Efficient Implementations of Residue Generators Mod $2^n + 1$ Providing Diminished-1 Representation

Stanisław J. Piestrak & Piotr Patronik

***Abstract*—The moduli of the form $2^n + 1$ belong to a class of low-cost odd moduli, which have been frequently selected to form the basis of various residue number systems (RNS). The most efficient computations modulo (mod) $2^n + 1$ are performed using the so-called diminished-1 (D1) representation. Therefore, it is desirable that the input converter from the positional number system to RNS (composed of a set of residue generators) could generate the residues mod $2^n + 1$ in D1 form. In this paper, we propose the basic architecture of the residue generator mod $2^n + 1$ with D1 output. It is universal, because its initial part can be easily designed for an arbitrary $p \geq 4n$, whereas its final block—the 4-operand adder mod $2^n + 1$—preserves the same structure for any p. If a pair of conjugate moduli $2^n \pm 1$ belongs to the RNS moduli set, the latter architecture can be easily extended to build p-input bi-residue generators mod $2^n \pm 1$, which not only save hardware by sharing $p - 4n$ full-adders, but also generate the residue mod $2^n + 1$ directly in D1 form.**

***Index Terms*—Residue arithmetic, residue number system (RNS), residue generation, modulo $2^n + 1$ arithmetic, diminished-1 representation, input converter, shared logic.**

I. Introduction

THE residue arithmetic modulo (mod) $2^n + 1$ has found numerous applications of which two distinct classes involve the non-positional residue number system (RNS) and the Fermat Number Transform (FNT) (where the Fermat number $F_n = 2^{2^n} + 1$). The RNS is defined by a set of pairwise prime natural numbers, called *moduli*, whose product determines its *dynamic range*. Its major attraction, which makes it competitive to the positional 2's complement system, is the possibility of particularly efficient (w.r.t. area, time, and power consumption) implementation of algorithms involving mostly sum of products operations. The numerous applications of RNS include implementations of the algorithms related to: RSA public-key cryptosystem [1], FIR filters [2], [3], microprocessors [4], [5], artificial intelligence [6], as well as many other DSP applications [7]. On the other hand, the FNT with diminished-1 representation was used to implement various DSP algorithms [8], to accelerate integer convolutional neural networks [9] and to reduce the computational complexity of the chromatic dispersion compensation in optical communication systems [10].

In general, any set of pairwise prime moduli could be used to form an RNS. Nevertheless, some moduli like those of the type 2^k and the conjugate moduli of the form $2^n \pm 1$ have particularly hardware-efficient and fast implementations of the residue datapaths, and hence are called *low-cost* moduli.

The authors are with Department of Computer Engineering, Faculty of Electronics (W-4/K-9), Wrocław University of Science and Technology, 50-370 Wrocław, Poland (email: stanislaw.piestrak@pwr.edu.pl; piotr.patronik@pwr.edu.pl).

Because only one even modulus can be used, of particular interest are the remaining two classes, of which $2^n - 1$ is unquestionably the best one. The next best class of odd moduli are those of the form $2^n + 1$ for which, however, some problems have been identified and resolved as follows. The normal representation of residues mod $2^n + 1$ requires $n + 1$ bits to represent all valid values from $[0, 2^n]$, which means that only $2^n + 1$ out of 2^{n+1} combinations are actually used. Amongst them, $(10 \ldots 0)$ is the only one out of $n + 1$ combinations with the Most Significant Bit (MSB) set to 1. Leibowitz [11] observed that executing arithmetic operations on $(n + 1)$-bit operands involves unnecessary hardware cost and delay. Assuming that any zero operand is recognized by a separate zero indication bit, the operations can be executed on n-bit operands, provided that each of them is decremented by 1. Such a notation was called the *diminished-1 (D1) representation*. However, to make possible execution of cheaper D1 operations, a designer must ensure two following features.

(i) The residue mod $2^n + 1$ is provided to the datapath mod $2^n + 1$ in D1 form.

(ii) Once the final result mod $2^n + 1$ is available, either it is converted from the D1 form to the normal residue representation—to make possible using generally available reverse converters, or it is applied directly to specially constructed reverse converter accepting the D1 from.

