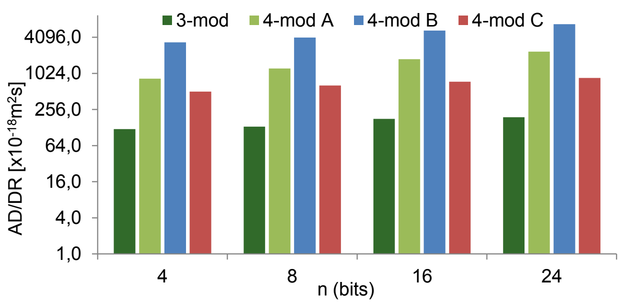
Fig. 8: Comparison with the state of the art for the ASIC; values normalized with respect to the DR for the 3-moduli set (following [15], with $x = n$) and for the several 4-moduli sets (following the RC≻Sc≻FC approach): *a*) area-delay product; *b*) energy per scaling; an improvement greater than 1 indicates that the proposed scaler is more efficient.

is also experimentally evaluated. As observed in the last lines of Table 5, the delay of the proposed scaler is approximately 33% lower and the area is approximately 37% smaller when the proposed scalers are implemented in ASIC. Similar values were obtained for FPGA, i.e., 35% and 30%, respectively. These reductions are even higher than those predicted using the U-G model. Moreover, an impressive reduction in power consumption was achieved by implementing the proposed scalers on ASICs: more than 50% for both values of n considered. This is an important aspect in designs that face power restrictions. For the 4-mod C set, the proposed scaler provides an area-delay product improvement that is always higher than 100% and reaches 146%, as observed in Fig. 7b). The U-G model overestimated the required area and the imposed delay for the proposed scaler. For some operators that include constant values, the synthesis tools can simplify the arithmetic circuits, and the impact of the interconnections, which is not considered in the U-G model, is more significant when the more complex RC≻Sc≻FC [25] scaling architecture is implemented.

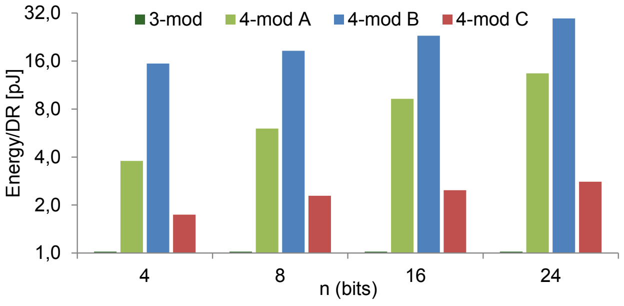
Fig. 8 depicts the improvements that were obtained in the experiments by designing ASICs for all of the scalers considered in this paper and for different values of n. The values of the improvements for energy are obtained by applying on (44) the experimental results for the power-delay product. For ASIC, the proposed scalers improve the area-delay product for the augmented 3-moduli set by approximately 45%, on average. This figure increases to more than 55%, reaching a maximum of 60% ($n = 16$) in the case of the 4-mod A set. For the 4-mod B set, the maximum and average values for the improvement in the area-delay product are 46% and 38%, respectively. As observed from the results depicted in Fig. 7b, the corresponding average improvement figure for the 4-mod C set is much higher, approximately 140%. The simplicity of the multiplicative inverse on the second level of this particular scaler leads to the simplified architecture in shown Fig. 6. For this simple architecture, the relative weight of avoiding computations to perform the reverse and forward conversions is more pronounced and, as predicted with the UG-model, leads to even more significant improvements.

By analyzing Fig. 8b, similar conclusions are reached for the improvements in energy per conversion. Using this metric, the average improvements increase to 54%, 68%, 58%, and 224% for the augmented 3-moduli, 4-mod A, 4-mod B and 4-mod C sets, respectively.

