Forward and Reverse Converters for the Moduli-Set $\{2^{2q+1}, 2^q + 2^{q-1} \pm 1\}$

Ghassem Jaberipur, Bardia Nadimi, R. Kazemi, and Jeong-A Lee, Member, IEEE

Abstract—Modulo-$(2^q + 2^{q-1} \pm 1)$ adders have recently been implemented using the regular parallel prefix (RPP) architecture, matching the speed of the widely used modulo-$(2^q \pm 1)$ RPP adders. Consequently, we introduce a new moduli set $\tau^+ = \{2^{2q+1}, 2^q + 2^{q-1} \pm 1\}$, with over $(2^{q+2}) \times$ dynamic range and adder speeds comparable to the conventional $\tau = \{2^q, 2^q \pm 1\}$ set. However, to fully leverage τ^+ in residue number system applications, a complete set of circuitries is necessary. This work focuses on the design and implementation of the forward and reverse converters for τ^+. These converters consist of four and seven levels of carry-save addition units, culminating in a final modulo-$(2^q + 2^{q-1} \pm 1)$ and modulo-$(2^{2q+1} + 2^{2q-2} - 1)$ adder, respectively. Through analytical evaluations and circuit simulations, we demonstrate that the overall performance of a sequence of operations—including residue generation, k additions, and reverse conversion—using τ^+ surpasses that of τ when k exceeds a certain practical threshold.

Index Terms—modular addition, parallel prefix adder, residue number system, forward and reverse conversions.

I. INTRODUCTION

THE decades-long classical moduli-set $\tau = \{2^q, 2^q \pm 1\}$ has been frequently used in numerous applications of residue number systems (RNS), including the popular deep neural network hardware accelerators [1]–[3], FIR filters [4], [5], and image processing [6]–[8]. However, on demand for additional dynamic range (DR), one has two options; namely 1) Increase the channel widths q, to the extent that additional delay is tolerable. 2) Introduce additional τ- balanced moduli (i.e., augmenting τ with some moduli, for which the same speed arithmetic operations are possible). In other words, as a necessary but not essentially sufficient condition, the residues of any added moduli should be representable with roughly the same q bits, as of the original moduli τ, as higher values of q will lead to increased delays and make the moduli set imbalance. For example, the two conjugate moduli $2^q + 2^{q-1} \pm 1$, have been recently studied in [9] and [10], where the corresponding modular adders, and that of 2^{2q+1}, with parallel prefix architectures, show to be equally fast as the similar modulo-$(2^q \pm 1)$ adders in regular parallel prefix (RPP, as named in [11]) realizations [12]–[15]. Therefore, $\tau^+ = \{2^{2q+1}, 2^q + 2^{q-1} \pm 1\}$ can be used as efficiently as the popular τ, with the benefit of (2^{q+2})X DR.

G. Jaberipur is with the Brain Pool Program for Chosun University, Gwangju, South Korea (e-mail: Jaberipur@chosun.ac.kr). B. Nadimi is a Ph.D. student at the University of South Florida, Tampa, Florida, USA (e-mail: bnadimi@usf.edu). R. Kazemi is a graduate of Computer Engineering from the Department of Computer Science and Engineering, Shahid Beheshti University, Tehran, Iran (e-mail: R.Kazemi@mail.sbu.ac.ir). Jeong-A Lee is with the Department of Computer Engineering, Chosun University, Republic of Korea (e-mail: Jalee@chosun.ac.kr).

However, for a truly effective use in the prospective applications, one needs to design and implement balanced multipliers with those of τ, and provide for reasonably efficient forward and reverse converters for τ^+.

In this work, we offer the required converters and leave the multipliers for another ongoing project. The remainder of this work contains the following sections. We provide a brief introduction to RNS essentials, including the Chinese remainder theorem (CRT) in the next section. Details of forward and reverse converters for τ^+ can be found in Section III, and analytical gate-level evaluation and the results of circuit simulation and synthesis are in Section IV. Finally, see the concluding remarks in Section V.
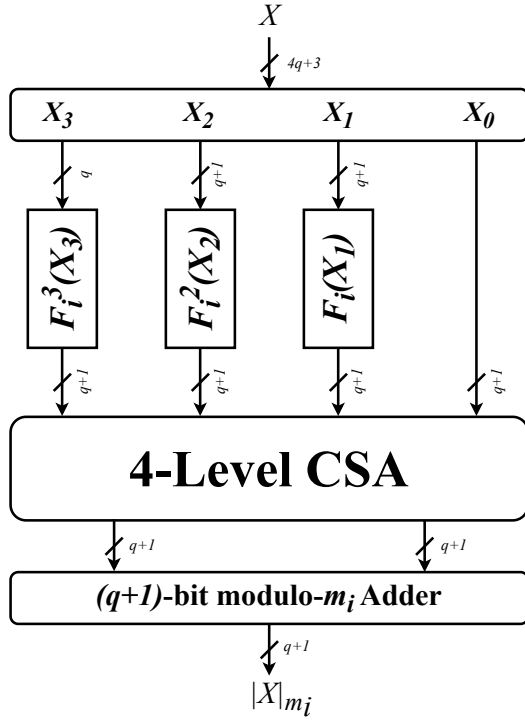
II. RNS ESSENTIALS

Binary numbers X, Y, and Z are represented in a k-moduli RNS $\{m_1, \cdots, m_k\}$, as $X = (x_1, \cdots, x_k)$, $Y = (y_1, \cdots, y_k)$, and $Z = (z_1, \cdots, z_k)$, where $x_i = |X|_{m_i}$, $y_i = |Y|_{m_i}$, and $z_i = |Z|_{m_i}$, denote the remainder of integer divisions X/m_i, Y/m_i, and Z/m_i, for $1 \leq i \leq k$, respectively. The most viable RNS operations are $+$ and $\times$, where the RNS equivalent of binary operations $Z = X + Y$ and $Z = X \times Y$, are obtained as $z_i = |x_i + y_i|_{m_i}$, and $z_i = |x_i \times y_i|_{m_i}$, respectively.