Because here we are particularly interested in the circuitry which involves the D1 representation, the following survey concentrates only on the contributions specifically taking into account the latter. The most efficient implementations of the arithmetic circuits mod $2^n + 1$ using D1 representation are: adders [12], [13], multi-operand modulo adders (MOMAs) [13], multipliers [14], and multiplier-accumulators (MACs—also called fused add-multiply units) [15], [16]. The reverse converters for the special 3-moduli set $\{2^n, 2^n - 1, 2^n + 1\}$ and its extensions which include the fourth modulus $2^{2n} + 1$, which were designed explicitly assuming that the datapath channels mod $2^n + 1$ and $2^{2n} + 1$ produce the D1 result, were proposed in [17]. An improved version of the reverse converter for the above 3-moduli set was proposed in [14] (Fig. 10).

The input (forward) converter for any RNS-based processor essentially consists of a set of residue generators for all moduli defining an RNS. Here, we are interested in designing residue generators with two features: (1) they generate the D1 output for the $2^n + 1$ modulus, and (2) they are amenable for hardware cost reduction by using shared logic with at least one residue generator for some other modulus. Indeed, relatively little works can be found on this subject. The most obvious scheme of the residue generator mod $2^n + 1$ which provides the output in D1 form consists of any normal

residue generator mod 2^n+1 (such as proposed in [18]) followed by the converter into D1 form, e.g. in the form of the modified D1 adder connected to the CSA tree [13] (Fig. 5b). The $3n$-bit input residue generator mod 2^n+1 (intended only for the 3-moduli set $\{2^n, 2^n-1, 2^n+1\}$) providing directly the D1 output was proposed in [14] (Figs. 4 and 8). For some moduli sets, including those which contain the pair of the conjugate moduli like $2^n \pm 1$, hardware savings can be obtained by using some shared circuitry to design input converters. The problem of hardware sharing between various residue generators for conjugate moduli $2^n \pm 1$ was considered in [19], [20]. However, these works deal only with the standard representation of residues mod 2^n+1. To our best knowledge, no design methods of the multi-residue generators for conjugate moduli $2^n \pm 1$ using shared logic and providing diminished-1 representation have been reported yet. Obviously, the most evident (but not necessarily the most efficient) solution would be to use the bi-residue generator mod $2^n \pm 1$ from [20] whose mod 2^n+1 output feeds the normal-to-D1 converter, such as for example one from [13].

Therefore, the goal of this paper is to study the possibility of designing efficient residue generators mod $2^n \pm 1$, which would produce directly the operand in D1 form for any number of input bits p and would be easily amenable for hardware sharing with the mod 2^n-1 residue generator.

This paper is organized as follows. In Section II, some theoretical background on the D1 representation and the periodicity properties of the series of $|2^k|_A$ is summarized. In Section III, a new architecture of the universal residue generator mod 2^n+1 providing the residue directly in D1 form, is detailed along with the possibility of its extension to the bi-residue generator mod $2^n \pm 1$. Conclusions are given in Section IV.

II. Preliminaries

A. Modulo 2^n+1 Diminished-1 (D1) Representation

The modulo 2^n+1 diminished-1 (D1) representation, which was introduced in [11], includes a zero indication bit. A number $X \in [0, 2^n+1)$ is represented as $X^* = (x_z\ x_{n-1} \dots x_0)$, where x_z is the zero indication bit and $X_{-1} = (x_{n-1} \dots x_0)$ is the diminished-1 magnitude of X. Formally, the terms x_z and X_{-1} are defined as

$$x_z = \begin{cases} 0 & \text{if } X \neq 0 \\ 1 & \text{if } X = 0 \end{cases} \tag{1}$$

so that $X = \bar{x}_z + X_{-1}$.