Fig. 9 presents the values of the area-delay product and energy per scaling, normalized with respect to the DR, that were experimentally obtained by implementing the proposed scalers on ASICs. Semi-log graphs are adopted by plotting the large range of values obtained for the area-delay product and energy per scaling on a logarithm to base 2 scale. The area-delay product per bit required by the scaler for the 3-moduli varies between $120 \times 10^{-18}\mathrm{m}^2\mathrm{s}$ and $190 \times 10^{-18}\mathrm{m}^2\mathrm{s}$, when the value of n increases. For the 4-mod C set, these figures are approximately four times greater, which represents the penalty of introducing the second level into the scaler architecture. As observed in Fig. 9a, the value of the area-delay product per bit for the mod-4 A set is approximately twelve times greater than that obtained for the 3-moduli set ($n = 24$). This number further increases to more than thirty times as large if we compare the area-delay product per bit of the scalers for the mod-4 B set and the 3-moduli set. The energy per bit evolution, represented as pJ in Fig. 9b, follows the same trend as the area-delay product, where the scalers of the 3-moduli set and the 4-mod C moduli set are the most efficient.



(a) Area×Delay



(b) Energy per scaling

Fig. 9: Performance; values normalized with respect to the DR on the ASIC for the 3-moduli set and several 4-moduli sets: *a*) area-delay product; *b*) energy per scaling.

7 Conclusions

This paper proposes efficient and accurate 2^n RNS scalers for important classes of moduli sets that have large dynamic ranges, such as a traditional 3-moduli set that has been augmented with the exponent of the power of two modulo increased by x or extended by an additional modulo m_4 ($\{2^n-1, 2^{n\underline{+x}}, 2^n+1[, \underline{m_4}]\}$). The new adopted approach is neither so general that it precludes an efficient solution nor so narrow that it only fits a single moduli set. It performs scaling directly in the RNS domain by operating both hierarchically and individually at the channel level and does not require reverse and forward conversions. From those formulations, simple memoryless VLSI architectures were proposed for several moduli sets of the considered classes. The proposed scalers were evaluated based on a technologically agnostic unit-gate model and tested experimentally by synthesizing both those scalers and those presented in the related state of the art using FPGA and ASIC technologies. The obtained experimental results suggest that significantly more flexible and efficient scalers are achieved. For a 90 nm CMOS ASIC technology and considering the merit area-delay product, the experimental results show that relative improvements of up to 57% and 146%, respectively, are achieved with the proposed scalers for the 3-moduli set with a dynamic range of $(4n-1)$-bit and the 4-mod set with a dynamic range of $(6n)$-bit. These improvements are further increased to 64.9% and 263% when the energy required per scaling is measured. The proposed unified scaling architectures allow not only the design of static scalers targeting different moduli sets but also the dynamic configuration of the RNS scalers to efficiently support several augmented and extended moduli sets in FPGAs. A final conclusion to be drawn from this work is that the proposed scalers are not only flexible and cost-effective but also represent a step forward in the design of energy-constrained devices, particularly mobile systems.

Acknowledgments

First and foremost, my deepest thanks to my lovely daughter Ana Cristina, whose courage and tenacity inspired me to write this paper during a quite difficult time of her life.

I would like to acknowledge Dr. Samuel Antão for his help in obtaining the experimental results.

This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013, and project EXPL/EEI-ELC/1572/2013.

References

[1] P. Mohan, *Residue Number Systems: Algorithms and Architectures*. New York: Kluwer Academic Publishers, 2002.
[2] S. Antao and L. Sousa, "The CRNS framework and its application to programmable and reconfigurable cryptography," *ACM Transactions on Architecture and Code Optimization*, vol. 9, no. 4, pp. 1–25, January 2013.
[3] 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.
[4] G. Cardarilli, A. Re., R. Lojacono, A. Nannarelli, and M. Re, "RNS implementation of high performance filters for satellite demultiplexing," in *IEEE Aerospace Conference*, vol. 3, 2003, pp. 1365–1379.
[5] G. Bernocchi, G. Cardarili, A. Nannarelli, and M. Re, "Low power adaptive filter based on RNS components," in *12th IEEE International Symposium on Circuits and Systems (ISCAS)*, 2007, pp. 3211–3214.
[6] R.Conway and J. Nelson, "Improved RNS FIR filter architectures," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 51, no. 1, pp. 26–28, 2004.
[7] M. Griffin, F. Taylor, and M. Sousa, "New scaling algorithms for the chinese remainder theorem," in *Twenty-Second Asilomar Conference on Signals, Systems and Computers*, vol. 1, 1988, pp. 375–378.
[8] M. Griffin, M. Sousa, and F. Taylor, "Efficient scaling in the residue number system," in *International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, vol. 2, 1989, pp. 1075–1078.
[9] Z. Ulman and M. Czyzak, "Highly parallel, fast scaling of numbers in nonredundant residue arithmetic," *IEEE Transactions on Signal Processing*, vol. 46, no. 2, pp. 487–496, 1998.
[10] M. Shenoy and R. Kumaresan, "A fast and accurate RNS scaling technique for high speed signal processing," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 37, no. 6, pp. 929–937, 1989.