Extracting the residues from the binary operand is called residue generation (aka forward conversion). This simple operation is commonly done in parallel for all the k moduli. However, the reverse conversion, as a function of all the k residues, is often hard to implement. This is through one or a mixture of alternative versions of the Chinese remainder theorem (CRT); namely the plain CRT, New CRT [16], and mixed radix conversion [17], where the latter is a sequential operation manipulating the residues one by one. Equation (1) provides for the New CRT, where $\mu_i = p_{2-i} p_{\hat{1}-i}$, $p_{2-i} = \prod_{j=2}^i m_j$, $p_{\hat{1}-i}$ is the multiplicative inverse of $p_{1-i} = \prod_{j=1}^i m_j$ (i.e., $|p_{1-i} p_{\hat{1}-i}|_{p_{i+1-k}} = 1$), $p_{i+1-k} = \prod_{j=i+1}^k m_j$ $(1 \leq i < k)$, $x_i = |X|_{m_i}$, and $M_1 = M/m_1$.

$$X = x_1 + m_1 \left| \sum_{i=1}^{k-1} \mu_i (x_{i+1} - x_i) \right|_{M_1} \tag{1}$$

For example, (2) describes the New CRT formula for τ, where $\mu_1 = 2^q$ and $\mu_2 = 2^{q-1} + 1$.

$$X = x_1 + 2^q |\mu_1(x_2 - x_1) + \mu_2(x_3 - x_2)|_{2^{2q}-1} \tag{2}$$

Fig. 1: Modulo-m_i forward converter ($i \in \{2, 3\}$)

III. FORWARD AND REVERSE CONVERTERS FOR τ^+

Forward conversion for the commonly utilized moduli $2^q \pm 1$, is straightforward, employing simple circuit designs as detailed in [11], [14]. However, designing residue generators for the more general case of $2^q \pm \delta$ where $3 \leq \delta \leq 2^{q-1} + 1$ (including $2^q + 2^{q-1} \pm 1$) has not been extensively explored in the literature. This is despite the fact that efficient modular adders have been developed for $\delta = 2^k \pm 1$, where $k \leq q - 1$ (including $2^q + 2^{q-1} \pm 1$) [18], [19]. Therefore, there is a clear motivation to study residue generators for τ^+.

A. Residue generators (forward converters) for τ^+

The DR of $\tau^+ = \{m_1 = 2^{2q+1}, m_2 = 2^q + 2^{q-1} - 1, m_3 = 2^q + 2^{q-1} + 1\}$ covers $(4q + 3)$-bit integers in $[0, 2^{4q+2} + 2^{4q-1} - 2^{2q+1})$ that can be represented as in (3), where $X_3 = 0x_{4q+2} \cdots x_{3q+3}$, $X_2 = x_{3q+2} \cdots x_{2q+2}$, $X_1 = x_{2q+1} \cdots x_{q+1}$, and $X_0 = x_q \cdots x_0$.

$$X = 2^{3q+3}X_3 + 2^{2q+2}X_2 + 2^{q+1}X_1 + X_0 \quad (3)$$

Let $F_2(Z) = |2^{q+1}Z|_{m_2}$ and $F_3(Z) = |2^{q+1}Z|_{m_3}$ be implemented via $2^{q+1}(q+1)$-bit look-up tables (LUT), where Z is a $(q+1)$-bit residue. Also, let $F_i^2(Z) = F_i(F_i(Z))$, and $F_i^3(Z) = F_i(F_i(F_i(Z)))$. Therefore, (4), and (5) are easily derived. Fig. 1 depicts the required circuitry for $|X|_{m_2}$ and $|X|_{m_3}$, where the critical delay path travels through one LUT, four levels of CSAs, and one modulo-m_i adder.

$$|X|_{m_1} = |2^{q+1}X_1 + X_0|_{2^{2q+1}} = x_{2q} \cdots x_{q+1}x_q \cdots x_0 \quad (4)$$

$$|X|_{m_i} = |F_i^3(X_3) + F_i^2(X_2) + F_i(X_1) + X_0|_{m_i} \quad (5)$$

B. Multiple-residue to binary (Reverse) converter for τ^+

We use the $\{m_1, \{m_2, m_3\}\}$ grouping of moduli, and apply the New CRT formula (1) on $\{m_2, m_3\}$, and on $\{m_1, m_2m_3\}$. However, after some tedious elaborations, we combine the two formulas leading to a more efficient implementation. Application of the New CRT on the moduli pair $m_2 = 2^q + 2^{q-1} - 1$, and $m_3 = m_2 + 2$, leads to (6), as follows, where the auxiliary definitions ⓐ to ⓗ can be found in the Appendix A.