B. Periodicity Properties of the Series of $|2^k|_{2^n \pm 1}$

In the designs considered here, we will need the following notions, which characterize the periodicity of the series of $\left|2^k\right|_{2^n \pm 1}$ and are particularly useful to design efficient arithmetic circuits mod $2^n \pm 1$ [18]. The practical importance of periodicity stems from the following equations, which hold for any nonnegative integer j:

$$\left|2^{jn+k}\right|_{2^n-1} = \left|2^k\right|_{2^n-1} \tag{2}$$

$$\left|2^{jn+k}\right|_{2^n+1} = (-1)^j \left|2^k\right|_{2^n+1}. \tag{3}$$

In particular, for $k=0$ the above equations take the form:

$$\left|2^{jn}\right|_{2^n-1} = 1 \tag{4}$$

$$\left|2^{jn}\right|_{2^n+1} = \begin{cases} 1 & \text{if } j \text{ even} \\ 2^n = |-1|_{2^n+1} & \text{if } j \text{ odd} \end{cases} \tag{5}$$

In [18], it was shown how to exploit these properties to simplify designing residue generators and multi-operand modulo adders (MOMAs) by using carry-save adders (CSAs) with end-around carry (EAC). In [18], it was shown that designing any arithmetic circuit taking advantage of Eqn (3) allows to invert all signals of weight $\left|2^{jn+k}\right|_{2^n+1}$ for odd j and handle them as signals of weight $\left|2^k\right|_{2^n+1}$, provided that the correction constant equal to $|-2^k|_{2^n+1}$ is added. To avoid unnecessary multiple additions of corrections, we can apply a simple general rule to calculate the cumulative correction value for any arithmetic circuit taking advantage of Eqn (3), given in [21]: COR_{2^n+1} is obtained as the cumulative sum of all inverted signals of weight $\left|2^k\right|_{2^n+1}$ that appear in the circuit taken mod 2^n+1, which can be added at some stage of computation. In all circuits which will be considered here, the total correction COR_A must be taken into account prior the final D1 representation is obtained.

Eqns (4) and (5) constitute the theoretical background for designing bi-residue generators for the conjugate moduli $2^n \pm 1$.

III. Design of Residue Generators mod 2^n+1 with D1 Output

A. Architecture Designed According to [18]

Here, we will first present the design method of residue generators mod 2^n+1 according to [18] and then we will consider the possibilities to generate the output in D1 form.

We assume that the input p-bit vector X is sufficiently large, so that it can be partitioned into $r = \lceil p/n \rceil > 2$ n-bit blocks B_j, beginning with the least significant bits (LSBs), i.e., $X = (B_{r-1} \dots B_1 B_0)$, where $B_0 = (x_{n-1} \dots x_1 x_0)$, $B_1 = (x_{2n-1} \dots x_{n+1} x_n)$, etc. If p does not divide n, the block B_{r-1} containing the most significant bits (MSBs) is padded with $r \cdot n - p$ leading 0s.

Due to (3) we have

$$|X|_{2^n+1} = \left| \sum_{j=0}^{r-1} 2^{j \cdot n} \cdot B_j \right|_{2^n+1} = \left| \sum_{j=0}^{r-1} (-1)^j B_j \right|_{2^n+1} \tag{6a}$$

$$= \left| \left(\sum_{j=0,\, j \text{ even}}^{r-1} B_j \right) - \left(\sum_{j=1,\, j \text{ odd}}^{r-1} B_j \right) \right|_{2^n+1}. \tag{6b}$$

For any odd j, we can benefit from the following equality to replace the second part of Eqn (6b) as follows

$$\left| -B_j \right|_{2^n+1} = \left| \bar{B}_j - \left(\sum_{i=0}^{n-1} b_{j,i} 2^i \right) \right|_{2^n+1} = \bar{B}_j + 2. \tag{7}$$

Eqn (7) indicates the correction constant equal to 2, which must be added mod 2^n+1 to the final result.

Basically, Eqns (6b) and (7) can be used as a starting point to design the residue generator mod 2^n+1 with D1 output.

TABLE I
THE CORRECTIONS REQUIRED DUE TO COMPLEMENTED SIGNALS.

p	B_1	B_3	B_5	CSA Stages 1–3	$COR(p,9)$
16	-7	-7	-1	$-2-1-1$	$\|-19\|_9 = 8$
17	-7	-7	-3	$-2-1-1$	$\|-21\|_9 = 6$
18	-7	-7	-7	$-2-1-1$	$\|-25\|_9 = 2$

However, the following example reveals some problems involved with such a design approach.