[11] F. Barsi and M. Pinotti, "Fast base extension and precise scaling in RNS for look-up table implementations," *IEEE Transactions on Signal Processing*, vol. 43, no. 10, pp. 2427–2430, 1995.

[12] A. Garcia and A. Lloris, "A look-up scheme for scaling in the RNS," *IEEE Transactions on Computers*, vol. 48, no. 7, pp. 748–751, 1999.

[13] K. Yinan and B. Phillips, "Fast scaling in the residue number system," *IEEE Transactions on Very Large Scale Integration (VLSI)*, vol. 17, no. 3, pp. 443–447, 2009.

[14] S. Ma, J. Hu, Y. Ye, L. Zhang, and X. Ling, "A 2^n scaling scheme for signed RNS integers and its VLSI implementation," *Science in China Series F: Information Sciences*, vol. 53, no. 1, pp. 203–212, 2010.

[15] C. H. Chang, J. Low, and S. Yung, "Simple, fast, and exact RNS scaler for the three-moduli set $\{2^n-1, 2^n, 2^n+1\}$," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 58, no. 11, pp. 2686–2697, 2011.

[16] T. Tay, C. H. Chang, and J. Low, "Efficient VLSI implementation of 2^n scaling of signed integer in RNS $\{2^n-1, 2^n, 2^n+1\}$," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 10, pp. 1936–1940, 2013.

[17] J. Y. S. Low and C. H. Chang, "A VLSI efficient programmable power-of-two scaler for $\{2^n-1, 2^n, 2^n+1\}$ RNS," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 12, pp. 2911–2919, 2012.

[18] M. Sheu, S. Lin, Y. Chen, and Y. Chang, "High-speed and reduced-area RNS forward converter based on $(2^n-1, 2^n, 2^n+1)$ moduli set," in *IEEE Asia-Pacific Conference on Circuits and Systems*, vol. 2, 2004, pp. 821–824.

[19] Y. Wang, X. Song, M. Aboulhamid, and H. Shen, "Adder based residue to binary number converters for $(2^n-1, 2^n, 2^n+1)$," *IEEE Transactions on Signal Processing*, vol. 50, no. 7, pp. 1772–1779, 2002.

[20] D. Gallaher, F. Petry, and P. Srinivasan, "The digital parallel method for fast RNS to weighted number system conversion for specific moduli $(2^k-1, 2^k, 2^k+1)$," *IEEE Trans. Circuits Syst. II*, vol. 44, no. 1, pp. 53–57, 1997.

[21] R. Chaves and L. Sousa, "$\{2^n+1, 2^{n+k}, 2^n-1\}$: a new RNS moduli set extension," in *Euromicro Symposium on Digital System Design (DSD)*, 2004, pp. 210–217.

[22] P. V. Mohan and A. B. Premkumar, "RNS-to-binary converters for two four-moduli sets $\{2^n-1, 2^n, 2^n+1, 2^{n+1}-1\}$ and $\{2^n-1, 2^n, 2^n+1, 2^{n+1}+1\}$," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 54, no. 6, pp. 1245–1254, 2007.

[23] L. Sousa, S. Antao, and R. Chaves, "On the design of RNS reverse converters for the four-moduli set $\{2^n+1, 2^n-1, 2^n, 2^{n+1}+1\}$," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 21, no. 10, pp. 1945–1949, 2013.

[24] K. Gbolagade, R. Chaves, L. Sousa, and S. Cotofana, "An improved RNS reverse converter for the $\{2^{2n+1}-1, 2^n, 2^n-1\}$ moduli set," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2010, pp. 2103–2106.