New CRT on $\{m_2, m_3\}$:
$X_{23} = x_3 + m_3|\mu_2(x_2 - x_3)|_{m_2} =$
$x_3 + |\mu_2 m_3(x_2 - x_3)|_{m_2 m_3}$, per ⓐ.
Let $X'_{23} = |\mu_2 m_3(x_2 - x_3)|_{m_2 m_3}$, per ⓓ - ⓔ $\Rightarrow$
$X'_{23} = |16\mu_2 m_3(x'_2 - x'_3) + \mu_2 m_3(x''_2 - x''_3)|_{m_2 m_3} =$
$|8(m_2 + 1)m_3(x'_2 - x'_3) + \mu_2 m_3(x''_2 - x''_3)|_{m_2 m_3}$, per ⓑ - ⓔ
$\Rightarrow$

$$X'_{23} = |8m_3(x'_2 - x'_3) + \mu_2 m_3(x''_2 - x''_3)|_{m_2 m_3} \quad (6)$$

For the 2nd application of New CRT the following are used:
New CRT on $\{m_1, m_2m_3\}$:
$X = x_1 + m_1 X', X' = |\mu_1(X_{23} - x_1)|_{m_2 m_3} =$
$|\mu_1(x_3 + X'_{23} - x_1)|_{m_2 m_3} = |X''_{23} + \mu_1(x_3 - x_1)|_{m_2 m_3}$, where
$X''_{23} = |\mu_1 X'_{23}|_{m_2 m_3}$.
$X''_{23} = |(3 \times 2^{q-4}(m_2 + 1) + 1)X'_{23}|_{m_2 m_3}$, per ⓗ $\Rightarrow$
$X''_{23} = |(3 \times 2^{q-4} + 1)X'_{23}|_{m_2 m_3} =$
$|((3 \times 2^{q-4} + 1)(8m_3(x'_2 - x'_3) + \mu_2 m_3(x''_2 - x''_3)))|_{m_2 m_3}$, per
(9) $\Rightarrow X''_{23} = |((3 \times 2^{q-1} + 8)m_3(x'_2 - x'_3) + \mu_2(3 \times 2^{q-4} + 1)m_3(x''_2 - x''_3))|_{m_2 m_3} = |((m_2 + 9)m_3(x'_2 - x_3) + (2\mu_2(3 \times 2^{q-5}) + \mu_2)m_3(x''_2 - x''_3))|_{m_2 m_3}$, per ⓖ $\Rightarrow$

$$X''_{23} = |9m_3(x'_2 - x'_3) + (3 \times 2^{q-5} + \mu_2)m_3(x''_2 - x''_3)|_{m_2 m_3} \quad (7)$$

$X' = |\mu_1(x_3 - x_1) + X''_{23}|_{m_2 m_3} = |(\mu_1(16x'_3 + x''_3) - \mu_1 x_1 + 9m_3(x'_2 - x'_3) + (3 \times 2^{q-4} + 1)m_3(x''_2 - x''_3))|_{m_2 m_3}$
The latter leads to (8) as found below, via replacing the negative terms with their positive complementary equivalents. The corresponding tedious elaborations can be found in the Appendix B.

$x_1 = |X|_{m_1} = 8x'_1 + x''_1, x'_1 = x_{1_{2q}} \cdots x_{1_3}, x''_1 = x_{1_2}x_{1_1}x_{1_0},$
$-x'_1 = \overline{x'_1} + 1 - 2^{2q-2}, \overline{x'_1} = \overline{x_{1_{2q}}} \cdots \overline{x_{1_3}}, -x''_1 = \overline{x''_1} - 7, \overline{x''_1} = \overline{x_{1_2}x_{1_1}x_{1_0}}, -x'_3 = \overline{x'_3} + 1 - 2^{q-2}, \overline{x'_3} = \overline{x_{3_q}} \cdots \overline{x_{3_3}}, -x''_3 = \overline{x''_3} - 7, \overline{x''_3} = \overline{x_{3_2}x_{3_1}x_{3_0}}.$
$|-9x'_1|_{m_2 m_3} = |-9 \times 2^{2q-2} + 9 + 9\overline{x'_1}|_{9 \times 2^{2q-2} - 1} = |9\overline{x'_1} + 8|_{9 \times 2^{2q-2} - 1}.$

$X = x_1 + 2^{2q+1}X', X' =$

$\left| \begin{array}{l} (2^{q+3} + 2^{q+2} + 2^q + 2^{q-1})(x'_2 + \overline{x'_3}) + x''_3 + \\ 9(\overline{x'_1} + x'_2 + x'_3) + 2^{2q-2}x_{2_2}x_{2_1}x_{2_0} + \\ (2^{2q-2} + 2^{2q-5} + 1)\overline{x''_1} + 2^{2q-6}(x''_2 + x''_3) + \\ 2^{2q-2}(\overline{x_{2_3}x_{2_2}x_{2_1}} + x_{3_3}x_{3_2}x_{3_1}) + 2^{2q-2}\overline{x_{2_3}} + x_{2_3} + \\ (2^{q-1} + 2^{q-2} + 2^{q-4} + 2^{q-5})(x''_2 + \overline{x''_3}) + \\ \overline{x_{3_3}x_{3_2}x_{3_1}} + 2^{2q}(x_{2_0} + \overline{x_{3_0}}) + x_{2_3}x_{2_2}x_{2_1} + \\ 2^{2q-4} + 2^{2q-5} + 2^{q-1} + 2^{q-2} + 2^{q-4} + 2^{q-5} - 9 \end{array} \right|_{9 \times 2^{2q-2} - 1}$
$\quad (8)$