Example 1: Consider the design of three residue generators mod 9 according to Eqn (6b) for $p = \{16, 17, 18\}$. Initially, the set of $p = 18$ input bits is partitioned into $r = \lceil p/3 \rceil$ 3-bit blocks B_0, B_1, B_2, B_3, B_4, and B_5 (with padded two and one 0's, respectively for $p = 16$ and $p = 17$), in which all nonzero bits of the odd-numbered blocks B_1, B_3, and B_5 are complemented. To construct the CSA tree, the bits of the blocks B_k, $0 \le k \le 5$, are partitioned into $HP(9) = 3$ disjoint sets G_k, $0 \le k \le 2$, containing the bits of the same weight $|2^k|_9$, i.e.:

$$\begin{aligned} G_0 &= \{x_0, \bar{x}_3, x_6, \bar{x}_9, x_{12}, \bar{x}_{15}\} \\ G_1 &= \{x_1, \bar{x}_4, x_7, \bar{x}_{10}, x_{13}, \bar{x}_{16}\} \\ G_2 &= \{x_2, \bar{x}_5, x_8, \bar{x}_{11}, x_{14}, \bar{x}_{17}\}. \end{aligned}$$

(Obviously, for $p = 16$ the bits $\bar{x}_{16}$ and $\bar{x}_{17}$ are omitted; similarly, for $p = 17$ the bit $\bar{x}_{17}$ is omitted.) The CSA parts of these residue generators can be described using the following shorthand notation introduced in [18]. The contents of a column G_k alternately indicates either how many bits of residue weight $|2^k|_9$ are present at a given stage of computation or specifies the number of full-adders (FAs) and half-adders (HAs) that operate on the bits from a given set G_k (the current number of such bits is provided by the entry in the same column in the preceding row).

(a)

G_2	G_1	G_0	
5	5	6	
FA HA	FA HA	2 FAs	CSA Stage 1
4	4	4	
FA	FA	FA	CSA Stage 2
3	3	3	
FA	FA	FA	CSA Stage 3
2	2	2	

(b)

G_2	G_1	G_0	
5	6	6	
FA HA	2 FAs	2 FAs	CSA Stage 1
4	4	4	
FA	FA	FA	CSA Stage 2
3	3	3	
FA	FA	FA	CSA Stage 3
2	2	2	

(c)

G_2	G_1	G_0	
6	6	6	
2 FAs	2 FAs	2 FAs	CSA Stage 1
4	4	4	
FA	FA	FA	CSA Stage 2
3	3	3	
FA	FA	FA	CSA Stage 3
2	2	2	

Fig. 1. Shorthand notation of the CSA tree for the residue generator mod 9 with: (a) $p = 16$; (b) $p = 17$; and (c) $p = 18$ inputs.

In Fig. 1, it is seen that the CSA trees are virtually identical for the three values of p with four identical inverted EACs (two for Stage 2 and one for Stage 2 and 3); the only difference results from one or two HAs replacing FAs for $p = 17$ and $p = 16$, respectively. The final column of Table I shows that, despite that the CSA tree reduces the input bits to the same set of six equally distributed bits, in each case the nonzero correction that must be added by the final adder mod $2^n + 1$ differs. Consequently, the final adder which generates the residue mod $2^n + 1$ in D1 form must be adapted to include the correction depending on the number of inputs p. ■

B. New Universal Architecture

The alternative new architecture of the residue generators mod $2^n + 1$ proposed here will not have the previously indicated drawback. Moreover, besides generating the D1 output for any arbitrary p without the need to add any correction due to complemented signals, it will also take into account the possibility of hardware sharing with the residue generator mod $2^n - 1$. The latter relies on using two following equations:

$$\left| |X|_{2^{2n}-1} \right|_{2^n-1} = |X|_{2^n-1} \quad (8)$$

$$\left| |X|_{2^{2n}-1} \right|_{2^n+1} = |X|_{2^n+1}, \quad (9)$$

which are the special cases of the well-known identity

$$|a|_b = ||a|_{bc}|_b. \quad (10)$$

We assume that the input p-bit vector X is partitioned into $q = \lceil p/(2n) \rceil$ $2n$-bit blocks D_j, beginning with the LSBs, i.e., $X = (D_{q-1} \dots D_1 D_0)$, where $D_0 = (x_{2n-1} \dots x_1 x_0)$, $D_1 = (x_{4n-1} \dots x_{2n+1} x_{2n})$, etc. If p does not divide n, the block D_{q-1} containing the MSBs is padded with $q \cdot n - p$ leading 0s. We assume that p is sufficiently large, so that X can be partitioned into

$$q = \lceil p/2n) \rceil \ge 4 \quad (11)$$

$2n$-bit blocks.

First, the q-operand $2n$-bit CSA with EAC reduces p input bits to a pair of $2n$-bit vectors D_C and D_S by realizing the equation

$$\left| \sum_{j=0}^{s-1} D_j \right|_{2^{2n}-1} = |D_C + D_S|_{2^{2n}-1}. \quad (12)$$

Now each of thus obtained $2n$-bit vectors D_C and D_S can be split into a pair of n-bit groups containing n MSBs and LSBs denoted, respectively, by the indexes H and L: $D_C = (D_{C,H} \| D_{C,L})$ and $D_S = (D_{S,H} \| D_{S,L})$. By taking into account that $D_C = 2^n D_{C,H} + D_{C,L}$, $D_S = 2^n D_{S,H} + D_{S,L}$, and $|2^n|_{2^n+1} = |-1|_{2^n+1}$, we obtain the identity

$$\begin{aligned} |X|_{2^n+1} &= \left| |D_C + D_S|_{2^{2n}-1} \right|_{2^n+1} \\ &= \left| D_C + D_S \right|_{2^n+1} \\ &= \left| (D_{C,H} \| D_{C,L}) + (D_{S,H} \| D_{S,L}) \right|_{2^n+1} \\ &= \left| 2^n D_{C,H} + D_{C,L} + 2^n D_{S,H} + D_{S,L} \right|_{2^n+1} \\ &= \left| - D_{C,H} + D_{C,L} - D_{S,H} + D_{S,L} \right|_{2^n+1} \end{aligned} \quad (13)$$

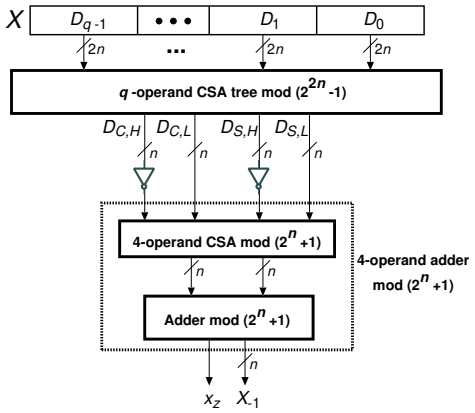

Fig. 2. New residue generator mod 2^n+1 with D1 output.

Eqn (13) can be realized using the 4-operand CSA mod 2^n+1 followed by the special adder mod 2^n+1 to obtain the $(n+1)$-bit vector X^*, which appears directly in D1 form (see the proof of Eqn (17) given below). (As the final adder mod 2^n+1 with D1 output can be used one described in [22], whose detailed structure can be found on Fig. 7 in [12].) Figure 2 shows the internal structure of the new residue generator mod 2^n+1 with D1 output, designed according to the above procedure. Obviously, for $p=4n$, the whole circuits reduces to the final 4-operand adder mod 2^n+1.

The theorem given below will prove that the output produced by the circuit of Fig. 2 is in the D1 form indeed. However, besides the following identity ($a \in \{0,1\}$)

$$-a = \bar{a} - 1, \tag{14}$$

we will need the following properties.