[25] A. S. Molahosseini and K. Navi, "A reverse converter for the enhanced moduli set $\{2^n-1, 2^n+1, 2^{2n}, 2^{2n+1}-1\}$ using CRT and MRC," in *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, 2010, pp. 456–457.

[26] L. Sousa and S. Antao, "MRC-based RNS reverse converters for the four-moduli sets $\{2^n+1, 2^n-1, 2^n, 2^{2n+1}-1\}$ and $\{2^n+1, 2^n-1, 2^{2n}, 2^{2n+1}-1\}$," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 59, no. 4, pp. 244–248, 2012.

[27] Y.-C. Kuo, S.-H. Lin, M. hwa Sheu, J.-Y. Wu, and P.-S. Wang, "Efficient VLSI design of a reverse RNS converter for new flexible 4-moduli set $(2^{p+k}, 2^p+1, 2^p-1, 2^{2p}+1)$," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2009, pp. 437–440.

[28] G. Chalivendra, V. Hanumaiah, and S. Vrudhula, "A new balanced 4-moduli set $\{2^k, 2^n-1, 2^n+1, 2^{n+1}-1\}$ and its reverse converter design for efficient FIR filter implementation," in *ACM 21st edition of the Great Lakes Symposium on VLSI (GLSVLSI)*, 2011, pp. 139–144.

[29] H. Pettenghi, J. Ambrose, R. Chaves, and L.Sousa, "Method for designing multi-channel RNS architectures to prevent power analysis SCA," in *IEEE International Symposium on Circuits and Systems (ISCAS)*, 2014, pp. 2233–2236.

[30] A. S. Molahosseini and K. Navi, *Applications of Digital Signal Processing*. InTech, 2011, ch. Study of the Reverse Converters for the Large Dynamic Range Four-Moduli Sets, pp. 337–350.

[31] R. Zimmermann, "Efficient VLSI implementation of modulo $2^n \pm 1$ addition and multiplication," in *14th IEEE Symposium on Computer Arithmetic (ARITH)*, 1999, pp. 158–167.

[32] G. Dimitrakopoulos, D. G. Nikolos, H. Vergos, D. Nikolos, and C. Efstathiou, "New architectures for modulo 2^n-1 adders," in *12th IEEE International Conference on Electronics, Circuits and Systems (ICECS)*, 2005, pp. 1–4.

[33] H. Vergos, C. Efstathiou, and D. Nikolos, "Diminished-one modulo 2^n+1 adder design," *IEEE Transactions on Computers*, vol. 51, no. 12, pp. 1389–1399, 2002.

[34] P. Matutino, R. Chaves, and L. Sousa, "Arithmetic-based binary-to-RNs converter modulo $2^n \pm k$ for jn-bit dynamic range," *IEEE Transctions on Very Large Scale Integration (TVLSI) Systems*, 2014.

Leonel Sousa received a Ph.D. degree in Electrical and Computer Engineering from the Instituto Superior Tecnico (IST), Universidade de Lisboa (UL), Lisbon, Portugal, in 1996, where he is currently Full Professor. He is also a Senior Researcher with the R&D Instituto de Engenharia de Sistemas e Computadores (INESC-ID). His research interests include VLSI architectures, computer architectures, parallel computing, computer arithmetic, and signal processing systems. He has contributed to more than 200 papers in journals and international conferences, for which he got several awards - such as, DASIP'13 Best Paper Award, SAMOS'11 'Stamatis Vassiliadis' Best Paper Award, DASIP'10 Best Poster Award, and the Honorable Mention Award UTL/Santander Totta for the quality of the publications in 2009. He has contributed to the organization of several international conferences, namely as program chair and as general and topic chair, and has given keynotes in some of them. He has edited two special issues of international journals, and he is currently Associate Editor of the IEEE Transactions on Multimedia, IEEE Transactions on Circuits and Systems for Video Technology and Springer JRTIP, and Editor-in-Chief of the Eurasip JES. He is Fellow of the IET, a Senior Member of both IEEE and ACM, and Member of IFIP WG10.3 on concurrent