TABLE I: The constituent bits of the $X^{'}$-expression

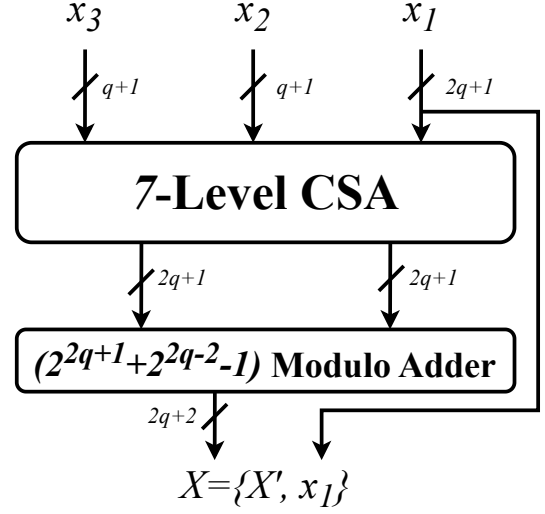|	2q	2q−1	2q−2	2q−3	2q−4	2q−5	2q−6	2q−7	⋯	q+3	q+2	q+1	q	q−1	q−2	q−3	q−4	q−5	q−6	⋯	4	3	2	1	0
1				$\overline{x_{1_{2q}}}$	$\overline{x_{1_{2q-1}}}$	$\overline{x_{1_{2q-2}}}$	$\overline{x_{1_{2q-3}}}$	$\overline{x_{1_{2q-4}}}$		$\overline{x_{1_{q+6}}}$	$\overline{x_{1_{q+5}}}$	$\overline{x_{1_{q+4}}}$	$\overline{x_{1_{q+3}}}$	$\overline{x_{1_{q+2}}}$	$\overline{x_{1_{q+1}}}$	$\overline{x_{1_q}}$	$\overline{x_{1_{q-1}}}$	$\overline{x_{1_{q-2}}}$	$\overline{x_{1_{q-3}}}$		$\overline{x_{1_7}}$	$\overline{x_{1_6}}$	$\overline{x_{1_5}}$	$\overline{x_{1_4}}$	$\overline{x_{1_3}}$
2	$x_{1_{2q}}$	$x_{1_{2q-1}}$	$x_{1_{2q-2}}$	$x_{1_{2q-3}}$	$x_{1_{2q-4}}$	$x_{1_{2q-5}}$	$x_{1_{2q-6}}$	$x_{1_{2q-7}}$		$x_{1_{q+3}}$	$x_{1_{q+2}}$	$x_{1_{q+1}}$	x_{1_q}	$x_{1_{q-1}}$	$x_{1_{q-2}}$	$x_{1_{q-3}}$	$x_{1_{q-4}}$	$x_{1_{q-5}}$	$x_{1_{q-6}}$					x_{1_4}	x_{1_3}
3	x_{2_2}	x_{2_1}	x_{2_0}	x_{2_q}	1	$\overline{x_{2_q}}$	$x_{2_{q-1}}$	$x_{2_{q-2}}$		x_{2_8}	x_{2_7}	x_{2_6}	x_{2_5}	x_{2_4}			x_{2_q}	$x_{2_{q-1}}$	$x_{2_{q-2}}$		x_{2_8}	x_{2_7}	x_{2_6}	x_{2_5}	x_{2_4}
4	$\overline{x_{2_3}}$	$\overline{x_{2_2}}$	$\overline{x_{2_1}}$			$x_{2_{q-1}}$	$x_{2_{q-2}}$	$x_{2_{q-3}}$		x_{2_7}	x_{2_6}	x_{2_5}	x_{2_4}	x_{2_q}	$x_{2_{q-1}}$	$x_{2_{q-2}}$	$x_{2_{q-3}}$	$x_{2_{q-4}}$	$x_{2_{q-5}}$		x_{2_5}	x_{2_4}	$\overline{x_{1_2}}$	$\overline{x_{1_1}}$	$\overline{x_{1_0}}$
5	x_{2_0}		x_{2_q}	$x_{2_{q-1}}$	$x_{2_{q-2}}$	$x_{2_{q-3}}$	$x_{2_{q-4}}$	$x_{2_{q-5}}$		x_{2_5}	x_{2_4}			1	1		x_{3_q}	$x_{3_{q-1}}$	$x_{3_{q-2}}$		x_{3_8}	x_{3_7}	x_{3_6}	x_{3_5}	x_{3_4}
6		x_{2_q}	$x_{2_{q-1}}$	$x_{2_{q-2}}$	$x_{2_{q-3}}$	$x_{2_{q-4}}$	$x_{2_{q-5}}$	$x_{2_{q-6}}$		x_{2_4}				x_{3_q}	$x_{3_{q-1}}$	$x_{3_{q-2}}$	$x_{3_{q-4}}$	$x_{3_{q-3}}$			x_{3_5}	x_{3_4}	x_{2_3}	x_{2_2}	x_{2_1}
7			$\overline{x_{2_3}}$			x_{3_q}	$x_{3_{q-1}}$	$x_{3_{q-2}}$		x_{3_8}	x_{3_7}	x_{3_6}	x_{3_5}	x_{3_4}	x_{2_3}	x_{2_2}	x_{2_1}	x_{2_0}				x_{3_3}	x_{3_2}	x_{3_1}	x_{3_0}
8	x_{3_3}	x_{3_2}	x_{3_1}		x_{3_q}	$\overline{x_{3_{q-1}}}$	$\overline{x_{3_{q-2}}}$	$\overline{x_{3_{q-3}}}$		$\overline{x_{3_7}}$	$\overline{x_{3_6}}$	$\overline{x_{3_5}}$	$\overline{x_{3_4}}$	$\overline{x_{2_3}}$	x_{2_2}	x_{2_1}	x_{2_0}						$\overline{x_{3_3}}$	$\overline{x_{3_2}}$	$\overline{x_{3_1}}$
9	$\overline{x_{3_0}}$		$\overline{x_{3_q}}$	$\overline{x_{3_{q-1}}}$	$\overline{x_{3_{q-2}}}$	$\overline{x_{3_{q-3}}}$	$\overline{x_{3_{q-4}}}$	$\overline{x_{3_{q-5}}}$		$\overline{x_{3_5}}$	$\overline{x_{3_4}}$	x_{2_3}	x_{2_2}	x_{2_1}	x_{2_0}		1								x_{2_3}
10		$\overline{x_{3_q}}$	$\overline{x_{3_{q-1}}}$	$\overline{x_{3_{q-2}}}$	$\overline{x_{3_{q-3}}}$	$\overline{x_{3_{q-4}}}$	$\overline{x_{3_{q-5}}}$			$\overline{x_{3_4}}$	x_{2_3}	x_{2_2}	x_{2_1}	x_{2_0}	$\overline{x_{3_3}}$	$\overline{x_{3_2}}$	$\overline{x_{3_1}}$	$\overline{x_{3_0}}$	1		1	0	1	1	0
11				x_{3_3}	x_{3_2}	x_{3_1}	x_{3_0}									$\overline{x_{3_3}}$	$\overline{x_{3_2}}$	$\overline{x_{3_1}}$	$\overline{x_{3_0}}$						
12	$\overline{x_{1_2}}$	$\overline{x_{1_1}}$	$\overline{x_{1_0}}$	$\overline{x_{1_2}}$	$\overline{x_{1_1}}$	$\overline{x_{1_0}}$								$\overline{x_{3_3}}$	$\overline{x_{3_2}}$	$\overline{x_{3_1}}$	$\overline{x_{3_0}}$				$\overline{x_{3_1}}$				
13		1		x_{2_3}	x_{2_2}	x_{2_1}	x_{2_0}				$\overline{x_{3_3}}$	$\overline{x_{3_2}}$	$\overline{x_{3_1}}$	$\overline{x_{3_0}}$											