Property 1: For three natural numbers $0 \le x, y, z < 2^n$, the following equation holds:

$$\begin{aligned} &|x+y+z|_{2^n+1} \\ &\quad \overset{\text{CSA}}{=} |2c+s|_{2^n+1} \\ &\quad = \left|2\cdot 2^{n-1}c_{n-1} + 2(c_{n-2}\ldots c_0) + s\right|_{2^n+1} \\ &\quad = |2^n c_{n-1} + 2(c_{n-2}\ldots c_0) + s|_{2^n+1} \\ &\quad = |2(c_{n-2}\ldots c_0) - c_{n-1} + s|_{2^n+1} \\ &\quad = |2(c_{n-2}\ldots c_0) + \bar{c}_{n-1} - 1 + s|_{2^n+1} \\ &\quad = |(c_{n-2}\ldots c_0 \| c_{n-1}) - 1 + s|_{2^n+1} \end{aligned} \tag{15}$$

Property 2: For three natural numbers $0 \le x, y < 2^n$ and $0 \le t \le 2^n$, the following equation holds:

$$\begin{aligned} |x+y|_{2^n+1} &= |2^n c_{n-1} + s|_{2^n+1} \\ &= |s - c_{n-1}|_{2^n+1} \\ &= |s + \bar{c}_{n-1} - 1|_{2^n+1} \\ &= |t-1|_{2^n+1}\,. \end{aligned} \tag{16}$$

Property 1 will be used twice to justify the computations executed by two subsequent CSA stages, whereas Property 2 will be used to evaluate the final result provided by the special version of the final adder mod 2^n+1.

Theorem 1: For the circuit of Fig. 2 the following two equations hold:

$$\begin{aligned} \left|X\right|_{2^n+1} &= \left|D_{C,L} - D_{C,H} + D_{S,L} - D_{S,H}\right|_{2^n+1} \\ &= \left|X^*+1\right|_{2^n+1} \end{aligned} \tag{17}$$

and

$$|X-1|_{2^n+1} = X^*. \tag{18}$$

Proof. Here, we will use the following identities: $|-D_{C,H}|_{2^n+1} = |\bar{D}_{C,H}+2|_{2^n+1}$ and $|-D_{S,H}|_{2^n+1} = |\bar{D}_{S,H}+2|_{2^n+1}$. Then

$$\begin{aligned} &\left|X\right|_{2^n+1} \\ &\quad \overset{(13)}{=} \left|D_{C,L} - D_{C,H} + D_{S,L} - D_{S,H}\right|_{2^n+1} \\ &\quad = \left|D_{C,L} + \bar{D}_{C,H} + 2 + D_{S,L} + \bar{D}_{S,H} + 2\right|_{2^n+1} \\ &\quad \overset{(15)}{=} \left|D_{C,1} + D_{C,2} - 1 + 2 + \bar{D}_{S,H} + 2\right|_{2^n+1} \\ &\quad = \left|D_{C,1} + D_{C,2} + \bar{D}_{S,H} + 3\right|_{2^n+1} \\ &\quad \overset{(15)}{=} \left|D_{C,3} + D_{C,4} - 1 + 3\right|_{2^n+1} \\ &\quad = \left|D_{C,3} + D_{C,4} + 2\right|_{2^n+1} \\ &\quad \overset{(16)}{=} \left|X^* - 1 + 2\right|_{2^n+1} \\ &\quad = \left|X^*+1\right|_{2^n+1} \end{aligned} \tag{19}$$

By subtracting 1 from both sides of Eqn (19), we obtain

$$\begin{aligned} \left|X-1\right|_{2^n+1} &= \left|X^*+1-1\right|_{2^n+1} \\ &= \left|X^*\right|_{2^n+1} = X^*. \quad \blacksquare \end{aligned}$$

We have shown that for any p, $COR(2^n+1, p) = 0$, i.e., no correction needs to be added to obtain the D1 output. This is because the initial s-operand CSA tree mod $(2^{2n}-1)$ contains no inverted signals, whereas the rest of the circuit remains identical for any p.

C. Design of Bi-residue Generators mod $2^n \pm 1$

The architecture of the bi-residue generator mod $2^n \pm 1$ results from the following straightforward application of Eqns (8) and (12)

$$\begin{aligned} |X|_{2^n-1} &= \left|\,|D_C + D_S|_{2^{2n}-1}\,\right|_{2^n-1} \\ &= \left|D_C + D_S\right|_{2^n-1} \\ &= \left|(D_{C,H}\|D_{C,L}) + (D_{S,H}\|D_{S,L})\right|_{2^n-1} \\ &= \left|2^n D_{C,H} + D_{C,L} + 2^n D_{S,H} + D_{S,L}\right|_{2^n-1} \\ &= \left|D_{C,H} + D_{C,L} + D_{S,H} + D_{S,L}\right|_{2^n-1}. \end{aligned} \tag{20}$$

Obviously, the q-operand $2n$-bit CSA with EAC, which reduces p input bits to a pair of $2n$-bit vectors D_C and D_S according to Eqn (12) can be shared. The detailed internal

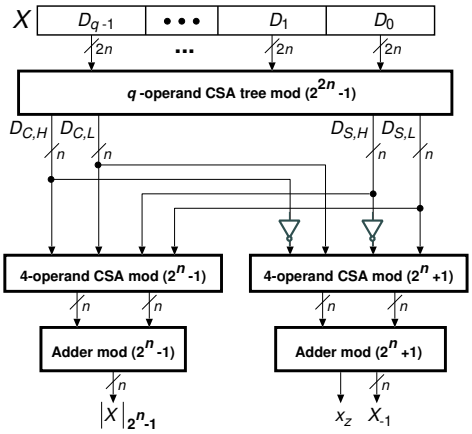

Fig. 3. New bi-residue generator mod $2^n \pm 1$ with D1 output.

structure of thus obtained bi-residue generator mod $2^n \pm 1$ is shown in Fig. 3. Its upper part allows to save $p-4n$ full-adders (FAs).

IV. Conclusions

The diminished-1 (D1) encoding has been known for several years as the efficient approach which could improve performance of residue arithmetic circuitry modulo $2^n + 1$ in arithmetic units using RNS. In this paper, we have proposed the new architecture of the p-input residue generator mod $2^n + 1$ with D1 output. It can be useful to build an input converter for any RNS moduli set containing one or more moduli of the form $2^n + 1$. The circuit is universal, because its initial part can be easily designed for an arbitrary $p \geq 4n$, whereas its final block—the 4-operand adder mod $2^n + 1$—preserves the same structure for any p. The latter feature was shown essential for the possible easy extension to build p-input bi-residue generators mod $2^n \pm 1$ with shared logic, which allows to save $p - 4n$ full-adders. The latter design can be useful for any set of RNS moduli containing a pair of conjugate moduli $2^n \pm 1$. As far as we know, to date no general design methods of residue generators mod $2^n + 1$ using shared logic with an arbitrary number of inputs p and providing the input in D1 form have been presented yet.

References

[1] S. Elango, P. Sampath, S. Raja Sekar, S. P. Philip, and A. Danielraj, "High-performance multi-RNS-assisted concurrent RSA cryptosystem architectures," *J. Circuits, Syst. & Comput.*, vol. 32, no. 15, p. 2350255, 2023.

[2] S. J. Piestrak and K. S. Berezowski, "Architecture of efficient RNS-based digital signal processor with very low-level pipelining," in *Proc. IET Irish Sign. & Syst. Conf.*, Galway, Ireland, 18–19 June 2008, pp. 127–132.

[3] G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, A. Nannarelli, M. Petricca, and M. Re, "Design space exploration based methodology for residue number system digital filters implementation," *IEEE Trans. on Emerging Topics in Computing*, vol. 10, no. 1, pp. 186–198, 2020.

[4] S. R. Barraclough, M. Sotheran, K. Burgin, A. P. Wise, A. Vadher, W. P. Robbins, and R. M. Forsyth, "The design and implementation of the IMS A110 image and signal processor," in *Proc. IEEE Custom Integrated Circuits Conf. (CICC)*, San Diego, CA, USA, 15–18 May 1989, pp. 24.5–1–24.5–4.

[5] P. Patronik and S. J. Piestrak, "Hardware/software approach to designing low-power RNS-enhanced arithmetic units," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 5, pp. 1031–1039, May 2017.

[6] B. Deng, B. Nadendla, K. Suo, Y. Xie, and D. C.-T. Lo, "Fixed-point encoding and architecture exploration for residue number systems," *ACM Trans. on Architecture and Code Optimization*, vol. 21, no. 3, pp. 1–27, 2024.