Table I contains all the weighted bits of (8), organized as thirteen $(2q+1)$-bit numbers to be added modulo-$(2^{2q+1}+2^{2q-2}-1)$. However, the cases of $q \leq 8$ require some changes before implementation (e.g., $2^{q-5}-9 < 0$). The number of required full adders (FA) are shown in Table II (black shaded figures, with $1/2$ for half adders), where the depth of all columns, after each of the consecutive reduction levels, are also indicated. However, the seven carries c that spill over the leftmost column reenter the Table as in the following relation, which is appropriately reflected in Tables I and II.

$$|2^{2q+1}c|_{2^{2q+1}+2^{2q-2}-1} = (-2^{2q-2}+1)c = -2^{2q-2}+2^{2q-2}\overline{c}+c$$

Note that the delay overhead of the reentrant carries is only equal to that of two CSA levels, as the non-modular $13:2$ reduction requires five CSA levels. The final modulo-$(2^{2q+1}+2^{2q-2}-1)$ addition is handled via a modular adder, whose architecture is pretty much the same as that of modulo-$(2^q+2^{q-1}-1)$ adder of [9]. Fig. 2 depicts the overall reverse converter circuitry.

IV. EVALUATION AND COMPARISON

The comparison between two RNS moduli sets is primarily focused on the speed of modular arithmetic within the residue channels. This focus arises because the number of arithmetic operations performed between the initial residue generation and the final conversion back to binary is typically in the range of hundreds (such as the number of taps in an RNS-based FIR filter [5]) or even thousands (for example, in matrix multiplications used in machine learning [20]). For the moduli set τ^+, it has been demonstrated that the speed of modulo-$(2^q + 2^{q-1} \pm 1)$ RPP adders is comparable to that of modulo-$(2^q \pm 1)$ adders in the $\tau = 2^q, 2^q \pm 1$ moduli set, as shown in previous studies [9], [10]. However, a significant advantage of the τ^+ moduli set is that its dynamic range (DR) is at least 2^q times greater than that of the commonly used τ moduli set for the same value of q ($2^{4q+2} + 2^{4q-1} - 2^{2q+1}$ vs. $2^{3q} - 2^q$). While the forward and reverse converters for τ^+ are indeed slower and costlier than those for τ, this is not a significant drawback. The reason

x_3 $\quad$ x_2 $\quad$ x_1

$q+1$ $\quad$ $q+1$ $\quad$ $2q+1$

7-Level CSA

$2q+1$ $\quad$ $2q+1$

($2^{2q+1}+2^{2q-2}$-1) Modulo Adder

$2q+2$

$X=\{X', x_{1_j}\}$

Fig. 2: Reverse converter for moduli set τ^+

is that the overhead from these slower and costlier conversions is offset by the superior performance of the adders in τ^+, especially as the number of operations increases. As a result, the overall efficiency gains in the modular arithmetic operations outweigh the additional cost and delay introduced by the conversions. To provide a more accurate assessment, considering that the residue channels for modulo $2^q + 2^{q-1} + 1$ in τ^+ and $2^{q'} + 1$ in τ are the slowest, we have calculated the delay associated with k consecutive modular additions. These additions are preceded by a forward conversion and followed by a reverse conversion (a sequence consisting of a forward conversion, k modular additions, and a final reverse conversion). The corresponding total delay formulas are provided in equation (9).

$$\tau_{delay} = (18 + 4\lceil \log q^{'} \rceil + k(3 + 2\lceil \log q^{'} \rceil))\Delta G,$$

TABLE II: Seven reduction levels represented by depth of each column and # of utilized FAs and HAs.