[7] C. Chang, A. Molahosseini, A. Zarandi, and T. Tay, "Residue number systems: A new paradigm to datapath optimization for low-power and high-performance digital signal processing applications," *IEEE Circ. & Syst. Mag.*, vol. 15, no. 4, pp. 26–44, Fourth Quarter 2015.

[8] A. Daher, E. Baghious, N. El Khouja, E. Radoi, and G. Burel, "Fast algorithm for optimal design of Fermat number transform based block digital filters," *Digital Signal Processing*, vol. 113, p. 103029, 2021.

[9] Z. Baozhou, N. Ahmed, J. Peltenburg, K. Bertels, and Z. Al-Ars, "Diminished-1 Fermat number transform for integer convolutional neural networks," in *Proc. IEEE 4th Int. Conf. on Big Data Analytics (ICBDA)*, Suzhou, China, 15–18 March 2019, pp. 47–52.

[10] Y. Xing, R. W. Luk, A. I. Sanka, Z. Ye, D. Chen, H. Yan, and R. C. Cheung, "Low-complexity chromatic dispersion compensation using high-radix Fermat Number Transform," *Journal of Lightwave Technology*, vol. 42, no. 15, pp. 5190–5203, 2024.

[11] L. Leibowitz, "A simplified binary arithmetic for the Fermat number transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 24, no. 5, pp. 356–359, Oct. 1976.

[12] H. T. Vergos, C. Efstathiou, and D. Nikolos, "Diminished-one modulo $2^n + 1$ adder design," *IEEE Trans. Comput.*, vol. 51, no. 12, pp. 1389–1399, Dec. 2002.

[13] H. T. Vergos and D. Bakalis, "On implementing efficient modulo $2^n + 1$ arithmetic components," *J. Circuits, Syst. & Comput.*, vol. 19, no. 5, pp. 911–930, 2010.

[14] G. Jaberipur, A. Belghadr, and S. Nejati, "Impact of diminished-1 encoding on residue number systems arithmetic units and converters," *Comput. Electr. Eng.*, vol. 75, pp. 61–76, May 2019.

[15] C. Efstathiou, N. Moshopoulos, I. Voyiatzis, and K. Pekmestzi, "On the design of modulo $2^n + 1$ dot product and generalized multiply–add units," *Comput. Electr. Eng.*, vol. 39, no. 2, pp. 410–419, Feb. 2013.

[16] K. Tsoumanis, K. Pekmestzi, and C. Efstathiou, "Fused modulo $2^n + 1$ add-multiply unit for diminished-1 operands," in *Proc. Int. Conf. Modern Circ. & Syst. Technologies (MOCAST)*, 2016, pp. 1–4.

[17] E. Vassalos, D. Bakalis, and H. T. Vergos, "Reverse converters for RNSs with diminished-one encoded channels," in *Proc. IEEE EUROCON*, Zagreb, Croatia, 1–4 July 2013, pp. 1798–1805.

[18] S. J. Piestrak, "Design of residue generators and multioperand modular adders using carry-save adders," *IEEE Trans. Comput.*, vol. 43, no. 1, pp. 68–77, Jan. 1994.

[19] F. Pourbigharaz and H. M. Yassine, "Simple binary to residue transformation with respect to $2^m + 1$ moduli," *IEE Proc. Circuits, Devices & Syst.*, vol. 141, no. 6, pp. 522–526, Dec. 1994.

[20] S. J. Piestrak, "Design of multi-residue generators using shared logic," in *Proc. IEEE Int. Symp. Circ. & Syst. (ISCAS)*, Rio de Janeiro, Brazil, 15–18 May 2011, pp. 1435–1438.

[21] ——, "Design of squarers modulo A with low-level pipelining," *IEEE Trans. Circuits Syst. II*, vol. 49, no. 1, pp. 31–41, Jan. 2002.

[22] R. Zimmermann, "Efficient VLSI implementation of modulo $(2^n \pm 1)$ addition and multiplication," in *Proc. 14th IEEE Symp. on Comput. Arithm.*, Adelaide, Australia, 14–16 Apr. 1999, pp. 158–167.