Level	Column #	2q	2q−1	2q−2	2q−3	2q−4	2q−5	2q−6	2q−7	⋯	q	q−1	q−2	q−3	q−4	q−5	q−6	⋯	5	4	3	2	1	0	Total # of FAs
I	Depth	7	8	10	10	11	13	12	10	⋯	10	13	11	8	11	8	7	⋯	7	7	7	8	8	8	
	# FA	2	2	3	3	4	4	4	3	⋯	3	4	3	2	3	2	2	⋯	2	2	2	2	2	2	5q+8
II	Depth	5	7	7+2	7	8	9	9	8	⋯	8	8	7	7	7	6	5	⋯	5	5	5	6	6	4+2	
	# FA	1	2	3	2	2	3	3	2	⋯	2	2	2	2	2	1		⋯	1	1	1	2	2	2	3q+12
III	Depth	5	6	5+1	5	7	6	5	6	⋯	6	6	5	5	5	4	4	⋯	4	4	5	4	4	2+1	
	# FA	1	2	2	1	2	2	1	2	⋯	2	2	1	1	1	1	1	⋯	1	1	1	1	1	1	3q
IV	Depth	5	4	3+1	5	5	3	5	4	⋯	4	3	4	4	4	3	3	⋯	3	3	4	3	3	1+1	
	# FA	1	1	1	1	1	1	1	1	⋯	1	1	1	1	1	1	1	⋯	1	1	1	1	1	0	2q
V	Depth	4	3	3+1	4	4	2	4	3	⋯	3	2	3	3	3	2	2	⋯	2	2	3	2	1	2+1	
	# FA	1	1	1	1	1	0	1	1	⋯	1	0	1	1	1	0	0	⋯	0	0	1	0	0	1	q+3
VI	Depth	3	2	3+1	3	2	3	3	2	⋯	1	3	2	2	2	2	2	⋯	2	1	3	1	2	1+1	
	# FA	1	0	1	1	0	1	1	0	⋯	0	1	1/2	1/2	1/2	1/2	1/2	⋯	1/2	1	0	0	0	0	q/2+4
VII	Depth	1	3	3+1	1	3	2	1	2	⋯	2	2	2	2	2	2	2	⋯	1	1	2	1	2	2+1	
	# FA	0	1	1	0	1	0	0	0	⋯	0	0	0	0	0	0	0	⋯	0	0	0	1/2	1/2	1	5
		2	2	2	2	2	2	2	2	⋯	2	2	2	2	2	2	2	⋯	2	1	2	2	2	1	
	Grand total # of FAs																								13.5q + 32
	Columns 0 and $(2q − 1)$ receive seven reentrant and inverted reentrant carries, respectively. HA is considered as 1/2 FA.																								

$$\tau_{delay}^{+} = (2q + 45 + 4\lceil \log q \rceil + k(4 + 2\lceil \log q \rceil))\Delta G \qquad (9)$$

To equalize the dynamic ranges (DR) of the corresponding moduli sets, different values of q are employed for each set. As illustrated in Table. III, with equivalent dynamic ranges, our proposed moduli set requires fewer bits. As a result, after a certain threshold value of k, the overhead from the forward and reverse conversions in τ^{+} is surpassed by the performance gains achieved through the use of faster adders. Hence, we recommend using τ^{+} instead of τ for applications with k greater than the turning points in Table. III, such as typical FIR filters with more than 100 taps.

TABLE III: Channel-widths of τ^{+} and τ with the same DR and the turning point of using τ^{+} instead of τ.

τ^{+}		τ		Minimum k for
q	ΔG	q'	ΔG	τ^{+}-delay $< \tau$-delay
4	$8k + 61$	7	$9k + 30$	31
8	$10k + 73$	12	$11k + 34$	39
16	$12k + 93$	23	$13k + 38$	55
32	$14k + 129$	44	$15k + 42$	87

We have also implemented the residue generators, adders, and reverse converters for the moduli-set τ and moduli-set τ^{+} using Verilog. These designs were synthesized using the Vivado 2023.1 software on the Xilinx Artix-7 AC701 Evaluation Platform. The findings, which also include the estimated and power consumption for both designs, are summarized in Table. IV. A speedup is anticipated with higher k values; for example, 8% speedup is observed when $k = 100$ and $q = 8$ for τ^{+} compared to τ with an equal dynamic range and $q' = 12$. Additionally, this configuration also results in 32% reduction in power dissipation. These results are due to the smaller channel width of the adders in τ^{+}. Note that the results shown in the Table. IV correspond to specific values of k. As k increases, the area deficiency remains constant; however, the improvements in delay and power become significantly more pronounced.

TABLE IV: Figures of merit of forward/reverse converters $+k$ additions

k	Design	q	Delay		Area		Power	
			(ns)	ratio	(# of LUTs)	ratio	(μW)	ratio
39	τ^{+}	8	401.5	1.0	1383	1.0	246	1.0
	τ	12	405.6	1.01	519	0.37	309	1.25
55	τ^{+}	16	774.6	1.0	4659	1.0	657	1.0
	τ	23	783.9	1.01	1335	0.29	826	1.26
87	τ^{+}	32	1298.9	1.0	10938	1.0	1881	1.0
	τ	44	1360.3	1.04	3772	0.34	2453	1.30
100	τ^{+}	8	911.6	1.0	1383	1.0	523	1.0
	τ	12	987.6	1.08	519	0.37	693	1.32

V. CONCLUSIONS AND FUTURE WORKS

Given the design and implementation of efficient RPP modulo-$(2^q + 2^{q-1} \pm 1)$ adders in [9] and [10], with compatible speed to similar designs for moduli $(2^q \pm 1)$, we presented the new moduli set $\tau^{+} = \{2^q, 2^q + 2^{q-1} \pm 1\}$, with at least 2^q X DR in comparison with the popular moduli set $\tau = \{2^q, 2^q \pm 1\}$. There are several practical RNS applications that require the same channel bit-widths, but higher DR than that of τ. Therefore, to accordingly make the new τ^{+} set readily available, circuit design for forward and reverse conversions and the modulo-$(2^q + 2^{q-1} \pm 1)$ multipliers are in order. Design of the required converters is presented in this work. Additionally, the experimental findings reveal that with an increase in k, enhancements in power, and delay metrics are more pronounced. For example, in the case where $k = 100$ and $q = 8$, in comparison to an equal DR τ with $q = 12$, the τ^{+} configuration results in 8% speed gain, and 32% decrease in power consumption. As for the future

relevant work, we plan to conclude the ongoing research on the design and implementation of the modulo-$(2^q + 2^{q-1} \pm 1)$ multipliers.

REFERENCES

[1] M.V. Valueva, N.N. Nagornov, P.A. Lyakhov, G.V. Valuev, and N.I. Chervyakov. Application of the residue number system to reduce hardware costs of the convolutional neural network implementation. *Mathematics and Computers in Simulation*, 177:232–243, 2020.

[2] Chan Hua Vun, Annamalai Benjamin Premkumar, and Wei Zhang. A new rns based da approach for inner product computation. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60(8):2139–2152, 2013.

[3] Jiaxin Peng, Yousra Alkabani, Shuai Sun, Volker J. Sorger, and Tarek El-Ghazawi. Dnnara: A deep neural network accelerator using residue arithmetic and integrated photonics. In *Proceedings of the 49th International Conference on Parallel Processing*, ICPP '20, New York, NY, USA, 2020. Association for Computing Machinery.

[4] Grande Naga Jyothi, Kishore Sanapala, and A. Vijayalakshmi. Asic implementation of distributed arithmetic based fir filter using rns for high speed dsp systems - international journal of speech technology, Feb 2020.

[5] Armin Belghadr and Ghassem Jaberipur. Fir filter realization via deferred end-around carry modular addition. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 65(9):2878–2888, 2018.

[6] Wei Wang, M.N.S. Swamy, and M.O. Ahmad. Rns application for digital image processing. In *4th IEEE International Workshop on System-on-Chip for Real-Time Applications*, pages 77–80, 2004.

[7] Evangelos Vassalos, Dimitris Bakalis, and Haridimos T. Vergos. Rns assisted image filtering and edge detection. In *2013 18th International Conference on Digital Signal Processing (DSP)*, pages 1–6, 2013.

[8] N. I. Chervyakov, P. A. Lyakhov, and M. G. Babenko. Digital filtering of images in a residue number system using finite-field wavelets - automatic control and computer sciences, Jul 2014.

[9] Ghassem Jaberipur and Bardia Nadimi. Balanced $(3+2 \log n)\delta g$ adders for moduli set $\{2^{n+1}, 2^n + 2^{n-1} - 1, 2^{n+1} - 1\}$. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 67(4):1368–1377, 2020.

[10] Ghassem Jaberipur, D. Badri, and Jeong-A Lee. A parallel prefix modulo-$(2^q + 2^{q-1} + 1)$ adder via diminished-1 representation of residues. *IEEE Transactions on Circuits and Systems II: Express Briefs*, 70(8):3104–3108, 2023.

[11] L. Kalampoukas, D. Nikolos, C. Efstathiou, H.T. Vergos, and J. Kalamatianos. High-speed parallel-prefix module 2/sup n/-1 adders. *IEEE Transactions on Computers*, 49(7):673–680, 2000.

[12] Haridimos T. Vergos and Giorgos Dimitrakopoulos. On modulo $2^n + 1$ adder design. *IEEE Transactions on Computers*, 61(2):173–186, 2012.

[13] C. Efstathiou, H.T. Vergos, and D. Nikolos. Fast parallel-prefix modulo $2^n + 1$ adders. *IEEE Transactions on Computers*, 53(9):1211–1216, 2004.

[14] H.T. Vergos, C. Efstathiou, and D. Nikolos. Diminished-one modulo $2^n + 1$ adder design. *IEEE Transactions on Computers*, 51(12):1389–1399, 2002.

[15] Ghassem Jaberipur and Saeed Nejati. Balanced minimal latency rns addition for moduli set $\{2^n - 1, 2^n, 2^n + 1\}$. In *2011 18th International Conference on Systems, Signals and Image Processing*, pages 1–7, 2011.

[16] Yuke Wang. Residue-to-binary converters based on new chinese remainder theorems. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 47(3):197–205, 2000.

[17] Huang. A fully parallel mixed-radix conversion algorithm for residue number applications. *IEEE Transactions on Computers*, C-32(4):398–402, 1983.

[18] Shang Ma, Jianhao Hu, and Chen-Hao Wang. A novel modulo $2^n - 2^k - 1$ adder for residue number system. *IEEE Transactions on Circuits and Systems I: Regular Papers*, 60:2962–2972, 2013.

[19] Seyed Hamed Fatemi Langroudi and Ghassem Jaberipur. Modulo-$(2^n - 2^q - 1)$ parallel prefix addition via excess-modulo encoding of residues. In *2015 IEEE 22nd Symposium on Computer Arithmetic*, pages 121–128, 2015.

[20] Javier Fernández, Jon Perez, Irune Agirre, Imanol Allende, Jaume Abella, and Francisco J. Cazorla. Towards functional safety compliance of matrix–matrix multiplication for machine learning-based autonomous systems. *Journal of Systems Architecture*, 121:102298, 2021.

$$X = x_1 + 2^{2q+1} \begin{vmatrix} \overline{x_1'} + x_2' + \overline{x_1} + x_2 + 2^{2q} + 2^{2q-2} + \\ (2^q + 2^{q-1} + 2^{q-3} + 2^{q-4})(x_2 + \overline{x_3}) + \\ (2^{2q-2} + 2^{2q-5})(\overline{x_1''} + x_2'') + 2^{2q-3} + 2^{2q-5} + \\ 2^q + 2^{q-1} + 2^{q-3} + 2^{q-4} - 2 \end{vmatrix}_{9 \times 2^{2q-2}-1}$$

Appendix

Appendix A (Auxiliary equations):

(a) $m|Z|_{m'} = |mZ|_{mm'}$ since $m|Z|_{m'} =$

$$m \begin{cases} Z & \text{if } Z < m' \\ Z - m' & \text{if } Z \geq m' \end{cases} =$$

$$\begin{cases} mZ & \text{if } mZ < mm' \\ mZ - mm' & \text{if } mZ \geq mm' \end{cases}$$

(b) $|\mu_2 m_3|_{m_2} = 1 \Rightarrow \mu_2 = 3 \times 2^{q-2}$

(c) $2\mu_2 = 3 \times 2^{q-1} = m_2 + 1$

(d) $x_2 = |X|_{m_2} = 16x_2' + x_2'', x_2' = x_{2_q} \cdots x_{2_4}, x_2'' = x_{2_3} x_{2_2} x_{2_1} x_{2_0}$

(e) $x_3 = |X|_{m_3} = 16x_3' + x_3'', x_3' = x_{3_q} \cdots x_{3_4}, x_3'' = x_{3_3} x_{3_2} x_{3_1} x_{3_0}$

(f) $|\mu_1 m_1|_{m_2 m_3} = 1 \Rightarrow$
$\mu_1 = 2^{2q-2} + 2^{2q-5} + 1 = 9 \times 2^{2q-5} + 1$

(g) $|8\mu_1|_{m_2 m_3} = |9 \times 2^{2q-2} + 8|_{9 \times 2^{2q-2}-1} = 9$

(h) $\mu_1 = 3 \times 2^{q-4} \times 3 \times 2^{q-1} + 1 = 3 \times 2^{q-4}(m_2 + 1) + 1$

Appendix B (Derivation of (8)):

$$X' = |X'' + X_{23}''|_{m_2 m_3} =$$

$$\begin{vmatrix} 9x_3' + \mu_1 x_3'' - 9x_1' - \mu_1 x_1'' + 9m_3(x_2' - x_3') + \\ (3 \times 2^{q-4} + 1)m_3(x_2'' - x_3'') \end{vmatrix}_{m_2 m_3} =$$

$$\begin{vmatrix} -x_3' \times 9(m_3 - 1) + 9\overline{x_1'} + 9 \times 2^{2q-5}(-7 + \overline{x_1''}) + \\ (9 \times 2^{2q-5} + 1 - (3 \times 2^{2q-4} + 1)(3 \times 2^{q-1} + 1))x_3'' \\ + (3 \times 2^{q-4} + 1)(3 \times 2^{q-1} + 1)x_2'' \\ + 8 - 7 + \overline{x_1''} + 9m_3 x_2' \end{vmatrix}_{m_2 m_3} =$$

$$\begin{vmatrix} 27 \times 2^{q-1}(\overline{x_3'} + 1) + 9 \times 2^{2q-3} - 2 - \\ 27 \times 2^{q-4}x_3'' + 9\overline{x_1'} + 8 - 63 \times 2^{2q-5} + \\ 9 \times 2^{2q-5}\overline{x_1''} - 7 + \overline{x_1''} + 27 \times 2^{q-1}x_2' + 9x_2' + \\ (9 \times 2^{2q-5} + 3 \times 2^{q-4} + 3 \times 2^{q-1} + 1)x_2'' \end{vmatrix}_{m_2 m_3} =$$

$$\begin{vmatrix} 27 \times 2^{q-1}\overline{x_3'} + 27 \times 2^{q-4}\overline{x_3''} - 7 \times 27 \times 2^{q-4} + \\ 9\overline{x_1'} + 9 \times 2^{2q-5}\overline{x_1''} + \overline{x_1''} + 27 \times 2^{q-1}x_2' + 9x_2' + \\ (9 \times 2^{2q-5} + 27 \times 2^{q-4})x_2'' + x_2'' + 27 \times 2^{q-1} + \\ 9 \times 2^{2q-3} - 2 + 8 - 63 \times 2^{2q-5} - 7 \end{vmatrix}_{m_2 m_3} =$$

$$\begin{vmatrix} 9(\overline{x_1'} + x_2') + 27 \times 2^{q-1}(\overline{x_3'} + x_2') + \overline{x_1''} + \\ x_2'' 9 \times 2^{2q-5}(\overline{x_1''} + x_2'') + 27 \times 2^{q-4}(x_2'' + \overline{x_3''}) + \\ 27 \times 2^{q-4} - 27 \times 2^{2q-5} - 1 + 9 \times 2^{2q-2} - 1 \end{vmatrix}_{m_2 m_3} =$$

$$\begin{vmatrix} 8\overline{x_1'} + \overline{x_1''} + 8x_2' + x_2'' + 9 \times 2^{2q-5}(\overline{x_1''} + x_2'') + \overline{x_1'} + \\ x_2' + 27 \times 2^{q-4}(8\overline{x_3'} + \overline{x_3''} + 8x_2' + x_2'') + 27 \times 2^{q-4} \\ + 45 \times 2^{2q-5} - 2 \end{vmatrix}_{m_2 m_3} =$$

$$\begin{vmatrix} \overline{x_1'} + x_2' + \overline{x_1} + x_2 + 27 \times 2^{q-4}(x_2 + \overline{x_3}) + \\ 9 \times 2^{2q-5}(\overline{x_1''} + x_2'') + 2^{2q} + 2^{2q-2} + 2^{2q-3} + 2^{2q-5} \\ + 2^q + 2^{q-1} + 2^{q-3} + 2^{q-4} - 2 \end{vmatrix}_{m_2 m_3} \Rightarrow$$